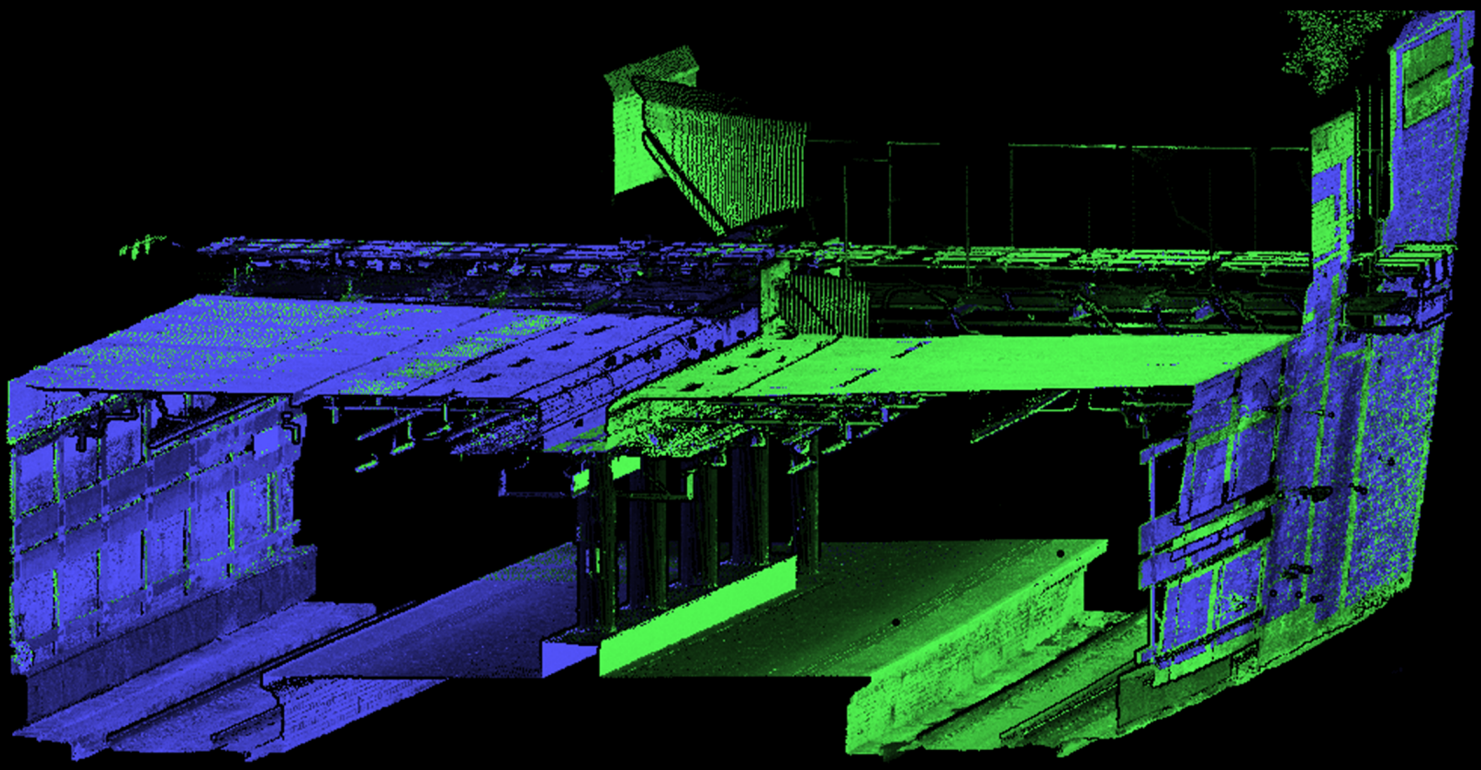
Point Cloud Feature Extraction for Trajectory Optimization

MSc Thesis

Arsenijs Nitijevskis

Delft University of Technology



TU Delft

Point Cloud Feature Extraction for Trajectory Optimization

MSc Thesis

by

Arsenijs Nitijevskis

Thesis committee:	Dr. R. C. (Roderik) Lindenbergh	TU Delft, supervisor
	Dr. A. R. (Alireza) Amiri-Simkooei,	TU Delft
	Ir. Luc Amoureus,	Fugro, supervisor
Student number:	4906578	
Project duration:	May, 2023 – March, 2024	

TU Delft

Summary

The railway industry is constantly growing to meet the demand of society for stable, accessible and sustainable transportation. With this growth, the need for the railway to be reliable increases, requiring frequent surveying and maintenance. Fugro's RILA (Rail Infrastructure aLignment Acquisition) mobile mapping system contributes by making the surveying more accessible to the relevant railway network stakeholders. However, the system has its limitations in environments where the GNSS (Global Navigation Satelite System) signal is occluded, such as tunnels and underground stations. The geo-data collected by RILA in those areas is poorly georeferenced due to the poorly tracked trajectory of the system, which introduces spatial data misalignment up to a meter or more. The current methods to fix data misalignment rely on manual data corrections, which is not cost-effective, or on automatic solutions, which have limited applicability.

Thus, the aim of this research is to develop an improved trajectory optimization method, thereby ensuring accurate geo-referencing and alignment of the survey data. This thesis proposes a newly developed methodology to achieve this aim: features are extracted from point cloud surveys, matched and utilized by g^2o optimizer and GNSS processing software to optimize the trajectory. The development is described and results are evaluated on two different scales - locally, within a point cloud tile and globally, within a sequence of tiles. It is done by using Glasgow's underground railway network as a test case.

Results from the implementation demonstrate significant improvements in trajectory accuracy - a misalignment of point cloud data was reduced from a 1.5 m to a cm level within an optimization time frame that took approximately 10 hours. This improvement in accuracy was present under different complex environments using both the local and global versions of the algorithm. However, the area near the railway tunnel entrance saw a limited benefit from the implementation of the proposed algorithm.

In conclusion, the developed trajectory optimization algorithm optimizes the trajectory and improves the alignment of the survey data. Moreover, the method outperforms the currently employed solutions by being automatic and applicable in different environments. However, further research is required to optimize the algorithm itself (accuracy and computationally speed of the algorithm) and to more accurately define its limitations in terms of the surveyed environments.

Contents

1 Introduction

This chapter (1) aims to introduce readers to the foundational elements of this research. It begins with Subsection 1.1, where the core system of our study, the mobile mapping system RILA, is presented. This includes a discussion on the problems RILA aims to solve and its technical specifications. Following this, Subsection 1.2 highlights the current limitations of RILA and the solutions that are currently employed in the processing chain. This segment introduces an examination of potential areas for advancement, which transitions to Subsection 1.3, where the research objectives and the research questions are presented.

1.1 Introduction to RILA system

Currently, rail transport plays a crucial role in global transportation, contributing significantly to the world's motorized passenger movements (8%) and freight transport (7%), while only accounting for 2% of energy use in the transportation sector (International Energy Agency, 2019). Moreover, rail transport is one of the greenest and most energy-efficient transportation alternatives. In terms of emissions, rail outperforms both road and air transport - for instance, transporting the same load by cars would lead to a 15% increase in oil consumption (International Energy Agency, 2019). Furthermore, the rail sector's high degree of electrification allows for a diverse range of energy sources, including renewable and nuclear energy, reducing reliance on fossil fuels. Therefore, looking at past trends and considering ongoing development projects, it is projected that rail transport utilization will grow in response to increasing economic demands (International Energy Agency, 2019; Knapcikova & Konings, 2018).

This anticipated growth brings into focus the importance of maintenance and expansion of the rail network. As the usage of the network grows, demand for its stability and reliance grows as well. Any disruptions in the railway network can have significant consequences, impacting stakeholders both directly and indirectly involved with the rail industry—ranging: from travel limitations for passengers and delays in freight goods to severe catastrophic events. To ensure efficient and safe utilization of the railway network, potential hazards must be mitigated beforehand. This necessitates proactive measures such as regular and thorough surveying of the rail network to identify areas that require immediate attention. This process involves two critical steps: identification of problematic segments through surveys, and subsequent mitigation activity.

The Rail department at Fugro contributes to keeping railway systems safe by providing an accurate and reliable surveying tool - their in-house developed mobile mapping system RILA. Established in 2009, Fugro's RILA (Rail Infrastructure aLignment Acquisition) system represents a unique train-borne track measurement solution, significantly enhancing the functionality and safety of rail infrastructure surveillance (Secco, 2022). Mounted on a train, RILA efficiently conducts surveys at line speeds of up to 200 km/h, collecting data without disrupting regular train operations and eliminating the need for on-track human presence (H. Wang & Berkers, 2019).

The system comprises several sensors. It includes a Global Navigation Satellite System (GNSS) antenna and an Inertial Measurement Unit (IMU), which aid in accurately positioning the system. Additionally, A 360°Light Detection And Ranging (LiDAR) scanner, three video cameras, and two laser triangulation systems (rail stripers, also known as railscanners) are responsible for infrastructure surveying. All sensors are housed within a carbon fiber unit for enhanced durability and mobility, which is shown in Figure 1.1. Moreover, the sampling rates for sensors are shown in Table 1.1 in order to present estimated data density provided by the system. This composition of sensors allows for precise data collection under normal conditions and it provides various data products for structural health assessment. To mention, RILA allows to collect absolute and relative track geometry (H. Wang & Berkers, 2019). Precise absolute track geometry is necessary for long-term monitoring of the rail assets through correct dataset correspondence. The relative track geometry tracks the relative rail asset correspondence, such as distance between railheads (track gauge), railway curvature or collision analysis. RILA provides this data with a standard deviation of less than 8 mm in the horizontal direction and less than 12 mm in the vertical direction (H. Wang et al., 2021).

1.2 Problematization

However, RILA's effectiveness is challenged in environments with poor GNSS tracking, such as underground stations and tunnels, leading to significant trajectory estimation errors (Secco, 2022). In such scenarios, the standard deviation and accuracy of the geo-referenced point cloud deteriorate, causing misalignment that can reach meters in magnitude. Figure 1.2 illustrates this issue, where two point clouds are misaligned due to the underground environment.

In GNSS-limited environments, the RILA system depends solely on its Inertial Measurement Unit sensors. While accurate in short-term measurements, IMUs record data relative to their previous measurements, a

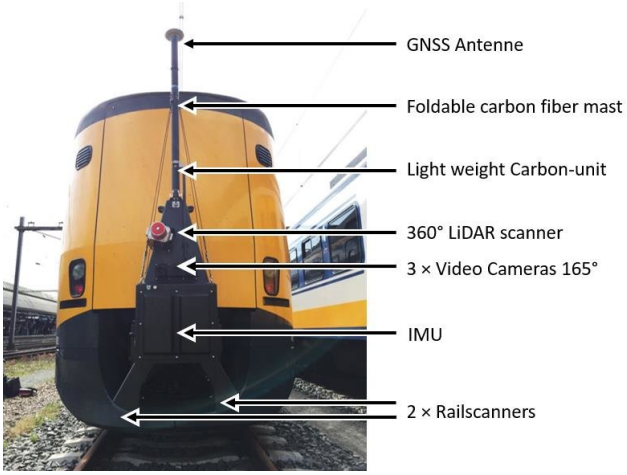


Figure 1.1: The RILA Mobile Mapping System featuring GNSS Antenna, LiDAR Scanner, Video Cameras, IMU, and rail stripers (H. Wang & Berkers, 2019).

Sensor	Sampling Frequency [Hz]	Quantity [-]	Sampling interval along the track [m]	
			Cruising speed 100 km/h	Cruising speed 160 km/h
GNSS antenna	5	1	5.56	8.89
IMU	300	1	0.09	0.15
LiDAR scanner	250 x 4000[1]	1	0.11	0.18
Rail striper	500 x 2352[2]	2	0.06	0.09
Video camera	15	3	1.85	2.96

[1]250 rounds per second and 4000 points per round.
[2]500 rail profiles per second and 2352 points per profile.

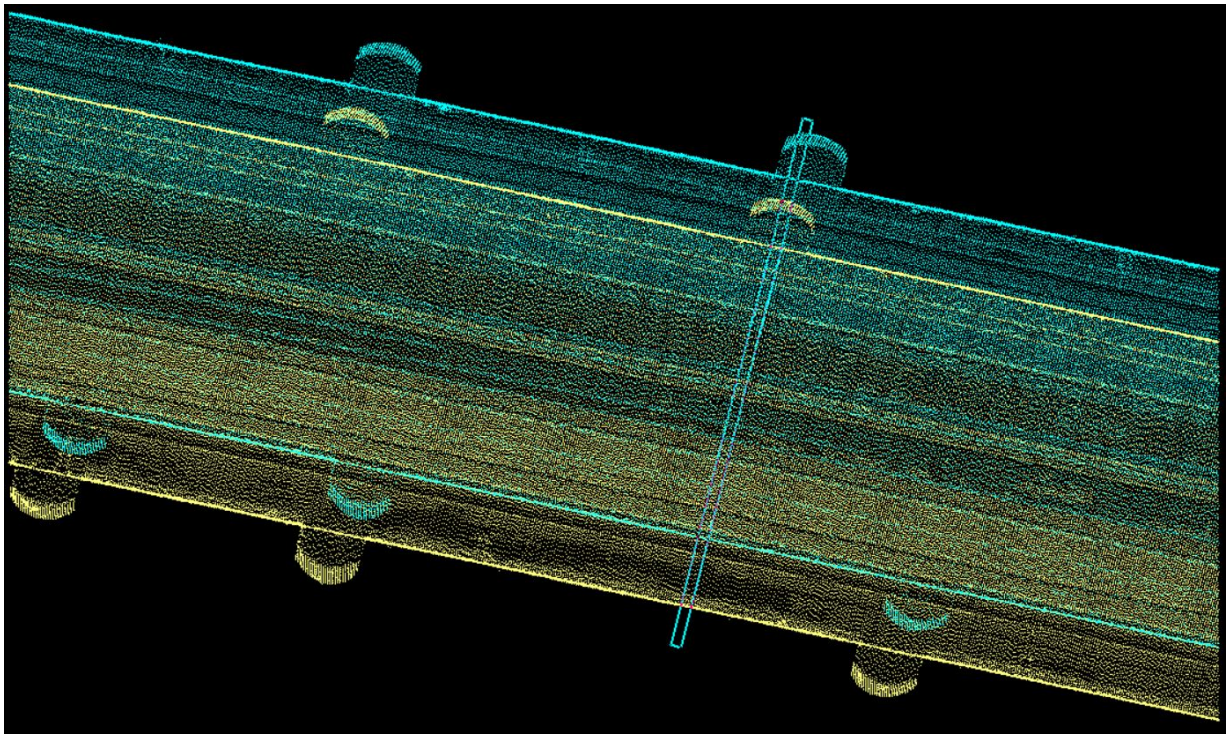
Table 1.1: Sampling frequency and expected sampling distance of RILA sensors (H. Wang et al., 2021).

process that inherently accumulates errors over time, a phenomenon also known as 'drift' (Elhashash et al., 2022; Kaplan & Hegarty, 2006). This drift continues and grows until the IMU can recalibrate with a GNSS signal. The resulting misaligned datasets are problematic for clients needing long-term, precise rail asset monitoring, since absolute track geometry measurements are impacted. The data is not only globally shifted due to the drift, but also, due to the growth of the drift, the shift itself is not uniform. This introduces different drift magnitudes at the beginning and the end of the survey.

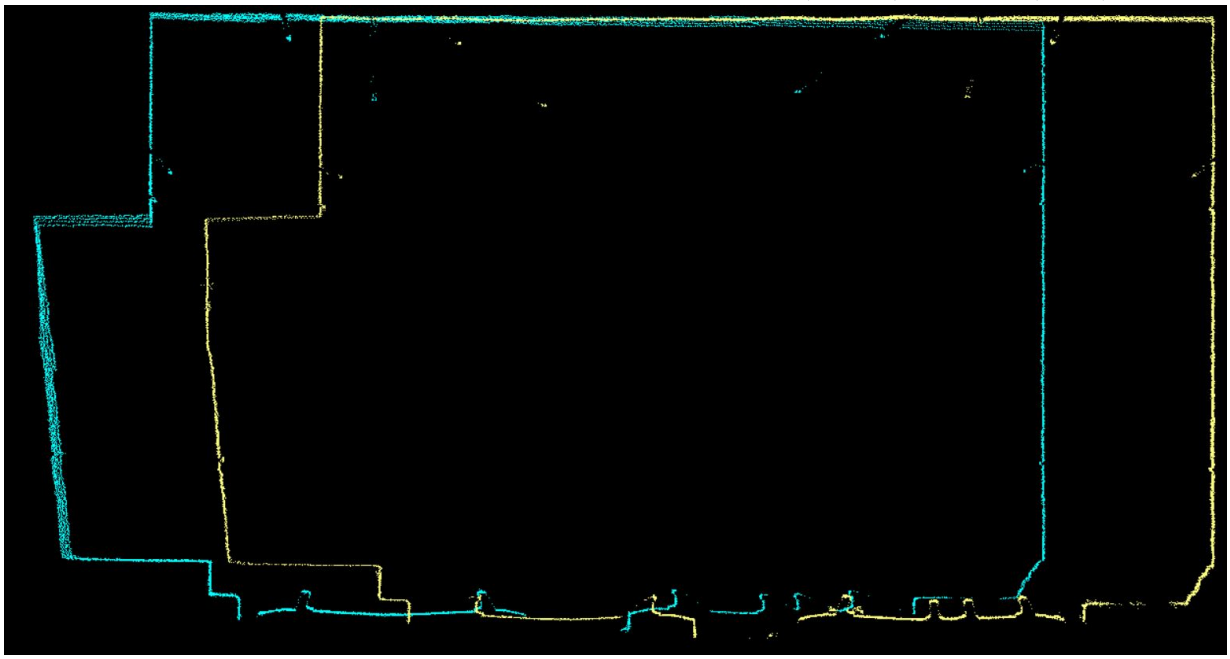
Due to the nature of this phenomenon and related survey aspects, some assumptions can be made about the present drift. More precisely, the drift affects only the positional updates of the sensor, while the IMU attitude remains reliable. Moreover, the rate of IMU drifting is relatively low, which allows us to assume it is uniform on a small scale, both time-wise and area-wise. Therefore, the misaligned point clouds can be well aligned by applying a translation vector, without the need for rotational adjustments. This misalignment vector can be split into two components: the misalignment in the normal direction (normal to the railway tracks) and the misalignment in the driving direction (along the railway tracks). This distinction is important, as some existing misalignment correction methods or visualization techniques may focus on these components separately.

Fugro Rail has implemented methods to mitigate these alignment issues, but each has its limitations. The main processing chain has already been automated sufficiently so that the current workflow produces accurate data alignment for the majority of the data. However, data collected in complex environments, such as tunnels and underground stations where IMU drift is present, currently requires a substantial amount of manual correction. It involves manually aligning trajectories by identifying and aligning common features in misaligned point clouds. This method is straightforward and does not require complex software. Additionally, it can be applied to correct both normal and driving direction mismatches. However, this method is labor-intensive, potentially requiring several months to correct a single dataset in complex environments. Furthermore, the manual nature of this process can introduce biases and errors into the corrections.

Another method utilizes the track centerlines of the surveyed and a parallel track, observed in both LiDAR point cloud and rail striper profiles. These centerlines are then used by the processing chain to correct the



(a) Top view of the surveyed point clouds. The cyan rectangle polygon corresponds to the location of the crossectional view visible in subfigure 1.2b. This view highlights the misalignment in both driving (≈ 10 cm) and normal (≈ 1 m) direction.



(b) Crossectional view of the surveyed point clouds. This view highlights the misalignment in normal (≈ 1 m) direction.

Figure 1.2: Example of the surveyed data misalignment. Cyan and Yellow point clouds correspond to the different acquisition campaigns performed in the same tunnel. The trajectories must be corrected to have properly georeferenced point clouds and to reduce the misalignment.

misalignment of the trajectories and the latest tool in such a chain is RailGraph. RailGraph is an in-house developed trajectory optimizer, which is based on the g^{2}o Optimizer. g^{2}o or "General Graph Optimization" is an open-source framework for optimizing graph-based non-linear error functions (Kummerle et al., 2011), and by relating trajectories with the observations of the track centerlines it is possible to use it for misalignment correction. While this approach has shown some improvements, its applicability is limited. It requires the presence and separate survey of a parallel track. In cases, when another track is not visible (for example, the

platform of an underground station hinders the visibility) or is just absent (a single track in a tunnel), then no corrections can be made with this method. Additionally, this trajectory correction method is mainly focused on normal direction corrections.

Finally, for correcting data in driving directions (assuming that the normal direction was corrected), the following algorithm exists in the data processing workflow:

1. The point clouds are projected through a top view.
2. A common object is identified in both point clouds (a mast or a pole, for example).
3. The projections of point clouds are treated as an original image and a shifted image. The shift is determined through image crosscorrelation and the identified object is used to assess it. The shift is constrained to happen parallel to the driving direction.

Thus, this algorithm fails, when no distinct objects are available for crosscorrelation assessment. Furthermore, it only corrects the driving direction misalignment.

1.3 Research Questions

These existing solutions, while partially effective, underscore the need for an automated and more complete approach to trajectory optimization in GNSS-poor environments. The goal of optimizing specifically trajectory is not governed only by aligning the point clouds, but rather by ensuring accurate geo-referencing of all collected data. By refining the trajectory, it ensures a consistent and smooth transition between point cloud segments, addressing both the geo-spatial alignment and the coherence of the data over larger areas. This approach mitigates the risk of discontinuities between point cloud tiles, ensuring a seamless and accurate transition between them and proper rail environment representation, which is not considered commonly with the point cloud alignment methods. Therefore, for this research, the aim is to develop an improved trajectory optimization method. More precisely, it will take the current results achieved with the g^{2}o optimizer and will tackle its current limitations. The focus will be on detecting and utilizing point cloud features in the g^{2}o optimizer other than track center lines.

Hence, the main research question is **"How to utilize point cloud features in rail-based tunnel environment for trajectory optimization?"**. It is expected to be answered during the development of this expanded g^{2}o workflow. The main question will involve answering the following sub-questions:

1. **How is the quality of trajectory solutions evaluated?**:
2. **What methods exist to optimize trajectory and how well do they suit rail-based tunnel environment?**
3. **What methods exist to identify and extract point cloud features in the tunnels?**
4. **What is the process for integrating the extracted features into a trajectory optimization workflow?**
5. **What improvements do these solutions offer over current methods and how generalizable the developed method is?**

To answer these questions, this research will conduct a thorough literature review, which will provide partial answers to research sub-questions 1 - 3. In addition to the literature review, answers to research sub-questions 4 - 5 (and partially to 1 - 3) will be derived from the practical work of this research. The research will involve developing an updated g^{2}o-based trajectory optimization workflow, to be tested on available RILA datasets. A complete software prototype will be developed, initially focusing on local point cloud processing, which includes local feature extraction and local trajectory optimization. This approach aims for robust development and easy scalability, as the developed algorithm subsequently is scaled to process longer trajectory segments by performing multi-tile feature extraction and global trajectory optimization.

2 Literature Review

In this chapter, an overview of the conducted literature study is presented. It begins by introducing the background theory related to Mobile Mapping Systems (MMS) in subsection 2.1, describing the main characteristics, components, and applications of systems like RILA. Additionally, the theory about the sensors used in this research - GNSS, IMU, and LiDAR - is presented. The following subsection 2.2 continues by describing core data-processing techniques and challenges for these sensors, including methods of sensor fusion. Subsection 2.3 reviews related research, focusing on trajectory optimization in GNSS-denied environments and how systems equipped with sensors similar to those in RILA, operating in similar environments, or systems deployed on the same platforms have addressed these challenges. This subsection continues the previous subsection by presenting works, which enhance the fundamental sensor fusion methods and how IMU drift is countered. The chapter concludes with section 2.4, highlighting the main takeaways from the literature for this research.

2.1 Mobile Mapping Systems and Sensors

Mobile Mapping Systems (MMS) are crucial in rapidly collecting accurate geo-data (Elhashash et al., 2022; Puente et al., 2013). This capability is important for a variety of applications:

- In railway infrastructure management there are reported projects performed with the help of MMS like RILA, Lynx, and RailMapper (Kremer & Grimm, 2012; Leslar et al., 2010; H. Wang & Berkers, 2019). These systems traverse the existing railway network (both outdoors and in tunnels) to collect information on its structural health and survey the surroundings for asset management, control of neighboring vegetation, and possible collision analysis. There are also special systems developed for tunnels surveying (rail and non-rail), to control the construction of tunnels and indicate any structural damage (Chapman et al., 2016; Sun et al., 2020).
- Other MMS have been effectively employed in non-rail environments, For example, for public infrastructure (car roads) in urban settings (Al-Bayari, 2019; Shen et al., 2024; K. Wang et al., 2021). This application is important for not only accumulating geo-data used to monitor road conditions, traffic dynamics, and infrastructure elements but also for developing autonomous driving vehicles.
- The MMS utilization also extends to monitoring natural environments, like forests, and water bodies. For surveying aquatic environments, MMS like FROG were developed, which successfully aids marine explorations and environmental studies (Menna et al., 2023). On the land, MMS were utilized to monitor forest assets, which are crucial for forest health assessment and preparing conservation strategies (Qian et al., 2016).
- Within mining operations, MMS are deployed and researched to enhance safety and operational efficiency. By providing detailed spatial data of underground environments, these systems are integral to the management and safety assurance in mining activities (Liu et al., 2022; Raval et al., 2019; Yoshida & Tadokoro, 2014).

In those examples, MMS were employed in different environments and on different platforms. However, despite the differences, all mentioned MMSs share the main features of an MMS. A typical MMS consists of an array of sensors mounted on a mobile platform. These sensors are fixed to the MMS frame, and their positions and orientations are precisely determined after a calibration process. The mentioned calibration is essential to correspond the data collected by multiple sensors accurately — each sensor samples data within its predefined reference frame and at a specific sampling frequency. Integrating data from multiple sensors necessitates a unified reference frame and time synchronization to ensure correct data correspondence (El-Sheimy, 2005).

The sensors in an MMS can be categorized based on their functional roles. For example, positional sensors, which are used for positional data collection and which are essential for georeferencing (Elhashash et al., 2022). The positional sensors form the cornerstone in determining the trajectory of a Mobile Mapping System. The MMS is conceptualized as a rigid body moving through three-dimensional (3D) space - A rigid body is a body with finite dimensions, which maintains the property that the relative positions of all its points, defined in a coordinate frame within the body, remain the same under rotation and translation. Therefore, the motion of the rigid body is characterized by six parameters: three for orientation and three for position, collectively known as a 'pose' (El-Sheimy, 2005). Estimating the pose of a rigid body in 3D space over time is the trajectory determination. The effectiveness of this determination depends on the MMS's proficiency in sensing the six defining quantities (position and orientation), which is typically achieved (but not limited to) through the GNSS and IMU sensors (El-Sheimy, 2005). The other classification of sensors includes sensors, that capture

detailed data of the environment: LiDARs capture the environment by sending laser pulses and collecting laser reflections resulting in 3D point clouds, cameras record the scene by controlling sensor or film exposure to the light, SONARs show the environment through sounds propagation and its reflection back through the medium, for example.

