USDOT Tier 1
University Transportation Center
on Improving Rail Transportation
Infrastructure Sustainability and Durability

Final Report UNLV-5

DEVELOPMENT OF MULTI-ROTOR-UAV-BASED RAIL TRACK IRREGULARITY MONITORING AND MEASURING PLATFORM WITH IMAGE AND LIDAR SENSORS

By

Lihao Qiu, Graduate Student, Ming Zhu, Ph.D., Lab Director, Yingtao Jiang, Ph.D., Professor
Department of Electrical and Computer Engineering, University of Nevada, Las Vegas

Jee Woong Park, Ph.D., Professor
Civil and Environmental Engineering and Construction, University of Nevada, Las Vegas

Han Li, Ph.D., Professor
College of Mathematics, Wenzhou University, Wenzhou, Zhejiang, China

Tianding Chen, Ph.D., Professor
Minnan Normal University, Zhangzhou, Fujian, China

Haijian Shao, Ph.D., Associate Professor
Jiangsu University of Science and Technology, Zhenjiang, Jiangsu, China

and

Hualiang (Harry) Teng, Ph.D., Professor
Civil and Environmental Engineering and Construction, University of Nevada, Las Vegas

September 2024

Grant Number: 69A3551747132

DISCLAIMER

The contents of this report reflect the views of the authors, who are responsible for the facts and the accuracy of the information presented herein. This document is disseminated in the interest of information exchange. The report is funded, partially or entirely, by a grant from the U.S. Department of Transportation's University Transportation Centers Program. However, the U.S. Government assumes no liability for the contents or use thereof.

CONTENTS

TABLE OF FIGURES

LIST OF TABLES

EXECUTIVE SUMMARY

The field of track geometry measurement has evolved from manual and visual methods to sophisticated Track Geometry Measurement Systems (TGMS), which use advanced digital instrumentation to record various parameters. Despite their ability to measure long distances with minimal human resources, TGMS still face a critical limitation: the need to close tracks during inspection.

This project aimed to develop a multi-rotor UAV-based track geometry measurement system using image and LiDAR sensors that does not require the closure of track during inspection. Key challenges, including UAV path planning, data collection, and data processing, were addressed. The research was divided into three stages: exploration of optimal path planning, development of a LiDAR-only track geometry measurement system, and development of a camera-LiDAR track geometry measurement system.

In the first stage, the authors identified reinforcement learning as a solution for optimal UAV path planning and proposed a modified Floyd-Warshall (FW) algorithm. Experiments demonstrated that the proposed algorithm effectively handles the exploration-exploitation tradeoff, showing that exploration and exploitation converge in the decision-making process. As a result, the graph-based algorithm finds the shortest path during exploration, leading to higher efficiency and faster convergence compared to the Q-learning algorithm and its variants. A key advantage of the proposed algorithm is its applicability in dynamic environments, such as UAV-based track geometry measurement, where value-oriented algorithms typically fail. However, this improved efficiency and convergence come at the cost of increased computational complexity.

In the second stage, the authors developed a UAV-LiDAR-based platform capable of performing track geometry measurements alongside normal rail operations. This system, built on a UAV equipped with a LiDAR sensor, utilizes machine learning for rail point segmentation, LiDAR SLAM for expanding the point cloud's field of view, and regression techniques for outlier removal and geometry calculations. Compared to traditional field measurements using specialized tools, the platform demonstrated high accuracy in gauge measurement, though it performed less effectively in curvature and profile measurements.

In the final stage, the authors explored integrating camera data with LiDAR to enhance track geometry measurement. Due to time constraints, a semi-assisted supervised image segmentation approach was used, yielding high accuracy when the rails were vertically aligned in the images. Calibration results were highly accurate with checkerboard data but showed significant errors when applied to rail data. As a result, the current data fusion approach is not yet suitable for the track geometry measurement platform. Future research will aim to achieve more comprehensive image segmentation and improve the accuracy of LiDAR-camera calibration.

Keywords: Lidar, Unmanned aerial vehicle, railroad geometry, image processing, vehicle routing

1 TRACK GEOMETRY BACKGROUND

1.1 Overview

Rail track geometry defines the properties and relations of points, lines, curves, and surfaces in the three-dimensional positioning of railroad track, and popular track geometries are tangent, horizontal and vertical curves, transition curves, superelevation, and gradient. Track irregularities are deviations from the original geometry of a railroad track, and the typical track irregularities are gauge, alignment, profile, cross level, twist. These irregularities are the second most frequent cause of derailments on Class I main line tracks in the United States (Liu, 2012). Accurate and timely measurement of track geometry is crucial for rail safety, passenger comfort, and preventing damage to rolling stock and cargo.

The field of track geometry measurement has evolved from manual and visual methods to sophisticated Track Geometry Measurement Systems (TGMS), which use advanced digital instrumentation to record various parameters.

1.2 The Basics of Track Geometry

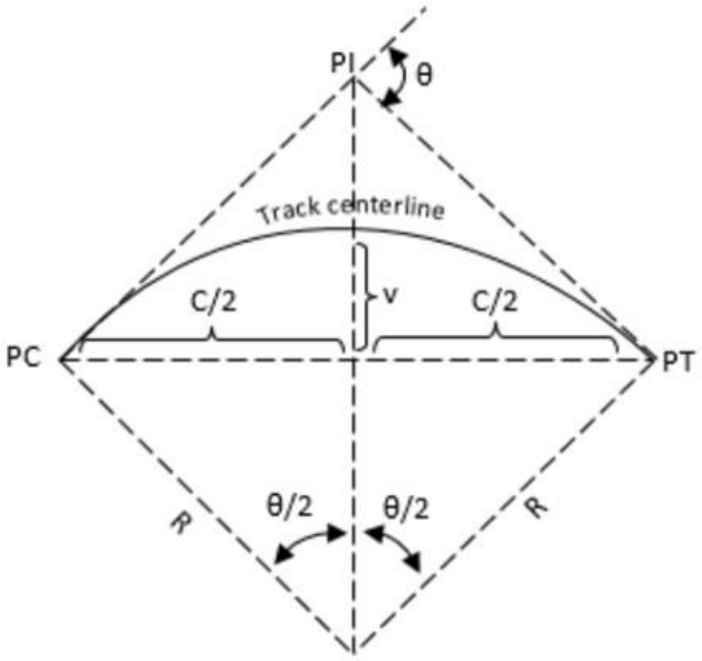

Figure 1 A simple curve

Track curvature can be expressed by the following four parameters:

a. Radius of curvature (R)
b. Inverse of radius of curvature (1/R).
c. Degree of curvature, the magnitude of the arc angle subtended by a chord of a specific length (typically 100 feet or 20 meters).

The relationship between radius of curvature, chord length, and degree of curvature is given by the formula:

$$R \sin\left(\frac{\theta}{2}\right) = \frac{C}{2} \tag{1}$$

where R is the curve radius, θ is the arc angle in radians, and C is the chord length.

d. Mid-chord offset (MCO) or versine: This is measured at the midpoint of a chord with a specified length:

$$v = R - \sqrt{R^2 - \frac{c^2}{4}} = R(1 - \cos(\frac{\theta}{2})) \tag{2}$$

where v is versine, R is curve radius, and C is chord length (all length dimensions in feet or meter; θ in radius).

Applying the small angle approximation and combining Equations (1) and (3) yields the following relationship (Ciobanu, 2016):

$$v \cong \frac{c^2}{8R} \cong \frac{c\theta}{8} \tag{3}$$

When using a 62-foot chord, the mid-chord offset (MCO) per degree of curvature is approximately 1 inch. This means a one-degree curve will have an MCO of about 1 inch, a two-degree curve will have an MCO of about 2 inches, and so on for higher degrees of curvature. (Figure 2)

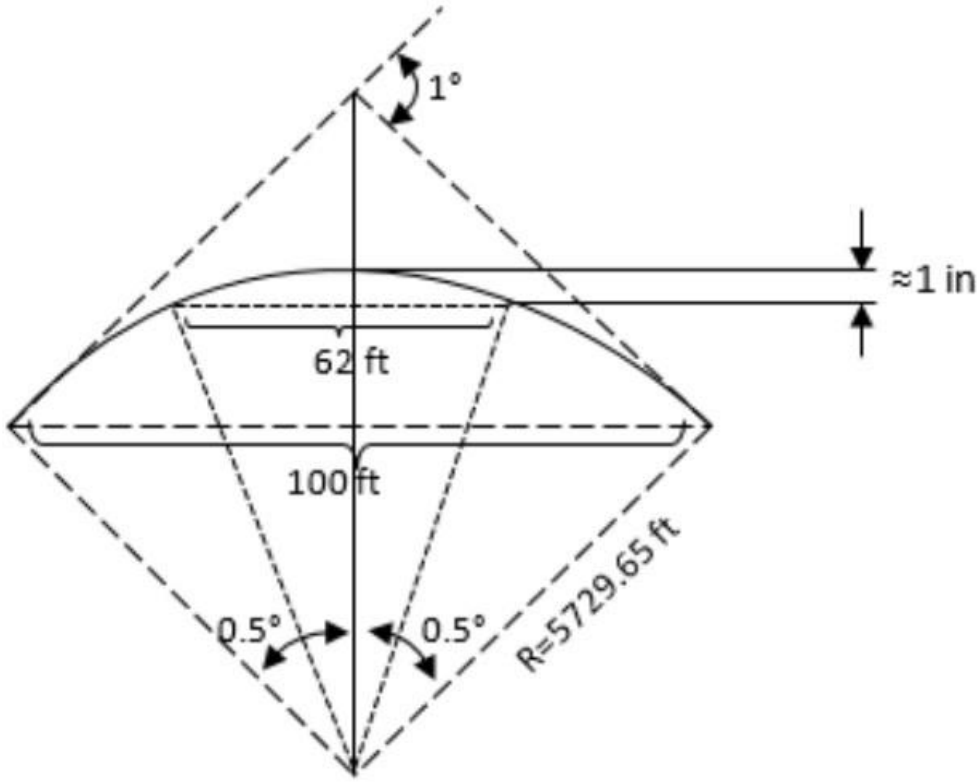

Figure 2 MCO measurement with a 62-foot chord

Here's a paraphrased version without itemization:

Transition curves, also called easement curves or spirals, are track sections connecting tangents and simple curves, as illustrated in Figure 3. These curves serve to prevent abrupt changes in superelevation and centripetal acceleration when vehicles transition between straight sections and curves, or between curves of different radii. However, transition curves might be omitted on low-speed tracks, in cases of shallow curves, in turnouts, on older rail lines, and where space is limited due to terrain features or structures along the track route.

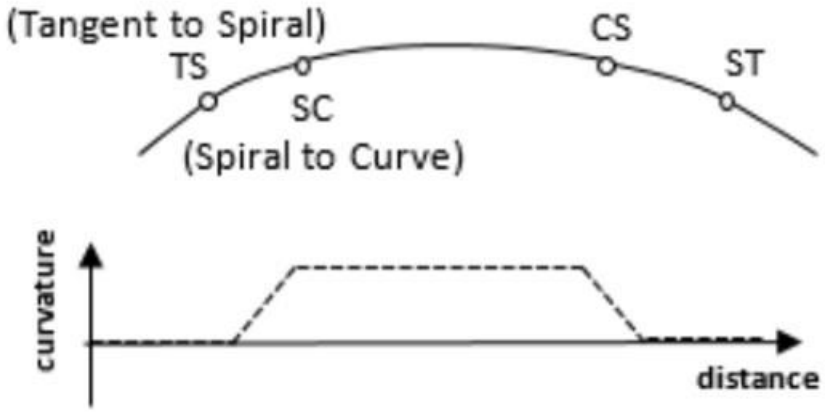

Figure 3 Curves without (left) and with (right) transition curves

Multicentered curves, depicted in
Figure *4*, consist of adjacent simple curves with either no tangent section or a short tangent section between them. These curves may or may not include transition curves. Multicentered curves encompass several special cases (Hay, 1982): compound curves (i.e., adjacent curves with the same curvature direction), reverse curves (i.e., adjacent curves with opposite curvature directions), and broken back curves (i.e., curves in the same direction separated by a short tangent).

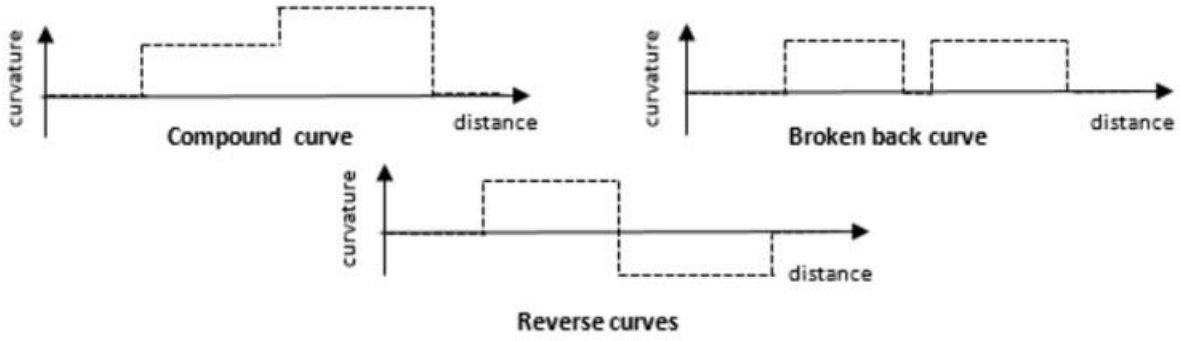

Figure 4 Multicentered curves.

Vertical track geometry is composed of constant grade sections and vertical curves, typically designed as quadratic parabolas. Grade represents the change in elevation (i.e., rise) relative to horizontal distance (i.e., run), often approximated as rise over distance traveled, and is usually expressed as a percentage.

Track gauge, illustrated in Figure 5, is the distance between the inner sides of rail heads, measured at a specific height below the top of the rail (5/8-inch in North America, 14 mm in Europe). The

standard gauge, used in most countries, is 56 ½ inches, or 1435 mm. In sharp curves, the gauge may be intentionally widened slightly from the nominal value to enhance vehicle steering.

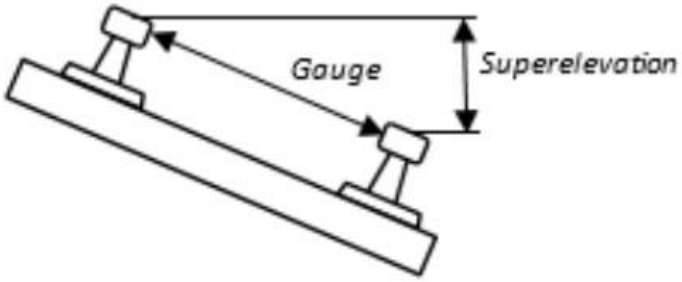

Figure 5 Track gauge and superelevation.

Superelevation refers to the intentional elevation of the outer (i.e., high) rail in a curve above the inner (i.e., lower) rail. This design feature compensates for the centrifugal acceleration experienced by vehicles as they navigate the curve. In a curve, the balance speed is the velocity at which the lateral components of centrifugal acceleration and gravitational acceleration neutralize each other. At this speed, a vehicle experiences no net lateral force, resulting in equal wheel loads on both left and right wheels, as depicted in Figure 6.

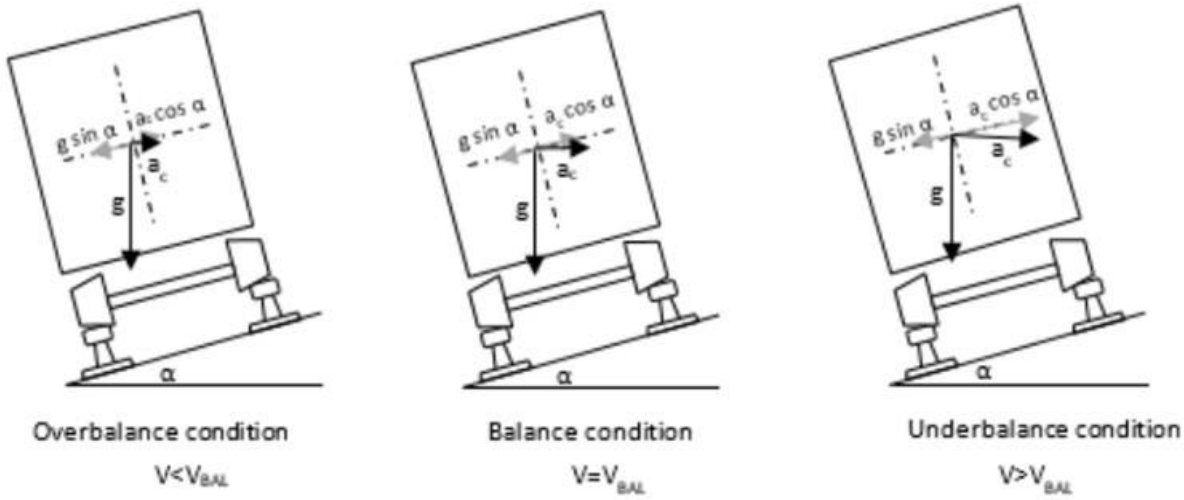

Figure 6 Balance conditions.

The formula for calculating balance speed in imperial units is:

$$V_{bal} = \sqrt{\frac{E_a}{0.00069\theta}} \tag{4}$$

where V_{bal} is balance speed in miles per hour, E_a is track superelevation in inches, and θ is track curvature in degrees.

1.2.1 Track Irregularities

Over time, track geometry deteriorates, with the track position deviating from its design geometry. These deviations are referred to as irregularities or excursions. Their cause includes (Puzavac et al., 2012; Haigermoser et al., 2015; Zarembski et al., 2015; Muinde, 2018):

- Track component manufacturing tolerances, including rail rolling defects
- Errors in initial surveying, construction, and track realignment measurements
- Degradation of crossties and fasteners
- Substandard weld geometry
- Issues with rail surface geometry
- Ground settlement
- Alterations in ballast density and stiffness, resulting from settlement, washout, scattering and tamping operations
- Vehicle-track interaction at both small scale (e.g., rail corrugation, rail squats, etc.) and large scale (e.g., uneven ballast and soil settlement under vehicle loads) with a notable positive feedback between increasing wheel-rail forces and degradation of track geometry degradation (Elkhoury et al., 2018)
- Lateral track displacement due to high lateral wheel forces and inadequate lateral track stiffness
- Track bucking caused by thermal loads and insufficient restraint

1.2.2 Absolute vs. Relative Track Geometry

Track geometry measurements are typically classified into two main categories: absolute and relative geometry. Absolute geometry, also known as outer geometry, refers to the position of the track in relation to an absolute reference point. This measurement is crucial for maintaining clearances around the track and establishing its location relative to other structures.

Measurement of absolute track geometry employs traditional or automated surveying methods, with results recorded in a surveying coordinate system (i.e., northings, eastings, and elevation). The origin of the coordinate system can be positioned anywhere on or off the track, with the XY surface parallel to the surface of the geoid and the Z axis perpendicular to this surface, aligning with Earth's gravitational force direction.

While absolute track geometry is essential for track construction and sometimes maintenance, it is not ideal for assessing the track's compliance with safety standards or predicting how rail vehicles will respond to track irregularities.

Relative or inner geometry is recorded in a track-centered coordinate system. Design geometry variables (i.e., curvature, superelevation, grade) and track irregularity variables (e.g., alignment, profile, gauge, cross level, and twist) are plotted against the distance along the track (i.e., chainage), often referred to as being in the distance domain. While relative track geometry doesn't provide information on the heading or position of the track with respect to absolute references, it describes curves and perturbations affecting vehicle behavior. Most TGMS measure relative track geometry,

which is suitable for verifying track quality, compliance with safety standards, and predicting vehicle response using multibody dynamics (MHD) simulation.

Absolute track geometry can be easily and accurately converted to relative track geometry. The reverse process, while theoretically possible through integration, is less accurate due to accumulating errors, unless combined with engineering survey measurements (Reedman, 2014).
Relative track geometry is recorded in two formats: space curve and chord offset (i.e., versine). The space curve format directly relates to vehicle performance and is used for multi-body dynamics (MBD) simulations and track quality assessment (Zhang et al., 2004; El-Sibaie and Zhang, 2004). Many countries base their track safety standards on space curve measurements (EN 13848-1,5,6). However, its complexity in measurement and processing leads some agencies to avoid using it (Malon, 2007).

Chordal offset measurements, while not directly related to vehicle performance, have been shown to perform similarly to space curve measurements in predicting vehicle response to track geometry (Keylin, 2019). These measurements are intuitive and can be taken with simple hand tools. Many countries, including the United States, based their track safety standards on chordal measurements.

1.2.3 Space Curve

The space curve concept separates track geometry features into long-length elements, such as curvature and gradient, and short-length elements, like alignment and profile deviations. This separation is achieved by establishing a reference track trajectory. The space curve describes both the deviations of the measured track from this reference trajectory and the overall geometry of the reference trajectory itself.

Figure *7* presents a plane view of a rail track. In this figure, solid line depict the actual positions of the rails and track centerline, while dashed lines represent the reference trajectory of the track. The space between these solid and dashed lines illustrates the space curve, which is further detailed in
Figure *8*.

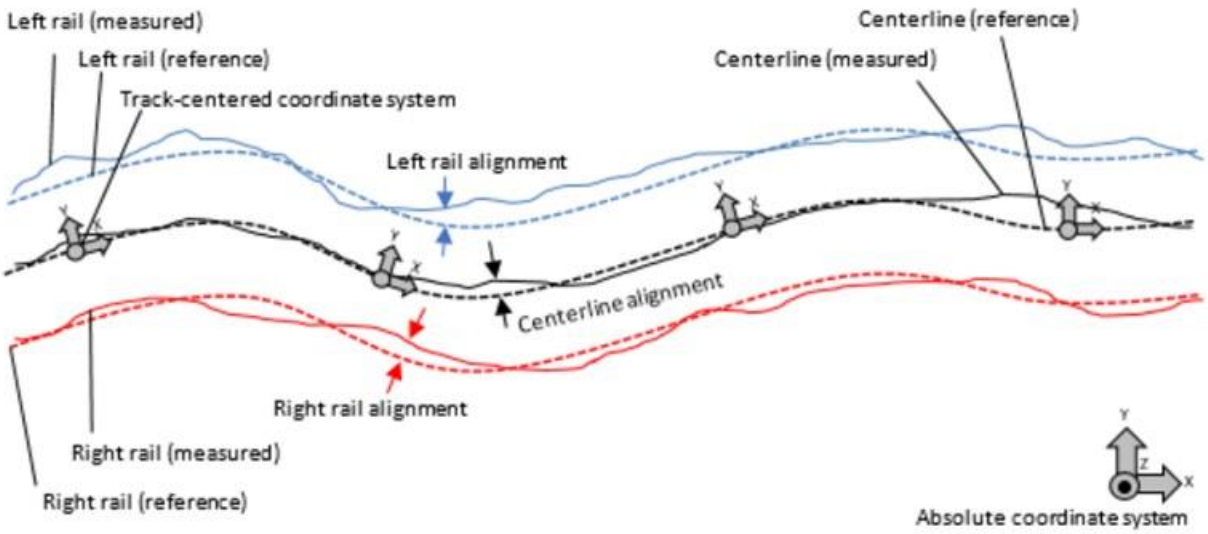

Figure 7 The relationship between absolute track geometry and alignment.

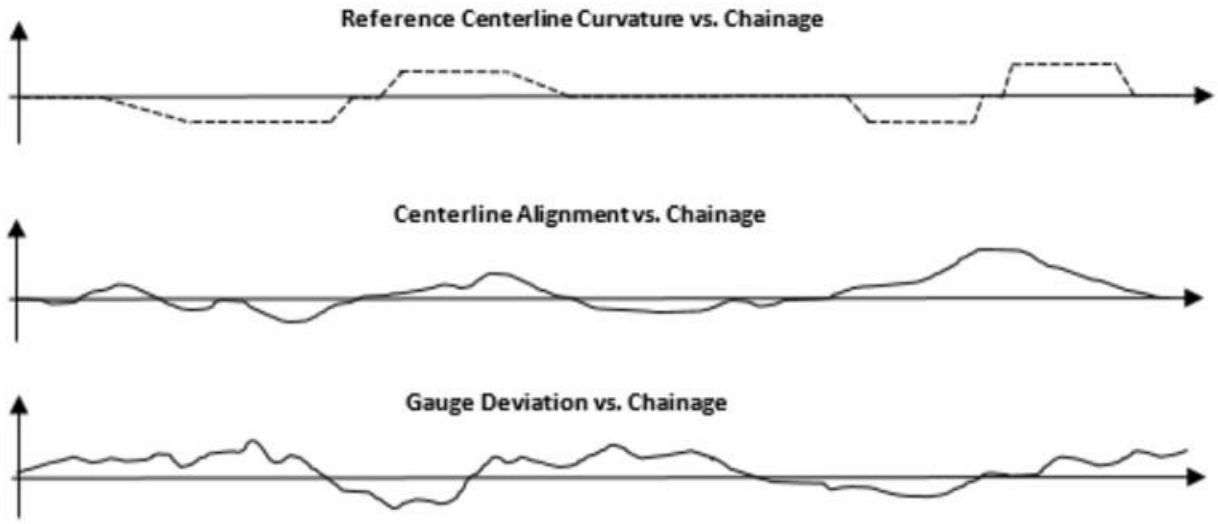

Figure 8 Space curve corresponding to the track in Figure 7.

Vertical plane track geometry is described in a similar manner to the horizontal plane, as illustrated in
Figure *9*. The reference centerline geometry is characterized by grade or vertical curvature, while the measured track geometry is described in terms of profile and cross level (or alternatively, left and right rail profiles).

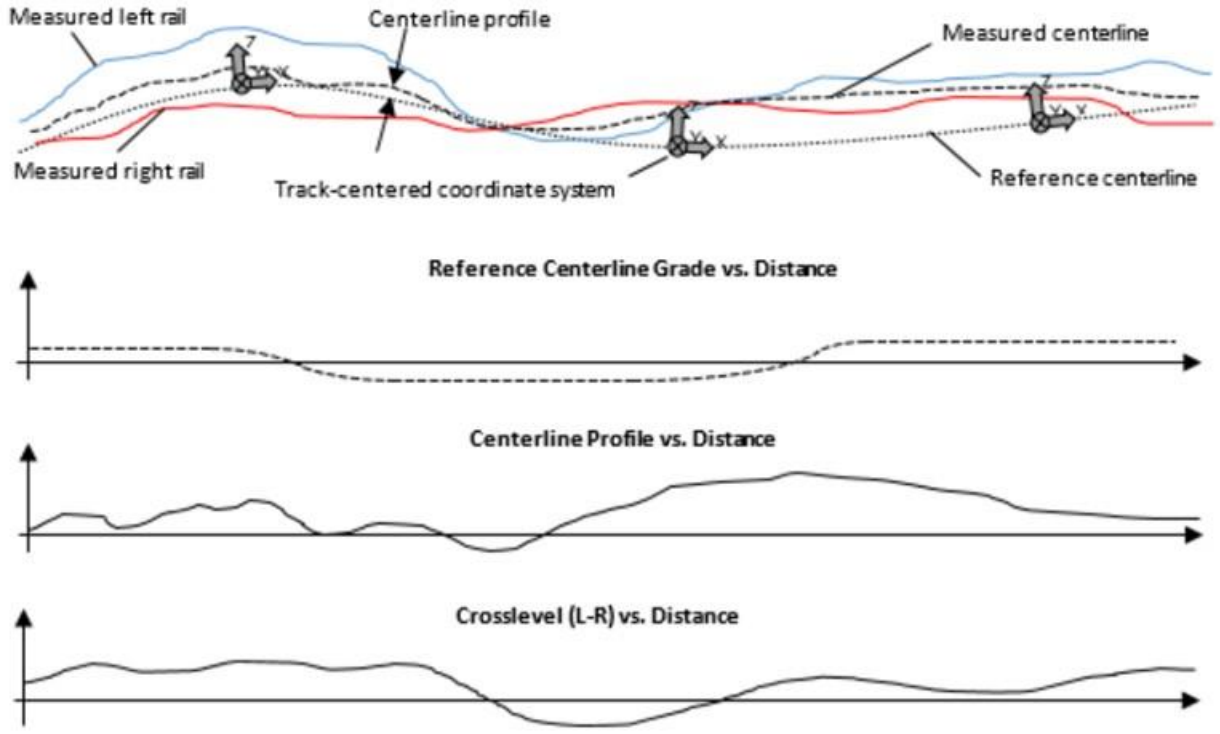

Figure 9 Absolute and relative vertical track geometry.

Space curve data is recorded in a track-centered coordinate system. While there is no universally agreed-upon system, one possible option is described as follows:

The origin of the system is on the reference centerline trajectory. The X axis aligns with the heading direction of the track, the Y axis points from the origin toward the nominal left rail position, and the Z axis points upward from the origin. This coordinate system can be either tilting or non-tilting, as shown in
Figure *10*.

In a non-tilting system, the Z axis always remains parallel to Earth's gravity. In a tilting system, the Z axis is always perpendicular to either the reference track plane (defined by nominal left and right rail positions) or the measured track plane (defined by measured left and right rail positions) (Lewis, 2011). This distinction is important when defining space curve variables.

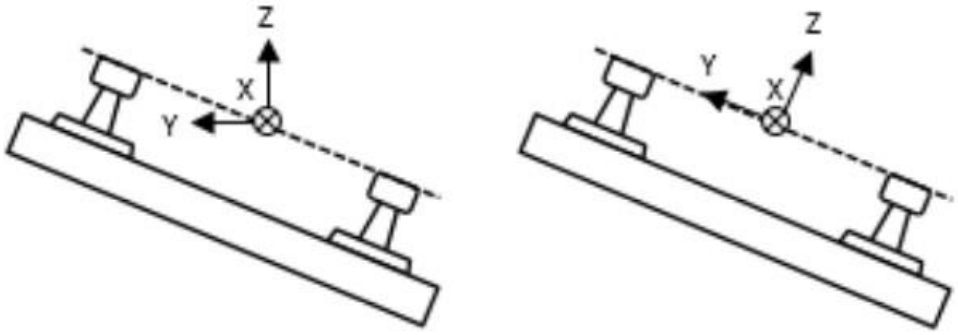

Figure 10 Non-tilting (left) and tilting (right) track-centered coordinate systems

Space curve data typically includes several key elements: chainage, curvature, cross level, alignment, gradient, superelevation, and profile.

Chainage measure the distance along the centerline of the track. Curvature describes how much the centerline of the track bends horizontally at a specific point. It can be expressed in various ways, such as the inverse of the radius of the curve, as MCO, or as an angle formed by a 100-foot chord (
Figure *1*).

In this context, gauge refers to the distance between the rails at a particular point on the track (Figure 5). It may differ from the standard gauge due to slight sideways shifts of the rails from their intended positions.

Cross level typically refers to the unplanned height difference between the left and right rails (Figure 5). When this difference is intentional, such as on curves, it is called superelevation. Some sources use "superelevation" to refer to both planned and unplanned height differences.

Cross level is calculated using a formula that considers the superelevation, the angle measured by an inclinometer, and the standard distance between rail centers (usually 1,500 mm or 59.055 inches for standard gauge tracks):

$$\Delta z_{XLV} = W\sin\alpha_{XLV} - \Delta z_{SE} \tag{5}$$

where Δz_{XLV} is the cross level, Δz_{SE} is the superelevation, α_{XLV} is the cross level angle measured with an inclinometer, and W is the nominal track width, or center to center rail distance. If the actual gauge differs from the standard, the calculated cross level may not exactly match the true height difference between the rails.

Centerline alignment measures how far the actual track centerline deviates sideways from its intended position. This deviation represents the combined lateral shift of both rails from where they should be.

Gradient, also known as grade, indicates how quickly the centerline of the track changes in elevation relative to its horizontal distance.

Centerline profile measures the vertical difference between the actual and intended track centerline, perpendicular to the intended line. This represents how both rails jointly deviate vertically from their planned positions. The intended centerline includes both the gradient and any designed vertical curves. Most TGMS provide separate alignment and profile data for each rail, rather than combined centerline measurements.

In a tilting reference system, the relationship between the alignment and profile of each track and the centerline alignment can be expressed as:

$$y_L \cong y_C + \frac{1}{2}\Delta g, y_R \cong y_C - \frac{1}{2}\Delta g \tag{6}$$

$$z_L \cong z_C + \frac{1}{2}\Delta z_{XLV}, z_R \cong z_C - \frac{1}{2}\Delta z_{XLV} \tag{7}$$

where y_L, y_R, y_C are left rail, right rail, and centerline alignments; z_L, z_R, and z_C are left rail, right rail, and centerline profiles, respectively; Δg is gauge deviation. These relationships aren't exact due to two factors

1. The plane used for gauge measurement may not be perfectly horizontal, depending on the cross level.
2. The cross level measurement does not precisely equal to the height difference between the rail tops.

Some of these track geometry variables are visually represented in
Figure *8*.