In RILA, the positioning is done by GNSS and IMU sensors, while environmental data is gathered through cameras, rail stripers, and LiDAR (H. Wang & Berkers, 2019). In this section, the fundamental principles behind GNSS, IMU and LiDAR are covered, while rail stripers, cameras and other sensors are occluded from the review due to the following reasons:

- Rail stripers - these sensors are based on the principle of laser triangulation and they capture an accurate shape of the railheads beneath them. They are useful for estimating the relative geometry of the railway, however the scope captured by the sensors is considered to be too narrow (within a meter from a railhead) and is not suitable for trajectory optimization.
- Cameras - are capable of capturing the environment in detail and can be used for trajectory optimization. For example, there is a successful use case of camera-aided surveying of tunnels (Chapman et al., 2016). However, cameras require good lighting conditions, while in tunnels mainly low lighting conditions are encountered. Therefore, the aforementioned survey utilized an LED light bank to counter it. In the case of RILA, the installation of additional equipment (such as an LED bank) is out of the scope of this research.
- Other sensors - this research focuses on trajectory optimization using the existing RILA sensors. More precisely, the LiDAR. Installation of additional sensors or any kind of hardware modification of the MMS is out of the scope of this research.

2.1.1 GNSS

The Global Navigation Satellite System (GNSS) is integral to Mobile Mapping Systems (MMS) like RILA, providing absolute positioning data in the global coordinate reference system (WGS84) through satellites orbiting Earth.

The GNSS signal sent by a satellite is an electromagnetic wave propagating at the speed of light. The distinct feature of the signal is that it is the modulation of the harmonic radio wave (termed the "carrier") with a characteristic pseudorandom noise (PRN) code (Kaplan & Hegarty, 2006). The GNSS receiver reads the transmitted signal and uses PRN code to distinguish the signals of different satellites. Additionally, the signal contains the navigational data (known as "navigation broadcast message"), which contains information about the transmitter satellite. The information on the orbit and clock offsets of the GNSS satellites allows the receiver to compute the position and velocity of the transmitting satellite at the signal transmission time (Teunissen & Montenbruck, 2017).

From the received carrier, the basic positional measurement can be made. The distance traveled by the signal can be estimated by calculating the time difference between the time, when the signal was broadcasted by the satellite, and the time, when the signal was received by the receiver. Scaled by the speed of light, it is called "pseudorange". Utilizing signals from four or more satellites, pseudorange observations provide localization accuracy to the meter or decimeter level (Teunissen & Montenbruck, 2017).

In addition, GNSS receivers measure the phase of the retrieved signal, referred to as the "Carrier phase". Given the signal frequency of 1.2-1.6 GHz, the signal wavelength ranges from 19-25 cm (Teunissen & Montenbruck, 2017). Therefore, accurate phase difference measurement between transmission and reception, along with correct estimation of the complete number of cycles in between, can enhance positioning accuracy to centimeter or even sub-centimeter levels. However, inaccuracies in cycle estimation can degrade results.

GNSS receivers also derive Doppler information from the signal, which is the frequency change of the signal due to the Doppler effect. This measures the range rate or line-of-sight velocity.

These measurements form the basis for computing the GNSS receiver's position and velocity. However, the accuracy of these calculations, for both pseudorange and carrier phase observations, depends on the signal's path from the satellites to the receiver. Various error sources influence the final accuracy of the positional solution and are shown in Table 2.1.

Some of the mentioned errors can be reduced or completely removed: proper satellite orbit modeling and troposphere information can reduce the corresponding errors, utilization of reference GNSS stations can remove the ionospheric delay, and investment into higher grade GNSS receiver can reduce the receiver noise (Teunissen & Montenbruck, 2017). However, some errors are environment-dependent. For example, multipath errors exist due to the reflection of the GNSS signal from the obstacles before it arrives at the receiver. Surveying in urban environments with tall buildings, in forests or canyons will produce large errors. Furthermore, GNSS signals are unavailable indoors, in tunnels, underwater, or underground.

Error Source	Contribution [m]
Signal-in-space (user) range error	
Broadcast satellite orbit	0.2 - 1.0
Broadcast satellite clock	0.3 - 1.9
Broadcast satellite orbit	0.0 - 0.2
User equipment error	
Unmodeled ionospheric delay	0 - 5
Unmodeled tropospheric delay	0.2
Multipath	0.2 - 1
Receiver noise	0.1 - 1
User equivalent range error	0.5 - 6

Table 2.1: Representative magnitudes of individual contributions to the GNSS user equivalent range error for estimates of the individual contributions (Teunissen & Montenbruck, 2017).

Despite these challenges, appropriate error mitigation techniques allow RILA to achieve sub-centimeter accuracy in GNSS positioning. GNSS is a valuable system for its worldwide satellite coverage and independent observation nature, meaning observational errors do not accumulate over time. This makes it particularly effective for railway data collection in open-sky environments. However, in locations like stations or tunnels where GNSS reception is poor, alternative positioning solutions are necessary.

2.1.2 IMU

An Inertial Measurement Unit (IMU) is a positional sensor that derives trajectory information based on the inertia of a body. Inertia is the propensity of bodies to maintain a constant translation and rotation velocity unless acted upon by external forces or torques.

IMUs measure variations in rotation and acceleration rates by incorporating gyroscopes and accelerometers (Grewal et al., 2001). The gyroscopes are responsible for sensing angular velocity - it describes the rotation of the body. Integrating over the angular velocities provides the data regarding the orientation changes of the body with respect to its initial orientation. Correspondingly, accelerometers measure the specific force exerted on them. These forces are essential for deducing the changes in the position of the body from an initial position, achieved through double integration over time. Typically, an IMU consists of at least three gyroscopes and three accelerometers to estimate the body's complete pose (El-Sheimy, 2005).

As mentioned, IMU measures the change in pose with respect to its previous measurement. Depending on the grade of the equipment, It can outperform the accuracy of GNSS due to the precision of IMU measurements. Moreover, it does not depend on the environment like GNSS - it can measure pose with the same accuracy in indoor and outdoor environments. Additionally, the sampling rates of IMU sensors are higher than typical GNSS sampling rates.

However, IMU is unable to survey the trajectory for a long time without any corrections from other sensors. Continuous estimation of new positions based on the previous estimations accumulates errors. This is also called 'drift'. Due to that, IMU positional measurements degrade over time and require calibration or synchronization with other sensors (such as GNSS) (El-Sheimy, 2005).

2.1.3 LiDAR

Light Detection and Ranging (LiDAR) is a remote sensing technology used for distance measurement by illuminating a target with laser light and recording the time for the reflected light to return to the sensor. The sent laser pulses travel with the known velocity - velocity of light, thus this time-of-flight system calculates distance as (Vosselman & Maas, 2014):

$$\rho = \frac{c}{n}\frac{\tau}{2} \tag{1}$$

ρ: Distance from the sensor to the reflection location

c: Speed of light (299 792 458 m/s)

τ: Time between pulse emission and reception

n: Refractive index correction factor, typically 1

By using an estimated range and knowing the LiDAR's orientation at the observation time, it is possible to provide 3D coordinates of the measured reflection location in the sensor frame. Adding the information from the navigational sensors allows to reference the points not only in the sensors frame but also in other global Coordinate Reference Systems (CRS). For that, both sensors should be time synchronized - a high-frequency navigation sensor is needed to match the sampling frequency of a LiDAR and provide sensors orientation and position at sampling time (El-Sheimy, 2005). This functionality is usually provided by IMU sensors. In RILA's case High frequency of IMU (300 Hz) sampling allows for synchronization with the the RILA's LiDAR (performing round scans at 250 Hz and 4000 points per round scan).

Typical LiDAR is capable of measuring thousands of points in a fraction of a second with a cm/mm accuracy. Resulting referenced measurements are termed point clouds, since every measurement can be visualized as a 3D point with additional information (not limited to and depending on the scanner) like intensity, number of pulse returns, and measurement time. This allows to accurately capture the 3D environment in short amounts of time, however the correct sampling of the points depends on the pulse reflection. Low or deflected reflection caused by material, by illumination angle or by disturbance within the pulse path will introduce errors in the measurement. Therefore, LiDAR performance deteriorates in hazardous weather conditions such as heavy rain, snow, and fog (Elhashash et al., 2022).

The pattern of how the consecutive measurements are performed is dependent on the structure of the LiDAR. There are rotating, solid-state and flash LiDARS (Elhashash et al., 2022):

- Rotating LiDARs - use rotating mirrors to redirect one or multiple laser beams. It is a popular choice for MMS due to the large Field-of-View (FoV), high Signal-to-Noise ratio (SNR) and dense point clouds.
- Solid-state LiDARs - use Micro-Electro-Mechanical Systems (MEMS) mirrors, embedded in a chip. Through it, the mirror can be controlled to follow a specific trajectory and is considered to be solid due to the absence of moving parts in the sensor.
- Flash LiDAR - illuminates the entire FoV with one pulse. It uses a 2D array of photodiodes to capture the laser returns. It usually has a smaller range than the rotating LiDARs and the FoV is limited by the sensor size.

RILA's LiDAR is a rotating LiDAR - as it rotates it captures the environment only in 2 dimensions. As the MMS moves, the 3rd dimension is introduced and the resulting helix-like scanning pattern is shown in Figure 2.1. Therefore, the density of the resulting point cloud depends on the velocity of the surveying train. Since RILA does not impose boundaries on the train movement, it varies through the acquisition campaign. Moreover, the wide coverage of LiDAR allows to capture the environment way beyond the train itself, which is impossible with the rail stripers, which cover only the nearest railheads. Additionally, the LiDAR can capture the environment in different environments and times of the day - while cameras require good lighting conditions, LiDARs are independent of that.

2.2 Fundamental Sensor Fusion Methods

The data provided from separate sensors can be valuable on its own. However, the quality of the final dataset can often be enhanced by fusing the outputs of multiple sensors. This enhancement is achieved by combining sensors, which complement each other's disadvantages. Moreover, the visualization of the collected data from some sensors, like LiDAR, is not possible without the provided trajectory, as it is essential for adding a third dimension in the case of RILA. Therefore, this section covers fundamental sensor fusion techniques and processing challenges. More precisely, it presents the integration of navigational sensors (GNSS + IMU), point cloud registration, and simultaneous localization and mapping (SLAM) solutions.

2.2.1 GNSS and IMU Data

Both navigational sensors GNSS and IMU are already sufficient for some navigational tasks and they both have their advantages. GNSS observations do not accumulate errors, however are sampled at low frequencies and depend on the environment. On the other hand, IMU provides frequent and accurate observations in any environment, however it tends to accumulate errors and requires frequent corrections to external sensors to counter the drift. To oppose the disadvantages of each sensor, methods were developed to combine the advantages of both sensors and reduce their limitations. The resulting fusion allowed to use of the independent GNSS observations to correct and limit the drift of the IMU and use accurate IMU measurements to provide navigational information in the absence of GNSS (due to signal occlusion or low sampling rate). This integration results in a more accurate trajectory, compared to the separate sensor solution (Puente et al., 2013).

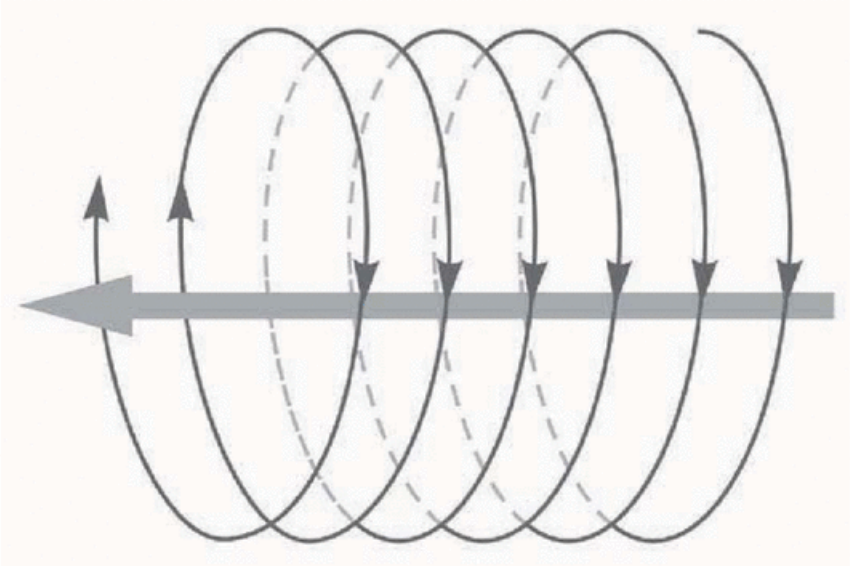


Figure 2.1: Point cloud scanning pattern. The big arrow shows the direction of the train, the helix structure shows the movement and scanning direction of the LiDAR scanner (Vosselman & Maas, 2014).

The integration of these sensors can happen at different levels. It is called 'Loosely Coupled', 'Tightly Coupled' and 'Ultra Tightly Coupled' systems (Qian et al., 2016; Teunissen & Montenbruck, 2017; Zhuang et al., 2023).

- Loosely Coupled (LC) system - integrates different sources of data, on a level of position and velocity, the raw data is not used.
- Tightly Coupled (TC) system - integrates data in ranging levels between transmitter and the receiver.
- Ultra Tightly Coupled (UTC) - integrates at a deeper level of raw measurements, such as raw carrier phase and code phase, from a GNSS receiver.

For fusing the sensors from various sensors in LC, TC or UTC manner, certain algorithms must be applied. There are two main groups of approaches: analytical-based and learning-based. An analytics-based approach uses an analytics function to model system states and external measurements, which are referred to as 'estimations'. Analytics-based methods were the first data fusion approaches and are still widely used for navigational data but also for data in other domains. The main analytical-based methods are Kalman Filter (KF) and Particle Filter (PF) (Zhuang et al., 2023).

One of the most common techniques to apply Kalman filter is the classical method to integrate GNSS and IMU data (Teunissen & Montenbruck, 2017). Developed in 1960, It is a statistical technique to combine knowledge of the system errors and the knowledge of the system dynamics. Since the carrying platform (a road vehicle, an aircraft, a vessel or handheld by a human operator) usually does not change during the survey, then the motion can be constrained by its physical behavior. It is a recursive procedure consisting of two main steps - time and measurement updates. In the time update, the knowledge of system dynamics is used to predict the system state ahead of time. In the measurement update, the newly received measurement is used to correct the predicted system step. By repeating those 2 steps the data of IMU and GNSS can be integrated together.

The Kalman filter proved to be efficient, however it has some disadvantages. The original method is limited to modeling the system with Gaussian noise and is built on a linear state space. For non-linear systems or non-Gausian noise other methods must be used. The particle filter is another method for INS and GNSS data fusion, which is based on the Monte Carlo method and can model any state. It uses a set of weighted random samples, which are referred to as 'particles', to estimate a posterior probability of an event. By using the particles of the previous state, the new particles of the next state are predicted by using the state transition equation. Then, upon receiving new measurements, the particle filter evaluates each particle's likelihood, assigning weights based on how well the particle agrees with the measurement. Particles that align closely with the actual measurement receive higher weights. The filter then resamples the particles, favoring those with higher weights. This resampling concentrates the particle cloud around the more probable states, refining the

system's state estimation. This process continues recursively and only the most likely particles are kept due to resampling. They will represent the final navigational solution (Zhuang et al., 2023).

The learning-based methods are more modern and these data fusion methods, such as Artificial Neural Network (ANN), Fuzzy Logic, and Support Vector Machine (SVM) are used, since they can model systems without prior statistical information about the process and measurement noise. Learning-based methods sometimes perform better than analytics-based methods, however, have their drawbacks such as training overhead and generalizability issues (Zhuang et al., 2023).

2.2.2 LiDAR Data

The main product of a LiDAR acquisition is the point cloud. Initially, the points are recorded by LiDAR in the scanner's frame. Therefore, a straight comparison of two acquisitions of the same location requires a common reference frame, the data of the same environment will misalign. The process of georeferencing can provide that, however it depends on the quality of the navigational solution. To improve the resulting alignment of the point clouds or align the point clouds without georeferencing them, registration of the point clouds must be performed.

Registration is the process of aligning and minimizing the mismatch of two point clouds. It involves two steps - determination of the registration parameters and performing transformation based on those (Vosselman & Maas, 2014).

The transformation of a point cloud can be of two types - rigid and non-rigid. In a rigid transformation, the relative distance between any two points within a point cloud is not changed. In contrast, in a non-rigid transformation, the relative point distance is modified due to additional scaling or warping (for example, due to projection or distortion) (Monji-Azad et al., 2023). Due to that, non-rigid registration of point clouds is more challenging. However, the transformation of point clouds is usually rigid and consists of just rotation and translation of a point cloud.

The popular determination of registration parameters technique is based on point-to-point correspondence between the point clouds (Vosselman & Maas, 2014). The point-to-point correspondence registration methods utilize the identification of the same points in both point clouds to estimate the registration parameters. It can be any salient features present in the scanned environment or specially deployed control points. For the method using control points, special targets are distributed in the environment before the survey and the survey is performed to capture the same control points in the data, which needs to be registered (Vosselman & Maas, 2014).

Registration algorithms that do not utilize point-to-point correspondence are commonly iterative closest point (ICP) based methods (Vosselman & Maas, 2014). It does not require identifying accurate point-to-point correspondence. In those methods, one point cloud is rigidly transformed to best fit the second point cloud or model shape. The correctness of the transformation or the fitness criterion in the original method is the sum of squared Euclidean distances between the nearest points between the two clouds. The ICP algorithm tries to find a point cloud configuration, where the fitness criteria is the smallest (Besl & McKay, 1992). Since this method can converge to a local optimal solution, it requires a prior coarse alignment of the point clouds.

Another point cloud registration method class is feature-based registration. This method uses common features detected in two misaligned point clouds for registration parameters estimation (Vosselman & Maas, 2014). Those features can be geometric primitives (planes, spheres, cylinders), salient points (keypoints) or objects. Unlike the ICP-based approaches little to no overlap is required of the point clouds.

The steps for a general feature-based approach are shown in Figure 2.2. As it can be seen, initially a set of the same type of features are extracted from the misaligned point clouds. Then, the extracted features are described with a selected descriptor - to later compare and match the features from two point clouds. After the descriptors are calculated and the matching of features is completed, the rejection of the outlier (incorrectly matched feature pairs) is done. Finally, the resulting feature pairs are used to estimate the transformation parameters.

1. **Feature Detection**: The first step is feature detection and it is required to identify common points (3D keypoints) for further point cloud matching. 3D keypoints, also known as interest points or salient points, are specific locations in a point cloud that exhibit unique or distinctive geometric characteristics (Tombari et al., 2013). Early approaches in detecting 3D keypoints were inspired by 2D image keypoint detection methods. SIFT (Scale-Invariant Feature Transform) (Lowe, 1999) or Harris corner detector (Harris & Stephens, 1988) were commonly used in the field of computer vision, which were then adapted for point cloud processing. For example, Harris3D (Sipiran & Bustos, 2011) extends the Harris corner detector to the 3D space. Moreover, such as ISS (Intrinsic Shape Signature) (Zhong, 2009) and NARF (Normal Aligned Radial Features) (Steder et al., 2010), further advanced the field by considering different

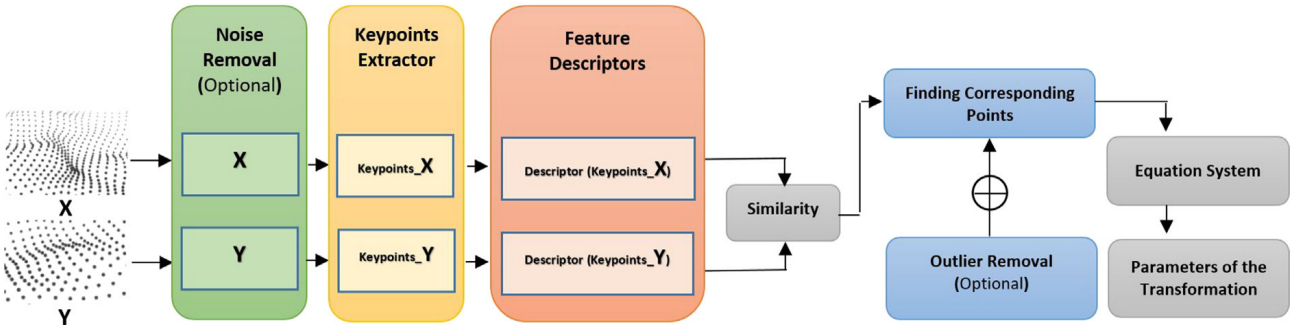


Figure 2.2: An overview of feature-based approaches to solve point cloud registration problem (Monji-Azad et al., 2023).

aspects of the point cloud data, such as local surface properties, scale-space extrema, and surface normals. Moreover, the growing popularity of deep learning methods resulted in the development of trainable models for keypoint detection and further utilization of them in point cloud matching. For instance, D3Feat (Bai et al., 2020) is one of the latest state-of-art models.

2. **Feature Description:**

   After keypoints were detected, a compact representation of them and their surroundings can be obtained via feature descriptors. Generally, feature descriptors can be split into Local Reference Frame free (LRF-free) and LRF-based descriptors (Yang et al., 2020). The first subcategory utilizes point geometrical positions, which result in geometric-based (curvature, planarity, point normals), density-based or statistical-based features. It allows those methods to be robust to variations in point cloud quality. However, these methods do not incorporate sufficiently higher-level geometric details within the descriptors. On the other hand, The LRF-based descriptors involve an extra step of Local Reference Frame estimation. This frame is then used to reference further extracted features, which can be shape-based or orientation-based. However, LRF-based descriptors tend to be sensitive to noise. Errors made in LRF estimation will propagate and will result in degraded performance for complex, novel, or noisy scenes.

   Then, both LRF-free and LRF-based methods can be split into hand-crafted and learned feature descriptors. Handcrafted feature descriptors are based on a general algorithm and mathematical formula, which does not involve model training and additional dataset preparation. Therefore, the advantage of these descriptors is their independence of training and ability to be applied out of the box. Additionally, the interpretability of those features is better, since the algorithm of their derivation is known beforehand and not learned during the training. However, hand-crafted features are usually application-tailored - each method is designed for certain use cases (Guo et al., 2016; Yang et al., 2020). Some of the popular handcrafted feature descriptors are FPFH (Fast Point Feature Histograms) (Rusu et al., 2009) and SHOT (Signature of Histograms of Orientations) (Salti et al., 2014), spin images (Johnson & Hebert, 1999) and RoPS (Rotational Project Statistics) (Guo et al., 2013).

   On the contrary, the learned descriptors are learned during a training process. They tend to outperform the hand-crafted features and can be more generalizable, but these descriptors are less explainable and require additional effort to train them (Fei et al., 2022). One of the first models was PointNet published in 2016, however since the follow-up research resulted in tens of new models with varying applications. For example, the latest models like Spin-net (Ao et al., 2021) and FCGF (Fully Convolutional Geometric Features) (Choy et al., 2019) learn keypoint detection and description within one network. Methods like these became the new state-of-art, outperforming the handcrafted detectors and descriptors.

3. **Outlier Rejection and Registration Parameter Estimation**

   After the features were extracted from the point clouds and matched by using their similarity from their descriptors, the outliers are removed. A popular method in the community to do it is Random Sample Consensus (RANSAC) (Fischler & Bolles, 1981). This method is probabilistic and the core idea of RANSAC is to repeatedly select a random minimal subset of the original data. Then, a model is fixed by the selected subset and the other datapoints and all other data points are tested against this model. More precisely, it is determined how many of them fit well (counted as inliers) and how many of them do not fit the model (counted as outliers). By repeating this procedure, the data can be kept or discarded based on the amount of outliers generated by the model. In case with point clouds, the kept features are used to used to estimate registration parameters.

2.2.3 SLAM

The output of other sensors during the surveying or after, during the postprocessing, can be used to enhance the localization of the system. This is more research in the field of robotics, where robots or other systems have to operate indoors or in other challenging environments, where GNSS and IMU do not provide sufficient navigation. It could be due the signal occlusion, IMU drift, or the necessity to precisely navigate through the environment without colliding with the obstacles. At the same time, robots are deployed in environments, for which local map is not available or does not provide sufficient navigational information. In those cases, the system has to solve the problem of simultaneous localization and mapping - to simultaneously create the map of the new environment and localize within it by using the observation through sensors such as cameras, SONARs, LiDARS (Durrant-Whyte & Bailey, 2006).

Many crucial tasks carried out by mobile robots require a map of the environment. It will allow robots to operate in complex environments only using onboard sensors and without utilizing external reference systems like GNSS. This problem of learning maps under pose uncertainty is referred to as a SLAM (Simultaneous Localization and Mapping) problem.

The problem was first defined in the 1990s and since that time multiple approaches to solving that problem were proposed (Durrant-Whyte & Bailey, 2006). The proposed solutions can be classified as filtering or smoothing solutions.

The filtering solutions model the problem as an online state estimation - estimating the current robot state (location and orientation) given the previous state. The state estimates are changed and updated as the new observations are received by the robot. It is done by utilizing the same methods used for GNSS and IMU integration - Kalman filter and particle filters. They are commonly referred to as online SLAM methods, since for a new position estimation only the previous state is used (Grisetti et al., 2010).

On the other hand, the smoothing approach estimates the trajectory by using the full set of observations. Those are commonly referred to as full SLAM problems and they utilize least square error minimization techniques and use the complete data array (Grisetti et al., 2010).

As mentioned, the solution to a SLAM problem consists of estimating the trajectory and the map of the environment as the robot moves. Due to the noise in sensor measurements, the SLAM problem is typically described through probabilistic tools:

- The robot is assumed to move in an unknown environment, along a trajectory described by the sequence of random variables $X_{1:T} = X_1, ..., X_T$.
- Along the way it acquires a sequence of odometry measurements (from a motion sensor like IMU or GNSS) $U_{1:T} = U_1, ..., U_T$
- Additionally, other sensors (like LiDAR) percept the environment around the robot $Z_{1:T} = Z_1, ..., Z_T$

Solving the full SLAM problem consists of estimating the posterior probability of the robot's trajectory and the map of the environment: $p(X_{1:T}, m|Z_{1:T}, U_{1:T}, X_0)$. The poses $X_{1:T}$ and the odometry $U_{1:T}$ are represented commonly as 2D or 3D transformations in SE(2) or in SE(3). SE stands for Special Euclidean space in 2D and 3D, which describes rotation and translation. Moreover, a map of the environment can be differently described - parametrized as a set of spatially located landmarks or by a dense grid (Grisetti et al., 2010).

The relation between the mentioned random variables can be visualized with a Dynamic Bayesian Network (DBN). An example is shown in Figure 2.3. The connectivity of the DBN is described by a pattern characterization of the state transition model (shown by 2 edges to X_t) and an observation model (shown by 2 edges going to Z_T). The state transition model represents $p(X_t|X_{t-1}, U_t)$ - the probability that the robot at time t is at X_t given both previous position X_{t-1} and the acquired odometry measurement U_t. The observation model represents $p(Z_t|X_t, m)$ - the probability of the environment observation Z_t given the current robot pose X_t and the map m (Grisetti et al., 2010).

The other modeling method of the SLAM problem utilizes pose graphs. The graph-based model will highlight the underlying spatial model. In such a model, the poses of a robot (X_t) are also nodes of the graph. However, the edges in this model define the spatial constraints to the nodes and are given by the environment perception (Z_t) or the odometry measurements (U_t). The constraints are commonly contradictory, since the all robot's measurements are affected by noise. Therefore, once the graph representation of a SLAM problem is constructed, it is necessary to find a node configuration, which is maximally consistent with the measurements. In other words, the adjusted nodes must solve the error minimization problem. Such graph modeling is a state-of-art approach in terms of the speed and accuracy in solving SLAM problems (Grisetti et al., 2010).

Let $X = X_1, ..., X_T{}^T$ be the vector of the recorded trajectory poses. Pose X_i will correspond to a node i in the graph. The LiDAR observation can be represented by the observation mean Z_{ij} and information

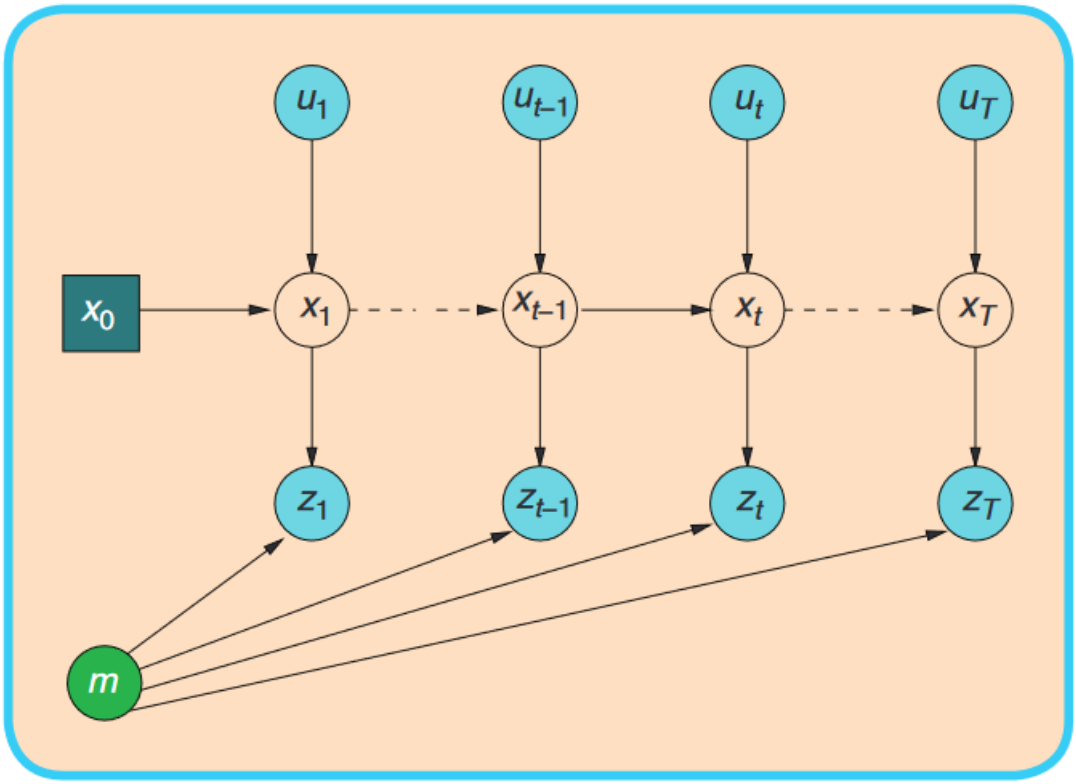


Figure 2.3: Representation of a SLAM problem with a Dynamic Bayesian Network. It describes a stochastic process with a directed graph - nodes correspond to random variables and directed edges between nodes correspond to a conditional dependence between them (Grisetti et al., 2010).

matrix Ω_{ij} of a virtual measurement between the nodes i and j. This observation is also estimated solely from the configuration of nodes X_i and X_j - denoted as $\hat{Z}_{ij}$. Due to the error in the measurements, the actual measurement Z_{ij} and virtual measurement $\hat{Z}_{ij}$ will be different. The difference between those values will define the error function for the poses X_i and X_j:

$$e(X_i, X_j, Z_{ij}) = e_{ij}(X_i, X_j) = e_{ij} = Z_{ij} - \hat{Z}_{ij}. \tag{2}$$

Figure 2.4 captures the relationship between the nodes, the observation and the error, which was described in the previous paragraph.

The resulting error function of the whole graph is expressed as the summation of the separate existing error functions weighted by their information matrix:

$$F(X) = \sum_{<i,j>\in C} e(X_i, X_j, Z_{ij})^T \Omega_{ij} e(X_i, X_j, Z_{ij}) = \sum_{<i,j>\in C} F_{ij}. \tag{3}$$

The graph optimization will find a state configuration, which locally optimizes X and the found solution is denoted as X^* and can be expressed as:

$$X^* = \underset{X}{\operatorname{argmin}} F(X). \tag{4}$$

The described graph construction and optimization have various implementations. One of the frameworks, that implements an interface for graph construction and optimization is g^{2}o (Kummerle et al., 2011). General graph framework or g^{2}o is a tool for modeling and solving graph-based SLAM problems. It is originally implemented in the C++ programming language and allows for simple and quick SLAM problem solutions. To use it, it is necessary to construct the graph for the posed SLAM problem. Additionally, the user has to define the corresponding error function(-s) and the corresponding update function(-s). User-defined error functions are required for error calculations and provide constraints and weighting for the states. User-defined update functions will define the steps, with which the states are modified, after an optimization iteration.

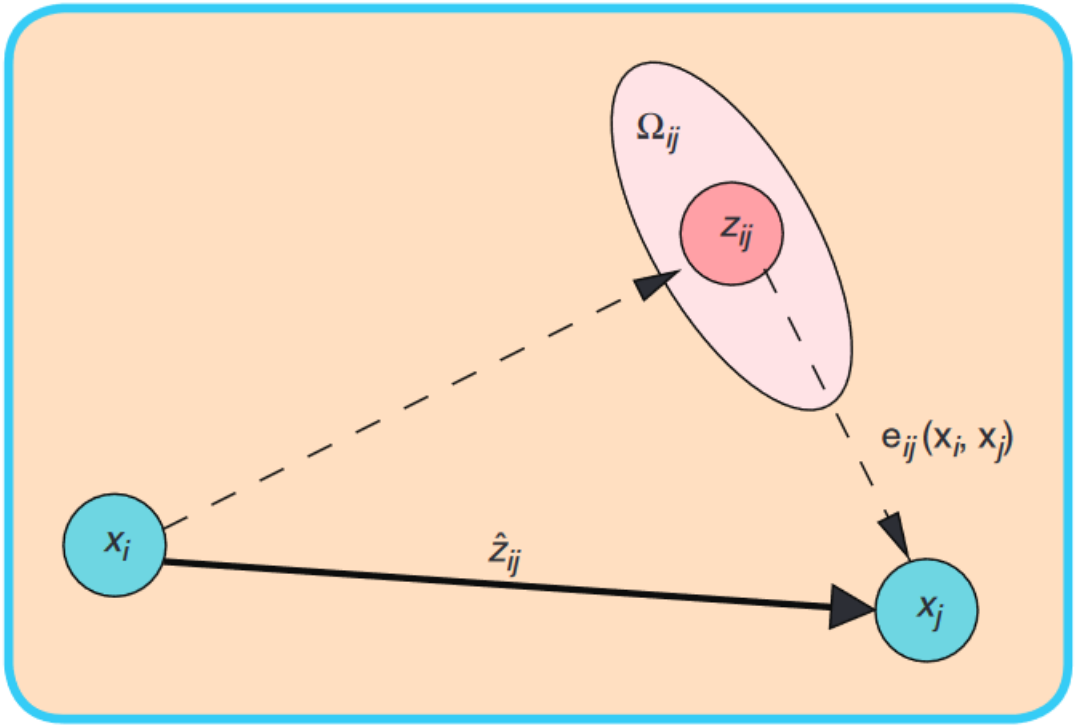


Figure 2.4: Visualization of the relationship between the nodes, the observation and the error in a graph-based SLAM model. The edge connecting the vertex X_i and the vertex X_j originates from the measurement Z_{ij}. From the relative position of the two nodes, it is possible to compute the expected measurement $\hat{Z}_{ij}$ that represents X_j seen in the frame of X_i. The error $e_{ij}(X_i, X_j)$ depends on the displacement between the expected and the real measurement (affected by noise). An edge is fully characterized by its error function $e_{ij}(X_i, X_j)$ and by the information matrix Ω_{ij} of the measurement that accounts for its uncertainty (Grisetti et al., 2010).

2.3 Research On Sensor Fusion for Enhanced MMS Positioning

The MMS community is continuously improving the quality of geo-data through the use of SLAM solutions and various sensor combinations. It is agreed that solely integrating GNSS and IMU is insufficient due to these sensor limitations. The incorporation of external sensors aligned with SLAM solutions is needed to enhance surveying quality and the surveying of challenging environments benefits the most from such research. Challenging environments can be defined as environments with poor or absent GNSS reception and not rich features. Those environments rely heavily on the positioning aid from other sensors like LiDARs or cameras. By using existing SLAM solutions, researchers were successful in improving the trajectory information by utilizing them.

For example, in a study focused on forest mapping accuracy, researchers addressed the challenge of weak GNSS signals by integrating LiDAR data with a GNSS/INS system (Qian et al., 2016). They developed an Improved Maximum Likelihood Estimation SLAM (IMLE-SLAM) approach, which additionally leveraged data from navigational sensors. This method enhanced trajectory accuracy to the order of decimeters. Notably, his approach did not rely on feature extraction or matching. Instead, it utilized a grid-based occupancy likelihood method for parameter estimation in registration.

Another study explored the use of a hand-held laser system, incorporating LiDAR, IMU, and a camera, for underground mine surveying, where GNSS signal is completely absent (Raval et al., 2019). They assessed its potential in combination with SLAM software and they concluded on the feasibility and benefits of Mobile Mapping Systems (MMS) use in mines. In separate research, the implementation of robust pose graph optimization and LiDAR data, integrated with place recognition techniques, was found to enhance the accuracy of mine mapping and data registration (Yoshida & Tadokoro, 2014). It offers significant improvements over systems relying solely on IMU technology.

In a more urban setting, similar work was performed to improve surveying in cities and on high-speed roads. In cities, the estimation of point cloud registration parameters with a method like ICP was proven to help with trajectory optimization problems due to the present multipath from existing buildings (Gressin et al., 2012). Additionally, research on road surveying in high-speed scenarios successfully obtained improvements from integrating a LiDAR/GNSS/IMU sensors (Shen et al., 2024). The main focus was on road surveying, however,

the research dataset included also a 420m long tunnel. Due to its smoothness and absence of distinctive features, the quality of the solution degraded.

Such featureless environments pose a challenge for SLAM-based MMS systems. A typical solution for those environments would involve placing control points. It was done by a Lynx system, when it was utilized to map the existing rail network of 1500km in Saudi Arabia and in order to to counter GNSS signal absence, ground control points were used for corrections (Al-Bayari, 2019). This was also applied to improve the georeferencing of point clouds in the underground environment. Research with experiments in the Beijing metro was performed to establish a correct procedure for control points spacing and usage in such environment (Liu et al., 2022). The research managed to receive a decimeter-level trajectory error, however, it was achieved by utilizing pre-installed targets with 50m spacing. Similar work was implemented for roadside tunnel monitoring. A truck-based MMS with camera sensors was used to collect images of tunnel walls and to capture targets for positioning aiding (Chapman et al., 2016). The disadvantage of using cameras in a dark environment was overcome by using an array of LED lamps. Another research specifically focused on tunnels presented a system capable of collecting accurate measurements of a tunnel by using a railcard-based system (Sun et al., 2020). However, it does not focus on trajectory correction - the presence of an IMU sensor is stated but nothing in the research mentions the problem of an IMU drift.

Since the scope of this research focuses on the RILA system, it is important to consider the specifics of rail-based surveying. The movement of a rail-based MMS platform is constrained by rails, which create a predefined path for the transport and will define its motion direction. This constraint allowed to use of some methods and sensors, which were not helpful in other environments.

For example, map-matching methods were used for train positioning. The underlying principle of map-matching algorithms is aligning surveyed trajectory data to existing road networks (Chao et al., 2019). Specific map matching algorithms were developed for train positioning and tested (Jiang et al., 2018; Kim et al., 2015; Saab, 2000, 2000). However, the map-matching algorithms have a disadvantage - their accuracy at best is as good as the accuracy of the used map (Chao et al., 2019). Typical map-matching applications do not require sub-cm accuracy as RILA does. Due to that both pioneering and more recent map-matching research for train applications showed accuracy of meters magnitude (Jiang et al., 2018; Saab, 2000).

Another benefit of a rail-based system is the possibility of sensors like odometers. Odometers provide one-dimensional measurements about the distance traveled by the sensor and can be used to aid the navigation in GNSS-denied environments (Kim et al., 2015). Odometers operate by tracking wheel rotations, translating this motion into an accurate distance measurement, and for that the conventional sensors must be integrated with the train. Additionally, special tags can be installed along the rail network, which can be accurately mapped and then can be read by passing by trains. The passing trains, which have a tag reader installed, will be able to use tags to identify their current location (Kim et al., 2015). However, sensor integration with the train itself or manual installation of the control tags along the way is not feasible for a system like RILA. There is a mention of another rail surveying system called RailMapper, which is reported to utilize the speedometer sensor to aid navigation in periods of GNSS signal occlusion (Kremer & Grimm, 2012). The RailMapper system was mentioned to be mountable on various platforms (cars, trams, trains), however, no information is given about the details of how the speedometer sensor aids the navigation and how it is mounted on different platforms.

However, promising results were obtained in the rail environment without using its distinctive features. In that research, LiDAR data was used to correct the trajectory (Jing et al., 2021). This research developed a semi-automatic approach to correct the trajectory without using any pre-installed control points. It is done by developing a novel method "Feature Extraction based Particle filter Point Cloud Aiding - **FEPPA**". For that method, the stable features (poles) are extracted from the point clouds. The features contain both absolute location information (position in an absolute coordinate system) and relative location information (distance between the system and other features). These features are then used within a particle filter along with the GNSS/IMU observations and the smoothened trajectory is generated. The resulting algorithm is tested against two datasets (data collected by tram and data collected by car), where GNSS observations originally were accurate and were artificially deleted (to simulate GNSS outage). The report shows an improvement from meter to decimeter trajectory error.

From the mentioned research, the benefits of using LiDAR data to optimize the trajectory are visible and can be feasible in the case of RILA. However, in some cases, the aid from a single data collection sensor is not sufficient for a SLAM algorithm. It happens due to the limitations of a sensor and in cases of multiple sensors, they can complement each other. Therefore, the latest research focuses on integrating multiple data collection sensors. For example, recent research proposed a sensor fusion framework for autonomous driving, where the SLAM solution utilizes both camera and LiDAR (K. Wang et al., 2021). The validation of the framework was revealed to be better than the state-of-art performance. Another work related to the multi-sensor fusion is utilizing both camera and LiDAR by a mobile robot in outdoor environments, where the proposed tight coupling

framework outperformed as well the single-sensor state-of-art methods (Xia et al., 2024). Nevertheless, the use of other sensors in the case of RILA will require hardware modifications, which are out of the scope of this research.

2.4 Literature Review Conclusion

From the researched literature, it is evident that the utilization of LiDAR data to optimize the trajectory has potential. Research has shown that utilization of point clouds without deploying control points was capable of improving the positioning of the system (Gressin et al., 2012; Jing et al., 2021; Qian et al., 2016). However, no specific research was done in the context of railway tunnels and LiDAR. Some methods were tested in car tunnels or on tram railways but not in the railway environment (Jing et al., 2021; Shen et al., 2024). Despite that, partial answers to the research questions were found:

1. **How is the quality of trajectory solutions evaluated?**:

   The quality of trajectory solutions can be evaluated through several methods. For example, the trajectory optimization solution can be compared with a trajectory, where the absence of GNSS signal is simulated (Jing et al., 2021; K. Wang et al., 2021). Another trajectory solution evaluation method is to consider the georeferenced point clouds and how they align with each other. More precisely, distances between set-up targets, features or points in the point clouds will quantify the misalignment of the point clouds locally (Jing et al., 2021; Vosselman & Maas, 2014).

2. **What methods exist to optimize trajectory and how well do they suit rail-based tunnel environment?**

   In the literature review, no methods were found to fully meet the specific requirements of the RILA system and this research. The desired method should function automatically within a rail-based tunnel environment and accommodate the RILA's expected cruising speed (up to 100 km/h and faster). Although no reviewed paper entirely matched these criteria, some studies offered partial solutions. For example, the FEPPA method optimized trajectory using a Particle Filter with poles extracted from point clouds, showing improvement Jing et al., 2021. However, the features are expected to be absent in tunnel environments and this method does not meet the automation criteria in feature extraction. However, this method shows a general workflow of utilizing point cloud features in trajectory optimization. It was also present in other studies, despite being conducted in different settings or using distinct platforms. Trajectory optimization in underground mines, forests or on high-speed roads utilizes point cloud data in a similar SLAM-based solution (Liu et al., 2022; Qian et al., 2016; Shen et al., 2024).

3. **What methods exist to identify and extract point cloud features in the tunnels?**

   Hand-crafted and deep learning methods exist for identifying features in point clouds. Methods such as ISS, Harris3D, and NARF use geometrical variations to identify salient points without requiring any training (Sipiran & Bustos, 2011; Steder et al., 2010; Zhong, 2009). More advanced deep learning methods, like D3Feat, tend to outperform hand-crafted methods (Bai et al., 2020). Although, these models must be trained and adjusted before use. Moreover, specifically for point cloud feature extraction in the tunnels, no benchmarks comparing these methods were found.

Therefore, this research will continue with describing the methodology, which was crafted based on the answers found in this chapter and will complete the questions partly answered above.

3 Methods

In this chapter, the methodology used to develop a trajectory optimization routine using point cloud features and steps taken to answer the proposed research goals are described. The methodology is split into 3 parts. In the subsection 3.1 the dataset used to develop and evaluate the methodology is presented. Following the dataset description, in the subsection 3.2 the local optimization algorithm and its parts are described. The last subsection 3.3 of this chapter presents how the local optimization algorithm is scaled to perform global trajectory optimization.

3.1 Dataset Description

To achieve the goal of this research it is necessary to assess the RILA data collected through tunnels, develop the new trajectory optimization method based on it and use it for evaluation. For this research, a dataset of surveyed data collected for the railway network around Glasgow is used. It covers the underground railway network going through Glasgow train station, hence is refered in text as "Glasgow Station dataset". The locations of the dataset is shown in Figure 3.1.

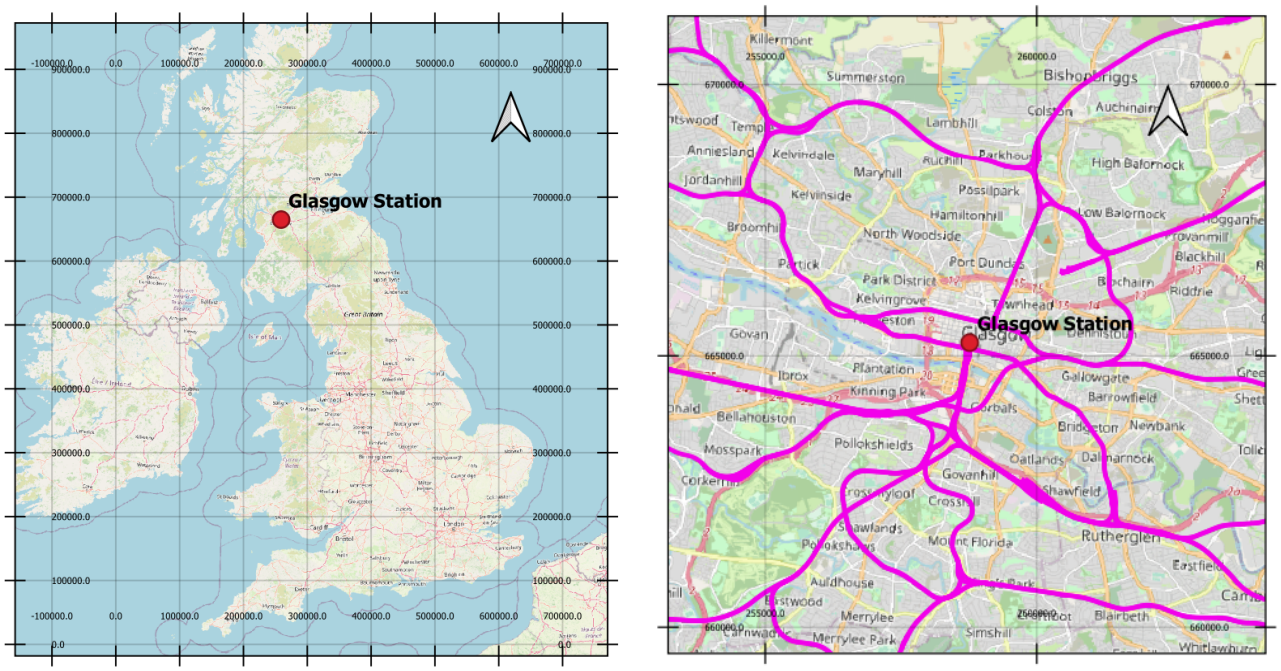


Figure 3.1: Location of the Glasgow Station datasets. The magenta lines show spatial distribution of the rail network. The map is shown in the British National Grid CRS (OSGB36 or ESPG:27700).

The data in this dataset does not differ from other RILA surveys, it contains the GNSS and IMU measurements, along with the LiDAR data. After a survey, the GNSS data is processed along with the IMU data with the processing software. In the software, the IMU and GNSS data are integrated by applying the Kalman filter, which estimates the final trajectory of the survey. The trajectory data point contains information about the measurement's IMU location, orientation, and GPS time. This trajectory is used to georeference the LiDAR data. The LiDAR data of RILA is collected by Riegl VUX-1HA, which has a reported sub-cm accuracy ($1\sigma = 5mm$). The resulting density of the point clouds will vary due to its dependence on train velocity and distance from the sensor. For example, with a cruising velocity of 100km/h, the driving direction separation between points is expected to be 0.11m (Table 1.1). After data is georeferenced, it is split into 25 by 25-meter tiles by the same reference grid, to make the data storage and processing easier. Points in the resulting point clouds contain fields like XYZ locations of the points (in British National Grid CRS or EPSG:27700), intensity and the time of the collection (GPS time).

Glasgow Station dataset is located in Glasgow, Scotland. It includes the underground section of the Glasgow Central railway station and the connecting underground tunnels. This dataset contains a wide variety of environments. Those include the underground station, the overlapping tunnels, the curved and normal tunnels (shown in Figures 3.2). The geometry of the tunnels tends to be mainly rectangular and the ceiling contains a

repetitive geometric pattern made from arcs. Additionally, the overlap of the point clouds varies. Therefore, the overlapping tunnels and underground station are expected to be the most challenging for trajectory optimization. Due to a smaller overlap, less data volume is shared for common feature detection and mapping. Mainly the platform and parts of the opposing walls. However, other objects (like tunnel pillars or platforms) are present and might be helpful.

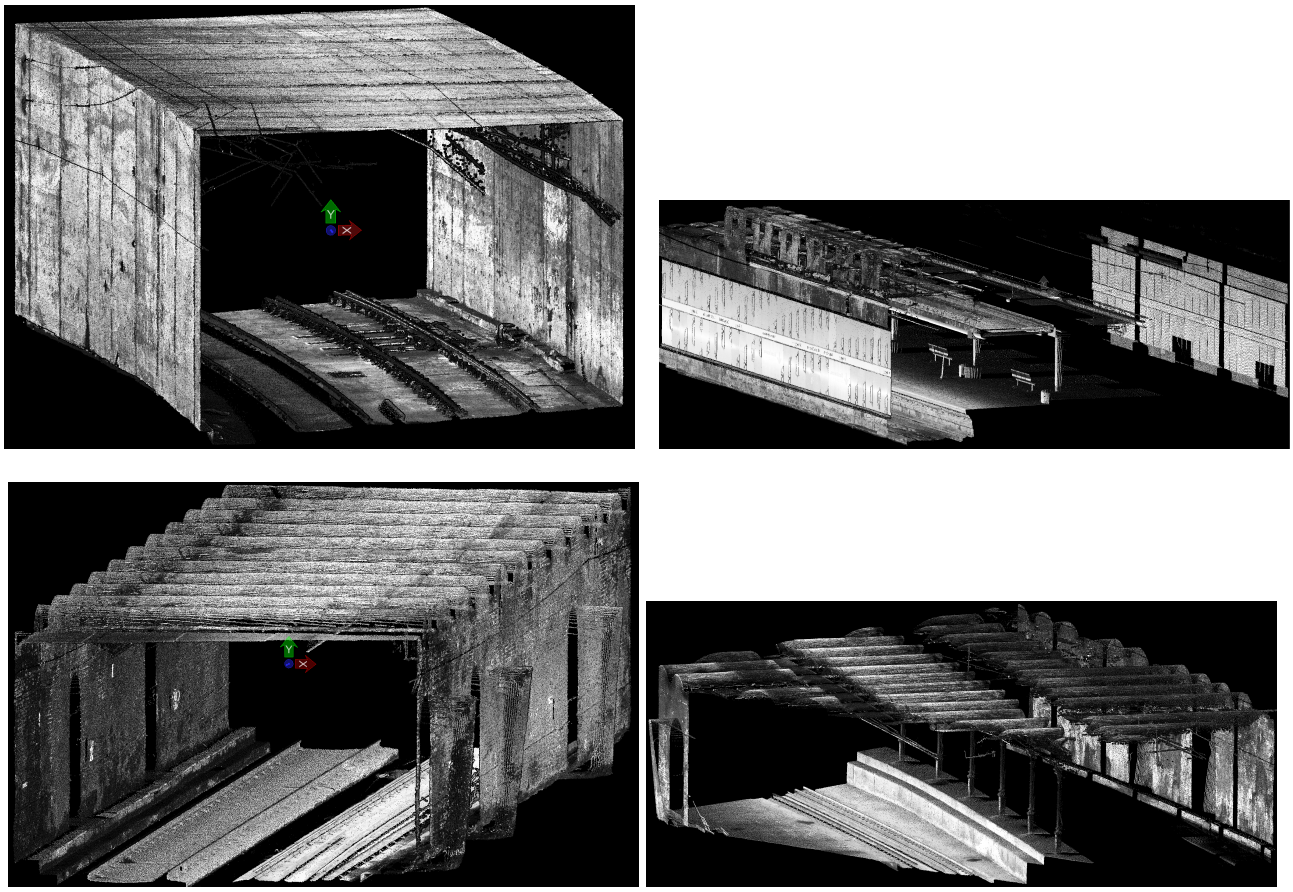


Figure 3.2: Glasgow Station point cloud data example colored by intensity (grayscale).

3.1.1 Data for Local Trajectory Optimization

To implement the described below local trajectory optimization algorithm, a single location (tile) is selected for the development, testing, and assessment of the chosen methodology. A tile featuring a straight tunnel (as shown in Figure 3.3) was chosen. This selection was made because the tile is not feature-rich, which is anticipated to pose more challenges than other underground environments. Therefore, focusing on this tile is expected to encompass the complicated scenarios, and the selected parameters are presumed to perform well in environments with more features.