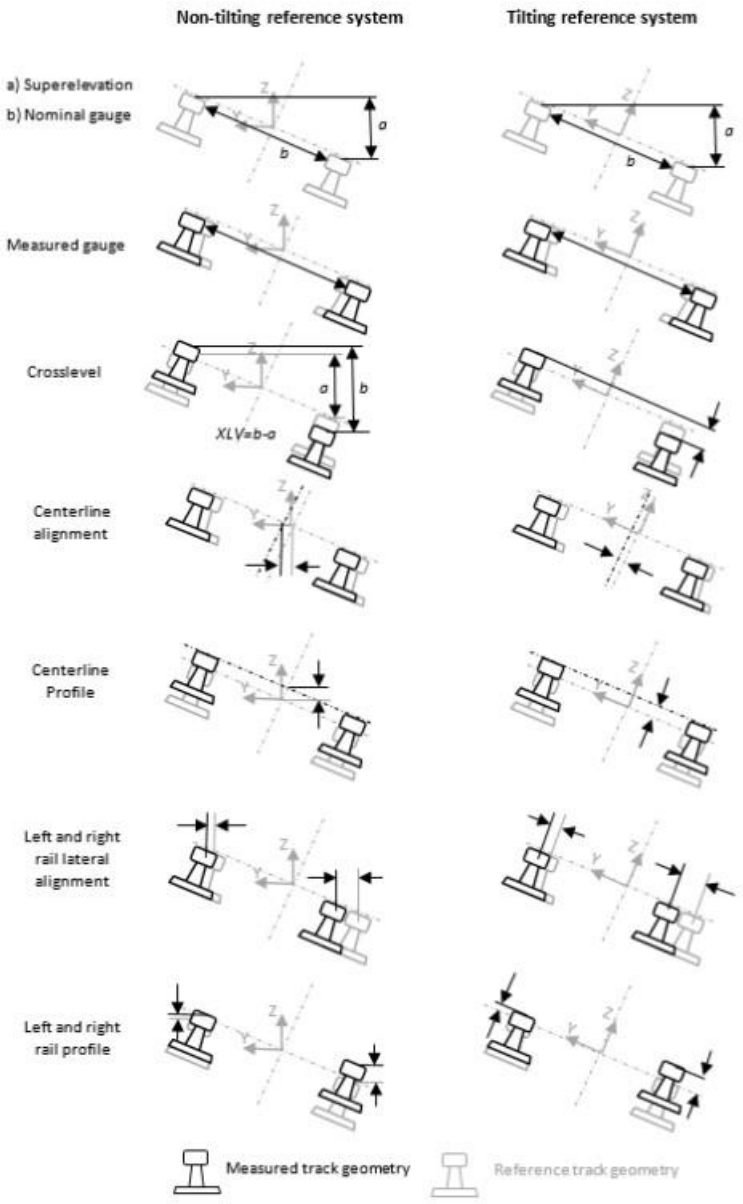

Figure 11 Space curve variable definitions

1.2.4 Chordal (Versine) Measurements

Many countries, including the United States and Canada, define their track safety standards using chordal measurements, also known as versine (
Figure *12*,
Figure *13*), rather than space curve data. This approach is outlined in specific regulations such as 49 CFR §213 Subpart C for the US and TC E-54 Subpart C for Canada. A versine with a ratio α of 0.5 is termed a symmetric versine, or more commonly, a mid-chord offset (MCO).

The main advantage of using chordal measurements is their practicality. They can be taken easily in the field using basic, handheld tools, making them convenient for on-site track inspections and maintenance.

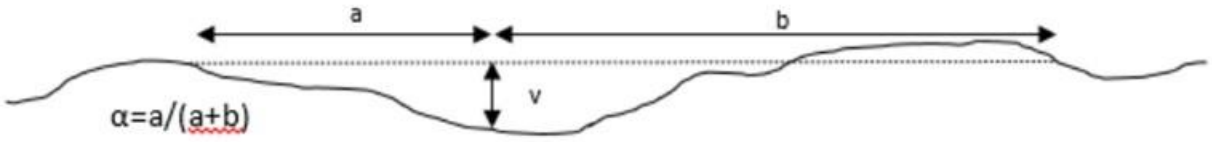

Figure 12 Versine (chordal) measurement

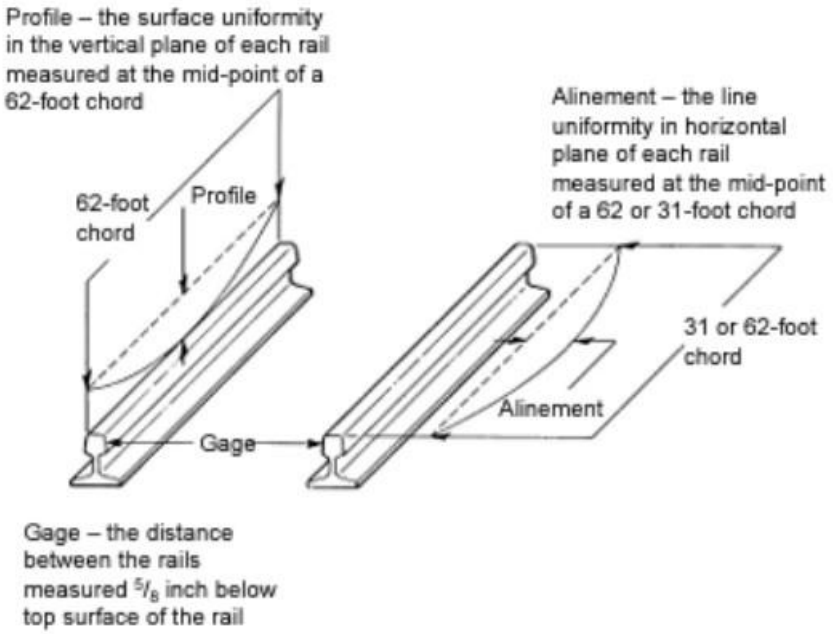

Figure 13 Mid-chord offsets (2017 FRA Compliance Manual, Vol. 2, Ch.1,p.2.1.29).

A chordal offset is influenced by two factors: the irregularities in track alignment (
Figure *12*) and intentional curvature in the track design (Figure 2). For example, a point in the body of a 10-degree curve has an MCO value of 10 inches for a 62-foot chord length if the rail at the point is perfectly aligned with its reference trajectory.

Because of this dual influence, track geometry standards are not based on the raw chordal measurements. Instead, they use the difference between the actual chordal values and the average values in the surrounding area. This approach allows for distinguishing between planned curvature and unintended irregularities.

1.2.5 Additional Track Geometry Variables

Beyond the basic track geometry variables, additional parameters can be derived:

Vertical curvature, which describes how the track centerline curves in the vertical plane, can be used to calculate the centrifugal acceleration experienced by a rail vehicle on a vertical curve. Twist and warp both measure how quickly cross level changes. Twist is defined as the cross level difference between two specific points, while warp is the maximum cross level difference between any points within a set distance. These can be calculated using chainage x and base distance L:

$$\mathrm{Twist}(x) = \Delta z_{XLV}\left(x + \frac{L}{2}\right) - \Delta z_{XLV}(x - \frac{L}{2}) \tag{8}$$

$$\mathrm{Warp}(x) = \max\left(\Delta z_{XLV}(x_1)\right) - \min\left(\Delta z_{XLV}(x_2)\right); x - \frac{L}{2} \leq (x_1, x_2) \leq x + \frac{L}{2} \tag{9}$$

Maximum gauge change is determined within a specified distance (i.e., between any two points less than a specified distance apart).

Runoff (ramp) refers to how much the elevation of a rail changes over a 31-foot segment at the end of raised section (49 CFR §213.63). Track recording vehicles typically measure this as the peak-to-peak amplitude within a 31-foot space curve window (Clouse, 2018).

Dip angle measures localized changes in the vertical gradient of a rail (
Figure *14*), often noticeable near rail joints. It is usually expressed in degrees or milliradians and is calculated from changes in vertical rail profile gradient over short distances. Dip angle is closely related to vertical wheel impact loads and can indicate potential rail end breaks. Its magnitude can vary based on vehicle speed, wheel load, and travel direction due to rail elasticity (Mandal et al., 2016; RAIB, 2014; prEN 13848-1:2016).

Kin angle, also called entry angle, is similar to dip angle but in the lateral direction (
Figure *14*). It is a design feature of turnouts and is not commonly measured by TGMS.

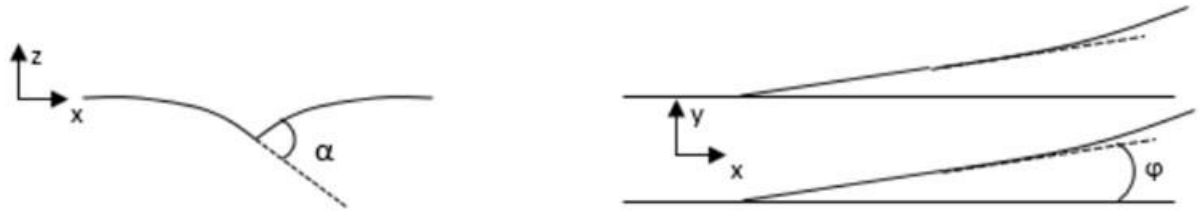

Figure 14 Rail dip angle (left); switch entry (i.e., kink) angle (right).

1.3 Types of Track Geometry Measurement Systems

Track geometry measurement systems are designed to collect comprehensive track geometry data. The data gathered typically includes measurements of curvature, grade, cross level, gauge, alignment, and profile.

1.3.1 Relative and Absolute TGMS

Relative TGMS, which typically employ either inertia-based or chordal methods, make up the majority of both trolley-based and vehicle-mounted systems. These systems are generally more cost-effective and easier to use compared to absolute TGMS of similar accuracy and portability. Most absolute TGMS use optical surveying methods. However, some newer TGMS overcome the limitations of optical surveying by combining inertial measurement systems (IMS), high-precision global navigation satellite system (GNSS), and/or machine vision technologies (Engstrand, 2011; Pinter, 2012; Chen et al., 2015; Chen et al., 2018; Trimble, 2017). While almost all absolute TGMS are mounted on small trolleys, there are also track recording vehicles equipped with absolute TGMS (Vogelaar, 2017).

1.3.2 Platform

TGMS can be installed on various platforms, including manually pushed trolleys, self-powered carts, road-rail (i.e., hi-rail) vehicles, specialized track geometry vehicles, and regular service vehicles. Beyond the obvious logistical and financial factors to consider, it is crucial to understand that TGMS are influenced by both the stiffness of the track and the weight of the measuring vehicle. TGMS can be mounted on hand-pushed trolleys, self-propelled carts, road-rail (i.e., hi-rail) vehicles, dedicated track geometry vehicles, and revenue service vehicles. Aside from obvious logistical and financial considerations, it is important to recognize that track geometry measurements are affected by the stiffness of the track and the weight of the measuring vehicle. However, the relationship between wheel load and track geometry irregularities is not simple nor direct. It is dependent on several factors, such as the condition of the ballast and soil, the type of track structure, and the kind of rail fasteners used. Moreover, the degree to which a truck bends

the rail (and consequently affects the measured amplitudes of gauge and alignment irregularities) is partly determined by its dynamic performance, including its behavior when negotiating curves.

1.3.3 Principle of Operation

Manual track geometry measurements, while straightforward in principle, are time-consuming and require significant labor. One manual method, chord surveying, measures relative track geometry using chordal offsets. This technique employs a chord of specific length and a ruler (
Figure *12*,
Figure *13*). It is often used to confirm track geometry defects that have been identified by vehicle-mounted systems.

Another manual approach, optical surveying, measures absolute track geometry. This method uses either a theodolite or an automated geodetic total station. It involves measuring the vertical and lateral coordinates of a point on one rail in relation to a global coordinate system (
Figure *15*, top). The corresponding coordinates for the opposite rail are then estimated using these measurements, along with gauge and cross level data obtained with a gauge bar at that location (Stow and Andersson, 2006).

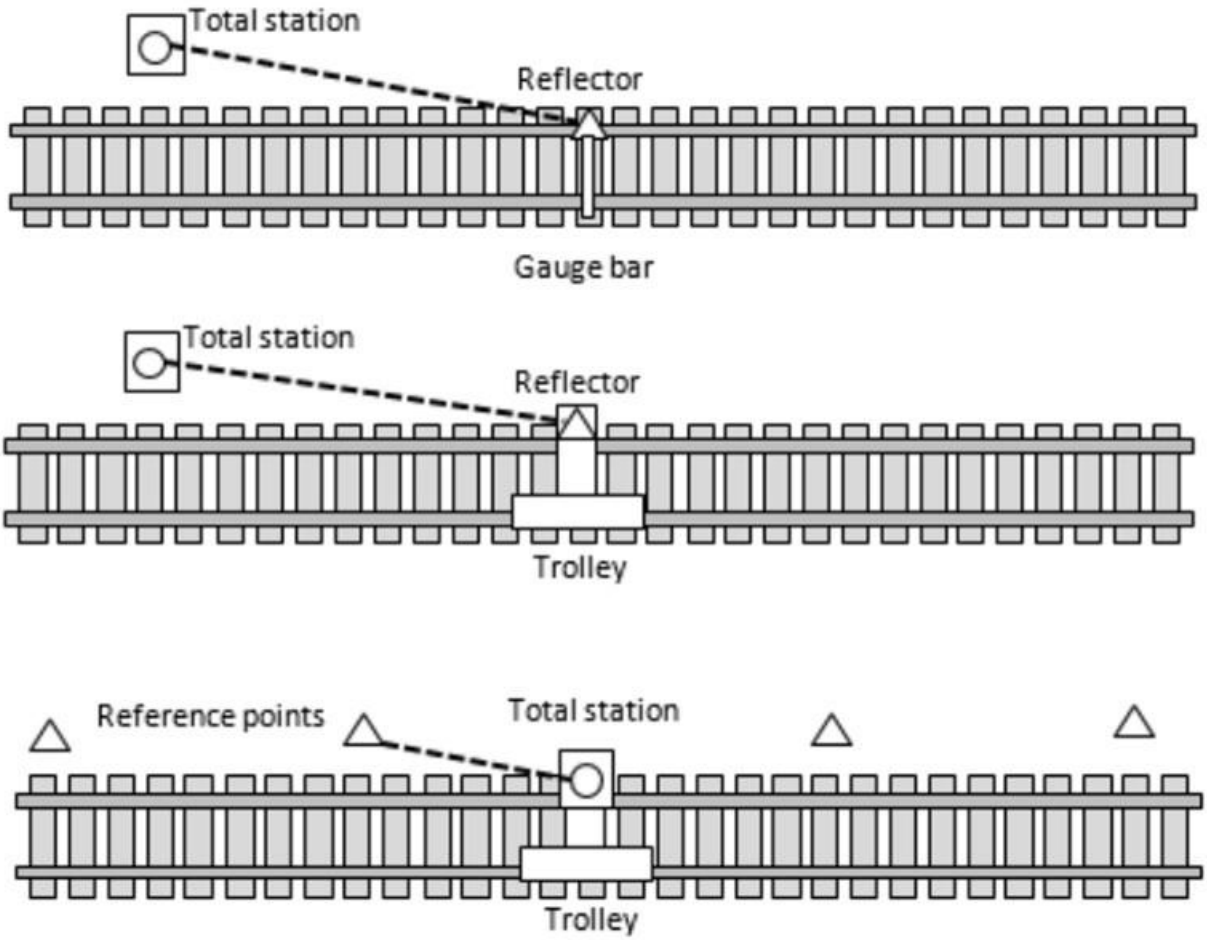

Figure 15 Manual and automated optical surveying.

Automated optical surveying uses specialized track surveying trolleys. A typical setup involves a stationary total station and a reflective target on a trolley, which also carries sensors for measuring cross level and gauge. In some configurations, the total station is mounted on a trolley and uses fixed reflectors along the track (
Figure *15*).

The trolleys can operate in two modes: kinematic (i.e., at walking speed) or stop-and-go, with the latter offering higher accuracy. However, this method has significant drawbacks, including slow measurement speed and limited range (typically tens to hundreds of meters). The range is further affected by topography and weather conditions, with higher temperatures and stronger winds reducing the effective distance.

While it is possible to extend the range and improve accuracy by relocating the total station and making overlapping measurements, this process is cumbersome and time-consuming. As a result, most absolute TGMS using this method are limited to trolleys designed for measuring short track sections.

Some optical surveying systems enhance their capabilities by incorporating data from inertial measurement systems (IMS). These IMS can be installed on various parts of a track recording vehicle, such as the carbody or truck frame, or less commonly, on a portable trolley (Chen et al., 2015; Sundaram and Wilson, 2016; Trimble, 2017).

These systems integrate data from multiple sensors:

- An odometer to measure chainage
- Non-contact (usually optical) or contact sensors to determine the relative position of rails to the sensor
- A combination of accelerometers and gyroscopes to measure the relative acceleration and velocity of the measuring device, which is then integrated to calculate its relative position

Traditional vehicle-mounted IMS have accelerometers and gyroscopes on the truck frame or carbody, with displacement sensors (such as linear variable differential transformers (LVDTs)) mounted across vehicle suspension components (
Figure *16*). In contrast, many modern IMS are self-contained units attached entirely to the carbody or truck frame, without measuring suspension displacement.

Each approach has its pros and cons. Sensors on upsprung masses like axle boxes provide a more direct connection to track irregularities but face harsh vibrations and must measure a wide

frequency range. Carbody-mounted TGMS with optical sensors may have limitations in measuring extreme track curvatures due to their offset from the rails.