Additionally, for this tile, a synthetic acquisition is generated to compare the obtained results with a known Ground Truth (GT). The data from the second acquisition was simulated by translating the point cloud with a known translation vector and distorting every point with Gaussian noise (zero mean, $\sigma = 2$cm, independently for X, Y, and Z). This approach was employed to simulate a separate data acquisition of that area, allowing the registration parameters obtained through feature extraction to be compared with a known ground truth (GT) translation vector and accuracy to be quantified. This data is assumed to provide a controlled tuning of the parameters for the algorithm.

The translation vector applied to the tile's point cloud was constructed from two orthogonal vectors - the driving and normal directions. For this tile, the applied translation vector to generate the simulated data was 0.5 * the driving direction unit vector + 1.5 * the normal direction unit vector, with the translation in the Z component selected to be 10 cm. This choice of translation vector was based on a visual assessment of the data, where differences in the normal directions were observed to be larger than corrections in the driving direction, and the Z component could be smaller or of the same magnitude as the driving direction.

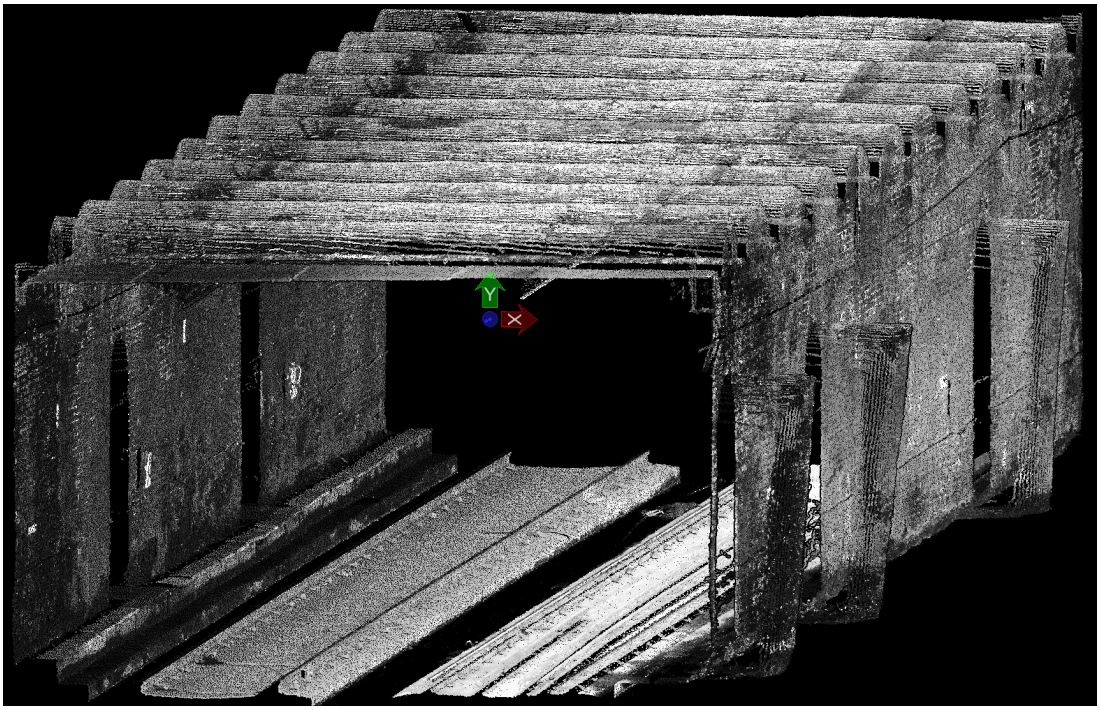


Figure 3.3: Selected tile for methodology development, representing a section of a straight tunnel from the Glasgow Station dataset colored by intensity (grayscale). Point cloud consists of approximately 2 million points.

After testing the local trajectory optimization on the synthetic data, the algorithm is applied again to the 2 real acquisitions covering the same selected tile.

3.1.2 Data for Global Trajectory Optimization

The global trajectory optimization algorithm uses the point cloud features from several tiles to optimize trajectory on a larger scope. The data for it is selected from surveys of 2 runs. Directories with the point cloud tiles are loaded, matched and filtered: only point cloud tiles present in both runs are selected. Additionally, the point clouds under a certain size limit are discarded. Then, for the selected set of point cloud tiles the local optimization procedure was applied consequently - for each pair of tiles covering the same area at one time. Figure 3.4 shows the spatial distribution of the selected tiles for the global optimization procedure. These tiles were selected from two train passes carrying RILA over the same location. Moreover, the selected tiles were selected only within the area of a tunnel and the tiles under the threshold size were discarded. This resulted in a selection of a 120 tiles, with each bigger than 3 MB (or 1 mil points).

3.2 Local Trajectory Optimization

The local trajectory optimizing algorithm focuses on optimizing a trajectory within one point cloud tile. This approach allows quick development and testing of the algorithm. Additionally, it provides straightforward scalability to optimize trajectory across multiple tiles. It partly follows the workflow outlined in the (Jing et al., 2021) - detect the features within the environment for trajectory correction and use them to aid the navigational solution. However, pole features in the underground environment are scarce (present rarely at the station) or absent (in the tunnels). Therefore, other features are selected to be applied in this environment. Those features are Intrinsic Shape Signatures (ISS).

The objective of the local trajectory optimization is to optimize the trajectory within a single-point cloud tile. The point cloud data covering that tile is retrieved from two runs (surveys), which are referred to as Run 1 and Run 2. Additionally, for those two runs, the trajectory data covering that tile is utilized (referred to as Trajectory 1 and Trajectory 2). These two point clouds (Point Cloud 1 and Point Cloud 2) and two trajectories (Trajectory 1 and 2) are used as input to the local trajectory optimization algorithm. The general steps of that workflow are shown in Figure 3.5. As mentioned in the Figure, first, the point cloud features are extracted from Point Cloud 1 and 2 separately. Then, the matching algorithm is applied to the extracted features - to match them and to remove outliers. The resulting matched feature pairs are used in the g^2o

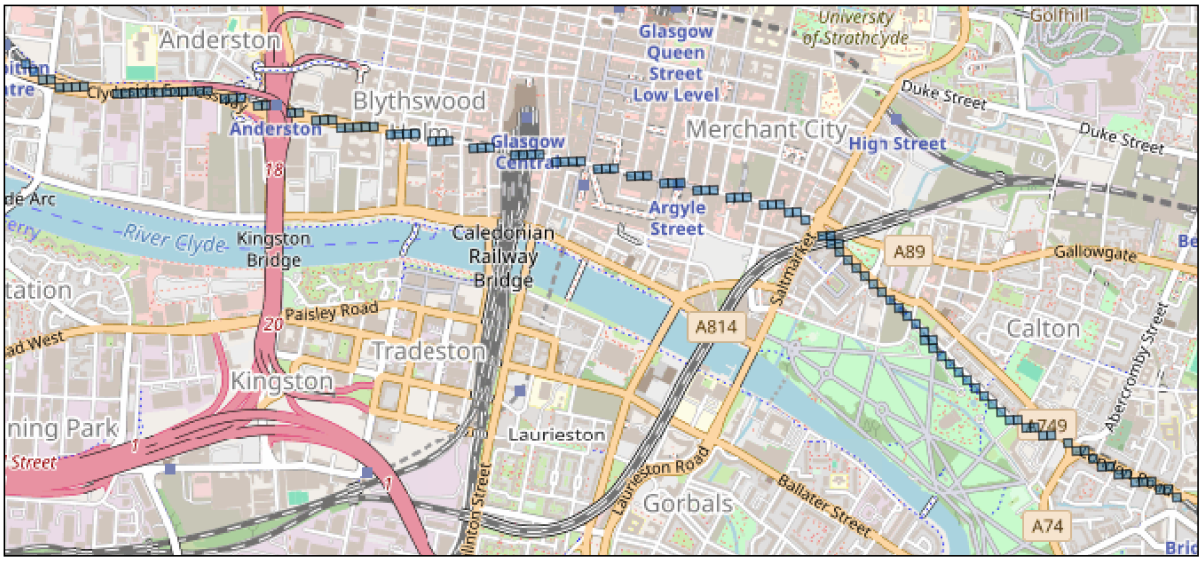


Figure 3.4: Visualization of the selected tiles for the global optimization (120 tiles were selected in total).

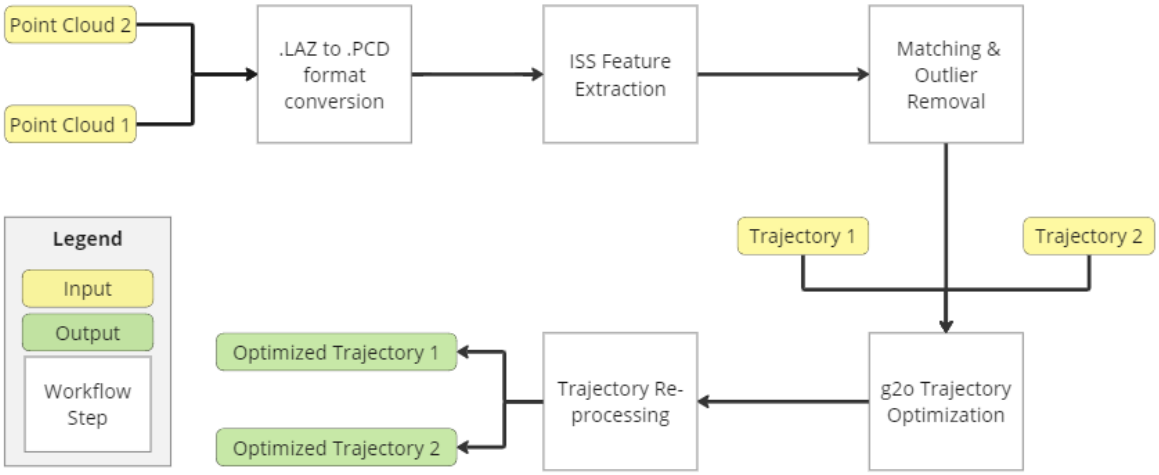


Figure 3.5: Steps of the local trajectory optimization algorithm.

optimizer. Finally, the optimization output (optimized IMU poses) is used to reprocess the trajectory with the GNSS processing software and new georeferenced point clouds are obtained. The newly georeferenced data will undergo an assessment to reveal any potential improvement or degradation of the data quality.

An additional step is also integrated into the methodology due to the specific requirements of the utilized feature detectors and descriptors implementations. The methodology employed functionality use of both PCL (Point Cloud Library) and Open3D point cloud processing libraries via their Python and/or C++ APIs. These libraries are incompatible with the .LAZ format. Therefore, the data was initially converted to the .PCD format. The conversion process from .LAZ to .PCD is supported by another point cloud processing library, PDAL (Point Data Abstraction Library), and the CloudCompare software. However, utilizing these tools for conversion resulted in a reduction of accuracy. The original coordinates, precise to three decimal places, were truncated to two decimal places post-conversion. During the research, no existing solution was found, which could address this issue. To overcome this limitation, a custom Python script was developed. This script utilized the Laspy library to read and decompress .LAZ point clouds. Subsequently, the point clouds were exported as ASCII .PCD files, preserving the three-decimal precision for point coordinates.

3.2.1 Feature Extraction

Intrinsic Shape Signatures (ISS) are selected as features (or keypoints) for extraction in this methodology. The ISS detector operates by analyzing the local neighborhood of each point in the cloud. It calculates the eigenvalues of the covariance matrix of the points within this neighborhood. ISS features are then selected based on specific criteria related to these eigenvalues, such as the variation and distinctiveness in the local surface geometry (Zhong, 2009).

There are several benefits of using ISS features. ISS features are less sensitive to noise and variations in point density. Moreover, the ISS features focus on intrinsic geometric properties, which makes them robust across different scales and orientations of objects within the point cloud. The features are also reported to be efficient and they provide a good trade-off between absolute and relative repeatability (Tombari et al., 2013). Lastly, ISS features are hand-crafted and they do not require any training.

The mentioned benefits make ISS features reliable for tasks such as feature matching and object recognition. As mentioned, RILA's point clouds vary in density due to the varying train speed, therefore a density-agnostic feature detector is preferred. Moreover, this research focuses mainly on the tunnel environments, however, those features can be present and found in a variety of other environments, providing possible generalizability to the developed trajectory optimization method. Furthermore, the practical benefit of utilizing ISS features is their popularity in the point cloud processing community. Due to that, various implementations of ISS feature detectors are available in point cloud processing libraries. However, more efficient deep learning feature detectors exist but the developed deep learning models require training before they can be applied. Preparation of a training dataset and performing a training cycle for a deep learning model is considered to be unfeasible in the scope of this research. There are also models, that are distributed and already pre-trained, however, their efficiency in the underground environment will be dependent on the data variety used for the models training. Lastly, deep learning models due to their complexity are harder to analyze and detect the reasons behind their strong or weak performance.

In this methodology, implemenentation of ISS feature detector from Open3D is used. It has the following parameters to tune:

- **Salient Radius**: It defines the radius used to search for neighbors around each point. Therefore, a larger radius considers a broader area for defining key features, which potentially captures more significant features. However, a bigger radius might include less relevant points of far-away structures, reducing the distinctiveness of the ISS key points. On the contrary, a smaller radius can capture finer details but may be more sensitive to noise. The optimal radius depends on the density of the point cloud and the scale of the features.

  For RILA, the density of the collected point clouds varies in the driving and normal direction, due to the point capture pattern and varying train cruising velocity. The biggest spacing is usually in the driving direction and can reach ≈ 10 cm (as shown in table 1.1). Therefore, the Salient radius should be bigger than that to capture variability in both driving and normal directions. Moreover, the upper bound for that radius is expected to be placed around 1.5 - 2 meters, to keep the resolution of features within meter magnitude. Additionally, considering a larger radius is expected to increase the computational time.

- **Non-maximum Suppression Radius**: This radius is used to suppress points from being selected as a keypoint in the vicinity of an already detected keypoint. This helps in reducing the clustering of keypoints. Therefore, a larger radius can spread out the keypoints more by suppressing keypoints that are too close to each other. On the contrary, if this radius is too small, you might get too many clustered keypoints. This value was decided to be equal to the Salient Radius parameter.

- **Gamma 21 (γ_{21})**: This parameter represents a threshold for the smallest eigenvalue to the second smallest eigenvalue ratio (as shown in the equation 5). This threshold helps in identifying points that have significant features. Lowering this threshold can increase the number of keypoints by accepting points with less distinct eigenvalue ratios. A higher threshold makes the criteria for keypoint selection more stringent, reducing the number of keypoints. Therefore, this parameter was decided to be kept at value 0.5 and only be increased if the amount of detected keypoints is too small.

$$\frac{\lambda_2}{\lambda_1} < \gamma_{21} \tag{5}$$

- **Gamma 32 (γ_{32})**: Similar to Gamma 21, this is a threshold for the second smallest eigenvalue to the largest eigenvalue ratio (as shown in the equation 6). A lower threshold includes more points as keypoints by accepting less distinct features, while a higher threshold will select only those points with

more pronounced features. Similarly to γ_{21}, this threshold was decided to be kept at value 0.5 and only altered in case of a small amount of detected keypoints.

$$\frac{\lambda_3}{\lambda_2} < \gamma_{32} \tag{6}$$

To find an optimal parameter set for the ISS feature detector, five different parameter configurations are tested, particularly varying the salient radius. The tested parameters are presented in the Table 3.1. The optimal set of parameters is determined by measuring the computation time required for it to complete as well as the amount of the detected features.

Config. #	$R_{salient}$ [m]	$R_{suppresion}$ [m]	γ_{12} [-]	γ_{23} [-]
1	0.3	0.3	0.5	0.5
2	0.6	0.6	0.5	0.5
3	0.9	0.9	0.5	0.5
4	1.2	1.2	0.5	0.5
5	1.5	1.5	0.5	0.5

Table 3.1: Parameter configurations tested for ISS feature detector.

3.2.2 Feature Matching

After the features are detected in both point clouds, they are matched. The matching is performed based on their similarity and the similarity of the features is based on their descriptors. The ISS method provides a method to detect a feature, however, it does not provide a descriptor for them. Thus, another algorithm is needed to calculate a descriptor for the selected features to correctly match them. For this methodology, such an algorithm is Fast Point Feature Histograms (FPFH).

As an output, this algorithm outputs a high dimensional (33D) descriptor vector for every point it was applied on. The similarity of the features can be estimated by calculating the difference between the descriptors of the points. In this methodology, the similarity is measured with the Euclidian distance between two descriptor vectors. The smaller the resulting distance, the more similar are considered the points to be.

FPFH operates by considering each point in the point cloud and its immediate neighbors within a specified radius. It computes a simplified point feature histogram for each point based on geometric relationships with its neighbors. These relationships are quantified in terms of angular variations and described using a histogram (Rusu et al., 2009). This FPFH descriptor was evaluated against other feature descriptors and it was reported that it provides a good balance between feature matching accuracy and computational efficiency (Guo et al., 2016).

The procedure to calculate an FPFH descriptor begins with generating a KD-tree for the provided point cloud - to perform efficient nearest-point queries. Subsequently, point cloud normals were estimated, as it is required as input for the FPFH descriptor calculation. The normals for each point were computed by considering neighboring points within a specified radius. Following this, a descriptor for each ISS keypoint index was calculated.

The primary variable parameter for the descriptor calculation is the search radius. It influences the degree to which neighborhood details or noise affects the calculation. This parameter bears similarity to the salient radius used in the ISS feature detection process. Therefore, various values for the FPFH feature descriptor search radius are explored, as detailed in Table 3.2.

Configuration #	$R_{normals}$ [m]	R_{FPFH} [m]
1	0.3	0.3
2	0.6	0.6
3	0.9	0.9

Table 3.2: Parameter configurations tested for FPFH feature descriptor.

The detection of the ISS features and their use in feature descriptor calculation is described in the following steps (The pseudo-code is shown in Listing 1):

1. **Feature Detection**: Initially, the algorithm processes each point cloud separately. First, in point cloud 1, ISS features are detected and saved. Then, this process is repeated identically for point cloud 2. As a result of this step, each point cloud has its own set of identified ISS features.

2. **Descriptor Generation**: Once features are identified in each point cloud, the next step involves generating descriptors for these features. For every detected feature in point cloud 1, an FPFH descriptor is computed and stored. This procedure is also applied to each feature in point cloud 2.

3. **Descriptor Matching**: The matching phase begins by examining each detected feature in point cloud 1. For every feature, the algorithm searches through all detected features in point cloud 2 to find the most similar one. The similarity between the two features is determined based on the Euclidean distance between their respective FPFH descriptors. The smaller the distance, the higher the similarity.

   After this, the process is mirrored: for each feature in point cloud 1, the most similar feature in point cloud 2 is identified following the same similarity criterion.

   A pair of features is considered a valid match only if each feature is the most similar match to the other in their respective point cloud. This condition ensures that only the most compatible pairs are selected as matches. These matched pairs of features are then saved as the final output of the algorithm.

Algorithm 1 ISS Features Matching

```
procedure MatchFeatures(PointCloud1, PointCloud2)
    Features1 ← DetectISSFeatures(PointCloud1)
    Features2 ← DetectISSFeatures(PointCloud2)
    Descriptors1 ← GenerateFPFHDescriptors(Features1)
    Descriptors2 ← GenerateFPFHDescriptors(Features2)
    MatchedPairs1to2 ← empty list
    MatchedPairs2to1 ← empty list
    for feature1 in Features1 do
        BestMatch ← null
        MinDistance ← ∞
        for feature2 in Features2 do
            Distance ← EuclideanDistance(Descriptors1[feature1], Descriptors2[feature2])
            if Distance < MinDistance then
                MinDistance ← Distance
                BestMatch ← feature2
        MatchedPairs1to2[feature2] ← BestMatch
    for feature2 in Features2 do
        BestMatch ← null
        MinDistance ← ∞
        for feature1 in Features1 do
            Distance ← EuclideanDistance(Descriptors2[feature2], Descriptors1[feature1])
            if Distance < MinDistance then
                MinDistance ← Distance
                BestMatch ← feature1
        MatchedPairs2to1[feature2] ← BestMatch
    for feature1 in Features1 do
        Match2 ← MatchedPairs1to2[feature1]
        if Match2 is not null and MatchedPairs2to1[Match2] == feature1 then
            Save (feature1, Match2) as a matched pair
```

However, the matching of descriptors does not guarantee the absence of incorrectly matched features. To counter that, an outlier removal algorithm was implemented, which was based on the core ideas of RANSAC. This algorithm has the following steps (The pseudo-code of the outlier removal algorithm is shown in Listing 2):

1. **Pairwise Registration and Transformation**: For a matched pair of features estimate the registration parameters (the translation vector) that align point cloud 2 to point cloud 1 by using only that feature pair. Then, the estimated registration parameters are applied to transform point cloud 2.

2. **Counting Inliers and Outliers**: After transforming point cloud 2, the number of feature pairs that have successfully aligned (inliers) is counted. Additionally, the number of feature pairs that remain misaligned (outliers) is found.

3. **Selection of Optimal Pairs**: This procedure is repeated for all matched feature pairs. Then, these pairs based on the number of inliers resulting from their respective transformations are ranked. From that ranking, select the top 10 matched pairs with the highest inlier count. Use these top 10 pairs for further optimization processes in the point cloud alignment task.

Algorithm 2 Outlier Removal for Point Cloud Alignment

```
procedure OutlierRemoval(MatchedPairs, PointCloud1, PointCloud2)
    InlierCounts ← empty list
    for pair in MatchedPairs do
        RegistrationParams ← EstimateRegistrationParameters(pair, PointCloud1, PointCloud2)
        TransformedPointCloud2 ← ApplyTransformation(PointCloud2, RegistrationParams)
        Inliers, Outliers ← CountInliersOutliers(TransformedPointCloud2, PointCloud1)
        InlierCounts[pair] ← Inliers
    TopPairs ← SelectTopNInlierPairs(InlierCounts, 10)
    for pair in TopPairs do
        Use pair for further optimization
```

3.2.3 Trajectory Optimization

The resulting matched features, with their IDs, XYZ coordinates, and GPS timestamps, and the corresponding trajectories are loaded into the g^{2}o optimizer. The feature timestamps are used to find corresponding IMU poses in the trajectory data, which are then used as states in the optimizer. Feature coordinates are transformed from real-world to sensor frame coordinates to connect them later with the IMU states, which is done via the error function. The error function is based on the variation in feature locations: the optimizer across iterations varies the IMU poses, which does not alter the relative (to the IMU) location of the features but alters their absolute location. The more detailed steps of the g^{2}o optimization are the following (the pseudo-code for point cloud feature utilization in the g^{2}o optimizer is given in Listing 3):

1. **Data Input and State Initialization:** Trajectories for runs 1 and 2 are loaded into the g^{2}o optimizer. Additionally, matched features, with their specific IDs, absolute XYZ coordinates, and GPS timestamps, are also loaded. Each feature timestamp is used to find and interpolate corresponding poses in the trajectory data, which are subsequently added as states in the optimizer for both runs.

2. **Observation Integration:** Features are transformed from real-world coordinates to sensor frame coordinates. The LiDAR observations (recorded points) are considered to be accurate (with a sub-cm accuracy) and therefore will be fixed. However, their georeferencing, which is based on the IMU pose, is erroneous and will be optimized.

   These transformed features serve as observations within the optimizer, constraining and tuning the states (related IMU poses).

3. **Error Function:** An error function is established based on the variation in feature absolute locations across the two runs. For that, during the optimization iterations, features are re-transformed from the sensor frame to the absolute frame using the corresponding poses and their new absolute position is retrieved. The error vector is then estimated as the differences in feature absolute locations, and minimizing this error is the main goal of the optimization process.

Then, trajectory re-processing is conducted using PosPac software, which already contains the main project for the Glasgow Station dataset. It is done to improve the original trajectory by using corrected IMU (from g^{2}o). This project file has a previously set-up project's scope, GNSS and IMU data linkage, calibration parameters of the system and various processing settings (input/output frames, processing settings of the Kalman Filter, selection of GNSS reference stations). It was previously fine-tuned to have an optimal trajectory processing routine for this dataset. Therefore, in this research, the project is only adjusted to incorporate the optimized IMU poses as 'anchor observations'. These observations are integrated into the trajectory processing with an assigned weight based derived from the standard deviation of each coordinate.

Therefore, before integration, the optimized IMU pose coordinates underwent conversion from EPSG:27700 to ETRS89, and their heights were transformed relative to the project's designated ellipsoid. An estimated standard deviation of 10 consecutive IMU poses is used to approximate the coordinates standard deviation and outlier presence.

Algorithm 3 Feature Utilization in g^{2}o Optimizer

```
procedure OptimizePointClouds(Trajectory1, Trajectory2, Matchedfeatures)
    Load trajectories for runs 1 and 2
    Load matched features with IDs, XYZ locations, and GPSTime
    for each feature in Matchedfeatures do
        Find corresponding pose in Trajectory1 using GPSTime
        Add pose to optimizer as state for run 1
        Find corresponding pose in Trajectory2 using GPSTime
        Add pose to optimizer as state for run 2
    for each matched pair of features do
        Transform features from real location to sensor frame
        Add features as observations in optimizer
    while not converged do
        for each state in optimizer do
            Update state based on observations and error minimization
        Calculate error vector based on feature location differences
        Minimize error to improve alignment
```

3.2.4 Optimized Trajectory Quality Assessment

After the optimization workflow concludes with an optimized trajectory, it is evaluated to identify any improvements made by the optimizer. For example, a method for that is to have a Ground Truth (GT) trajectory for comparison with the optimized trajectory - to assess how the optimized trajectory deviates from the GT trajectory. However, this method is not directly applicable, since trajectory corresponds to the position and orientation of MMS, which depends on a train motion. Due to that, trajectories do not have to align with each other completely. However, the georeferenced data (assuming no significant errors in sensor calibration) must. Therefore, locally assessing the misalignment of generated point clouds will implicitly assess the quality of the resulting trajectory alignment. This assessment is done in different ways:

- **Considering Cloud-to-Cloud (C2C) distances:** By calculating the distance from points in one point cloud to the nearest points in the other cloud, it is possible to estimate the order of the alignment. This metric tends to correlate with the alignment of clouds and was also utilized in research for the same reason (Jing et al., 2021). However, the overlap of the used point clouds affects the effectiveness of this metric. Evaluation of the point cloud misalignment with the help of C2C distances will fail if the point clouds do not have a small overlap. In those cases, manual selection of the overlapping regions is necessary.
- **Considering Feature Distance:** If there are correctly identified and matched features (automatically or manually), then the distance between the corresponding features before and after trajectory optimization can be used as a metric to assess point cloud misalignment. These could be the same features used by the optimizer or some other manually identified and properly matched features such as identifiable objects.
- **Considering GNSS processing software metrics:** The PosPac software offers a variety of quality assessment metrics for processed trajectory data. Among these, the smoothed performance metrics specifically address the positional errors for each component of a position in the final processed solution. These metrics represent the accuracy of the navigation errors that have been modeled and removed, thereby indicating the reliability of different parts of the solution.

Additionally, it is possible to simulate a GT trajectory (Jing et al., 2021). However, in that research, the original dataset was collected with a good GNSS signal, resulting in cm-level positioning, and the GNSS observations were excluded on a continuous segment of data during the trajectory processing part. It is done to artificially simulate GNSS signal occlusion present in the indoor environments (tunnels, underground stations) and, therefore, allow IMU to drift. For this research, this method is not applicable, since the focus of the research is on the tunnel environment and the developed methodology was not created with a focus on the outdoor environment.

3.3 Global Trajectory Optimization

The local optimization algorithm was scaled to cover a larger scope. Instead of focusing on a selected tile, a sequence of tiles of a run is used in the global optimization algorithm. This scaling of the algorithm represents its

intended use scenario. Additionally, it reveals potential performance enhancements and challenges that might emerge from incorporating features across multiple tiles.

The global optimization algorithm gets as an input a sequence of tiles and the larger trajectory, which covers the area of the input tiles. Then the same steps are performed with the data with small deviations from the local optimization algorithm:

1. **Feature Extraction & Feature Matching:** These steps are identical to the corresponding steps described in the local trajectory optimization. Features are extracted and matched for each tile in the provided set of tiles consequently.

2. **Trajectory Optimization:** The matched features from each tile are merged in one dataset. Then, the dataset along with the trajectory is provided to the g^2o optimizer, where the same procedure is performed as described for the local trajectory optimization algorithm. The output of the g^2o optimizer is projected to the ETRS89 and given to the GNSS processing softare

3. **Optimized Trajectory Quality Assessment:** The global trajectory optimization's outcomes were evaluated using the same QC tool used for the local optimization algorithm. Yet, due to the expansive nature of the dataset, a comprehensive qualitative analysis was not undertaken.

4 Results

The aforementioned methodology was implemented and the corresponding results are presented in this chapter. First, subsection 4.1 will cover the results of developing the local trajectory optimization algorithm. The last subsection 4.2 of this chapter will conclude with presenting the results from the global trajectory optimization algorithm.

4.1 Local Trajectory Optimization

4.1.1 Feature Extraction

The ISS feature detector implementation from Open3D point cloud processing library was used and the parameter configurations mentioned in the Table 3.1 were tested. The outcomes of these tests on the selected and synthetic tiles are detailed in Table 4.1. As it was expected, configurations with larger salient radii required more time for completion. Furthermore, Figure 4.1 illustrates the ISS keypoints detected on the selected tile for configuration #1, showing that features were mainly identified in areas where the geometry of the environment changes, such as the intersection of walls and ground, along the railways and wires, and around the contours of emergency exits. This pattern was consistent across both the selected tile and the generated synthetic acquisition.

From the tested parameters configurations, configuration #2 was chosen for further proceeding as it generated a satisfactory number of keypoints. Additionally, it covered a broader region than configuration #1 by considering more neighbors, and was deemed capable of providing adequate registration capabilities (for example, even if only 10% of keypoints matched correctly). The computational time of just over a minute was considered acceptable compared to configurations #3 through #5.

Config. #	Execution Time [s]	Run1 features #	Synthetic Run1 features #
1	20.32	708	887
2	73.80	218	237
3	175.50	84	97
4	357.761	48	51
5	545.24	14	24

Table 4.1: Results of the ISS Feature extraction for run1 and synthetic run1. The execution time is an average of run1 and synthetic run1 times.

This step performed on 2 real data acquisitions resulted in the following outcomes. The ISS detection step detected 218 and 197 ISS features for Run1 and Run2. The step took 71 and 85 seconds for the acquisitions.

4.1.2 Feature Matching

Upon detecting keypoints using the ISS feature detector (configuration #2), the next step involved calculating the FPFH feature descriptors. This process utilized the Point Cloud Library's (PCL) existing implementation. The calculation required the full point clouds along with the set of detected ISS keypoint indices. Various values for the FPFH feature descriptor search radius were explored, as detailed in Table 3.2. The results are shown in Table 4.2. It was observed that FPFH calculations are notably computationally expensive, with execution times extending several minutes even for smaller search radii. Therefore, configuration #1 was selected for FPFH due to the lowest computational time.

Configuration #	Execution Time [s]
1	139.37
2	788.7
3	2697.0

Table 4.2: Results of FPFH Feature descriptor calculation for different parameters. The execution time is an average of run1 and synthetic run1 times.

Following the calculation of descriptors using configuration #1, the next step involved matching features based on the similarity of the descriptors. This step was executed relatively quickly - matching approximately 200 features from each acquisition required only about 11-12 seconds. The accuracy of this matching procedure was quantifiable, since the synthetic acquisition involved a known translation vector. Figure 4.2 illustrates the

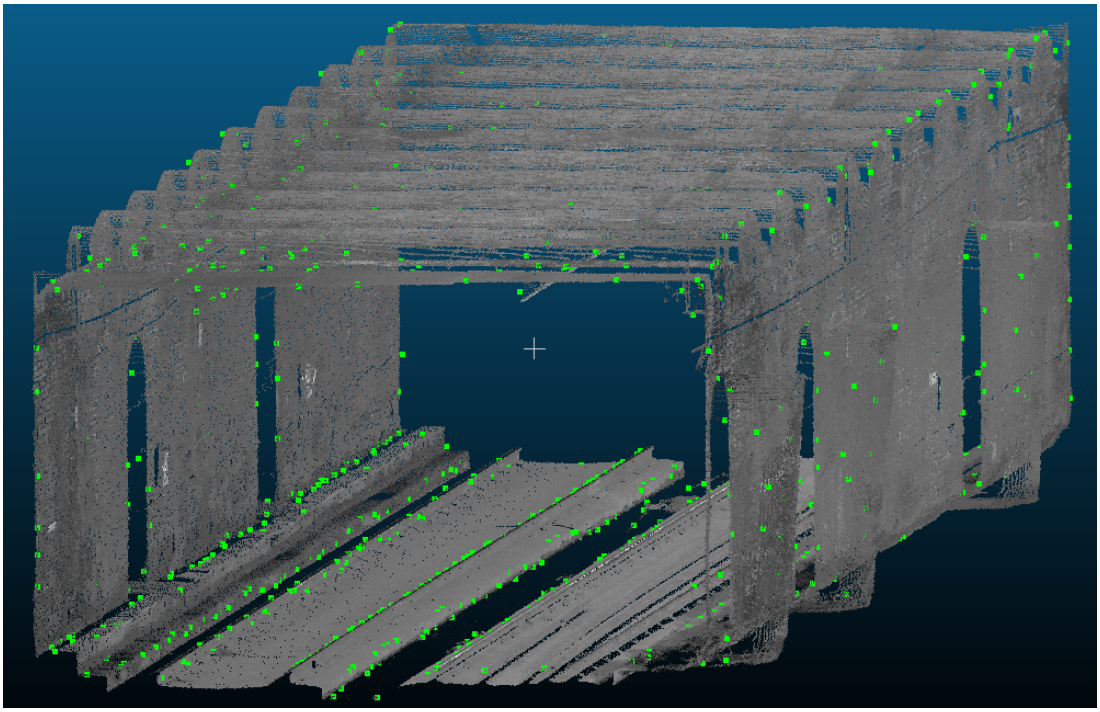


Figure 4.1: Keypoints (in green) detected by the ISS detector for the straight tunnel from the Glasgow Dataset using parameters from configuration #1.

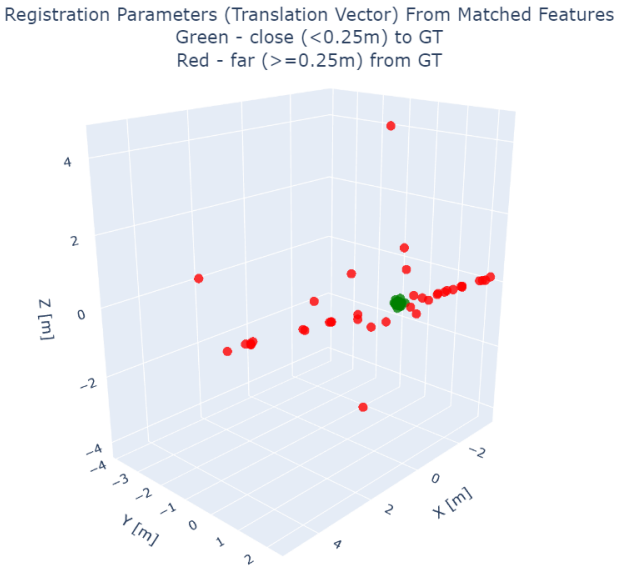
distribution of registration parameters (translation vectors) determined by the matched feature pairs. Only ≈ 50 % of the matched keypoints yielded registration parameters close to the Ground Truth (GT). Additionally, some registration parameters are aligned linearly, which may be attributed to the homogeneous nature of objects in the environment (continuous in the driving direction), as also visible on Figure 4.1, where many points align along the intersections of planes, railheads, and wires. Due to the continuous and homogeneous nature of wires, railheads, and walls, or the absence of any other descriptive feature around them, the descriptors of two points on these objects will be similar, even though they could be separated by several meters.

Additionally, a comparison of derived registration parameters deviation from the Ground Truth and descriptors similarity revealed no correlation. Considering this observation alongside the lengthy computational times required for descriptor calculations, it was decided to exclude the FPFH descriptor calculation from the methodology. Instead, detected keypoints were directly forwarded to the outlier removal step.

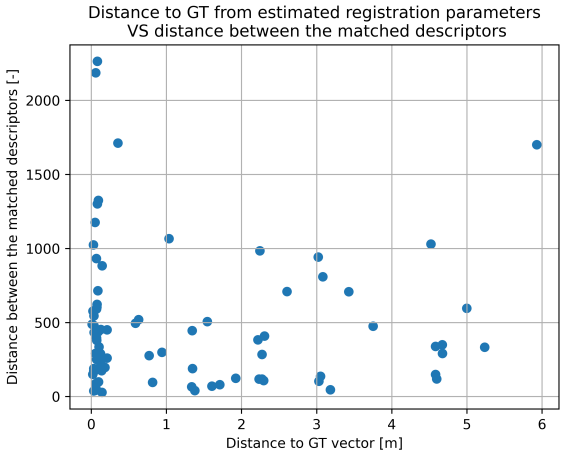
Given the undemonstrated utility and considerable computational resources required for FPFH feature descriptor computation, the workflow was adapted to utilize the detected ISS keypoints immediately in the outlier removal stage. This step involved generating all possible matching pairs from the detected keypoints. Subsequently, registration parameters were derived from these pairs, which were then ranked according to the number of inliers produced. The top ten feature pairs, based on inlier count, were selected for further analysis. Figure 4.3 displays the derived translation vectors from the selected feature pairs in comparison to the GT translation vector. The derived parameters closely align with the GT, deviating by less than a centimeter in each dimension, which is within the magnitude of noise applied to the synthetic acquisition. Additionally, the outlier removal process was significantly less computationally demanding than the FPFH descriptor calculation - it was completed in roughly 60 seconds.

For the real data acquisitions, the feature descriptor calculation step was also skipped and outlier removal was applied directly to the detected ISS keypoints. The procedure took 44.4 seconds and the registration parameters from the top 10 matched feature pairs are shown in Figure 4.4. The derived registration parameters are clustered together with a standard deviation of 1.5 and 1.4 cm for X and Z axis. The standard deviation for Y axis is 0, since all derived parameters happen to be on the same Y coordinate.

The ground-truth translation is not known for the non-synthetic data. Therefore, the derived registration parameters can be incorrect. To evaluate these parameters, Run2 was translated using the average of the derived registration parameters, as indicated in Figure 4.4. Additionally, Cloud2Cloud (C2C) distances were calculated before and after the translation. Both results are shown in Figure 4.5. As can be seen, this translation visually aligned the point clouds. Moreover, calculated C2C distances decreased after the alignment - the average value reduced from 0.8 to 0.03m and the standard deviation reduced from 0.311m to 0.062m.



(a) 3D scatter plot of registration parameters (translation vectors) from matched features, color-coded by proximity to Ground Truth (GT): Green points represent matches within 0.25m of GT, red points indicate matches further than 0.25m from GT. In total 81 feature pairs were matched, however only 42 or 51.85% of them were close (<0.25m) to the known ground truth translation.



(b) Scatter plot comparing the distance of estimated registration parameters from GT with the Euclidean distance between the matched descriptors. No correlation ($\rho = -0.006$) was found.

Figure 4.2: Analysis of Feature Matching Accuracy and Descriptor Similarity.

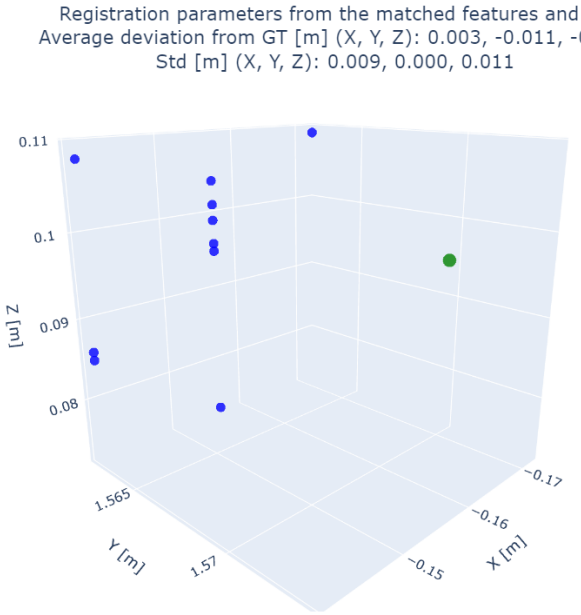


Figure 4.3: 3D scatter plot displaying the registration parameters obtained from the outlier removal process (denoted in blue) in comparison to the Ground Truth (denoted in green).

Additionally, 2 vertical cross-sections were done to the aligned point clouds - perpendicular to the driving direction and normal direction (shown in Figure 4.6). These cross-sections are supposed to highlight the misalignment (or alignment) of the point cloud in the corresponding directions. As can be seen from the Figures, the point clouds align well in both directions.

The combined evaluation—taking into account the narrowed C2C distribution and the visual alignment—suggests that the registered acquisitions have an error margin at the centimeter level. Due to that, the developed pipeline to extract point cloud features was considered sufficient. This degree of precision validates the feature extraction

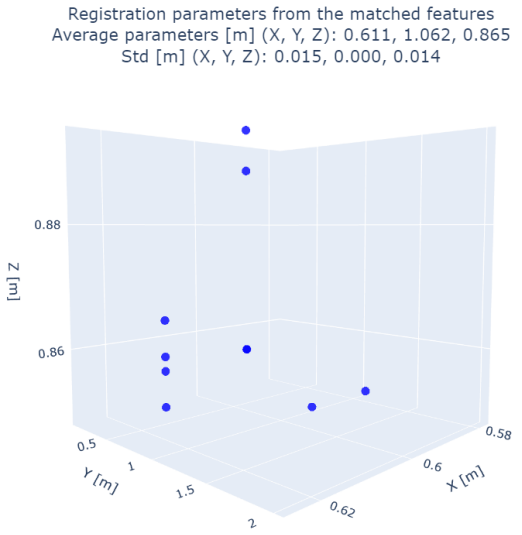
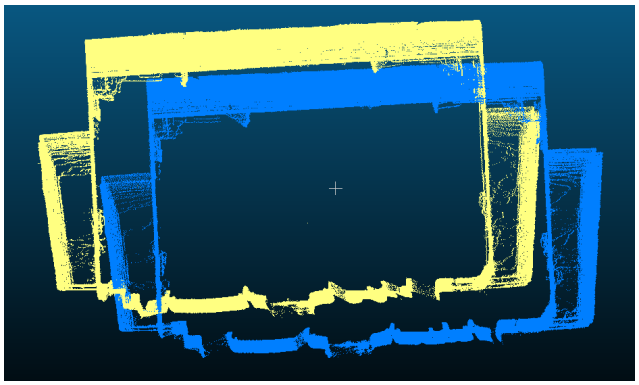
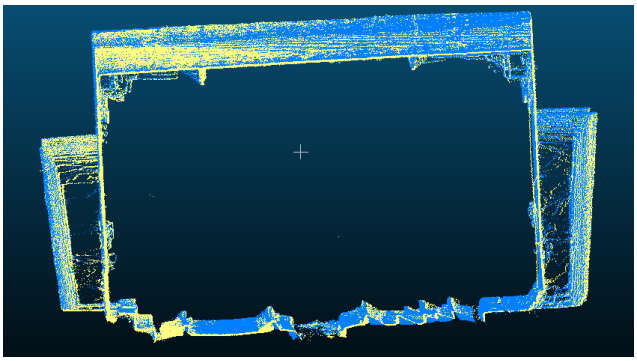


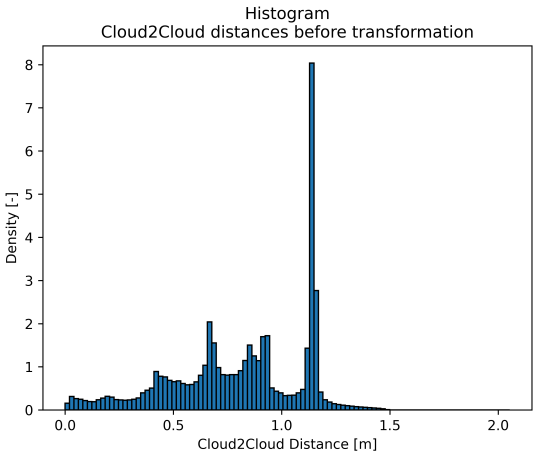
Figure 4.4: Derived registration parameters from the top 10 matched feature pairs after the outlier removal step.



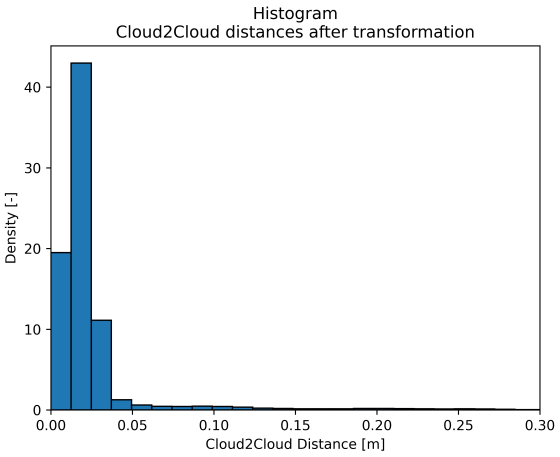
(a) Two data acquisitions (run 1, blue and run2, yellow), which are originally misaligned due to the drift in IMU.



(b) The yellow acquisition (run2) was shifted by the average vector estimated from top 10 selected features. Due to that, its alignment with the blue acquisition (run1) was visually improved.

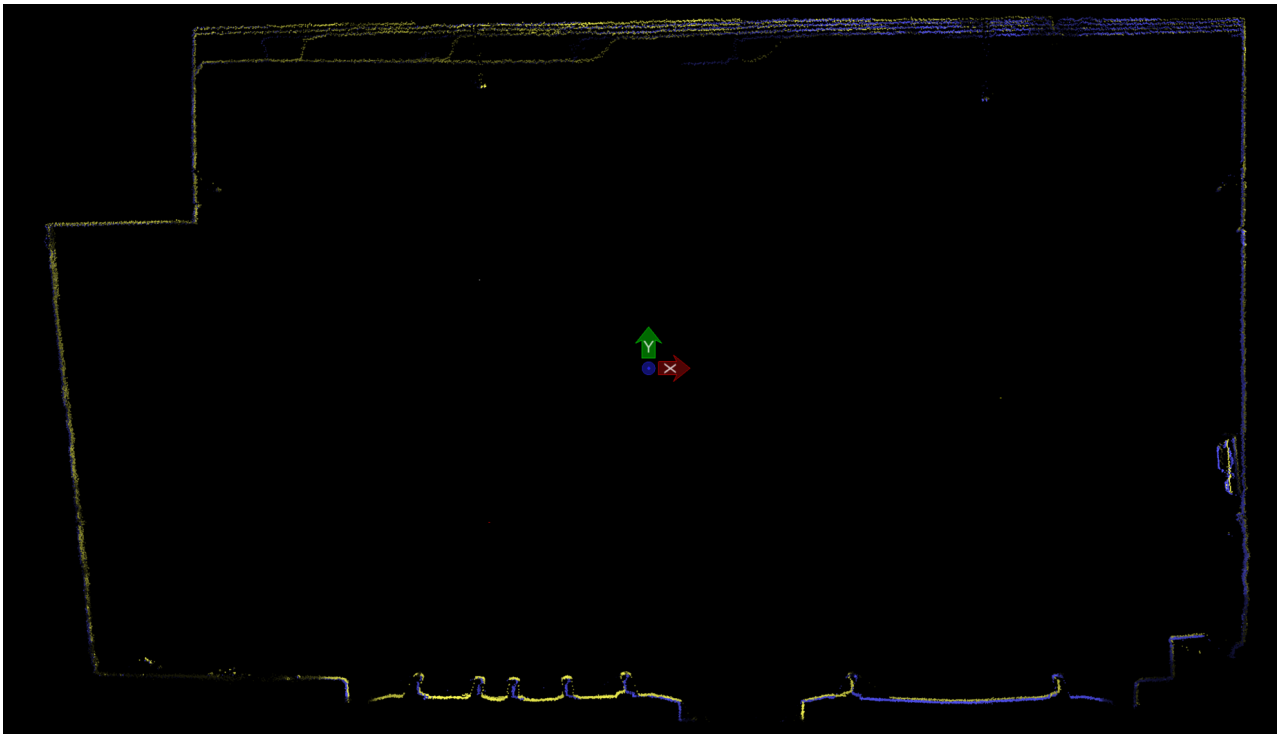


(c) Distribution of the Cloud2Cloud computed distances before the translation of the yellow acquisition (run2). The mean Cloud2Cloud distance is 0.807m with a standard deviation of 0.311m.

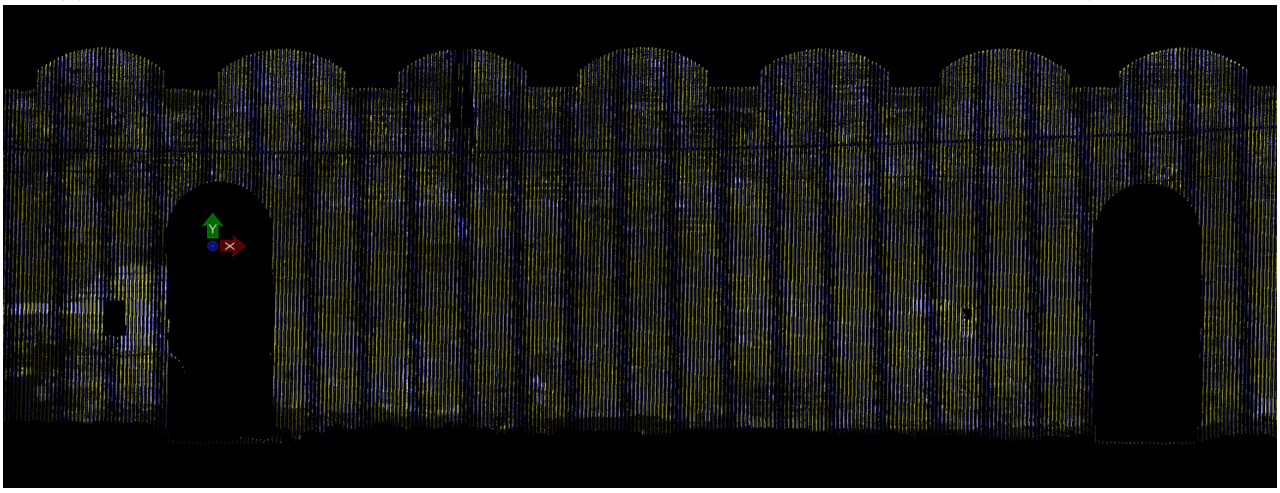


(d) Distribution of the Cloud2Cloud distances computed after the translation of the yellow acquisition (run2). The mean value of the distribution is 0.031m with a standard deviation of 0.062m. Both values were significantly decreased after the translation.

Figure 4.5: Manual alignment of point cloud acquisitions by using the average registration parameters from matched top 10 features.



(a) Crossection in the normal direction. The railheads, the walls and the ceiling of the scene align well (within 1 cm).



(b) Crossection in the driving direction. The emergency exits along with the general geometry of the tunnel align well.

Figure 4.6: Crossections (≈ 10 cm thick) of the point clouds (run1, yellow and run2, blue) after the manual alignment of point clouds by using the average registration parameters from top 10 features.

pipeline developed in this study, which was proficiently used to detect, extract, and match features subsequently supplied to the trajectory optimization stage.

4.1.3 Trajectory Optimization

The optimization was executed using the g^{2}o framework, which achieved convergence after eight iterations, resulting in an average error of $1.78E-9$ meters and completing in 0.05 seconds. The transition of IMU poses, driven by the optimization, is depicted in Figure 4.7. The optimization modified only the spatial locations of the poses, IMU attitude was fixed. The present uniform adjustment originated from the equal weighting of features within the optimization process, leading to the features' convergence at a central point between their initial locations.

The repositioning of features themselves after the optimization is shown in Figure 4.8. It highlights the achieved alignment of the features. The optimized poses then were utilized for the subsequent GNSS processing step: to reprocess the trajectory leveraging the optimized features to correct the IMU drift and to re-georeference the point clouds.

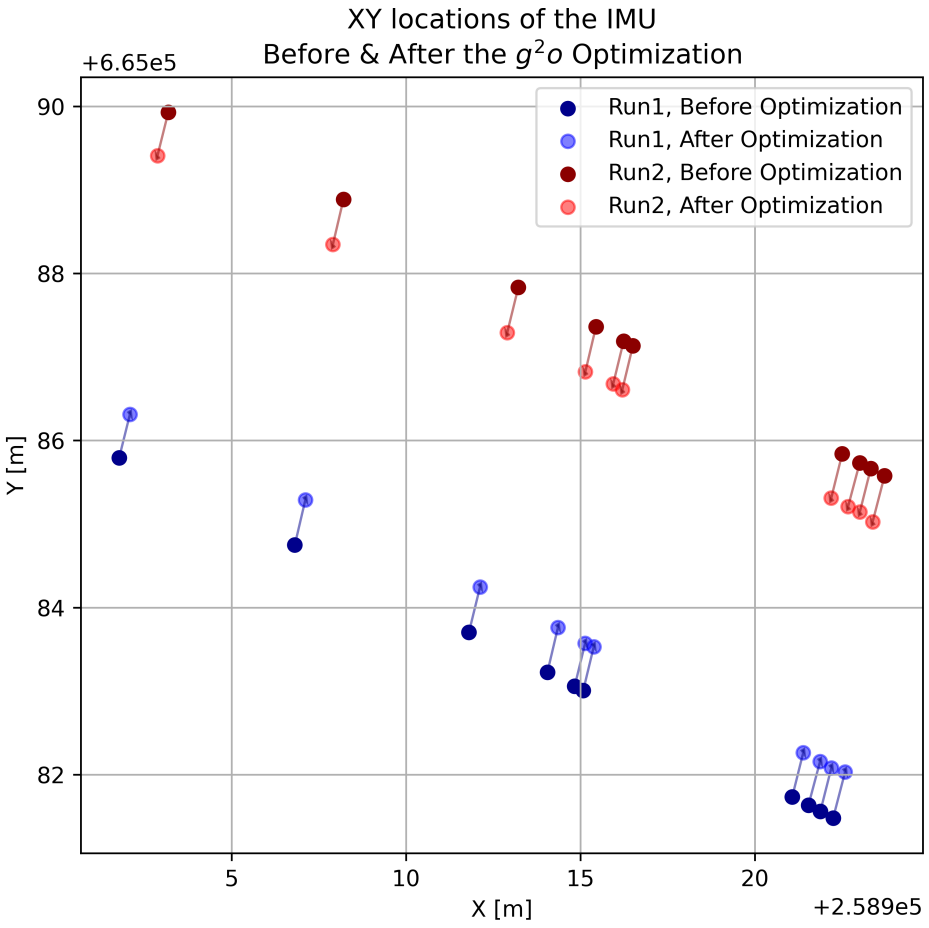


Figure 4.7: Visualization of the IMU locations before and after the optimization. Thin dark blue and dark red lines visualize the corresponding initial and final locations of the IMU poses.