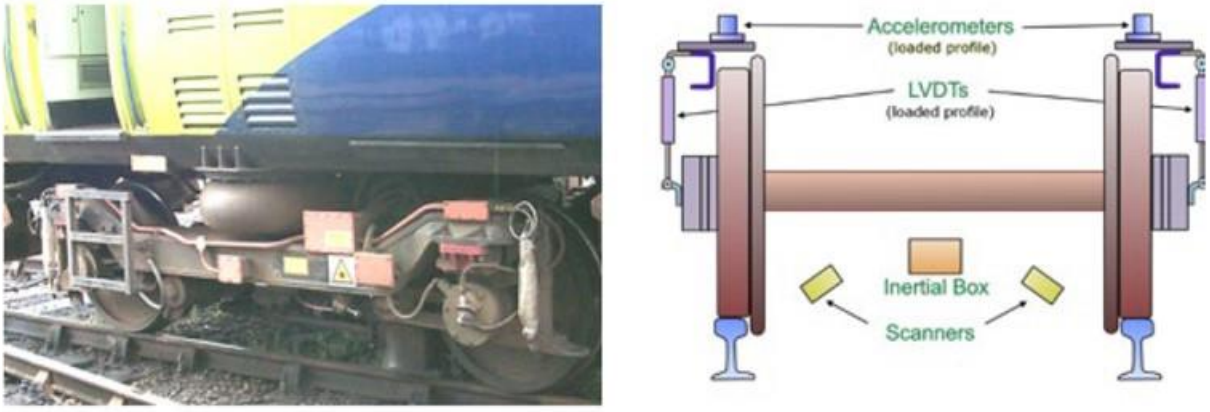

Figure 16 Traditional vehicle-mounted IMS and its schematic (Lewis, 2011)

A significant limitation of IMS is the minimum vehicle speed required for accurate measurements. At lower speeds, the outputs from accelerometers and gyroscope sensors decrease, leading to a poor signal-to-noise ratio below a certain speed threshold (Lewis, 2011). This means that the slower the vehicle moves, the shorter the maximum wavelength of track irregularity that the IMS can accurately measure. Typically, measuring long wavelengths demands much higher speeds than shorter wavelengths. However, recent advancements in sensor design and data processing have led to new IMS capable of measuring track geometry at lower speeds (i.e., around 5 mph) and even during brief stops.

Chordal systems are modernized versions of manual chord surveys (
Figure *13*). They measure the lateral and vertical distances between each rail and the frame of the track recording vehicle or trolley at a minimum of three points (
Figure *17*). When necessary, adjustments can be made to account for the bending of the vehicle frame (Haigermoser et al., 2015). The resulting chordal measurements can be converted into a space curve through a restoration process. Asymmetric chordal systems have fewer “zeros” (i.e., wavelengths with zero gains that cannot be restored) but introduce phase distortion. Even for asymmetric systems, the larger the ratio of track defect wavelength to chord length, the more challenging it becomes to accurately restore that wavelength due to decreasing signal-to-noise ratio. Consequently, small trolley-based chord systems are limited to short wavelengths, making them

unsuitable for high-speed rail track measurements. Vehicle-mounted chord-based systems, however, can measure longer wavelengths.

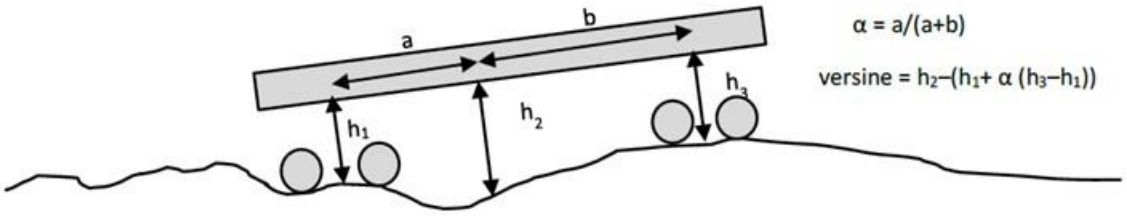

Figure 17 Vehicle-mounted chordal TGMS

Despite these limitations, chordal systems offer several advantages over inertial systems. Chordal systems are generally simpler in mechanical design and less expensive than IMS. Their accuracy is not dependent on vehicle speed, and they can perform static measurements. However, due to ongoing improvements in IMS design, vehicle-mounted chordal systems are becoming less common. Trolley-mounted chordal systems, on the other hand, remain widely used.

Some chordal systems have been developed to overcome typical disadvantage by combining chordal measurements with other measurement types. For instance, one trolley-mounted system a "differential difference method" (Naganuma and Yada, 2016), which integrates versine and slope measurements. Other systems combine chordal and inertial measurements (Yazawa and Takeshita, 2002; Yada et al., 2017).

Global Navigation Satellite System/Global Positioning Systems (GNSS/GPS) alone lack the precision required for track geometry measurements, with even high-accuracy differential GNSS having an accuracy of only 0.5-1.0 inch in this application (Szwilski et al., 2003). Typically, GNSS on track recording vehicles are used to record the location of track geometry defects, not to measure their magnitude.

However, some newer TGMS combine data from high-accuracy GNSS with inertial or chordal systems. This allows for measurement of both absolute and relative track geometry while addressing some issues associated with optical surveying systems (Luck et al., 2001; Kreye et al., 2004; Chen et al., 2015; Trimble, Inc., 2017). Other systems integrate GNSS data with optical surveying systems (Mahalakshmi and Joseph, 2013; Jiang et al., 2017).

Research has explored using Doppler Light Detection and Ranging (LiDAR) systems to measure track chainage and curvature. These offer advantages over traditional methods, including accuracy at low speeds and resistance to wheel slip and tread wear (Wrobel, 2013; Andani, 2016; Andani et al., 2018).

More sophisticated LiDAR-based machine vision systems have also entered the TGMS market. These sensors can create 3D point clouds, enabling measurement of track geometry (both absolute

and relative) and clearances between track and nearby structures (Vogelaar, 2017; Burton, 2018). Non-LiDAR machine vision systems are also in development (Gabara and Sawicki, 2018).

Accurate position determination remains a challenge in TGMS design and operation. Odometers often lack sufficient accuracy, while GNSS does not function in tunnels and is not precise enough to identify specific tracks in multi-track areas. This may require more complex methods, such as installing radio-frequency identification 9RFID) tags or specifically processing accelerometer data (Broquetas et al., 2012).

1.3.4 Autonomous TGMS

Over the past decade, advancements in electronics and software have led to the development of autonomous track geometry measurement systems (ATGMS). These systems, mounted on regular service trains, operate without human supervision, enabling more frequent and cost-effective inspections compared to traditional TGMS. The collected track geometry data is wirelessly transmitted to a remote operator (Morant, 2016; Stuart, 2017; Higgins and Liu, 2018). While railroads, transit agencies, and regulatory bodies have shown great interest in ATGMS and conducted extensive testing, their widespread implementation is currently hindered by several issues (Morell, 2017), including:

- Challenges in precisely locating defects, especially in distinguishing between tracks in multi-track areas
- Difficulties in ensuring data quality and preventing false positive without real-time human data examination
- Logistical issues, such as underdeveloped procedures for hardware maintenance and data handling
- Regulatory uncertainties, particularly regarding requirements for addressing defects identified during automated inspections

Additional technical challenges may arise depending on the specific vehicles hosting the ATGMS. For instance, installing an ATGMS on a freight railcar may expose it to harsh load conditions with high accelerations and varying natural frequencies based on loading conditions. Furthermore, the weight of the vehicle and curving characteristics can cause ATGMS measurements to differ from those taken by a traditional track geometry vehicle on the same route.

1.3.5 Vehicle Response Measurement

The end goal of track geometry measurement is to ensure vehicle safety and passenger comfort. As a result, it is often valuably to directly measure vehicle performance by recording wheel-rail forces and/or accelerations of the carbody and truck, and to draw conclusions about track condition from these measurements.

Vehicle response measurement systems (VRMS) complement traditional track geometry measuring system, but do not replace them for several reasons:

- Vehicle response does not clearly distinguish between different types of irregularities or

measure their magnitudes accurately

- The response is specific to each vehicle. A lack of response from one vehicle to a track defect does not guarantee safety for other vehicle types or even for the same vehicle at a different speed; the correlation is not straightforward

Directly measuring wheel-rail forces requires the use of instrumented wheelsets, which are costly and require significant effort to design, use, and maintain. Accelerometers do not have these drawbacks, but the readings do not directly represent wheel-rail forces.

1.4 References

1. Liu, X., Saat, M., & Barkan, C. (2012). Analysis of causes of major train derailments and their effect on accident rates. Journal of the Transportation Research Board, 2289, 154–163.
2. Ciobanu, C. (2016). The versine formulae. Accessed: December 12, 2018.
3. Hay, W. (1982). Railroad engineering, 2nd Edition. John Wiley & Sons, New York.
4. Puzavac, L., Popović, Z., Lazarević, L. (2012). Influence of Track Stiffness on Track Behaviour under Vertical Load. Promet – Traffic & Transportation, 24(5), 405–412.
5. Haigermoser, A., Luber, B., Rauh, J., & Grafe, G. (2015). Road and track irregularities: measurement, assessment and simulation. Vehicle System Dynamics, 53(7).
6. Zarembski, A.M., Grissom, G.T., Euston, T.L., & Cronin, J.J. (2015). Relationship between missing ballast and development of track geometry defects. Transportation Infrastructure Geotechnology, 2, 167–176.
7. Muinde, M.S. (2018). Railway track geometry inspection optimization. M.S. Thesis, Luleå University of Technology.
8. Reedman, M. (2014). Kinky Tracks. Conference on Railway Excellence (CORE-2014), Adelaide, Australia, May 5–7, 2014.
9. Zhang, Y., El-Sibaie, M., & Lee, S. (2004). FRA track quality indices and distribution characteristics. Proceedings of AREMA Annual Conference, Nashville, TN, September 19-22, 2004.
10. Zhang, Y., El-Sibaie, M., & Lee, S. (2004). FRA track quality indices and distribution characteristics. Proceedings of AREMA Annual Conference, Nashville, TN, September 19-22, 2004.
11. Malone, J. (2007). Performace and testing requirements for portable track geometry inspection systems. TCRP Research Results Digest, 83.
12. Keylin, A. (2019). Measurement and Characterization of Track Geometry Data: Literature Review and Recommendations for Processing FRA ATIP Program Data.
13. Lewis, R. (2011). Track geometry recording and usage: Notes for a lecture to Network Rail.
14. Clouse, A. (2018). Federal Railroad Administration. Personal communication. April 6, 2018
15. Mandal, N.K., Dhanasekar, M., & Sun, Y.Q. (2016). Impact Forces at Dipped Rail Joints. Proceedings of the Institution of Mechanical Engineers, Part F: Journal of Rail and Rapid Transit, 230(1), 271–282.
16. European Committee for Standardization (CEN) (2016). PrEN 13848-1: Railway applications – Track – Track geometry quality – Part 1: Characterization of track geometry.
17. Engstrand, A. (2011). Railway Surveying – A Case Study of the GRP 5000. MS Thesis, Royal Institute of Technology, Stockholm.
18. Pinter, O. (2012). Using of Trimble® GEDO CE system for absolute track positioning. [in Czech], Diploma Thesis, Czech Technical University in Prague, Prague.
19. Chen, Q., Niu, X., Zhang, Q. & Cheng, Y. (2015). Railway track irregularity measuring by GNSS/INS integration. Navigation, 62(1), 83-93.
20. Chen, Q., Niu, X., Zuo, L., Zhang, T., Xiao, F., Liu, Y., & Liu, J. (2018). A railway track geometry measuring trolley system based on aided INS. Sensors, 2018(18).

21. Trimble Inc. (2017). Trimble adds inertial-based trolley solution to its track survey and scanning rail portfolio. Press Release.
22. Vogelaar, J. (2017). Absolute track geometry, what is it and how does it help me? Rail Infrastructure and Vehicle Inspection Technology Conference, University of Illinois Urbana Champaign, June 20-21, 2017.
23. Stow, J. & Andersson, E. (2006). Field testing and instrumentation of railway vehicles. Handbook of railway vehicle dynamics, ed. S. Iwnicki, CRC Press, Boca Raton, FL.
24. Sundaram, N. & Wilson, R. (2016). Portable track geometry measurement system: a unique derailment investigation tool. Presentation, 2016 APTA Rail Conference, 2016.
25. Naganuma, Y. & Yada, T. (2016). Development of truly portable track geometry recording trolley and accompanying new measurement principle. Proceedings of the 15th International Conference on Railway Engineering Design and Operation (CR 2016), 2016.
26. Yazawa, E. & Takeshita, K. (2002). Development of measurement device of track irregularity using inertial mid-chord offset method. Quarterly Report of RTRI, 43(3).
27. Yada, T., Soda, Y., & Naganuma, Y. (2017). Improvement of the realignment performance for short wavelength track irregularity on a tamping machine. WIT Transactions on Engineering Sciences, 118.
28. Szwilski, T.B., Begley, R., Dailey, P., Sheng, Z., & Rahall, N.J. (2003). Determining rail track movement trajectories and alignment using HADGPS. Proceedings of the AREMA Annual Conference, Chicago, Ill, USA, October 2003.
29. Luck, T., Eissfeller, B., Kreye C., & Meinke, P. (2001). Measurement of Line Characteristics and of Track Irregularities by Means of DGPS and INS. International Symposium on Kinematic Systems in Geodesy, Geomatics and Navigation, Banff, Alberta, Canada, June 5-8, 2001.
30. Kreye, C., Eissfeller, B., & Ameres, G. (2004). Architectures of GNSS/INS Integrations Theoretical Approach and Practical Tests. Institute of Geodesy and Navigation, University FAF Munich, Neubiberg, Germany. Accessed Dec. 12, 2018.
31. Mahalakshmi, V. & Joseph, K.O. (2013). GPS Based Railway Track Survey System. International Journal of Computer Applications in Engineering Sciences, Volume III, Special Issue on National Conference on Information and Communication (NCIC'13).
32. Jiang, Q., Wu, W., Li, Y., & Jiang, M. (2017). Millimeter Scale Track Irregularity Surveying Based on ZUPT-Aided INS with Sub-Decimeter Scale Landmarks. Sensors, 2017(17).
33. Wrobel, S. (2013). Multi-function LIDAR sensors for non-contact speed and track geometry measurement in rail vehicles. M.S. Thesis, Virginia Polytechnic Institute and State University, Blacksburg, VA.
34. Andani, M. T. (2016). The application of Doppler LIDAR technology for rail inspection and track geometry assessment. PhD Dissertation, Virginia Tech, Blacksburg, Virginia.
35. Andani, M.T., Peterson, A., Munoz, J., & Ahmadian, M. (2018). Railway track irregularity and curvature estimation using doppler LIDAR fiber optics. Journal of Rail and Rapid Transit, 232(1), 63–72.
36. Burton, T. (2018). RILA Train Mounted Survey System, Opportunities & Constraints.
37. Gabara, G. & Sawicki, P. (2018). A New Approach for Inspection of Selected Geometric Parameters of a Railway Track Using Image-Based Point Clouds. Sensors, 18(3).
38. Broquetas, A., Comerón, A., Gelonch, A., Fuertes, J.M., Castro, J. A., Felip, D., López, M. A., & Pulido, J.A. (2012). Track Detection in Railway Sidings Based on MEMS Gyroscope Sensors. Sensors.
39. Morant, S. (2016). Automating geometry measurement offers real-time benefits. International Railway Journal.
40. Stuart (2017). Autonomous track geometry measurement system: technical development and short line demonstration. Rail Infrastructure and Vehicle Inspection Technology Conference, University of Illinois Urbana-Champaign, June 20-21, 2017.

41. Higgins, C. & Liu, X. (2018). Modeling of track geometry degradation and decisions on safety and maintenance: A literature review and possible future research directions. Journal of Rail and Rapid Transit, 232(5), 1385–1397.
42. Morell, J. (2017). Evaluation of the Federal Railroad Administration's Autonomous Track Geometry Measurement System Research and Development Program (Report No. DOT/FRA/ORD-17/05). Federal Railroad Administration.

2 DEVELOPING OPTIMAL UAV FLIGHT PATH

2.1 Overview

The researchers concluded that an unmanned aerial vehicle (UAV) would be the optimal carrier for this automated track geometry measurement system (TGMS). However, the implementation of UAVs presents a significant challenge due to the intricate environment surrounding railway tracks. Common obstacles in these areas include power lines and other infrastructure. Consequently, it is crucial to develop an algorithm that enables the UAV to navigate effectively and avoid collisions while determining its flight path. The authors deemed Reinforcement Learning (RL) to be a suitable algorithm for this task.

2.2 Introduction

Reinforcement Learning (RL) has found its great use in a lot of practical applications, ranging from problems in mobile robot (Mataric, 1997; Smart and Kaelbling, 2002; Huang et al., 2005), adaptive control (Sutton et al., 1992; Lewis et al., 2012; Lewis and Varbie, 2009), AI-backed chess playing (Silver et al., 2017a; Silver et al., 2017b; Silver et al., 2016), among many others. The idea behind reinforcement learning, as illustrated in
Figure *18*, is that an agent learns from the environment by interacting with it and receives positive or negative rewards for performing calculated actions, and the cycle is repeated. The key issue of the whole process is to learn a way of controlling the system so as to maximize the total award.

When the agent begins to sense and learn a completely or partially unknown environment, it involves in two distinct tasks: exploration which attempt to collect as much information about the environment as possible, the exploitation which attempts to receive positive rewards as quickly as possible.

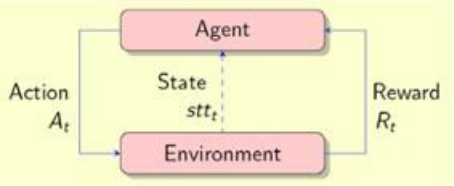

Figure 18 In reinforcement learning, the agent observes the environment and interact with the environment, update its own state and receives reward

Nevertheless, there is a dilemma of choosing between the two tasks of exploration and exploitation. Too much exploration will adversely influence the efficiency and convergence of the learning algorithm, while putting too much emphasis on exploitation will increase the possibility of falling into a locally optimal solution. The existing RL algorithms all attempt to balance out these two tasks in their learning cycles (
Figure *18*), but there is no guarantee that the best result can always be obtained.

Besides the exploration and exploitation dilemma, the RL algorithms have to employ value distributions that inexplicitly assume that environment is static (i.e., no change), or it changes very

slowly and/or insignificantly. However, in many real applications, the environment rarely stays unchanged. More than likely, the environment can be described in terms of states (
Figure *18*) changes over the course of exploration. In this case, value distribution has nothing to do with the problem at hand, and all the information obtained from the previous exploration efforts become less, or totally irrelevant.

To effectively solve the aforementioned problems in reinforcement learning for a better UAV exploration plan, the authors proposed a new algorithm based on the partitioning of the states set and search of the shortest path in a directed graph that represents a RL method.

2.3 Preliminaries and Background

In this section, the basic structure of reinforcement learning (RL) algorithms will be surveyed, with a particular focus on value-oriented methods. The exploration versus exploitation tradeoff in RL will be formally defined. The literature demonstrates that RL can be mapped to various graph representations, and these methods will be briefly described. Utilizing graph representations allows RL to benefit from the extensive results in graph algorithms.

2.3.1 Value-oriented Method for the Exploration-Exploitation Tradeoff in RL

Most RL problems can be formalized using Markov Decision Processes (MDPs), and there are a few key elements in RL as defined below:

1. Agent: An agent takes actions.
2. Environment: The physical world through which the agent operates.
3. States: A state refers to a specific and immediate situation in which the agent is located. In this paper, stt_i denotes the state of the agent at time instance i, with set S containing all the possible states that the agent can operate on. Thus, $\mathrm{stt}_i \in S$.
4. Action: Agents choose among a list of possible actions. Denote a_i as the action that agent might perform at time instance i. Actions is defined as the set of all possible moves of the agent can make (i.e., $a_i \in \mathrm{Actions}$).
5. Reward: A reward is the feedback that is used to measure the success of failure of an agent's action. Here a reward at time instance t is defined as R_t. Actions may affect both the immediate reward and, through the next situation, all the subsequent rewards (Sutton and Barto, 2017).
6. Exploitation: A task that makes the best decision given all the current information.
7. Exploration: A task that gathers more information to be used for making the best decision in the future.
8. An episode: The behavior process cycle of the agent from the beginning of the exploration to the beginning of the next exploration. When the interaction between the agent and the environment breaks naturally into subsequences, which are referred as episodes. Each episode ends in a special state called the terminal state, followed by a reset to a standard starting state or to a sample from a standard distribution of starting states (Sutton and Barto, 2017).

In RL, the exploration-exploitation tradeoff refers to a decision-making process that chooses between exploration and exploitation. Value-oriented RL methods have to deal with such exploration-exploitation tradeoff through the value distribution as defined by the value function or a probability that decides the chain of actions that lead to the target state all the way from the start state through a series of awards. A decision chain refers to a series of decision-making taken by an agent.

In order to strike a balance between exploration and exploitation, there are two main decision methods that can be followed, the ϵ-greedy method, and softmax.

In ϵ-greedy method, the action is selected by, one has

$$a_{stt} = \begin{cases} a^*_{stt} & \text{with proability } 1-\epsilon \\ \text{random action} & \text{with probability } \epsilon \end{cases} \tag{10}$$

where a^*_{stt} is the action in which of the value function assumes the highest value:

$$a^*_{stt} = \text{argmax}_{a \in \text{Actions}} Q(stt, a) \tag{11}$$

where $Q(stt, a)$ is action-value function which evaluates each possible action while in the current state. One drawback of ϵ-greedy action selection is that when it explores, all the possible actions are given the equal opportunity, as indicated in Eq. (1). In a simple term, this method is as likely to choose the worst-appearing action as it is to choose the next-to-best action. This gives rise to the so-called softmax method that can vary the action probabilities through a graded function below:

$$\pi(a|stt) = \frac{e^{\frac{Q(stt,a)}{\tau}}}{\sum_{a' \in \text{Actions}} e^{\frac{Q(stt,a')}{\tau}}} \tag{12}$$

Where $\pi(a|stt)$ is the probability policy to choose an action from the specific state stt, and τ is a "computational" temperature, and is an active-value function that evaluates each possible action in the current state.

The problem of value-oriented method is due to its weak ability to eliminate exploration blindness resulting from a large number of repeated explored states introduced by the values distribution structure. The stochastic factors that are added to help the search process jump out of the loops and balance exploration and exploitation actually come at the expense of more blindness of exploration.

2.3.2 RL over Graphs

A RL can be represented as a directed graph, G<V, E>, where a vertex, $v_i \in V(G)$, corresponds to a state stt_i in reinforcement learning and an edge, $e_{ij} \in E(G)$, connects two vertices (two states stt_i and stt_j) with a decision action a_i in reinforcement learning. In this graph, a path can be regarded as decision sequences in reinforcement learning.

In literature, many RL methods are related to their graph representation. In PartiGame Algorithm (Moore, 1994), the environment of RL is divided into cells modeled by kd-tree, and in each cell, the actions available consist of aiming at the neighbor cells (Kaelbling et al., 1996). Dayan et al. (Dayan et al., 1993) achieved speedup of reinforcement learning by creating a Q-learning managerial hierarchy in which high-level managers learn how to set tasks for their lower level managers. A hierarchical Q-learning algorithm (Dietterich, 1998) proves its convergence and shows experimentally that it can learn much faster than ordinary "flat" Q-learning. None of these methods, however, can solve the root problem concerning the dilemma of exploration and exploitation.

2.3.3 Floyd-Warshall Algorithm

Denote $SSA(G, e_i, e_j)$ as a shortest path search algorithm that is applied to G from vertice e_i to vertice e_j that represents a RL. The classical shortest path algorithms like Dijkstra (Dijkstra, 1959), and A* (Hart et al., 1968) are single starting point algorithms for the path-finding. The Floyd-Warshall algorithm (Floyd, 1962) (FW), which is purseued to use in this study, provides the shortest path between any two vertices in specified graph and it is found to be adaptive to the change of the graph.

In the standard FW algorithm, two matrices (DIST and NEXT) are used to express the information of all the shortest path in the graph. The matrix DIST records the shortest path length between two vertices. The matrix NEXT contains a name of the intermediate vertex through which the two vertices are connected through the shortest path. Because of the optimal substructure property of the shortest path, no matter how many immediate vertices the shortest path passes through, simply recording one of the intermediate vertices is sufficient to express the entire shortest path.

2.4 Convergence of Exploration and Exploitation

This section first defines the concept of a completely explored graph, which serves as the foundation for a graph-based iterative framework for reinforcement learning. Within this framework, the knowledge acquired from the RL exploration task can be recorded by the graph, allowing for the shortest path search to determine the next decision chain. This new approach effectively tracks changes in the graph caused by exploration and sometimes by a changing environment. It will be demonstrated that with this framework, which involves the shortest path search, exploration converges to exploitation. In simple terms, exploration will identify the shortest path to achieve the same reward as exploitation.

2.4.1 Completely Explored Graph

Definition 1: A Completely Explored State is a state of which all its possible successor states have already been explored. If one of a state's successor states has been explored and at least one of its successor states has not yet been explored, the state is called a Partially Explored State.

Definition 2: If the vertex set V in connected G<V, E> includes the start states of episodes and all these states have been completely explored, and the edge set E represents all the action that need

to be taken to connect all the different states, graph G is called a Completely Explored Graph (CEG).

Figure *19* shows an example of a CEG where each state ($\mathrm{stt_{xx}}$) is linked with up to 4 possible actions: $\mathrm{a_0}$, $\mathrm{a_1}$, $\mathrm{a_2}$, $\mathrm{a_3}$. There are some explored edges which are omitted for simplicity, such as action $\mathrm{a_1}$ for ($\mathrm{stt_{20}}$) and ($\mathrm{stt_{01}}$) with $\mathrm{a_0}$; they point to nonexistent state transitions.

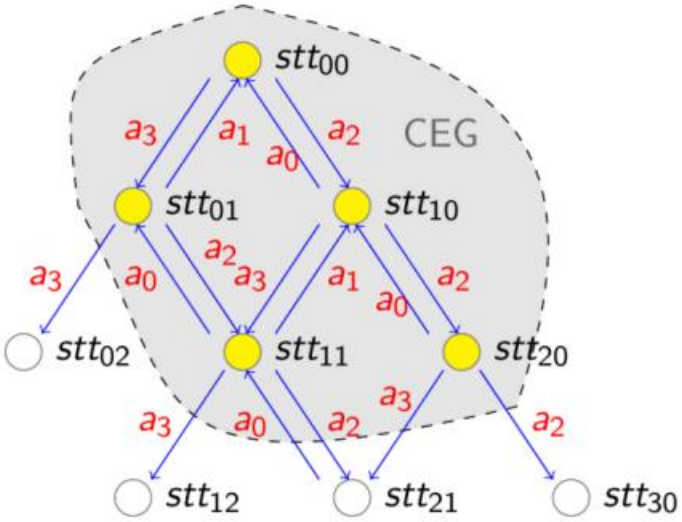

Figure 19 An example of a completely explored graph. Nodes represent states and directed edges between nodes represent actions. The shadowed area that includes all the State nodes (colored yellow) and all the associated directed edges represents the CEG. The unfilled nodes outside the shadowed area represent incompletely explored states, even though they connect to the CEG.

Suppose the complete action set $\mathrm{Actions} = \{\mathrm{a_i}: \mathrm{a_0}, \ldots, \mathrm{a_n}\}$ is known. State $\mathrm{stt_i}$ is a completely explored state if any reachable next state of $\mathrm{stt_i}$, denoted as $\mathrm{stt_{i+1}}$, by taking a possible action $\mathrm{a_i} \in \mathrm{Actions}$, is traversed. Let GU<V, E> represent a graph that contains all the traversed states, including both the completely and partially explored states. We can define the predecessor state set $\mathrm{Set_{predecessor}(stt)}$ as $\mathrm{Set_{predecessor}(stt)} = \{\mathrm{s}: (\mathrm{s} \in \mathrm{SE}) \wedge (\{\mathrm{s}, \mathrm{stt}\} \in \mathrm{E(GU)})\}$ and the successor state set $\mathrm{Set_{predecessor}(stt)}$ is defined as $\mathrm{Set_{predecessor}(stt)} = \{\mathrm{s}: (\mathrm{s} \in \mathrm{SE}) \wedge \{\mathrm{stt}, \mathrm{s}\} \in \mathrm{E(GU)}\}$ where SE denotes the set V(CEG). If an environment feedback is denoted by Env, then for a given action $\mathrm{a_k}$, the next state $\mathrm{s_{i+1}}$ can be determined as $\mathrm{s_{i+1}} = \mathrm{Env}(\mathrm{s_i}, \mathrm{a_k})$.

2.4.2 Exploration Converges to Exploitation

From CEG, it can be proven that exploration converges to exploitation. Here, $\mathrm{stt_{rwd}}$ is denoted as a reward state.

Lemma 1. Suppose that exploration of each episode starts with state stt_0, and ends in reward state $\mathrm{stt}_{\mathrm{rwd}}$ after passing some intermediate states through a series of episodes. Of all the possible episodes, one can see $\mathrm{stt}_{\mathrm{rwd}} \notin \mathrm{SE}$.

Proof: Once the agent has landed in state $\mathrm{stt}_{\mathrm{rwd}}$, the episode ends, so there will be no further exploration originated from $\mathrm{stt}_{\mathrm{rwd}}$. That is, the graph is a completely explored graph and the states are completely explored states, according to definitions 1 and 2. Thus $\mathrm{stt}_{\mathrm{rwd}}$ cannot be a member of the SE set. (End of proof)

Define the envelope set of set SE as $\mathrm{SEE} = \{\mathrm{stt}_i : (\mathrm{stt}_i \mathrm{SE}) \wedge (\mathrm{Set}_{\mathrm{sucessor}}(\mathrm{stt}_i) \not\subset \mathrm{SE})\}$
Corollary 1. For $\mathrm{stt}_i \in \mathrm{SEE}$ consists of members in SE, there exists at least one of the successor states of stt_i does not belong to SE.

Proof: It comes directly from the definition of SEE. (End of proof)

Corollary 2. If $\mathrm{stt}_i \in (\mathrm{Set}_{\mathrm{predecessor}}(\mathrm{stt}_{\mathrm{rwd}}) \cap \mathrm{SE})$ is in an exploring episode, then irrespective of the explore strategies adopted in the future, stt_i will always be part of SEE.

Proof: From lemma 1, one can see that if $\mathrm{stt}_i \in \mathrm{SE}$ has a successor state $\mathrm{stt}_{\mathrm{rwd}}$ and it is impossible for $\mathrm{stt}_{\mathrm{rwd}}$ to be a member of SE. By definition of SEE it is known that stt_i is always a member of SEE. (End of proof)

Theorem 1. Assume there are a finite number of states and a SSA(CEG) is able to find the shortest path in CEG, exploration becomes finding a path from the start state stt_0 to the $\mathrm{stt}_{\mathrm{rwd}}$. In other words, exploration converges to exploitation.

Proof:

i During exploration, state stt_0 can reach the SEE through SSA(CEG). That is, one needs to find the shortest path, path k, among all the paths, such that:

$$\mathrm{k} = \underset{\mathrm{j}}{\operatorname{argmin}}\{\mathrm{L}(\mathrm{stt}_0, \mathrm{stt}_\mathrm{j}) : \mathrm{stt}_\mathrm{j} \in \mathrm{SEE}\} \tag{13}$$

Where $\mathrm{L}(\mathrm{stt}_i, \mathrm{stt}_j)$ is the length of the shortest path between state stt_i and state stt_j. If $\mathrm{stt}_k \in \mathrm{Set}_{\mathrm{predecessor}}(\mathrm{stt}_{\mathrm{rwd}})$, then $(\mathrm{stt}_0, \ldots, \mathrm{stt}_k, \ldots, \mathrm{stt}_{\mathrm{rwd}})$ from SSA(CEG) algorithm marks the shortest path from stt_0 to $\mathrm{stt}_{\mathrm{rwd}}$. In this case, the conditions concerning exploration convergence (defined in Definition 3) are met, and the exploration converges to the exploitation.

ii If $\mathrm{stt}_k \notin \mathrm{Set}_{\mathrm{predecessor}}(\mathrm{stt}_{\mathrm{rwd}})$, CEG continues to evolve as exploration progresses.

iii As exploration continues, new members are added into SEE and they replace the old ones, extending the shortest path, and according to Corollary 2, any new member $\mathrm{stt}_i \in (\mathrm{Set}_{\mathrm{predecessor}}(\mathrm{stt}_{\mathrm{rwd}}) \cap \mathrm{SE})$ will always be part of SEE.

iv When exploration ends, stt_k that satisfies Eq. (4) will eventually meet the condition: $stt_i \in Set_{predecessor}(stt_{rwd})$.

v The agent is bounded to pass the state stt_k associated with the shortest path in the $Set_{predecessor}(stt_{rwd})$. If not, there would have a different $stt_{ks} \in Set_{predecessor}(stt_{rwd})$ from stt_k that otherwise makes $L(stt_0, stt_{ks}) < L(stt_0, stt_k)$. If $stt_{ks} \in SEE$, it is impossible for SSA(CEG) to choose stt_k as a state in the shortest path. If $stt_{ks} \in SEE$, there must be a state $stt_m \in SEE$ in stt_{ks} 's predecessor chain that makes $L(stt_0, stt_m) < L(stt_0, stt_k)$. The algorithm does not converge during this episode.

vi Putting all things together, one can see that exploration by SSA(CEG) must converge to the shortest path from start stt_0 to stt_{rwd}.

As indicted in corollary 3, once the algorithm has found the shortest path from stt_0 to stt_{rwd}, the path will be repeated with no change in the following episodes. In this case, the exploration is readily to be halted. (End of proof)

2.5 Algorithm Implementation

Based on Theorem 1 described in the previous section, a framework for reinforcement learning is proposed that addresses the exploration and exploitation dilemma. The framework consists of two major components: CEG and incompletely explored states. It operates through two iterative steps, as illustrated in Fig. 3.

i Based on the current CEG, an action decision, in the form of a single decision or a chain of multiple decisions, will be made to guide the next exploration.

ii Update the CEG with the new knowledge acquired from the latest exploration. In a static or nearly static environment, exploration will help continue to grow CEG, while in a changing environment, CEG members can be added or deleted according to the exploration result. Note that when the CEG is updated, nodes or edges can be added or deleted from the graph. In a static environment, as the exploration progresses, the number of nodes and edges tends to increase, while in a dynamic environment, the number of nodes and edges may increase or decrease.