4.1.4 Optimized Trajectory Quality Assessment

The anchor observations were constructed from the g^{2}o optimized IMU poses and imported to the project. The trajectory was then reprocessed.

The outcome of this reprocessing, specifically the Smoothed Performance Metrics indicating the North, East, and Down position errors, are presented in Figure 4.9. A reduction in RMS metrics is observed of the original trajectory around the epochs of the applied corrections. Mainly, it can be observed in the RMS peaks, which decreased from 3-3.5m to $\approx$ 1.5 m, inidicating a more reliable solution due to the introduction of aiding to the trajectory processing in the GNSS occluded section. Figure 4.10 provides a better zoomed-in view of the RMS plot at the epochs corresponding to the corrected epochs, demonstrating the reduction of the RMS for all free components due to the local corrections. Additionally, the smoothing effect of the processing also extended the

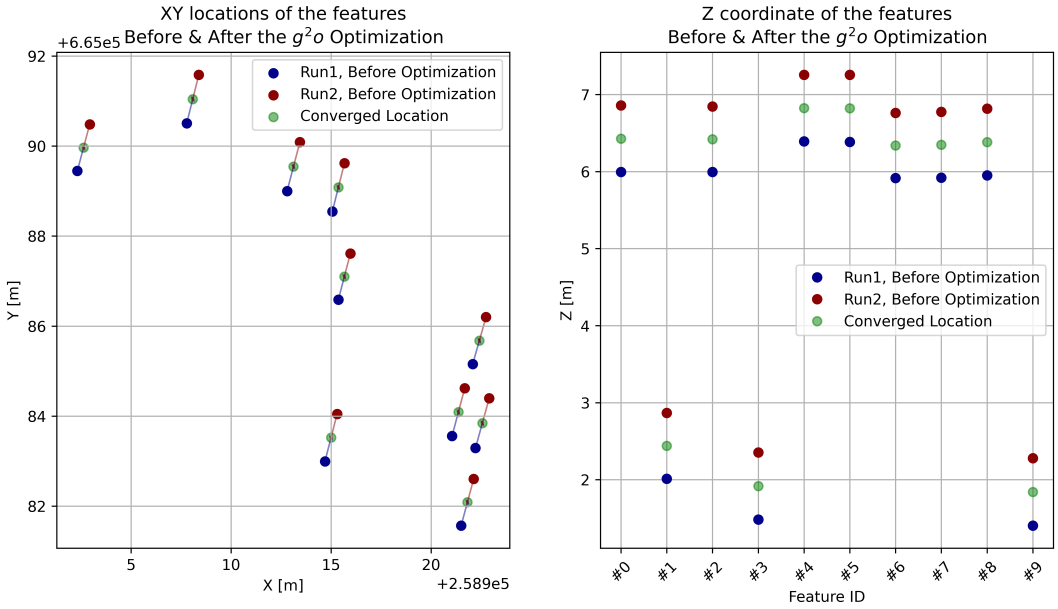


Figure 4.8: Visualization of the extracted point cloud feature locations before and after the optimization. Thin dark blue and dark red lines visualize the corresponding initial and final locations of the features.

impact of these corrections to neighboring epochs, affecting RMS values beyond 100 seconds from the corrected poses.

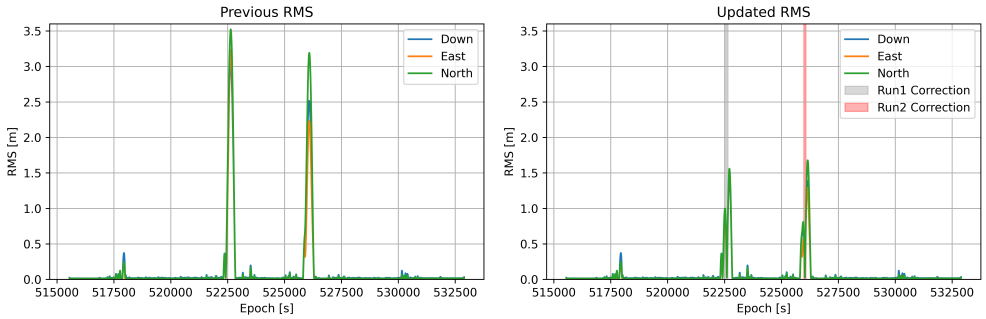
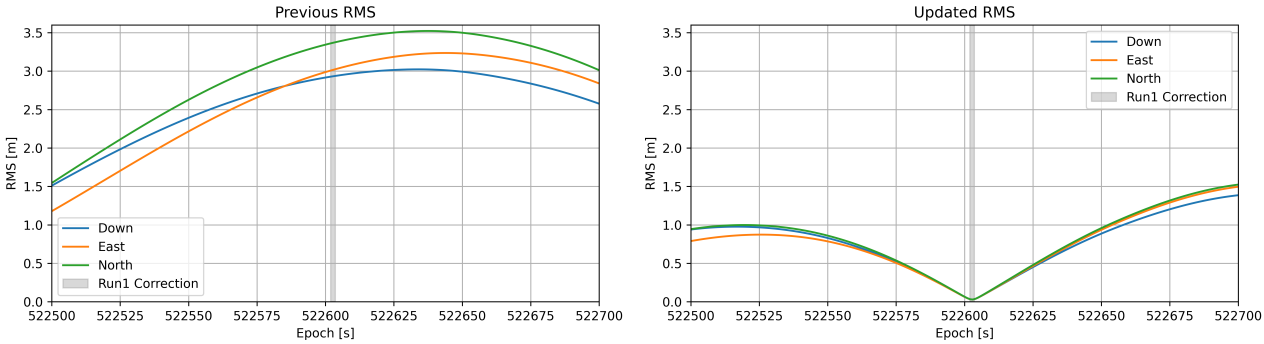


Figure 4.9: Root Mean Square (RMS) error of the trajectory's positional accuracy before and after the reprocessing. The 'Updated RMS' plot demonstrates the reduction in these errors, with the time intervals of the corrected IMU observations for Run1 and Run2 distinctly marked (indicating where the trajectory adjustments were applied).

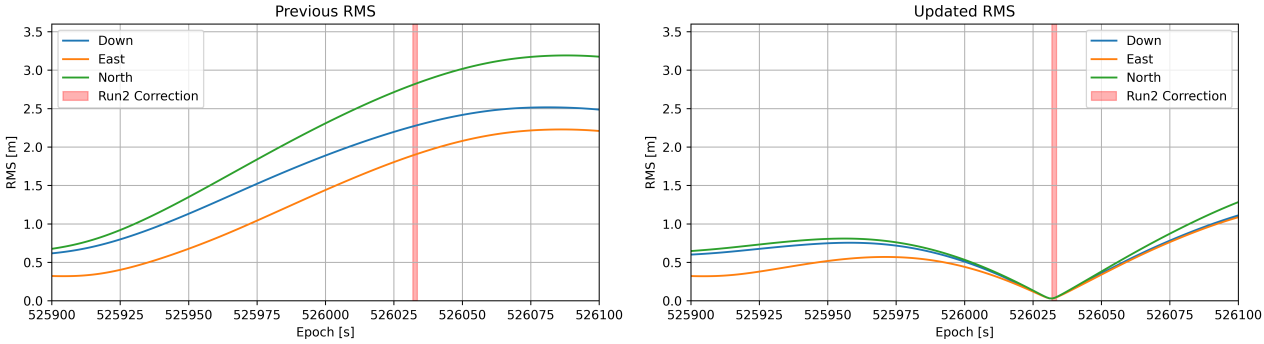
Figure 4.11 illustrates the component-wise corrections applied to the trajectory (separate corrections to the X, Y and Z components). It is evident from the plots that the corrections in all components have an opposing trend, due to the influence of the optimizer and its relative spatial locations - the optimizer "pulled" the trajectories closer to each other, hence the opposite corrections. Moreover, while the magnitudes of these adjustments are similar, they are not completely identical, reflecting the additional considerations, which are present in the trajectory processing workflow.

After the trajectory optimization, the straight tunnel tile underwent a georeferencing process utilizing the updated trajectory. The results, depicted in Figure 4.12, highlight an improvement in alignment. This enhancement is apparent both visually and through C2C distance metrics, which now reflect a mean of 0.033 m and a standard deviation of 0.068 m. Moreover, misalignments in both the normal and driving directions have been reduced from approximately 1.5 m to a magnitude of 1-2 centimeters. These results are close to those derived from manual alignment using feature extraction directly.

The aforementioned Figures 4.9 and 4.11 illustrate that the localized corrections have had a broader impact, influencing a wide range of epochs within the trajectory data. This effect is also visible in the georeferenced point clouds. For instance, the georeferencing of a section approximately 150 meters from the selected tile also shows alignment improvements. Figure 4.13 depicts the alignment of those acquisitions (150m apart from the



(a) The epochs corresponding to the corrected Run1 IMU.



(b) The epochs corresponding to the corrected Run2 IMU.

Figure 4.10: Detailed RMS plot of the processed trajectory.

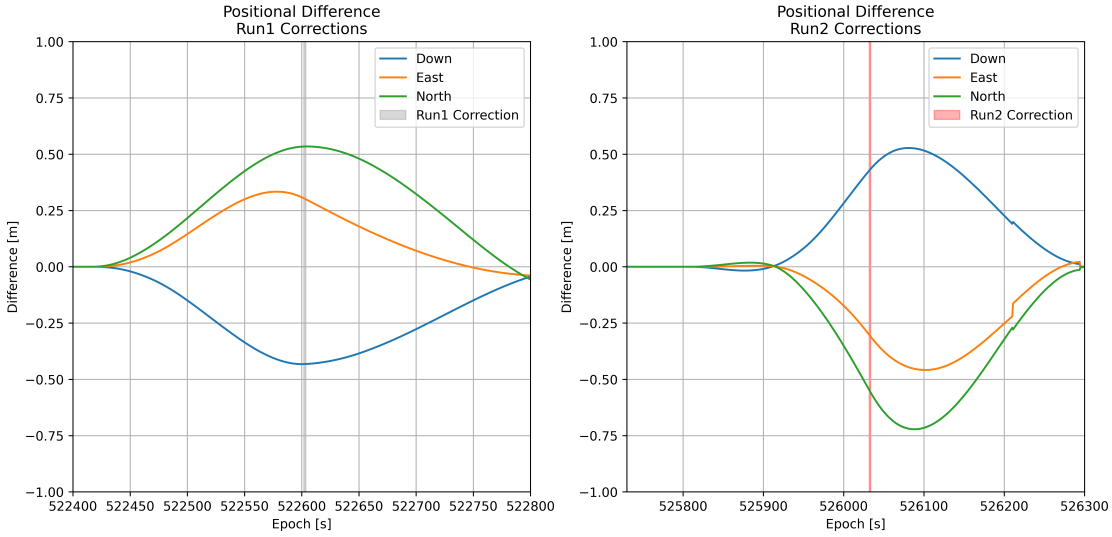
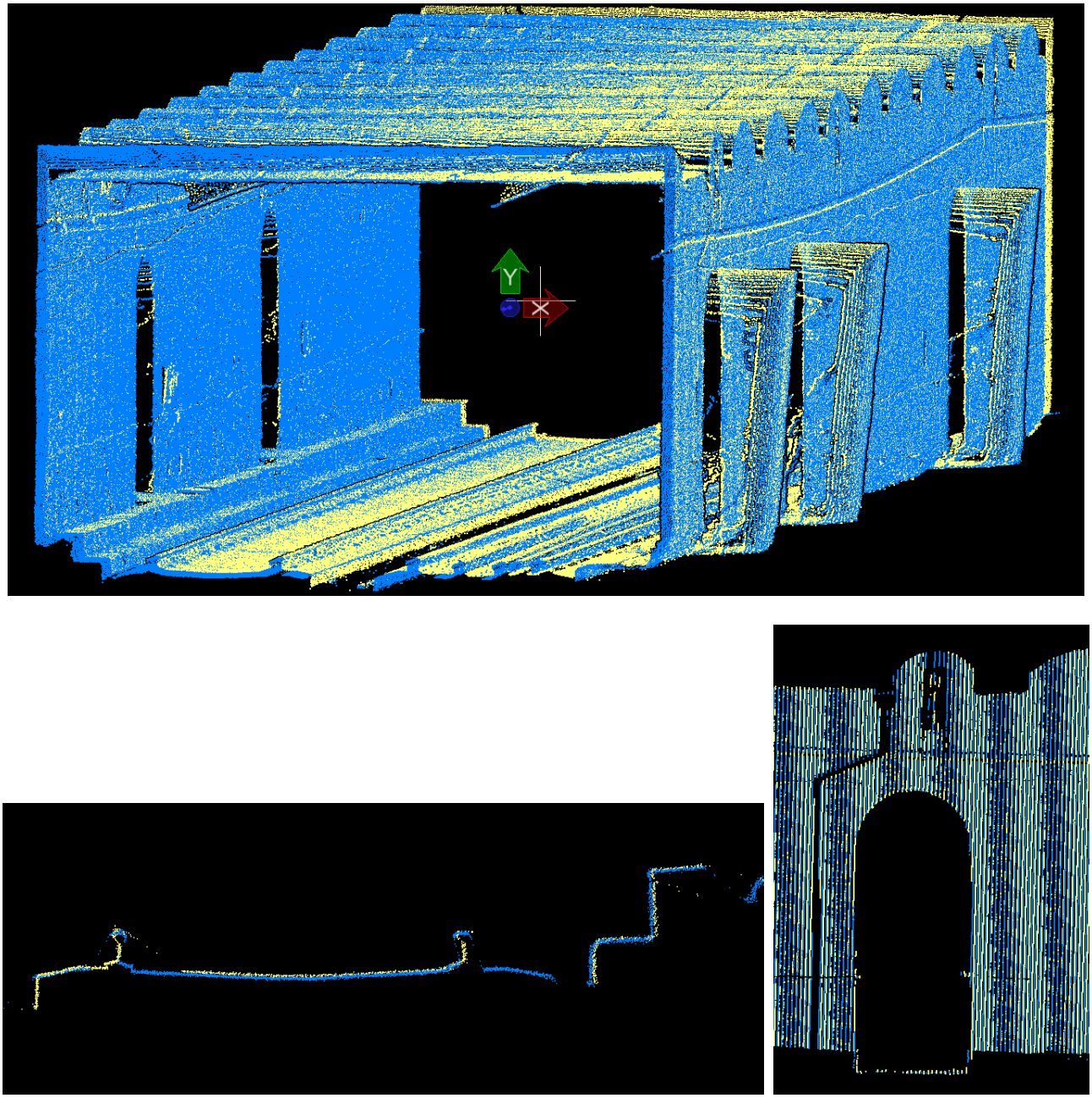


Figure 4.11: Trajectory position difference due to the added corrections.

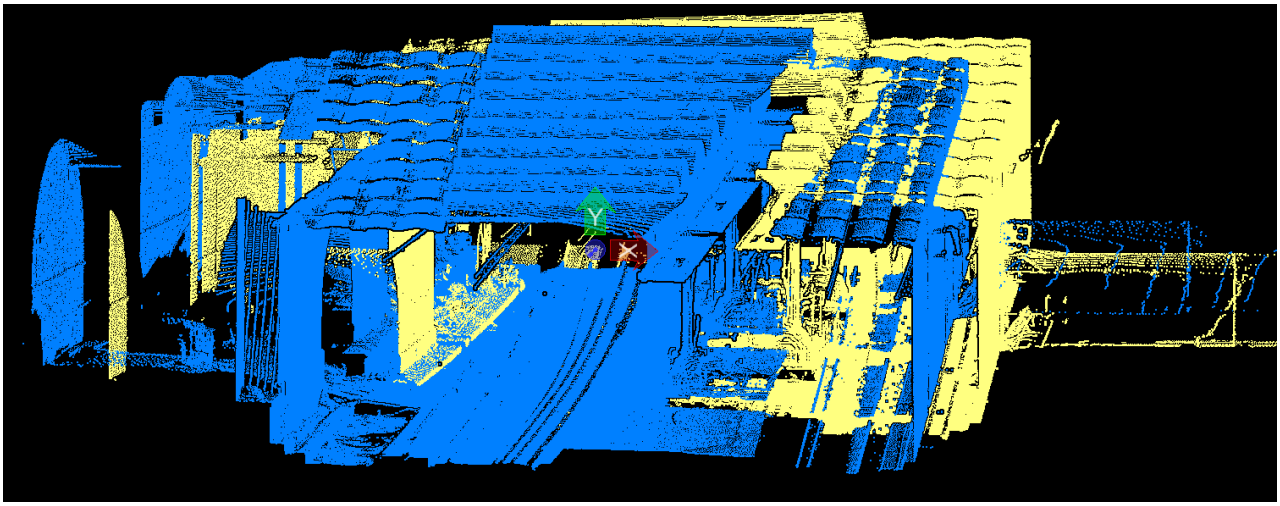
straight tunnel tile) in both the normal and driving directions before and after trajectory optimization. The final alignment is not as precise as that of the tile from which features were extracted. However, it still presents a significant improvement over the original state, with residual misalignments on the order of 8-10 cm (reduced from $\approx$ 1.5m). Despite these small remaining discrepancies, the optimized trajectory yields a net enhancement in point cloud alignment.



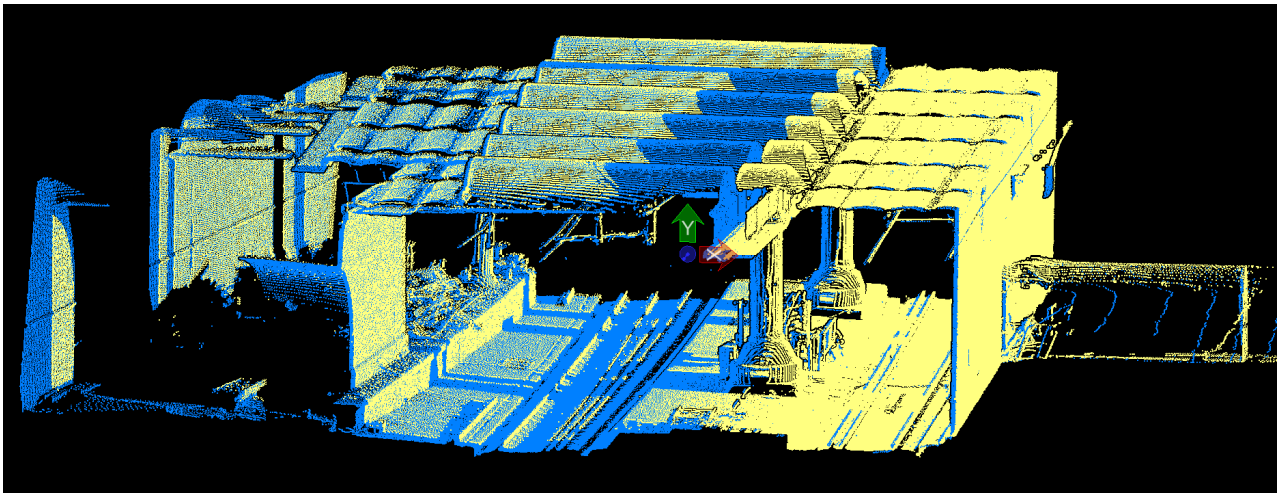
(a) Point cloud crossection highlighting the normal direction alignment (within a cm error).

(b) Point cloud crossection highlighting a driving direction alignment.

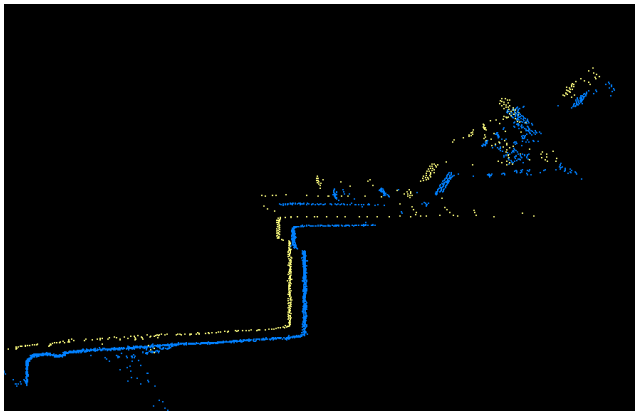
Figure 4.12: Re-georeferenced acquisitions of Run1 (yellow) and Run2 (blue).



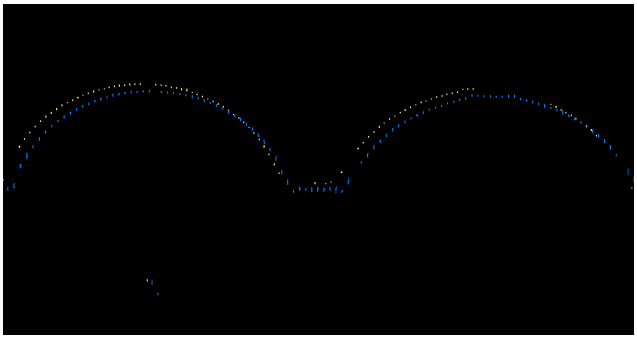
(a) Misalignment of the point clouds before the trajectory correction



(b) Improved alignment of the point clouds after the trajectory corrections.



(c) Small (8 - 10cm) misalignment still present after trajectory correction in normal direction.



(d) Small (8 - 10cm) misalignment still present after trajectory correction in driving direction.

Figure 4.13: Original and re-georeferenced acquisitions of tunnel 150m furthen from the straight tunnel section used for optimisation. Two acquisitions: Run1 (yellow) and Run2 (blue).

4.2 Global Trajectory Optimization

The global trajectory optimization was performed on the 120 tiles from the Glasgow dataset with the parameters selected during the development of the local trajectory optimization. The total processing of all tiles required 37980 seconds (or 10.55 hours). The median processing time per tile is just over 3 minutes, which aligns with the expectations from the local trajectory optimization results. However, some of the tiles in this dataset required significantly more time - more than an hour - to complete (complete processing time statistics shown in Table 4.3).

	Min	Median	Average	Max	Std
1 Tile Processing Time [s]	90.76	187.92	316.50	4819.84	607.07

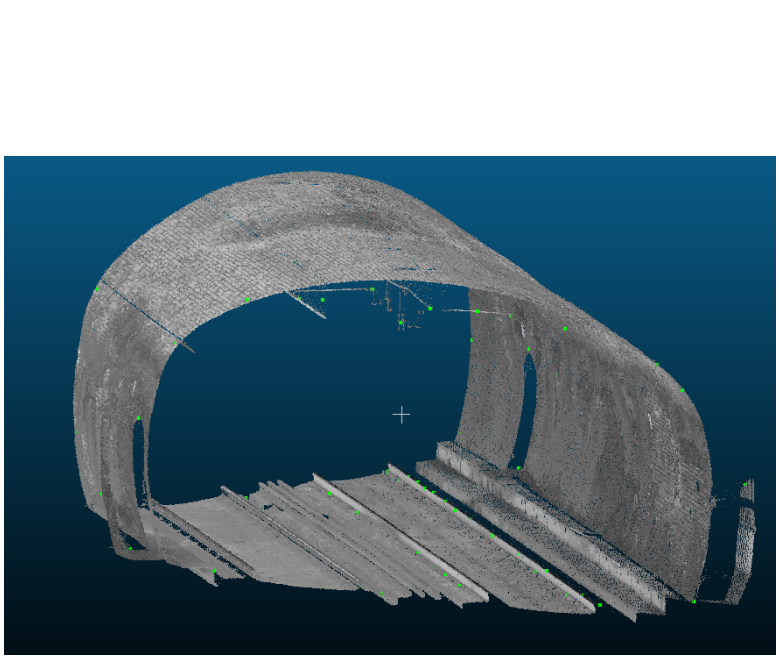
Table 4.3: Statistics of complete processing time for a tile.

4.2.1 Feature Extraction

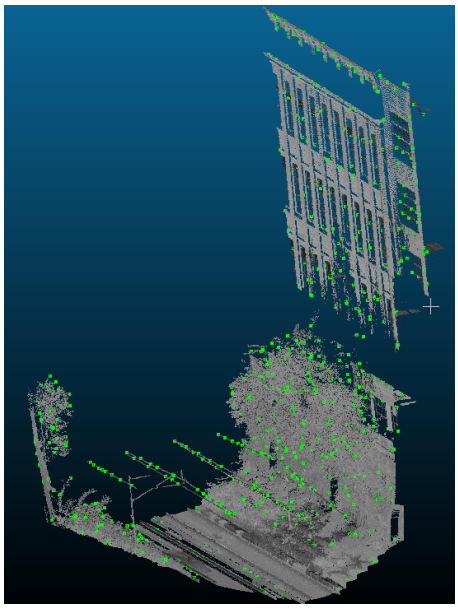
The results of the feature extraction step are summarized in Table 4.4. The time required for this step does not vary as much as the total processing time shown above, which indicates a low impact of data variability on the processing time of the feature extraction step. Moreover, the amount of extracted features varied per tile, which was caused by the feature-richness of the tile. By assessing the tiles with the largest amount of keypoints, it was revealed that they correspond to the section, where the tunnel is temporarily exposed to the outdoor environment. The vegetation (such as trees) or neighboring buildings with windows are captured by the LiDAR and keypoints are detected in them. On the contrary, the smallest amount of features was detected in cases, where the train passed a tunnel with a circular geometry. In those tunnels, the amount of features does not exceed 100, most likely due to the smooth and feature-poor geometry of the tunnel. An example of feature over-detection and under-detection is shown in Figure 4.14.

	Min	Median	Average	Max	Std
Feature Extraction Time [s]	60.27	140.50	136.82	241.74	33.37

Table 4.4: Statistics of processing time in a feature extraction step (for 2 acquisitions of a tile).



(a) Point cloud of a feature-poor tunnel (colored in grey, by intensity). Detection of the features in the smooth and circular tunnel resulted in approximately 60 features in total (colored in green).



(b) Point cloud of a feature-rich tunnel (colored in grey, by intensity). Detection of the features in the tunnel with an absent ceiling resulted in approximately 700 features in total (colored in green).

Figure 4.14: Example of the most feature-poor and feature-rich tiles in the Glasgow Station dataset.

4.2.2 Feature Matching

Results of the feature matching and outlier removal step for the global trajectory optimization algorithm are shown in Table 4.5. This step is highly variable in the processing times, which results in increased total processing times. Figure 4.15 shows the relation between the needed time to perform that step and the number of features. The figure indicates the exponential growth of the processing time with the increased amount of detected features.

	Min	Median	Average	Max	Std
Outlier Removal & Matching Time [s]	0.60	17.71	166.89	4722.69	617.25

Table 4.5: Statistics of processing time in a feature matching and outlier removal step (for 2 acquisitions of a tile).

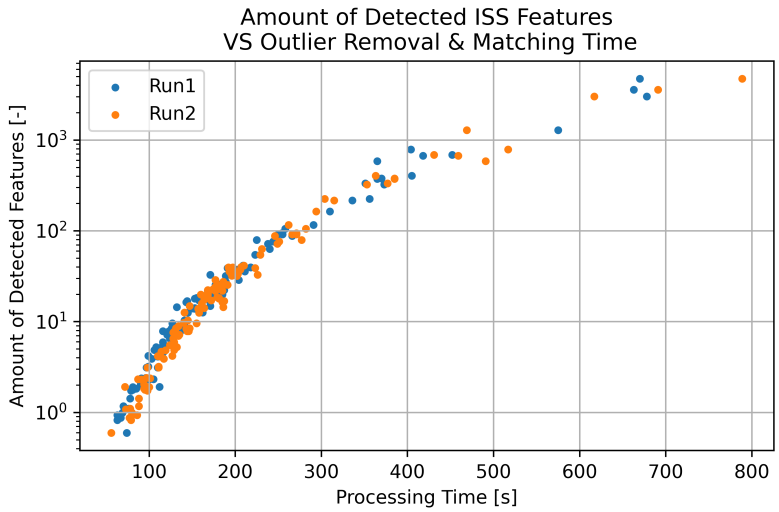


Figure 4.15: The impact of the number of detected features on the timespan of the outlier removal & feature matching step.

4.2.3 Trajectory Optimization

After the features were extracted and matched, they were exported to the g^{2}o to optimize the poses. This procedure took a similar amount of iteration and convergence error compared to the local trajectory optimization results - the optimizer finished after the 9th iteration with an average error of $1.38E-9$ m. The differences between the original and the optimized poses are shown in Figure 4.16. In both plots the actual drift signal can be seen, however, it is cluttered by the abundant amount of outliers.

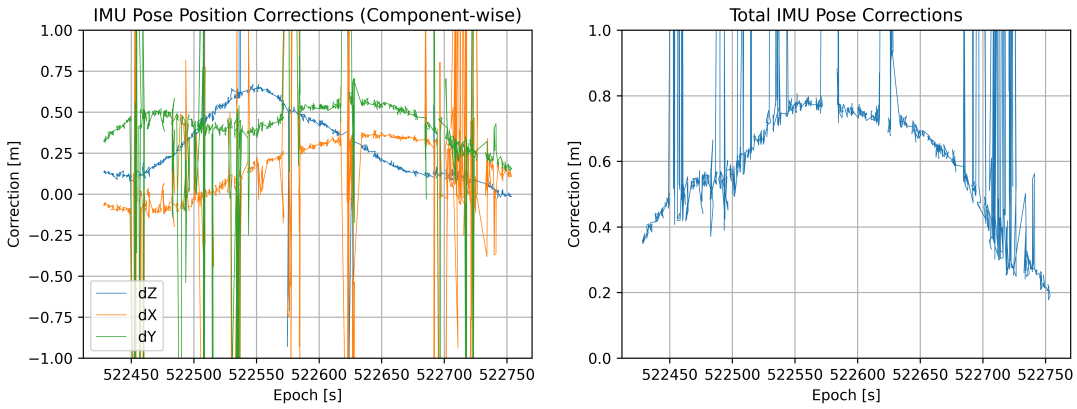


Figure 4.16: Run1 IMU poses corrections after the g^{2}o optimization.

4.2.4 Optimized Trajectory Quality Assessment

The anchor observations were constructed from the g^{2}o optimized IMU poses and imported to the project. The trajectory was then reprocessed and resulting RMS values are shown in Figure 4.17. For global trajectory optimization, a bigger reduction in RMS metrics is observed compared to the local trajectory optimization. The RMS values decreased from 3.5m to ≈ 0.5 m. Figure 4.18 highlights the RMS values at the epochs corresponding to the corrected poses - the reduction of the RMS for all three components is present.

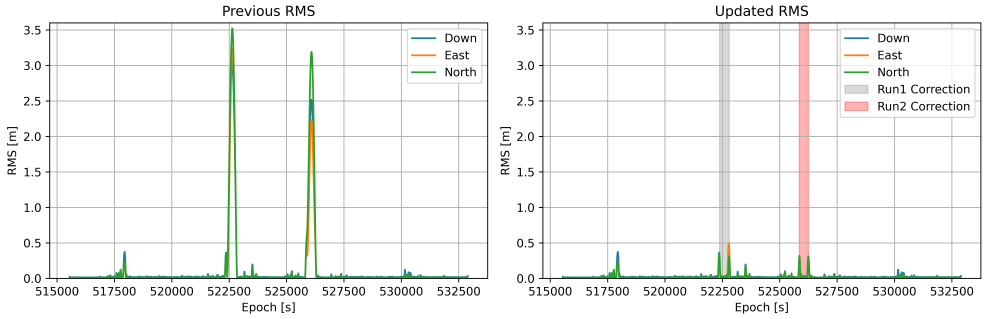
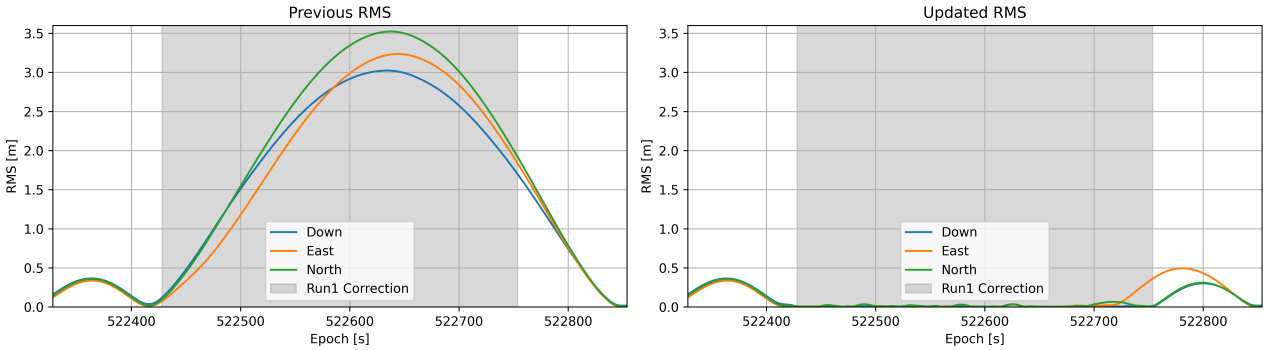
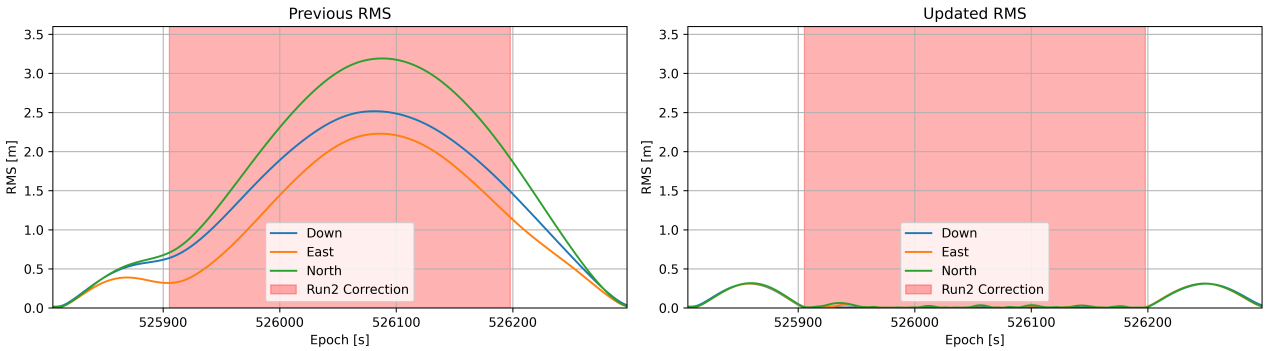


Figure 4.17: RMS error of the trajectory's positional accuracy before and after the reprocessing. The 'Updated RMS' plot demonstrates the reduction in these errors, with the time intervals of the corrected IMU observations for Run1 and Run2 distinctly marked (indicating where the global trajectory adjustments were applied).



(a) The epochs corresponding to the corrected Run1 IMU.



(b) The epochs corresponding to the corrected Run2 IMU.

Figure 4.18: Detailed RMS plot of the processed trajectory after the global trajectory optimization.

Figure 4.19 illustrates the component-wise corrections applied to the trajectory (separate corrections to the X, Y and Z components). The corrections for the global trajectory optimization also have an opposing trend. However, bigger discrepancies between corrections are present compared to the local trajectory optimization - the East component of the Run1 has a significantly larger correction than the same component of Run2. Moreover, the corrections given by the software do not completely repeat the IMU pose corrections from the g^{2}o. This behaviour could be caused by the complex processing chain of the GNSS processing software, however it could also be influenced by the presence of outliers.

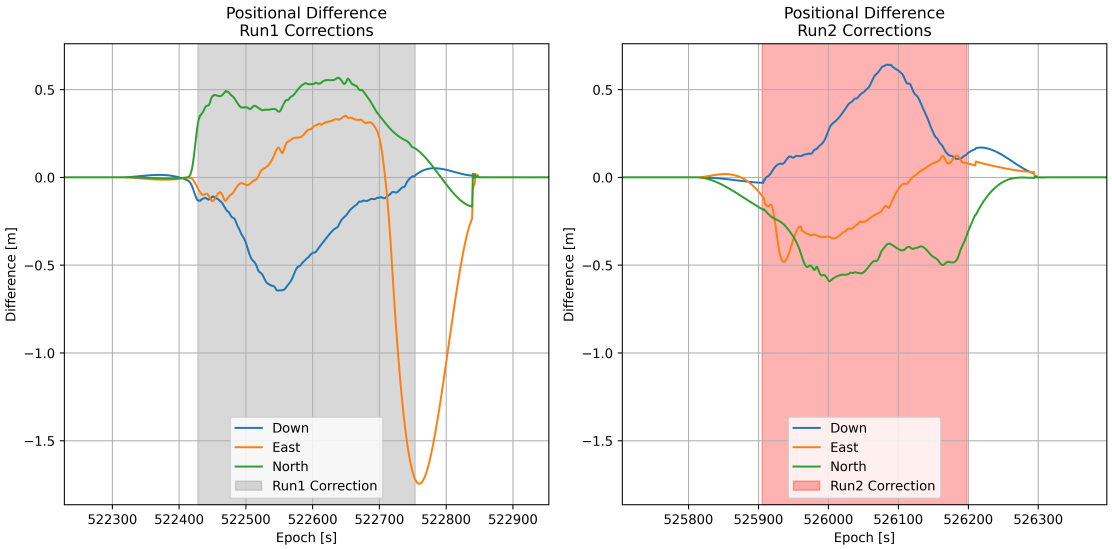
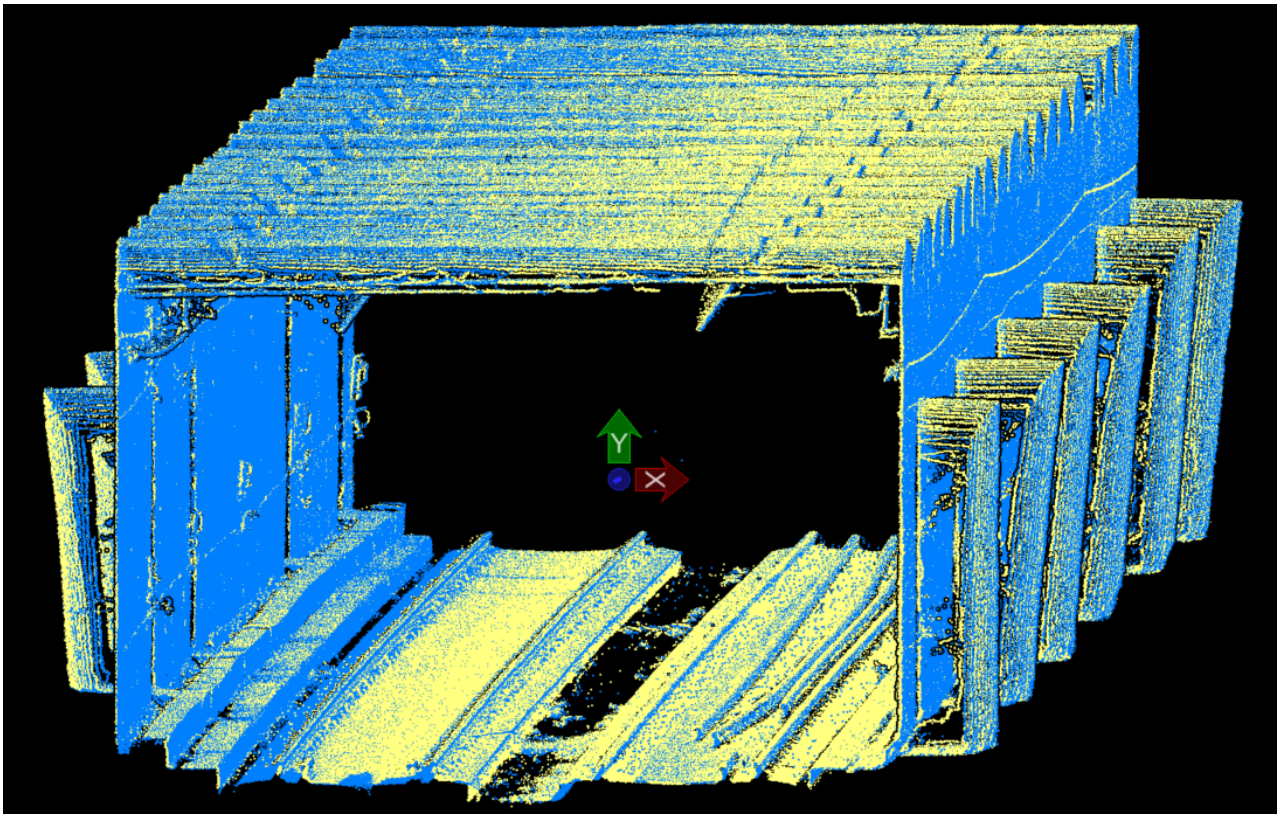
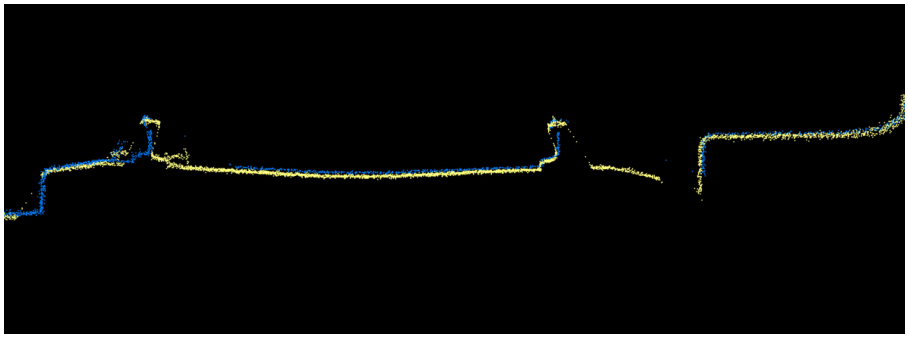


Figure 4.19: Trajectory position difference due to the added corrections.

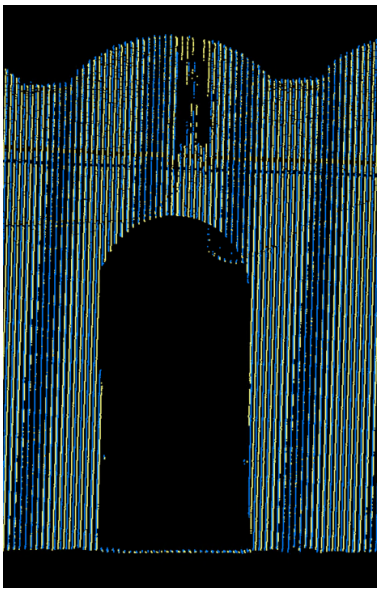
The straight tunnel segment from where the tile for the local trajectory optimization was used was re-georeferenced with the trajectory obtained from the global trajectory optimization algorithm. Figure 4.20 shows the re-georeferenced results, which are similar to the results of the local trajectory optimization algorithm - both driving and normal direction alignment are improved. C2C distance metric for the completely overlapping tunnel segment of 200 meters ($\approx$ 15 mil. points) has a mean of 0.022 m and a standard deviation of 0.026 m. This level of results indicates that present outliers did not impact the data, where the outliers are absent.

The point clouds collected within the epochs, which correspond to the abnormal East component correction were re-georeferenced as well. That data was found to correspond to the survey of a tunnel with a round geometry close to the entrance of the tunnel (shown in Figure 4.21). The re-georeferenced point clouds have improved alignment (cm level error) in the normal directions, however the alignment in the driving direction is of a meter level error. Additionally, driving direction misalignment disappears as the data is located further inside from the tunnel entrance and within 200-400 meters it aligns with a cm level error. This initial misalignment is assumed to be by the multipath error of the GNSS, since no signal occlusion was done next to the tunnel entrance. the GNSS data deteriorated due to the sudden change from the open sky to the indoor environment, which was still used as a reliable observation by the software. Additionally, the geometry of the tunnel could cause the poor performance of the algorithm due to the feature-poor environment being present for $\approx$ 200 - 400 meters.



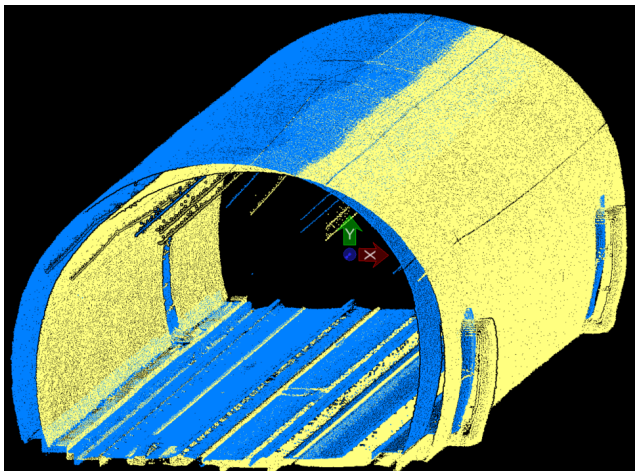


(a) Point cloud crossection highlighting the normal direction alignment (within a cm error).

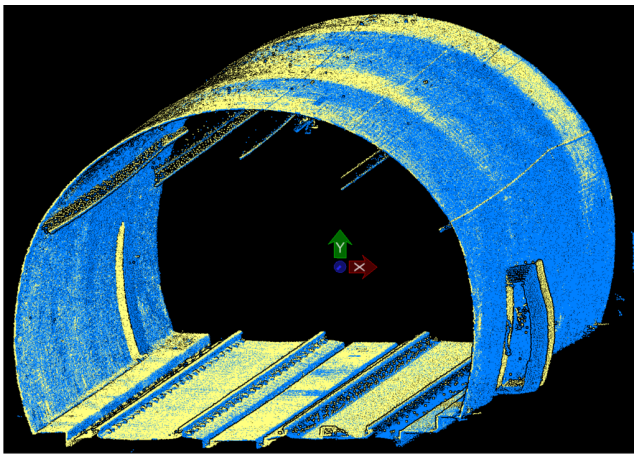


(b) Point cloud crossection highlighting a driving direction alignment.

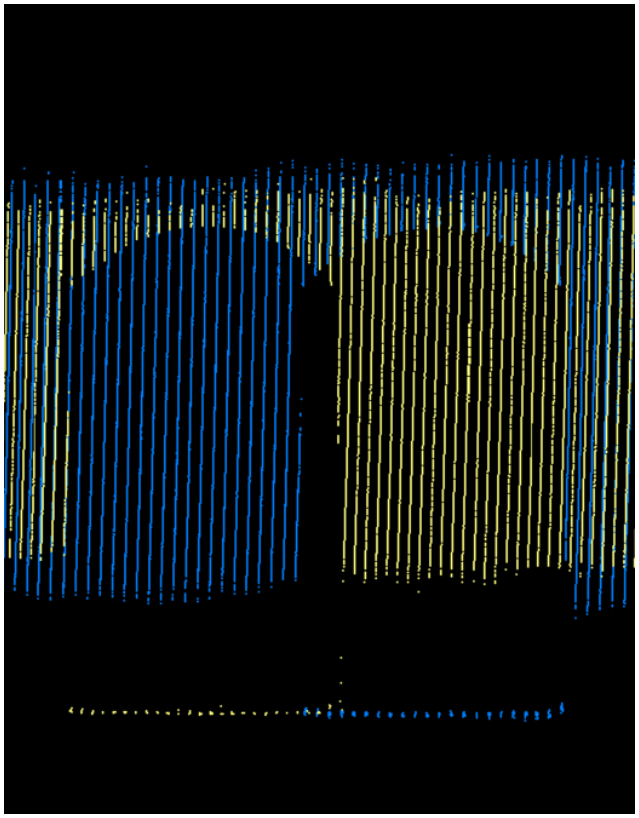
Figure 4.20: Re-georeferenced acquisitions of Run1 (yellow) and Run2 (blue) after the global trajectory optimization.



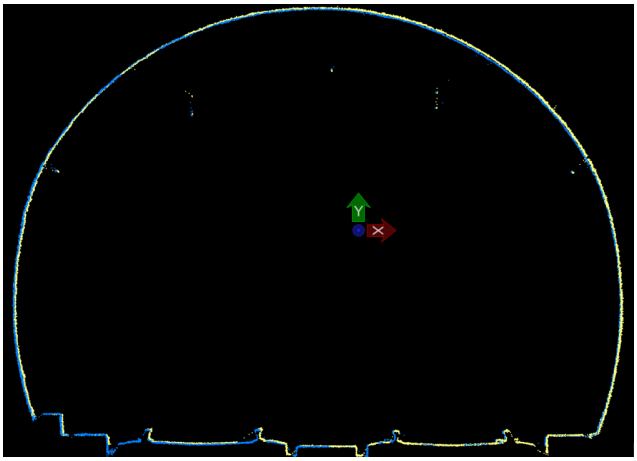
(a) Misalignment of the point clouds before the trajectory correction



(b) Improved alignment of the point clouds after the trajectory corrections.



(c) Significant (≈ 1 m) misalignment still present after trajectory correction in driving direction.



(d) Cm level misalignment after trajectory correction in driving direction.

Figure 4.21: Original and re-georeferenced acquisitions of the tunnel, which corresponds to the peak of the abnormal East component correction. Two acquisitions: Run1 (yellow) and Run2 (blue).

5 Discussion

The goal of this research was to develop an improved trajectory optimization method. The existing methodology in the data processing chain relies heavily on manual corrections, which, despite their high quality, are labor-intensive and time-consuming. Additionally, other autonomous methods, such as trajectory optimization with track distance corrections, face limitations. They cannot be applied in environments without parallel tracks. This research aimed to overcome these challenges. More precisely, it aimed to develop an autonomous and more generalized trajectory optimization solution, focusing primarily on rail-based environments, especially in GNSS-poor areas like underground tunnels and stations. The proposed methodology optimizes trajectories by utilizing point clouds of different acquisitions. It utilizes georeferenced point clouds to extract Intrinsic Shape Signature (ISS) features and match them. Within the methodology, these matched features are then used in a generalized graph framework (g^2o) to obtain optimized discrete poses. The resulting poses would be used to correct the trajectory in the GNSS processing software. Testing on datasets demonstrated the methodology's capability to significantly improve trajectory alignment and the re-georeferenced point clouds had an improvement in the alignment.

5.1 Feature Extraction

The utilization of handcrafted Intrinsic Shape Signature (ISS) features within the developed methodology underscored a significant achievement in trajectory optimization in GNSS-poor environments. Despite their simplicity compared to advanced deep learning detectors, ISS features proved to be effective: with the help of ISS features, the point cloud misalignment was refined through trajectory optimization and was decreased from a misalignment of 1.5 meters down to a 1 centimeter. This result was attained without necessitating any training phase, which would be required for learned detectors. Nevertheless, the application of ISS features was not without its challenges - optimal utilization of ISS detector required identification of the optimal parameter settings. Through an iterative process involving five distinct parameter configurations the optimal settings were found. The chosen configuration provided a balanced trade-off between the quantity and quality of detected features against the required computational demands. However, the variability within the parameter space suggests that a more optimized set of parameters might exist, potentially reducing both the number of features (retaining the feature correspondence) and the computational time required. Such optimization is crucial, since the global trajectory optimization results showed a significant increase of processing times for an increased amount of features. This scalability factor underscores the need for continuous refinement of parameter settings to enhance efficiency and applicability across larger areas.

Moreover, the ISS feature detector, initially validated in tunnel environments, is versatile and, therefore, can have other applications. Its reliance solely on geometric properties suggests potential utility across diverse environmental settings, which potentially broadens the scope of the developed methodology. Therefore, the developed methodology could be employed either independently or in conjunction with other methods, such as parallel track distance corrections, to enhance trajectory optimization further. However, the exploration into the generalizability of this feature detector and methodology across varied environments was beyond the scope of this research. The generalizability of the method was partially highlighted within the global trajectory optimization - some of the provided tiles contained the surveys done in outdoor environments. Moreover, the tiles used by the algorithm covered a variety of environments - overlapping tunnels of rectangular or round geometry, and partially overlapping stations. However, no deep analysis was done to assess whether the outdoor features contributed to the trajectory optimization or were used with high positional standard deviation and were considered outliers. Therefore, future studies should focus on the validation of this potential, bearing in mind that different environments may pose challenges that could impact the performance of the ISS detector and the chosen settings. For instance, more demanding conditions might result in fewer detected features and reduce feature correspondence, which potentially affects the performance of the trajectory optimizer.

Additionally, this research primarily concentrated on the ISS feature detectors, a core of the developed trajectory optimizer. However, further research could focus on the exploration of additional feature detectors. It could include both handcrafted detectors, known for their direct applicability and efficiency, and machine learning-based detectors, which, despite necessitating a more labor-intensive process of dataset preparation and potential model training, might provide improved performance.

5.2 Feature Matching

In the exploration of feature descriptors, the Fast Point Feature Histogram (FPFH) was initially considered due to its handcrafted nature, which, like the Intrinsic Shape Signature (ISS) feature detector, leverages surrounding geometry without requiring model training. Despite its possible potential, the practical application of FPFH

descriptors was not successful. Similarly to the ISS feature detector, various parameter configurations were tested. However, it was found that the computational demands of FPFH descriptor calculations are high. The calculations required over several minutes. The reason behind the long execution times was not identified within this research - this step was implemented by utilizing PCL library, which provides already optimized implementations of various feature descriptors calculators. As a possibility, the speed increase might be achieved by parallelizing the procedure but it was not tested within the scope of this research. From the tested parameters configurations, configuration #1 was chosen for its relatively shorter execution time.

Then, the process of matching detected ISS features utilized the FPFH descriptors to assess feature similarity. While the implementation of this matching step was straightforward and executed fast (approximately 12 seconds), the outcome of this matching procedure was insufficient. From the matching process, 81 pairs of matched features were identified. However, validation against known Ground Truth (GT) data showed the weakness of this approach: only half of the feature pairs were accurately matched within a 0.25cm proximity to the GT vector. Further analysis revealed an absence of correlation between the descriptor's similarity scores and the proximity of the retrieved matched features registration parameters from the GT. It underscores that the selected descriptor type or the parameters configuration is insufficient for the given tunnel environment.