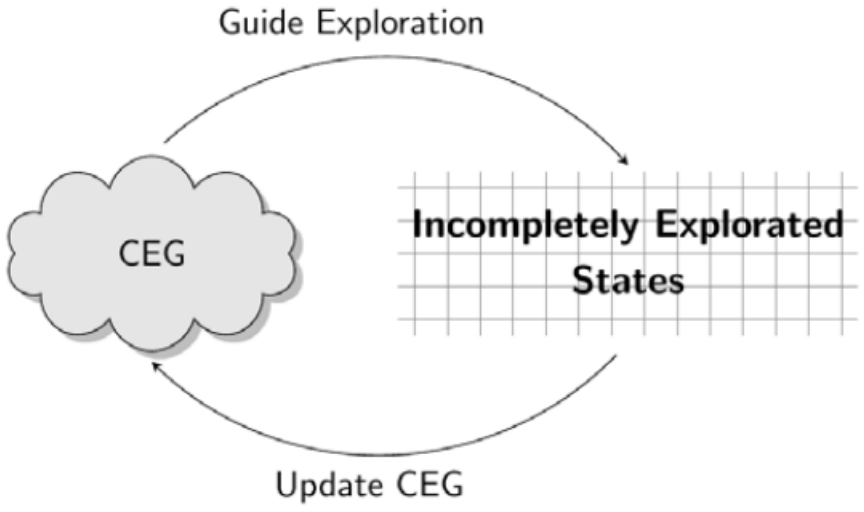

Figure 20 Graph iterative frame

2.5.1 Shortest Path Search in Dynamic Environment

The standard Floyd-Warshall Algorithm calculates matrices DIST and NEXT in batch divided by the length of short path (the number of relay vertices here) for each vertex pair. For constantly changing of vertices and graph structure that engages in exploration all the time, a more efficient method is needed and thus proposed in this section.

During exploration in reinforcement learning, the completely explored states are discovered in sequence, and subsequently, they are added to the CEG, after which the corresponding edges are also added. In addition, if the agent wants to adapt to the dynamic environment, the removal of vertex must also be taken into account. In this section, a modified FW algorithm is presented, which is able to search for the shortest path in a graph that represents a dynamic environment.

In SFW, each time when a new vertex is added, it is not only to add the shortest path associated with the new vertex directly tied to the two matrices as defined in FW, but also to compare the length of the new path introduced by the new vertex against that of the shortest path obtained from the prior iteration. These operations may result in the update of the two matrices. The major steps of SFW are summarized follow:

If current state stt is to be added to set SE, do the following steps:

i Add and initialize a new row in matrices DIST and NEXT.
ii Add and initialize a new column in matrices DIST and NEXT.
iii Update the new column by computing the shortest paths from all the vertices to this new vertex.
iv Update the new row by computing the shortest paths from the new vertex to all the other vertices.
v Update matrices DIST and NEXT by comparing the length of each vertices pair between

the old shortest path recorded in the matrices and that of the new paths with the new vertex added.

Denote $L_{set}(stt_i, Set_{predecessor}(stt))$ as the length of the shortest path from arbitrary state stt_i to $Set_{predecessor}(stt)$ as defined:

$$L_{set}\left(stt_i, Set_{predecessor}(stt)\right) = \min_j\{L(stt_i, stt_j): stt_j \in Set_{predecessor}(stt)\}$$

Matrices DIST and NEXT are updated by performing the following operations:

$$DIST(stt_i, stt) = L_{set}\left(stt_i, Set_{predecessor}(stt)\right) + 1$$
$$NEXT(stt_i, stt) = stt_{j_{min}}$$

where j_{min} is the result of $\min_j\{L(stt_i, stt_j): stt_j \in Set_{predecessor}(stt)\}$.

In the same token, one can update the stt's predecessor states set $Set_{predecessor}(stt)$ with stt's successor states set $Set_{successor}(stt)$. That is:

$$L_{set}(Set_{successor}(stt), stt_i) = \min_j\{L(stt_j, stt_i): stt_j \in Set_{successor}(stt)\}$$
$$DIST(stt, stt_i) = L_{set}(Set_{successor}(stt), stt_i) + 1$$
$$NEXT(stt, stt_i) = stt_{j_{min}}$$

Correspondingly, matrix DIST in this case can be updated by

$$DIST(stt, stt_i) \leftarrow \min\{DIST(stt_i, stt_j), DIST(stt_i, stt) + DIST(stt, stt_j)\}$$

If a path that passes through state stt is shorter than the previously obtained shortest path, then matrix NEXT is updated by:

$$NEXT(stt_i, stt_j) = stt$$

2.5.2 Guided Exploration

As proven, exploration finally converges to the shortest path that connects with the target state. Since exploration and exploitation basically produce the same result, the proposed algorithm only needs to consider one single task, exploration.

Table 1 Exploration algorithm

01	Set the current state to stt, if $\exists a_i \in Actions \Rightarrow stt_{rwd} \equiv Env(stt, a_i)$:
02	Return $\{a_i\}$
03	If $stt \notin SE$:
04	$A_{s_{uxpl}}(stt) = \{a: Env(stt, a)$ is unexplored, $a \in Actions\}$

```
    If A_{s_uxpl}(stt) ≠ ∅:
        Choose {a_k} randomly from A_{s_uxpl}(stt)
    Else:
        Choose {a_k} randomly from Actions
Else:
    If stt ∈ SEE:
        As_outside(stt) = {a: Env(stt, a) ∉ SE, a ∈ Actions}
        Choose {a_k} randomly from As_outside(stt)
    Else:
        Get the nearest edge state stten by SFW from current state stt
        Build the shortest actions chain As_chain by SFW from stt to stten
        Get As_outside(stten)
        Choose a_k randomly from s_outside(stten)
        As_chain ← As_chain ∪ (a_k)
        Return As_chain
Return {a_k}
```

There are several major steps in the algorithm listed in Table 1:

Step 1: If stt is a neighbor of $\mathrm{stt_{rwd}}$, the agent can take action $\mathrm{a_i}$ directly, which transitions the state to $\mathrm{stt_{rwd}}$.
Step 2: If stt is not a member of SE, a decision is randomly made by calling $\mathrm{As_{uxpl}(stt)}$.
Step 3: If stt is not a member of SE, a decision is randomly made by calling $\mathrm{As_{outside}(stt)}$.
Step 4: If stt is a member of SE but not a member of SEE, the shortest path is obtained using SFW.

This path represents the decision chain by which the agent can exit SE in the most efficient way.

2.5.3 Update of the CEG

Before exploration starts, SE is empty, and the agent has no a priori knowledge of the environment. Denote $\mathrm{stt_0}$ as the start state of each exploration episode. Once exploration begins, from the initial state $\mathrm{stt_0}$, for each successor state $\mathrm{stt_i}$, an action $\mathrm{a_k}$ is selected from the actions set according to SFW, after which the agent moves to the next state $\mathrm{stt_{i+1}}$ by taking action $\mathrm{a_k}$ obtained from the feedback of the environment.

Exploration gets repeated. Whenever a new state is found, it will be added to the graph GU immediately. When the current state is completely explored, it will be added to set SE, sometimes to the SEE simultaneously. This algorithm is listed in Table 2. One can see that when a new completely explored state, corresponding to a vertex in the graph can be added to the CEG, it must generate some action decision reflected as edge changes in the graph. The new SEE by definition can be readily derived from the updated CEG.

Table 2 Algorithm: update CEG

```
Initialize: SE = ∅
Repeat:
```

03 Agent takes action a_k in state stt_i
04 Get next state stt_{i+1} from environment
05 If $(stt_i, stt_j) \notin E(GU)$:
06 $E(GU) \leftarrow E(GU) \cup \{(stt_i, stt_{i+1})\}$
07 If $\{(stt_i, stt_j): stt_j = Env(stt_i, a_k), a_k \in Actions \subset E(GU)$:
08 If $stt_i \notin SE$:
09 $SE \leftarrow SE \cup \{stt_i\}$
10 Update CEG
11 Update SEE

2.6 Experimental Results

The new graph-based algorithm detailed has been applied to solve a maze. Maze solving has been widely adopted for the testing of RL algorithms. The agent in the experiment can be seen as a ground robot roaming in a maze, and it can always sense its current position (state) as it moves around. At the beginning of an experiment, the agent knows nothing about the maze, and it needs to find the reward (target) position and complete its journey by passing through a path from a specified starting position.

2.6.1 Setup of Experiment

The maze has a size of 16 rows by 16 columns for a total of 256 blocks. There are 4 types of blocks, namely target, trap, obstacle, and ordinary pass. The fixed start position is treated as a normal pass block. The agent gets a reward of 1 when it researches the target, but if the agent falls into a trap, it will get a reward of -1. Both conditions will lead to the finish of the current episode, and the agent will have to return to the start position and restart its exploration. Note that the agent can keep the exploration information from all the previous episodes. There are 975 mazes in the experiment, and they differ from each other in terms of the locations of the obstacle blocks. In our experiment, there are 46 obstacles for each of the 975 mazes.

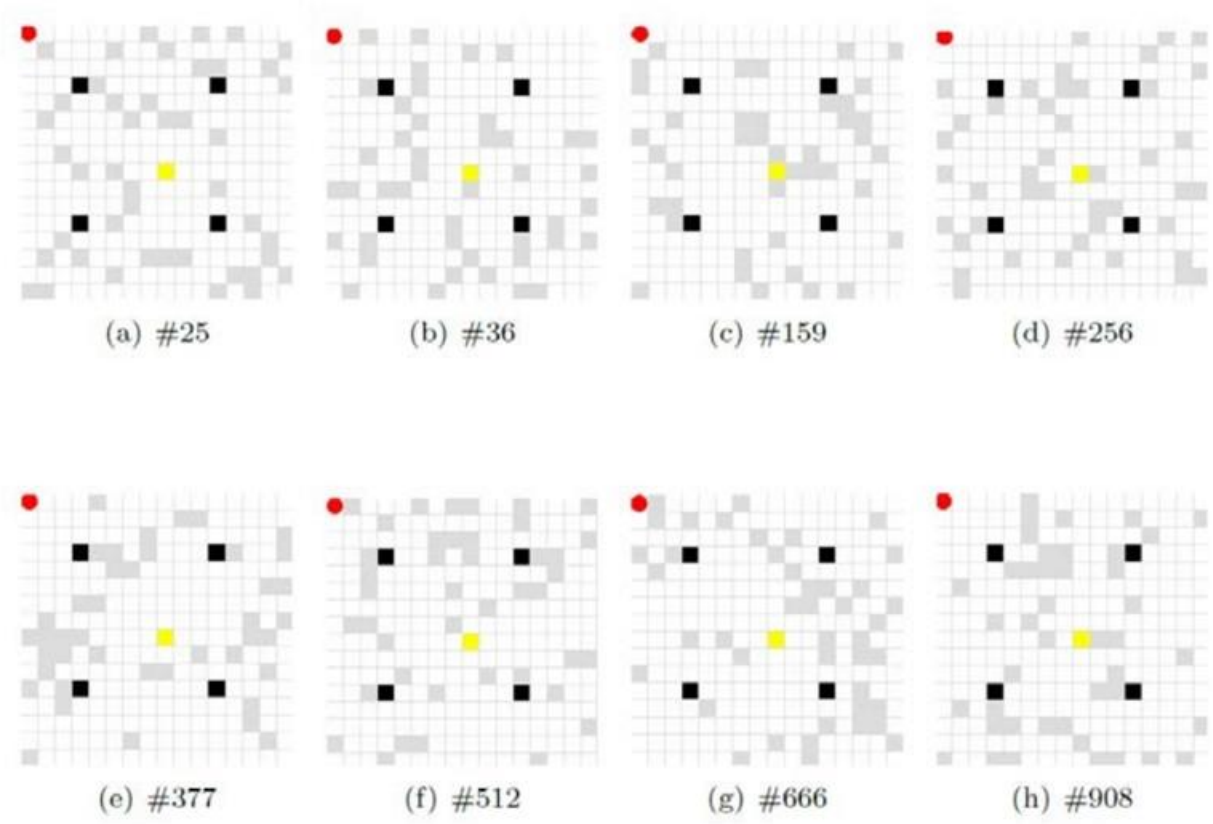

Figure 21 Maze settings

For each maze, the initial position of the agent is at the upper left corner (1, 1), and the target position is set to be (9, 9). There are 4 fixed traps, located at (4, 4), (12, 4), (4, 12), and (12, 12). Table 3 summarizes the main characteristics of the maze.
Figure *21* illustrates a sample of mazes. In these mazes, the red circle represents the agent, the gray blocks represent obstacles, the black blocks represent the traps, and the rest are normal pass blocks.

Table 3 Maze design parameter

Parameter	**Value**
Map Size	16×16
Map Amount	975
Target Amount	1
Trap Amount	4
Rate of Obstacles	0.18
Total Number of Episodes	100

The proposed algorithm, referred as SFW, is compared against the classical Q-learning algorithm (ql) and an improved Q-learning algorithm (qlm). Table 4 tabulates the main parameter values for ql. The main improvement of qlm over ql is that qlm can remember the locations of the obstacles and traps found during exploration., and avoid them during the subsequent explorations. Even if

the next action is randomly selected based on some probability, qlm can filter out the obstacles and traps.

Table 4 Q-learning parameter

Parameter	Value
Learning Rate (α)	0.01
Discount (γ)	0.9
ϵ-greedy	0.9

2.6.2 Performance Comparison with Q-learning Algorithms

2.6.2.1 Single Maze Comparison

All three algorithms are compared in terms of number of steps per episode when they are applied to solve all the mazes, and the results from maze 25 and 908 are plotted in
Figure *22* and
Figure *23*, respectively. In solving both mazes, SFW is found to converge quickly than the other two algorithms, and it requires less number of steps during the exploratory process. As expected, the performance of qlm is better than that of classical ql.

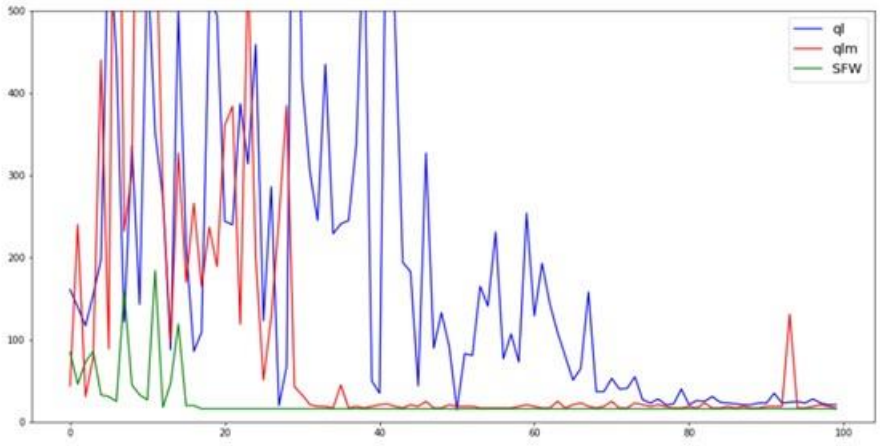

Figure 22 The steps amount in the #25 maze per episode comparison. The X axis is the episode number, and the Y axis represents the number of steps in each episode

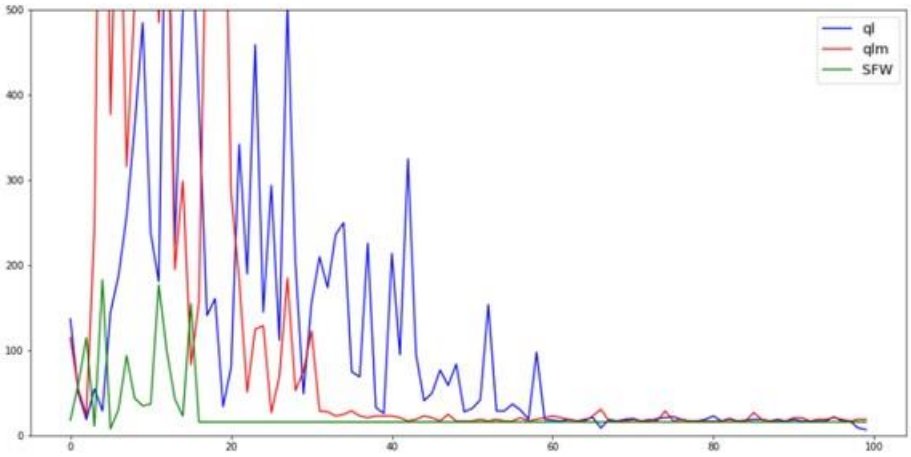

Figure 23 The number of steps in the #908 maze per episode

All three algorithms are compared in terms of average exploration efficiency of each step in every episode when they are applied to solve all the mazes, and again, the results from maze 25 and 908 are plotted in
Figure *24* and Figure 25, respectively. One can see that SFW is more efficient in exploration and converges faster than the other two algorithms.

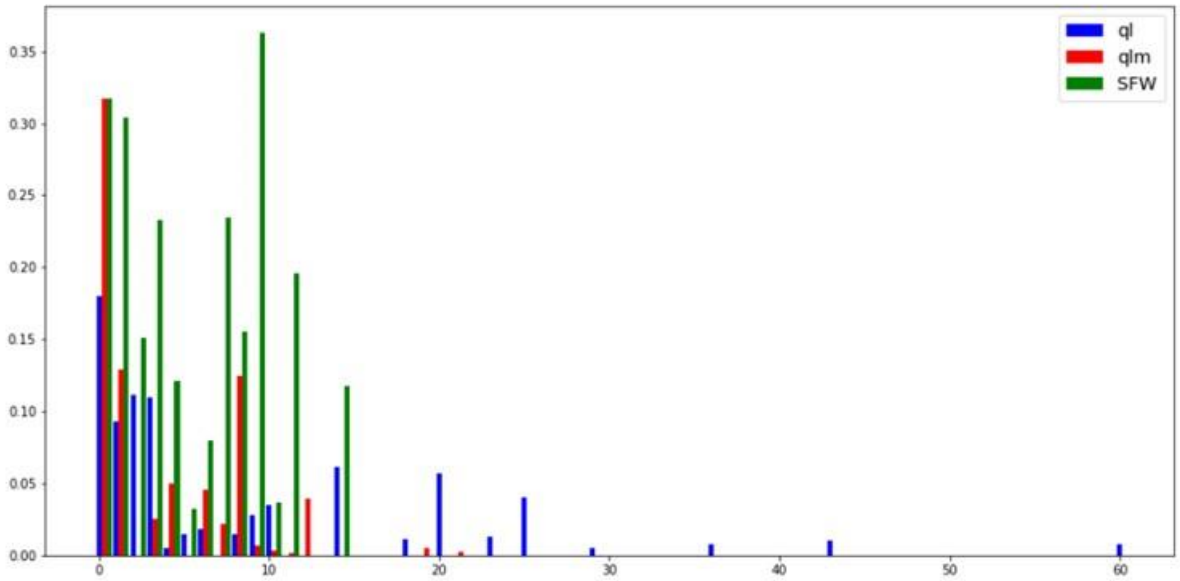

Figure 24 Average exploration efficiency while solving maze 25

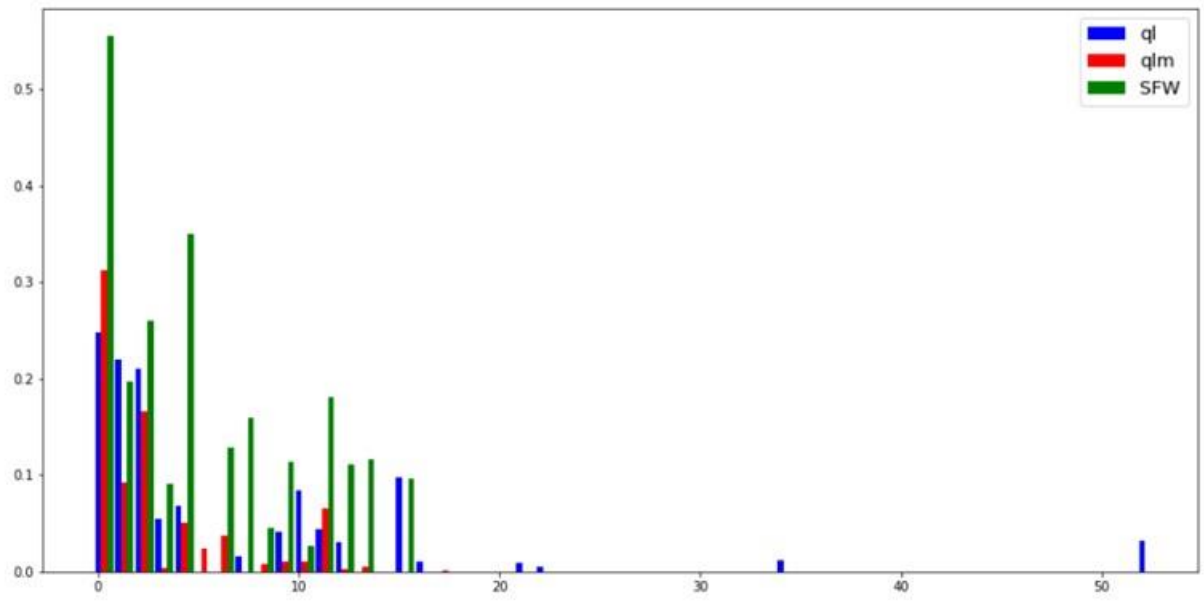

Figure 25 Average exploration efficiency while solving maze 908

2.6.2.2 Statistical Performance Comparisons for All Mazes

The experiments in this section include all 975 mazes. The X axis of each figure corresponds to the maze number.

The comparison of convergence speed of every maze is shown in Figure 26 and Figure 27, where ql and qlm are compared with SFW separately. The Y axis of each figure is the number of episodes when the agent for the first time arrives at convergence. In both figures, one can see that the proposed algorithm converges more quickly than the other two algorithms, especially true when there are a large number of episodes. Actually, the SFW algorithm converges after no more than 20 episodes, while the other two need as many as 100+ episodes.

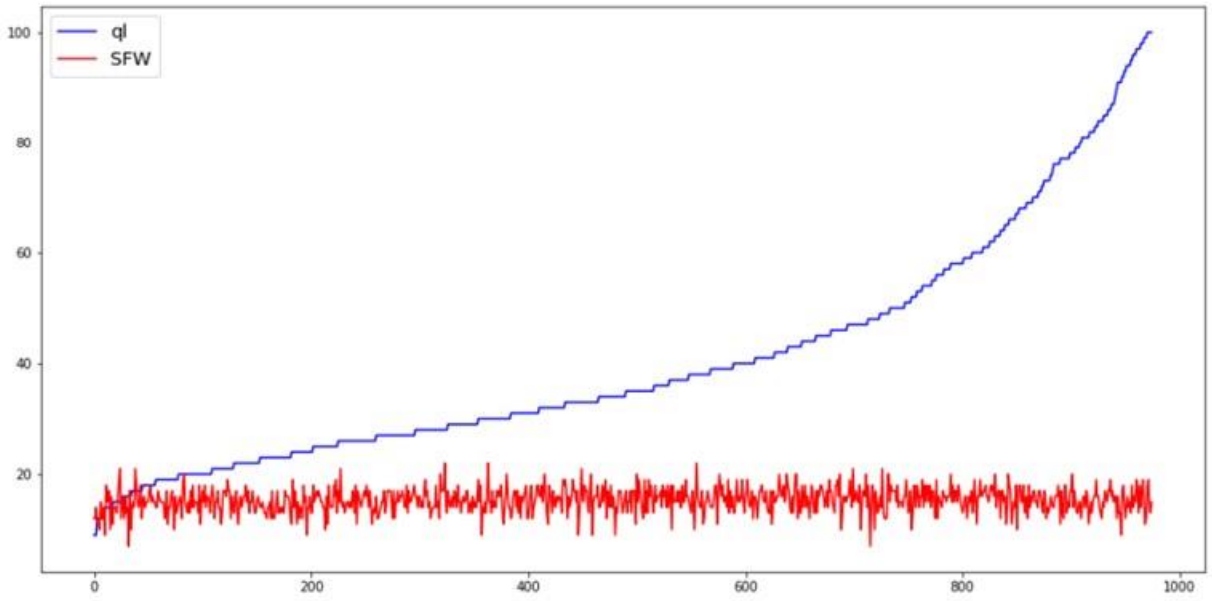

Figure 26 Convergence speed comparison: ql and SFW

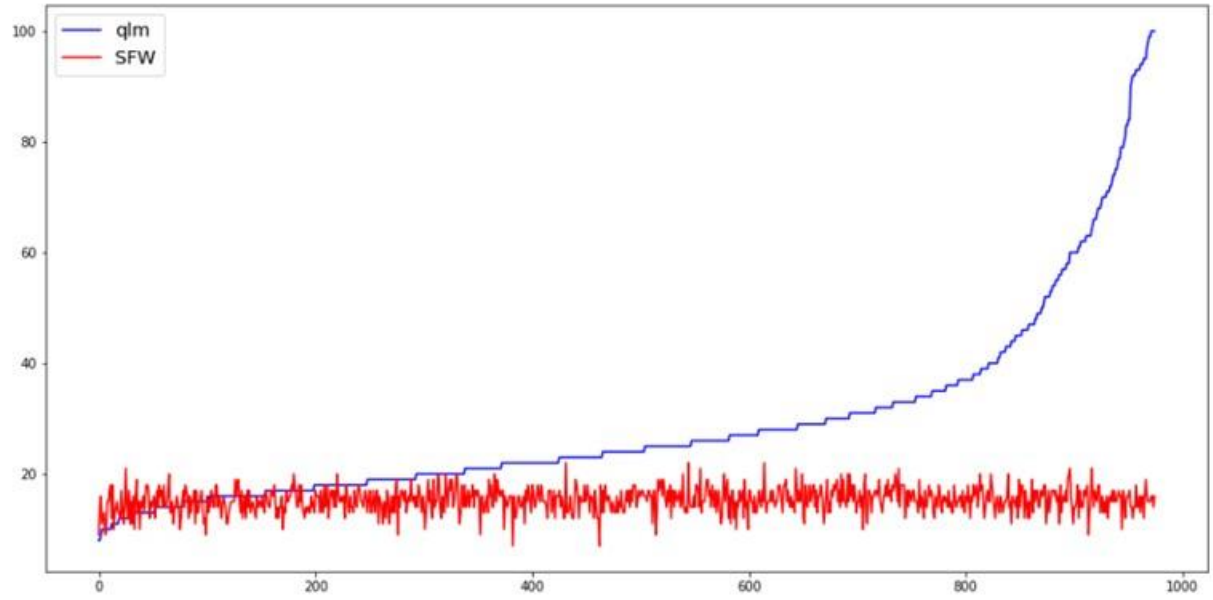

Figure 27 Convergence speed comparison: qlm and SFW

The lengths of the finally discovered paths by are reported in Figure 28 and Figure 29. One can see that the paths found by SFW have shorter lengths, in the range of 15 to 20, while the paths found by the other two algorithms are much longer. In quite a few cases, the paths found by ql and qlm are twice longer than those found by SFW.

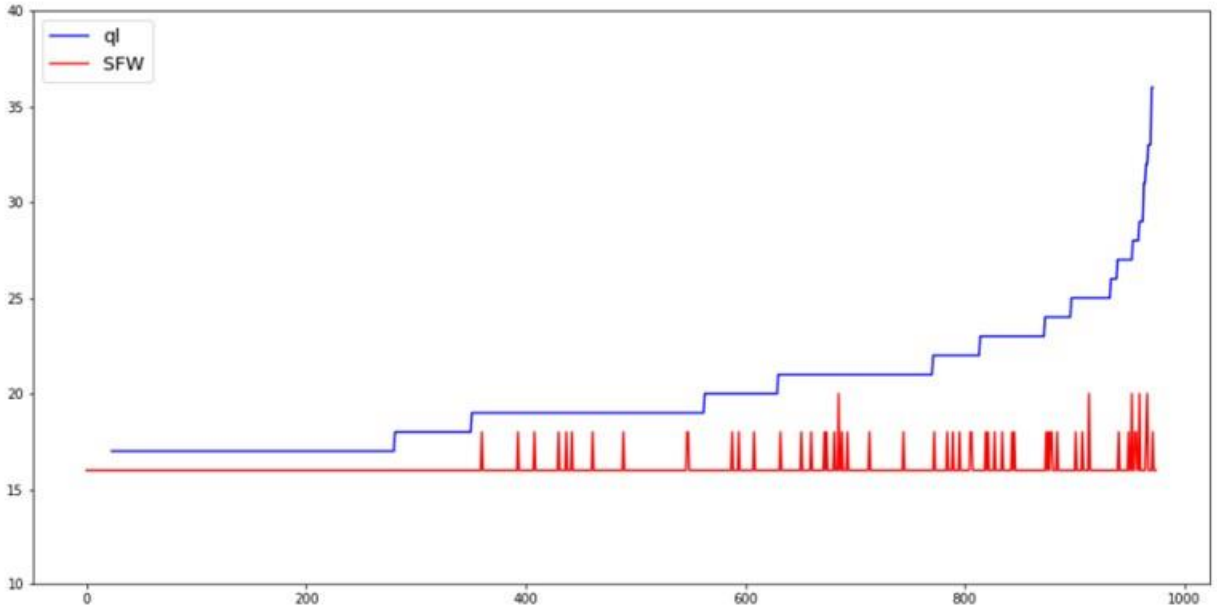

Figure 28 Steps length of convergence comparison: ql and SFW

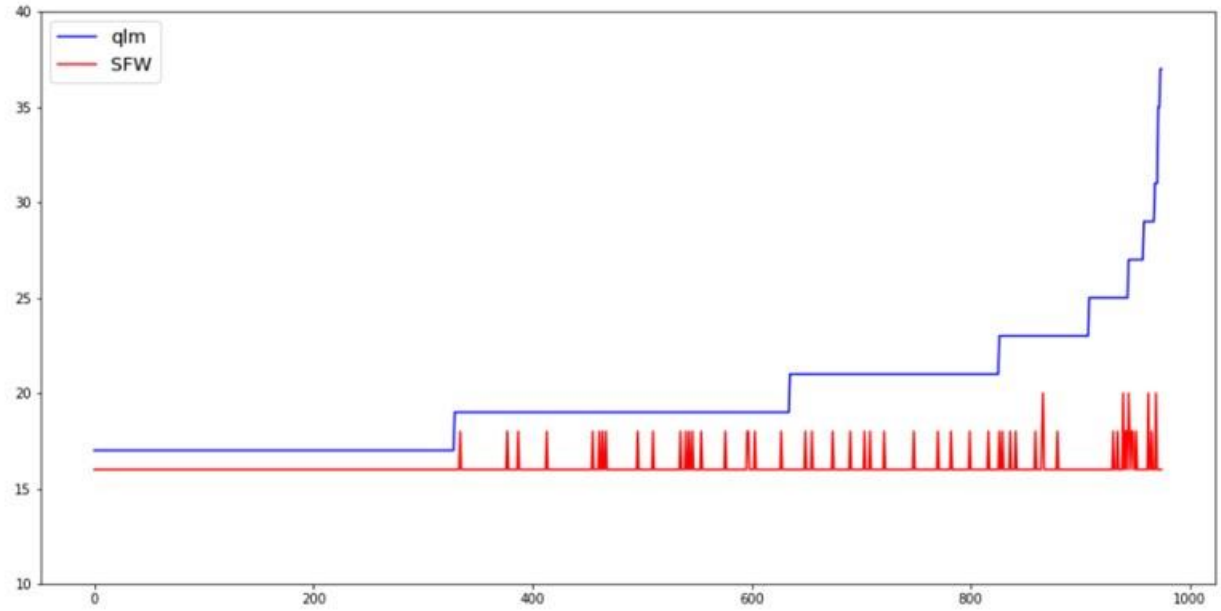

Figure 29 Steps length of convergence steps comparison: qlm and SFW

The exploration efficiency obtained from solving every maze is shown in Figure 30, One can see SFW outperforms qlm in this regard, and both algorithms are significantly better than ql. The X axis represents the maze number. The Y axis is the ratio of the total number of explored states to the total number of steps when the agent for the first time arrives at convergence.

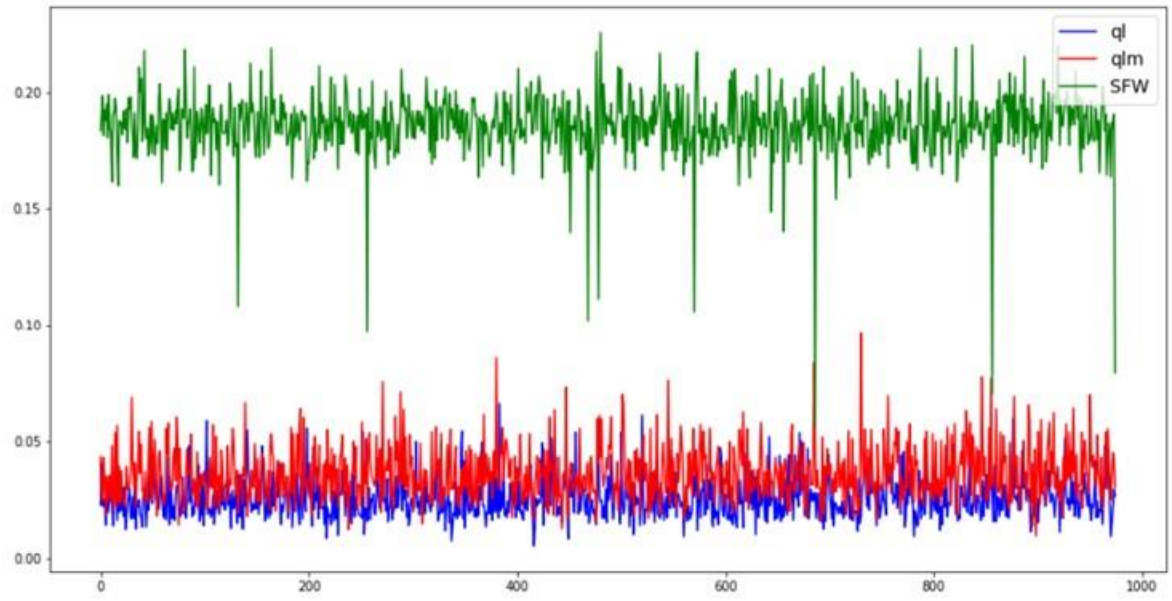

Figure 30 Explore efficiency comparison

2.6.3 The Maze in the Dynamic Environment

Changes of environment are categorized as obstacle change and target position change. In this subsection, the focus will be on examining how these changes may impact the performance of the three algorithms.