In this research, the utilized parameters were already minimized to reduce computational load, yet they still resulted in extended processing times. Therefore, a proper configuration of the parameters, which produces adequate matching results, is expected to be also computationally expensive. Given these findings, the prospect of refining parameter selections for FPFH descriptors appears limited, not due to a lack of effort, but rather to the constraints imposed by computational practicality and the parameters' inherent limitations. Thus, as further research other feature descriptors, whether handcrafted or derived from machine learning techniques, can be considered for this optimizer. Such alternatives may offer the dual advantages of computational efficiency and improved matching accuracy. Additionally, other distance metrics or additional feature scaling steps can be implemented in the workflow to possibly enhance the matching with the feature descriptors.

As for the present research, the feature descriptors were simply omitted from the methodology. However, the trajectory optimization results indicated that bypassing the feature descriptor calculation phase did not negatively impact the outcome (not in terms of the trajectory quality), suggesting the possibility of achieving aimed results without this step. The modified version of the Random Sample Consensus (RANSAC) algorithm for outlier removal in this study employs a simplified yet effective implementation. This process, which performs both matching and outlier discarding, generates all possible pairs from detected ISS features. From each newly created feature pair, the registration parameters between point cloud acquisitions are estimated. By ranking these registration parameters according to the number of inliers produced, only the top 10 parameter sets are used later for the subsequent optimization step. Testing across synthetic and real data acquisition scenarios demonstrates the method's capability to find optimal matching pairs and registration parameters, and completes that within 45 seconds. Notably, the parameters derived from synthetic tiles showed cm level proximity to the Ground Truth (GT) translation vector, which affirms the outlier removal step's efficiency in this context. In the case of global trajectory optimization algorithm, the feature matching beared similar results - the median time for the feature matching was 17.71 seconds.

However, there is a need for broader empirical testing to confirm the technique's reliability in diverse scenarios. For example, the methodology's reliance on identifying the top 10 feature pairs, lacks a mechanism to explicitly safeguard against the inclusion of outliers within the selection. From the global trajectory results, it was shown, that next to capturing the present IMU drift in the pose corrections, an abundant amount of outliers were present in the data. Future improvements of this procedure could incorporate more stringent criteria for feature pair selection. Potentially, the focus on the lower amount of the selected features (e.g. top 5 feature pairs) could decrease the chances of outlier selection. However, the decreased amount of selected features could also potentially impact and deteriorate the present trajectory optimization performance but this is deemed to be unlikely due to the observed wide impact of local corrections on the trajectory.

Finally, the algorithmic foundation of the outlier removal process has a straightforward implementation and, therefore, is expected to have scalability challenges attributed to its computational intensity. For N detected features there will be $N * (N - 1)$ pairs, and for each feature pair, the amount of generated inliers and outliers is counted. The current implementation of the inliers and outliers counting is also $O(N^2)$ due to the simplicity of the algorithm and this simplified approach lacks optimization. Due to that, the total complexity of the outlier removal is expected to be $O(N^4)$, which for a doubled amount of the detected features will require approximately 8 times more computations. The results of the global trajectory optimization confirmed the poor scaling of the matching algorithm - the matching of ≈ 700 features took more than an hour, despite the median being just 17 seconds. Therefore, this inlier part should be optimized to reduce the overall running time of the developed methodology and to be able to scale better for the increased amount of features. As an additional solution, separate tile processing is independent and can be parallelized. This will not reduce the time required

to perform the matching step, however the total processing time of all tiles will be reduced to the the worst time required to process a tile within the dataset.

5.3 Trajectory Optimization

From the outlier removal step, the selected features, were then exported to the g^2o graph optimizer. Each feature, along with its assigned Feature ID, X, Y, and Z coordinates from the georeferenced point cloud, and the GPS Time of acquisition was exported for further processing. Additionally, each feature had an assigned weight, which was kept equal for every data point. This information was then loaded with the trajectory data as input to the optimizer. The pre-optimization steps involved interpolating IMU poses for each feature based on the input trajectory and GPS Time, creating states in the graph optimizer with fixed IMU attitudes (yaw, pitch, heading). Furthermore, the distance between points corresponding to the same ISS feature was used as an error to create constraints for the corresponding poses, forming the edges of the graph. Then, the optimization started and it converged in approximately 0.05 seconds over 8-9 iterations, with a convergence error of $1.78E-9$ meters, indicating successful alignment of feature pairs at their median location in between their pre-optimization locations. Similar results were obtained for the global trajectory optimization. Both results was expected since the weights of the provided features were kept the same in the matched feature pair. The uniform weighting of detected features was chosen due to the absence of additional data to evaluate the original feature's location's accuracy. This approach simplified the optimization problem and it did not leverage potential insights from the feature detection or matching process that could inform the weighting. For example, a more confident feature pair or the presence of a well-established trajectory can be used can be weighted more and, thus, can be used to correct the less accurate data.

Moreover, another reason for the perfect alignment of the optimization results with the expectations lies in the linearity of the posed problem. Adjusting discrete IMU poses with fixed attitudes and only altering its location to match the observations in the sensor is equivalent to finding the median point between the matched feature pair, which is trivial to achieve without the optimizer. Despite the current problem being too simple and incapable of leveraging the complete power of the non-linear optimizer, it creates a solid base for further research, where the current optimizer setup can be scaled and additional (both linear and non-linear) observations can be added as constraints. As an example, this study outsourced smooth trajectory modeling to GNSS processing software, but the optimizer could incorporate this directly by adding constraints between consecutive IMU poses. However, the exploration of various models for these constraints was beyond this study's scope but represents a significant opportunity for advancing the optimization framework and building upon the current results.

The output of the optimizer - optimized IMU poses - was used as anchor observation in the GNSS processing software PosPac. To utilize them as anchor observations, the optimized IMU poses were imported in a specific format. The format required to have the X and Y coordinates in the corresponding CRS (ETRS89 with 1989 as Datum), the Z coordinate referenced to the project's ellipsoid. Additionally, for each coordinate of the anchor point, the standard deviation was specified. In this trajectory re-processing, the positional standard deviations for the coordinates were set to be equal to the positional standard deviation of the 10 consecutive corrections. It was done in order to weigh the more consistent corrections more, since the outliers were assumed to be present in a more chaotic order. With this method employed, the final georeferencing results showed well-aligned point clouds. The misalignment next to the entrance of the tunnel is assumed to be caused by the GNSS multipath errors being present and not neglected by the Kalman filter in the software - the pose correction plot(Figure 4.16) does not indicate the complete presence of the outliers at the epochs corresponding to the abnormal East component correction (Figure 4.19). However, it can be attempted to improve the alignment quality by performing additional outlier filtering at this step. For example, by applying a median filter (with various kernel size) or other robust filtering methods.

5.4 Optimized Trajectory Quality Evaluation

After the coordinates and the standard deviations for the coordinates were determined, they were imported into the software, and the trajectory was re-processed. The resulting trajectory (both local and global) has shown a decrease in the RMS metrics for all components (North, East, and Down) in comparison with the previous processing result (Figure 4.18). It indicates more reliability in the obtained trajectory data, however it did not highlight data misalignment as strongly as the correction plot (Figure 4.19). Moreover, comparing the new trajectory with the old trajectory coordinates showed positional updates at the expected epochs - at the epochs where the IMU corrected poses were added. Additionally, the magnitudes and the direction of the positional updates mainly correspond relatively between Run1 and Run2 corrections: the magnitudes of the corrections are of the same order due to the equal weighting of the corrections and the directions are opposite, because the

continuous trajectory represents two opposite passes of the train over the same location and the trajectories were pulled towards each other.

Furthermore, the trajectory correction process, facilitated by Forward and Backward Kalman Filtering, ensures that adjustments are smoothly distributed across a broad range of epochs. The most substantial corrections typically occur near the epochs associated with the adjusted IMU poses. However, the asymmetry in corrections, misalignment of correction peaks with the epochs of corrected poses, differences between the corrections in Run1 and Run2, and occasional data jumps, are attributed to the characteristics of the Kalman Filtering algorithm and data acquisition route. This algorithm updates state predictions based on previous states, a predefined kinematic model, and new observational data, meaning the corrections are influenced by both the trajectory the train follows and the data (GNSS and IMU observations) collected during its journey. In this scenario, the train approaches the location of interest from two opposing directions, thus, the path leading up to each location and the magnitude of the resulting IMU drift differs. This difference in approach paths contributes to the variability observed in trajectory corrections. This explanation can be used to both explain small discrepancies between the Run1 and Run2 corrections but also - to explain the abnormal East component correction at Run1 and its smaller presence at Run2. Additionally, to assess the assumption that the abnormal correction was caused by the GNSS multipath error it is possible to compare the results to the re-processed trajectory with a manually occluded GNSS signal around the tunnel entrance epochs. Moreover, such a big anomaly was useful in finding data misalignment and these correction differences can be further researched about their use as data alignment quality indicators.

Finally, a comprehensive assessment of the global trajectory results is essential. In the research, only several data locations were manually assessed, and selected based on anomalies detected in trajectory-related plots (RMS and trajectory corrections data). These plots served to approximate potential problem areas; however, it was not confirmed nor tested that these plots would consistently reflect problematic areas in the data.

6 Conclusion & Recommendations

In conclusion, the goal of this research was to develop an improved trajectory optimization method. This was accomplished by utilizing the ISS feature detector in combination with the developed outlier removal algorithm to detect, extract and match features, which were later used to obtain corrected IMU poses by utilizing a g^2o graph optimizer. With this developed methodology, the corrected IMU poses were used to enhance trajectory processing. After the methodology was developed, the complete chain was then applied to the selected point cloud tile, and, as a result, newly georeferenced point clouds exhibited significantly improved alignment. The reduction in misalignment for the selected point cloud tile from around 1.5 meters to 1 centimeter signifies the research's success for the applied data, directly meeting its primary aim. Furthermore, throughout this study, various research questions were addressed and the findings were summarized in the list below:

1. **How is the quality of trajectory solutions evaluated?**:

   From the literature review, it was found that the quality of trajectory solutions can be evaluated through comparison with a trajectory, where the absence of GNSS signal is simulated (Jing et al., 2021; K. Wang et al., 2021). However, this approach was not employed in this research due to the focus on the tunnel environment. Another trajectory solution evaluation method was found to be to consider the georeferenced point clouds and how they align with each other. More precisely, distances between set-up targets, features or points in the point clouds will quantify the misalignment of the point clouds locally (Jing et al., 2021; Vosselman & Maas, 2014). It was employed and correlated with the visual assessment of the alignment of the point clouds. Moreover, RMS plots and the trajectory correction plots from GNSS processing software were found to work well as a proxy to find the location of the data misalignment. However, more research is needed to reliably confirm that.

2. **What methods exist to optimize trajectory and how well do they suit rail-based tunnel environment?**

   No methods were found to be directly employed by this research. The desired method needed to function automatically within a rail-based tunnel environment and accommodate the RILA's expected cruising speed (up to 100 km/h and faster). No reviewed paper entirely matched these criteria, although, some studies offered partial solutions. For example, the FEPPA method optimized trajectory using a Particle Filter with poles extracted from point clouds, showing improvement Jing et al., 2021. However, the features are were absent in tunnel environments of the Glasgow Station dataset and the proposed method does not meet the automation criteria in feature extraction. However, this method was used to develop a general workflow of utilizing point cloud features in trajectory optimization. It was also present in other studies, despite being conducted in different settings or using distinct platforms. Trajectory optimization in underground mines, forests or on high-speed roads utilizes point cloud data in a similar SLAM-based solution (Liu et al., 2022; Qian et al., 2016; Shen et al., 2024). Based on this, a new adjusted methodology was developed in this research - by utilizing ISS features of the environment in the g^2o optimizer and the GNSS processing software to optimize the trajectory.

3. **What methods exist to identify and extract point cloud features in the tunnels?**

   In the literature, hand-crafted and deep learning methods exist for identifying features in point clouds. Methods such as ISS, Harris3D, and NARF use geometrical variations to identify salient points without requiring any training (Sipiran & Bustos, 2011; Steder et al., 2010; Zhong, 2009). Moreover, more advanced deep learning methods exist, like D3Feat, which tend to outperform hand-crafted methods (Bai et al., 2020). Although, these models must be trained and adjusted before use. Despite the methods of both types being mentioned in the literature, specifically for point cloud feature extraction in the tunnels, no benchmarks comparing these methods were found. Therefore, this study was the first one to utilize ISS features in such an environment. ISS features are extracted using the ISS feature detector implemented in the Open3D point cloud processing library. This method leverages the geometric properties of point clouds to identify distinctive features. The process involves analyzing the local neighborhood of each point in the cloud to compute a signature based on its shape, which aids in identifying features robust to environmental changes and noise.

4. **What is the process for integrating the extracted features into a trajectory optimization workflow?**

   The features are extracted from the point cloud tiles. Then, the extracted features can be used as anchor observations in the trajectory processing chain by the GNSS processing software. For that, the extracted features are matched by undergoing a RANSAC-inspired approach. This matching and outlier removal

procedure considers all possible pairs of features, derives possible registration parameters for the point cloud alignment and ranks them based on the amount of inliers these registration parameters generate. From the obtained ranking, top 10 features are selected and then optimized with the g^2o optimizer, resulting in the optimized feature locations and the optimized corresponding IMU poses. Finally, the optimized IMU poses are then used as anchor observations in the GNSS processing software to re-process the trajectory.

5. **What improvements do these solutions offer over current methods and how generalizable the developed method is?**

   The aforementioned steps were implemented in the methodology and a re-processed trajectory was obtained successfully. The re-georeferenced point clouds with the obtained trajectory from the global trajectory optimization results showed a decrease of misalignment from $\approx$ 1.5m - to cm level error within the proximity of the corrections, which covered complex and variable environments (overlapping and non-overlapping tunnels and stations, partially outdoors). Moreover, this process was done automatically. Therefore, the improvements provided by this method that it can be employed automatically, it can resolve the misalignment in both directions and it can more feature than just parallel track distance correction. Additionally, the method shows potential to be utilized in the variable environments, however a deeper and independent assessment of each environment should be employed to confirm this reliably. For example, it was found that the method failed to perform next to the entrance of the tunnel.

6.1 Recommendations

Reflecting on the answers provided to the research questions, it becomes evident that while substantial progress has been made, opportunities for further refinement and exploration remain. Therefore, further exploration is recommended to address existing limitations and expand the scope of its applicability. First, the investigation on the robustness of the method and generalizability to reliably confirm the detected problems with the "entrance-of-the-tunnel" environment. Another improvement can be made at the outlier removal step, which is identified as a potential bottleneck due to its current scalability issues. Therefore, it requires particular attention for optimization - for example, the algorithm can be parallelized. Optimizing the outlier removal step will ensure the methodology's efficiency and effectiveness in processing a larger amount of detectable features. Moreover, various weighting schemes can be explored in the steps of graph optimization and anchor observation utilization. Adding properly calculated weight might reduce the possible impact of outliers and improve the results of trajectory optimization. Moreover, in this research, only two runs were optimized and the actual survey will consist of several runs. Assessment on how the method performs with the several runs should be made. Finally, another feature detector can be explored and an improvement can be made by integrating advancements in feature detection. For example, exploring machine learning approaches for feature extraction could provide valuable improvements, however it will require additional efforts in data preparation, pre-processing and model training.

References

Al-Bayari, O. (2019). Mobile mapping systems in civil engineering projects (case studies). *Applied Geomatics*, *11*(1), 1–13. https://doi.org/10.1007/s12518-018-0222-6

Ao, S., Hu, Q., Yang, B., Markham, A., & Guo, Y. (2021). SpinNet: Learning a General Surface Descriptor for 3D Point Cloud Registration.

Bai, X., Luo, Z., Zhou, L., Fu, H., Quan, L., & Tai, C.-L. (2020). D3Feat: Joint Learning of Dense Detection and Description of 3D Local Features.

Besl, P., & McKay, N. D. (1992). A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *14*(2), 239–256. https://doi.org/10.1109/34.121791

Chao, P., Xu, Y., Hua, W., & Zhou, X. (2019). A survey on map-matching algorithms.

Chapman, M. A., Min, C., & Zhang, D. (2016). Continuous Mapping of Tunnel Walls in a GNSS-Denied Environment. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *XLI-B3*, 481–485. https://doi.org/10.5194/isprsarchives-XLI-B3-481-2016

Choy, C., Park, J., & Koltun, V. (2019). Fully Convolutional Geometric Features. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 8957–8965. https://doi.org/10.1109/ICCV.2019.00905

Durrant-Whyte, H., & Bailey, T. (2006). Simultaneous localization and mapping: Part I. *IEEE Robotics & Automation Magazine*, *13*(2), 99–110. https://doi.org/10.1109/MRA.2006.1638022

Elhashash, M., Albanwan, H., & Qin, R. (2022). A Review of Mobile Mapping Systems: From Sensors to Applications. *Sensors*, *22*(11), 4262. https://doi.org/10.3390/s22114262

El-Sheimy, N. (2005). An Overview of Mobile Mapping Systems.

Fei, B., Yang, W., Chen, W., Li, Z., Li, Y., Ma, T., Hu, X., & Ma, L. (2022). Comprehensive Review of Deep Learning-Based 3D Point Cloud Completion Processing and Analysis. *IEEE Transactions on Intelligent Transportation Systems*, *23*(12), 22862–22883. https://doi.org/10.1109/TITS.2022.3195555

Fischler, M. A., & Bolles, R. C. (1981). Random sample consensus. *24*(6).

Gressin, A., Cannelle, B., Mallet, C., & Papelard, J.-P. (2012). Trajectory-Based Registration of 3D LiDAR Point Clouds Acquired with a Mobile Mapping System. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *I-3*, 117–122. https://doi.org/10.5194/isprsannals-I-3-117-2012

Grewal, M. S., Weill, L. R., & Andrews, A. P. (2001). *Global positioning systems, inertial navigation, and integration*. John Wiley.

Grisetti, G., Kummerle, R., Stachniss, C., & Burgard, W. (2010). A Tutorial on Graph-Based SLAM. *IEEE Intelligent Transportation Systems Magazine*, *2*(4), 31–43. https://doi.org/10.1109/MITS.2010.939925

Guo, Y., Bennamoun, M., Sohel, F., Lu, M., Wan, J., & Kwok, N. M. (2016). A Comprehensive Performance Evaluation of 3D Local Feature Descriptors. *International Journal of Computer Vision*, *116*(1), 66–89. https://doi.org/10.1007/s11263-015-0824-y

Guo, Y., Bennamoun, M., Sohel, F. A., Wan, J., & Lu, M. (2013). 3D free form object recognition using rotational projection statistics. *2013 IEEE Workshop on Applications of Computer Vision (WACV)*, 1–8. https://doi.org/10.1109/WACV.2013.6474992

Harris, C., & Stephens, M. (1988). A Combined Corner and Edge Detector. *Procedings of the Alvey Vision Conference 1988*, 23.1–23.6. https://doi.org/10.5244/C.2.23

International Energy Agency. (2019). *The Future of Rail: Opportunities for energy and the environment*. OECD. https://doi.org/10.1787/9789264312821-en

Jiang, W., Chen, S., Cai, B., Wang, J., ShangGuan, W., & Rizos, C. (2018). A Multi-Sensor Positioning Method-Based Train Localization System for Low Density Line. *IEEE Transactions on Vehicular Technology*, *67*(11), 10425–10437. https://doi.org/10.1109/TVT.2018.2869157

Jing, H., Meng, X., Slatcher, N., & Hunter, G. (2021). Efficient point cloud corrections for mobile monitoring applications using road/rail-side infrastructure. *Survey Review*, *53*(378), 235–251. https://doi.org/10.1080/00396265.2020.1719753

Johnson, A., & Hebert, M. (1999). Using spin images for efficient object recognition in cluttered 3D scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *21*(5), 433–449. https://doi.org/10.1109/34.765655

Kaplan, E. D., & Hegarty, C. J. (Eds.). (2006). *Understanding GPS: Principles and applications* (2nd. ed). Artech House.

Kim, K., Seol, S., & Kong, S.-H. (2015). High-speed Train Navigation System based on Multi-sensor Data Fusion and Map Matching Algorithm.

Knapcikova, L., & Konings, R. (2018). European Railway Infrastructure: A Review. *Acta logistica*, *5*(3), 71–77. https://doi.org/10.22306/al.v5i3.97

Kremer, J., & Grimm, A. (2012). The *RailMapper* – A Dedicated Mobile LiDAR Mapping System for Railway Networks. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *XXXIX-B5*, 477–482. https://doi.org/10.5194/isprsarchives-XXXIX-B5-477-2012

Kummerle, R., Grisetti, G., Strasdat, H., Konolige, K., & Burgard, W. (2011). G^2o: A general framework for graph optimization. *2011 IEEE International Conference on Robotics and Automation*, 3607–3613. https://doi.org/10.1109/ICRA.2011.5979949

Leslar, M., Perry, G., & McNease, K. (2010). Using Mobile LiDAR to Survey a Railway Line for Asset Inventory. *San Diego*.

Liu, H., Li, Y., Wang, B., & Guo, Z. (2022). Research on Mapping Error Control of Underground Space Mobile LiDAR Constrained by Cooperative Targets (Y. Li, Ed.). *Journal of Sensors*, *2022*, 1–12. https://doi.org/10.1155/2022/8690532

Lowe, D. (1999). Object recognition from local scale-invariant features. *Proceedings of the Seventh IEEE International Conference on Computer Vision*, 1150–1157 vol.2. https://doi.org/10.1109/ICCV.1999.790410

Menna, F., Battisti, R., Nocerino, E., & Remondino, F. (2023). FROG: A Portable Underwater Mobile Mapping System. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *XLVIII-1/W1-2023*, 295–302. https://doi.org/10.5194/isprs-archives-XLVIII-1-W1-2023-295-2023

Monji-Azad, S., Hesser, J., & Löw, N. (2023). A review of non-rigid transformations and learning-based 3D point cloud registration methods. *ISPRS Journal of Photogrammetry and Remote Sensing*, *196*, 58–72. https://doi.org/10.1016/j.isprsjprs.2022.12.023

Puente, I., González-Jorge, H., Martínez-Sánchez, J., & Arias, P. (2013). Review of mobile mapping and surveying technologies. *Measurement*, *46*(7), 2127–2145. https://doi.org/10.1016/j.measurement.2013.03.006

Qian, C., Liu, H., Tang, J., Chen, Y., Kaartinen, H., Kukko, A., Zhu, L., Liang, X., Chen, L., & Hyyppä, J. (2016). An Integrated GNSS/INS/LiDAR-SLAM Positioning Method for Highly Accurate Forest Stem Mapping. *Remote Sensing*, *9*(1), 3. https://doi.org/10.3390/rs9010003

Raval, S., Banerjee, B. P., Kumar Singh, S., & Canbulat, I. (2019). A Preliminary Investigation of Mobile Mapping Technology for Underground Mining. *IGARSS 2019 - 2019 IEEE International Geoscience and Remote Sensing Symposium*, 6071–6074. https://doi.org/10.1109/IGARSS.2019.8898518

Rusu, R. B., Blodow, N., & Beetz, M. (2009). Fast Point Feature Histograms (FPFH) for 3D registration. *2009 IEEE International Conference on Robotics and Automation*, 3212–3217. https://doi.org/10.1109/ROBOT.2009.5152473

Saab, S. S. (2000). A map matching approach for train positioning. II. Application and experimentation. *IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY*, *49*(2).

Saab, S. (2000). A map matching approach for train positioning. I. Development and analysis. *IEEE Transactions on Vehicular Technology*, *49*(2), 467–475. https://doi.org/10.1109/25.832978

Salti, S., Tombari, F., & Di Stefano, L. (2014). SHOT: Unique signatures of histograms for surface and texture description. *Computer Vision and Image Understanding*, *125*, 251–264. https://doi.org/10.1016/j.cviu.2014.04.011

Secco, P. (2022). *Point cloud based improvement of the trajectory of a Railway Mobile Mapping System* [MSc Thesis]. TU Delft.

Shen, Z., Wang, J., Pang, C., Lan, Z., & Fang, Z. (2024). A LiDAR-IMU-GNSS fused mapping method for large-scale and high-speed scenarios. *Measurement*, *225*, 113961. https://doi.org/10.1016/j.measurement.2023.113961

Sipiran, I., & Bustos, B. (2011). Harris 3D: A robust extension of the Harris operator for interest point detection on 3D meshes. *The Visual Computer*, *27*(11), 963–976. https://doi.org/10.1007/s00371-011-0610-y

Steder, B., Rusu, R. B., Konolige, K., & Burgard, W. (2010). NARF: 3D Range Image Features for Object Recognition.

Sun, H., Xu, Z., Yao, L., Zhong, R., Du, L., & Wu, H. (2020). Tunnel Monitoring and Measuring System Using Mobile Laser Scanning: Design and Deployment. *Remote Sensing*, *12*(4), 730. https://doi.org/10.3390/rs12040730

Teunissen, P. J., & Montenbruck, O. (Eds.). (2017). *Springer Handbook of Global Navigation Satellite Systems*. Springer International Publishing. https://doi.org/10.1007/978-3-319-42928-1

Tombari, F., Salti, S., & Di Stefano, L. (2013). Performance Evaluation of 3D Keypoint Detectors. *International Journal of Computer Vision*, *102*(1-3), 198–220. https://doi.org/10.1007/s11263-012-0545-4

Vosselman, G., & Maas, H. (2014). *Airborne and Terrestrial Laser Scanning*. Whittles Publishing. OCLC: 1136417519.

Wang, H., & Berkers, J. (2019). Absolute and Relative Track Geometry: Closing the Gap.

Wang, H., Berkers, J., Van Den Hurk, N., & Layegh, N. F. (2021). Study of loaded versus unloaded measurements in railway track inspection. *Measurement*, *169*, 108556. https://doi.org/10.1016/j.measurement.2020.108556

Wang, K., Cao, C., Ma, S., & Ren, F. (2021). An Optimization-Based Multi-Sensor Fusion Approach Towards Global Drift-Free Motion Estimation. *IEEE Sensors Journal*, *21*(10), 12228–12235. https://doi.org/10.1109/JSEN.2021.3064446

Xia, Y., Wu, H., Zhu, L., Qi, W., Zhang, S., & Zhu, J. (2024). A multi-sensor fusion framework with tight coupling for precise positioning and optimization. *Signal Processing*, *217*, 109343. https://doi.org/10.1016/j.sigpro.2023.109343

Yang, J., Quan, S., Wang, P., & Zhang, Y. (2020). Evaluating Local Geometric Feature Representations for 3D Rigid Data Matching. *IEEE Transactions on Image Processing*, *29*, 2522–2535. https://doi.org/10.1109/TIP.2019.2959236

Yoshida, K., & Tadokoro, S. (Eds.). (2014). *Field and Service Robotics: Results of the 8th International Conference* (Vol. 92). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-40686-7

Zhong, Y. (2009). Intrinsic shape signatures: A shape descriptor for 3D object recognition. *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*, 689–696. https://doi.org/10.1109/ICCVW.2009.5457637

Zhuang, Y., Sun, X., Li, Y., Huai, J., Hua, L., Yang, X., Cao, X., Zhang, P., Cao, Y., Qi, L., Yang, J., El-Bendary, N., El-Sheimy, N., Thompson, J., & Chen, R. (2023). Multi-sensor integrated navigation/positioning systems using data fusion: From analytics-based to learning-based approaches. *Information Fusion*, *95*, 62–90. https://doi.org/10.1016/j.inffus.2023.01.025