2.6.3.1 Obstacle Change

Take maze #8 as an example, the changes of the obstacles are tabulated in Table 5.

Table 5 Dynamic obstacles, maze #8

Change episode	Change State	Convergence Episode #	The Corresponding Figure
0	Init	16	Figure 31 (a) and (c)
18	(2,8)	18	Figure 31 (d)
24	(3,6), (3,7)	25	Figure 31 (e) and (f)
29	(9,8), (10,8)	34	Figure 31 (g) and (h)

Figure 31 shows the snapshot of exploration, convergence, environment change and adaptation. The maze has undergone three major changes that occur to the locations of the obstacles in the experiments. The green square in Figure 31 represent the member of SE, and the purple squares represent dynamically increased obstacles that are located within the current convergent path.

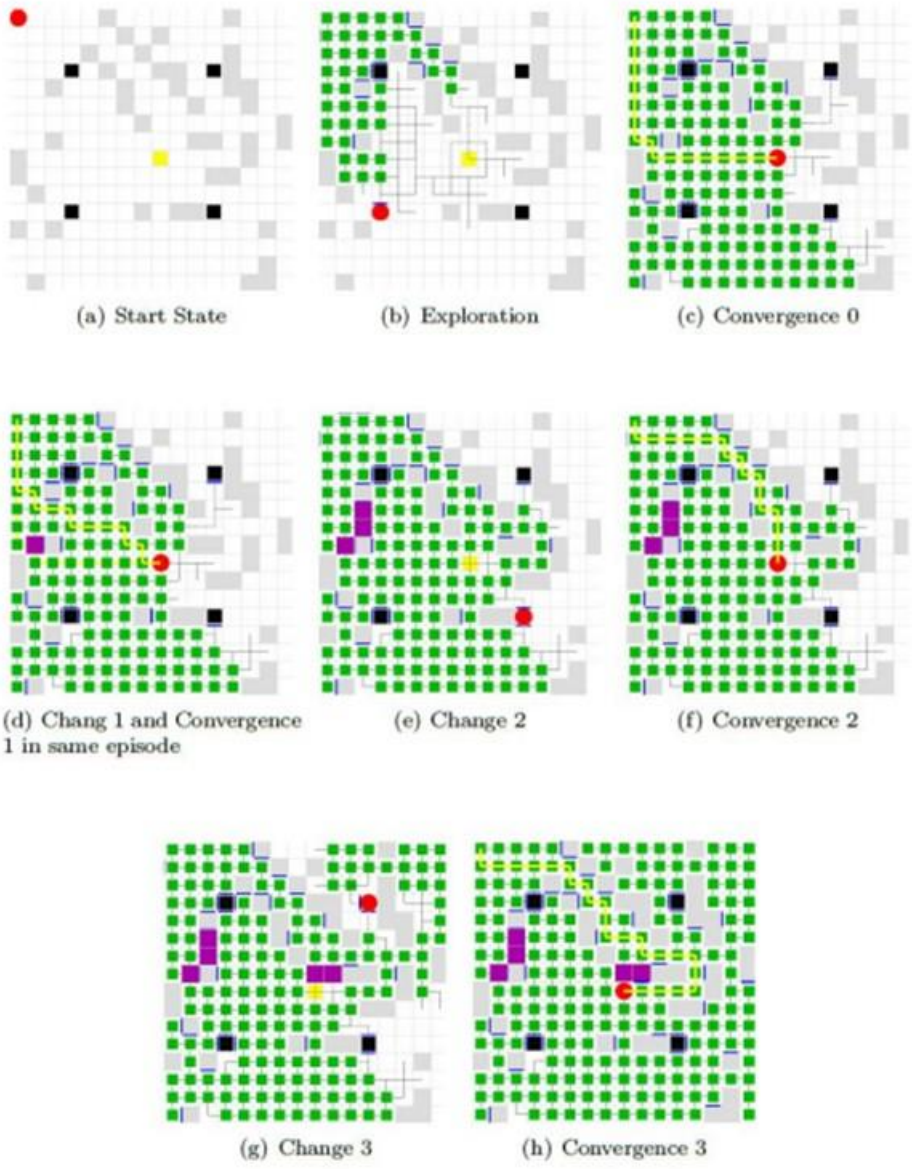

Figure 31 Dynamic obstacles, maze #8

2.6.3.2 Changes of Target Positions

Table 6 summarizes the changes that occur in maze 243. Other mazes have gone through similar changes. One can see that the target position is changed once, relocated from the center of the maze to its lower left corner.

Table 6 Dynamic target, maze #243

Change episode	Change State	Convergence Episode #	The Corresponding Figure

0	Init	14	Figure 31 (a) and (c)
18	(9,9)⇒(1,16)	21	Figure 31 (d)

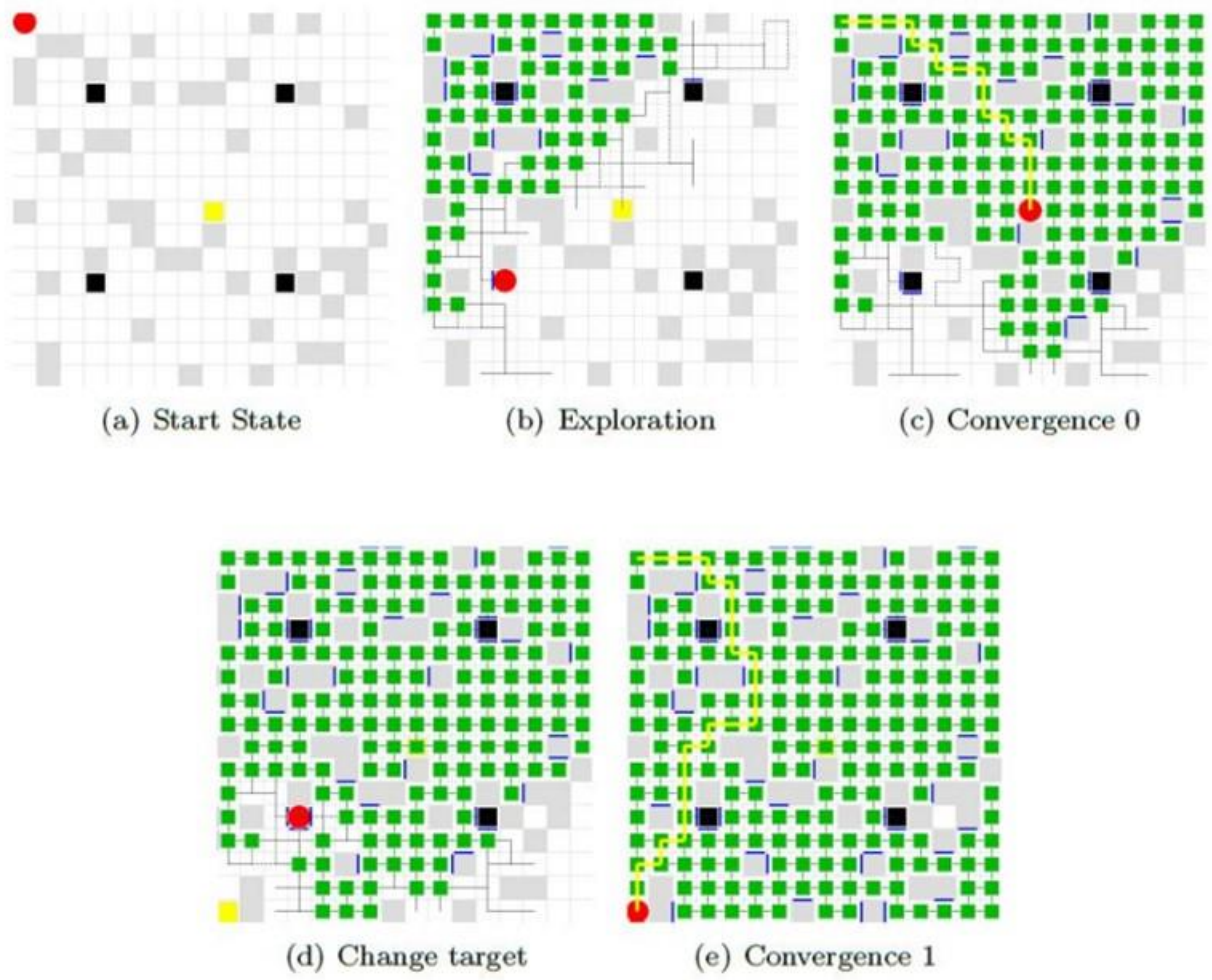

Figure 32 Dynamic target, maze #243

The results of exploration, convergence due to target position changes and re-convergence are shown in Figure 32. In episode 0, the reward is claimed at block (9, 9) (Figure 32 (a)). The exploration converges in episode 14 (Figure 32 (c)). In episode 18 where the reward is moved to block (1, 16) (Figure 32 (d)), the agent finds the right path to the target, after three episodes (Figure 32 (e)).

2.6.4 Computation Efficiency

All three algorithms are compared for their respective computation efficiency under the same computation platform. The hardware used in the experiments has an Intel® Core™ i5-3210M CPU running at 2.5 GHz, and a RAM size of 8 GB. The operating system is Ubuntu 64 bits. The tools used to test CPU time and memory occupation are line_profiler and memory_profiler, respectively. The average CPU time reported in Table 7 is the average time of solving all 975 mazes. The basic

memory usage in Table 7 refers to the stable memory usage collected from solving select 62 mazes. One can se that SFW requires more memory space than the other two algorithms; the memory usage for both ql and qlm is comparable. The peak memory usage of SFW is also higher than that of ql or qlm.

Table 7 Algorithm complexity comparison

Parameter	ql	qlm	SFW
Average CPU Time (s)	0.5382	0.8472	4.9307
Basic Memory Usage (MB)	60.634	60.352	70.148
Peak Memory Usage (MB)	61.024	61.500	74.135

2.7 Conclusion

In this phase, the authors presented a new graph-based reinforcement learning method for optimal UAV path planning. Unlike classical Q-learning algorithm and improved Q-learning algorithm, the proposed algorithm does not struggle with the exploration vs. exploitation tradeoff, as it was proved that the two tasks of exploration and exploitation actually converge in the decision-making process. As so, the proposed graph-based algorithm finds the shortest path during exploration, which gives higher efficiency and faster convergence than the Q-learning algorithm and its variant. Another big advantage of the proposed algorithm is that it can be applied to the dynamic environment where the value-oriented algorithm fails to work. The efficiency and convergence performance of the proposed algorithm comes at a cost of increased computational complexity.

2.8 References

1. Dayan, P., & Hinton, G. E. (1992). Feudal reinforcement learning. Advances in neural information processing systems, 5.
2. Dietterich, T. G. (1998, July). The MAXQ Method for Hierarchical Reinforcement Learning. In ICML (Vol. 98, pp. 118-126).
3. Dijkstra, E. W. (2022). A note on two problems in connexion with graphs. In Edsger Wybe Dijkstra: his life, work, and legacy (pp. 287-290).
4. Floyd, R. W. (1962). Algorithm 97: shortest path. Communications of the ACM, 5(6), 345-345.
5. Hart, P. E., Nilsson, N. J., & Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. IEEE transactions on Systems Science and Cybernetics, 4(2), 100-107.
6. Huang, B. Q., Cao, G. Y., & Guo, M. (2005, August). Reinforcement learning neural network to the problem of autonomous mobile robot obstacle avoidance. In 2005 International conference on machine learning and cybernetics (Vol. 1, pp. 85-89). IEEE.
7. Kaelbling, L. P., Littman, M. L., & Moore, A. W. (1996). Reinforcement learning: A survey. Journal of artificial intelligence research, 4, 237-285.
8. Lewis, F. L., Vrabie, D., & Vamvoudakis, K. G. (2012). Reinforcement learning and feedback control: Using natural decision methods to design optimal adaptive controllers. IEEE Control Systems Magazine, 32(6), 76-105.
9. Lewis, F. L., & Vrabie, D. (2009). Reinforcement learning and adaptive dynamic programming for feedback control. IEEE circuits and systems magazine, 9(3), 32-50.

10. Matarić, M. J. (1997). Reinforcement learning in the multi-robot domain. Robot colonies, 73-83.
11. Moore, A. (1993). The parti-game algorithm for variable resolution reinforcement learning in multidimensional state-spaces. Advances in neural information processing systems, 6.
12. Smart, W. D., & Kaelbling, L. P. (2002, May). Effective reinforcement learning for mobile robots. In Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292) (Vol. 4, pp. 3404-3410). IEEE.
13. Sutton, R. S., Barto, A. G., & Williams, R. J. (1992). Reinforcement learning is direct adaptive optimal control. IEEE control systems magazine, 12(2), 19-22.
14. Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., ... & Hassabis, D. (2017). Mastering chess and shogi by self-play with a general reinforcement learning algorithm. arXiv preprint arXiv:1712.01815.
15. Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., ... & Hassabis, D. (2017). Mastering the game of go without human knowledge. nature, 550(7676), 354-359.
16. Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., ... & Hassabis, D. (2016). Mastering the game of Go with deep neural networks and tree search. nature, 529(7587), 484-489.
17. Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction. MIT press.

3 UAV-LIDAR-BASED GEOMETRY MEASUREMENT SYSTEM

3.1 Overview

After determining the optimal flight path for the UAV, the study was focused on assembling a platform that included sensors, processors, and other necessary components for geometry measurement.

3.2 Methodology

3.2.1 Data collection platform

The platform hardware (Figure 33) includes a DJI Matrix 600 as the carrier, an Ouster OS-1 LiDAR for point cloud data generation, a 3DM-GQ7 IMU for flight status tracing, and an Nvidia Jetson Xavier Development Kit (or Jetson in short) for on-board data acquisition and processing.

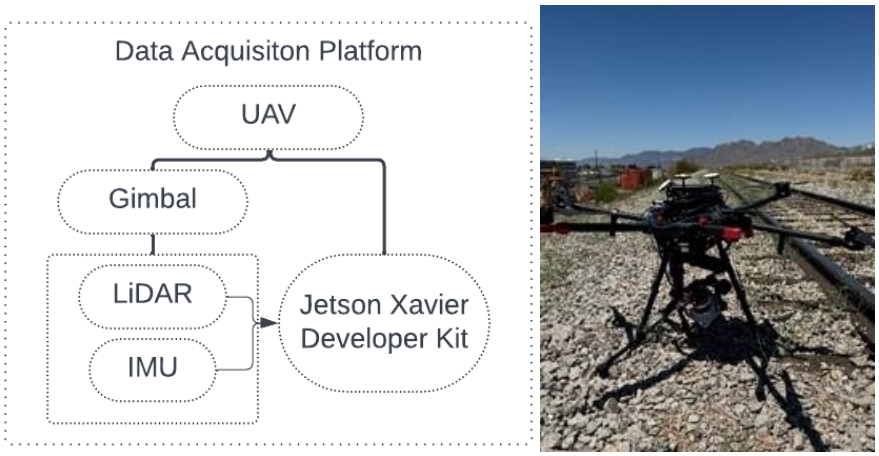

Figure 33 Left: Data collection prototype description. Right: Data collection hardware at the test site.

The UAV spans a dimension of 65 inches × 60 inches × 30 inches after propellers, frame arms, and GPS mount are unfolded. It carries the LiDAR with its gimbal mount, the IMU, and the Jetson (a total payload of about 11 lbs) for a continuous 20 minutes' flight. The maximum takeoff weight is approximately 33 lbs.

The Ouster OS-1 time-of-flight (ToF) LiDAR scans with a 360° rotational field of view (FOV) horizontally and emits 128 laser channels covering 45°FOV vertically (key specifications in TABLE 8). This LiDAR outputs the point cloud data in 3D Cartesian coordinates (x_L, y_L, z_L) that represent the surfaces of surround objects and their corresponding reflection intensity measurements, each linked to a channel angel, corresponding channel ID ranging from 1 to 128, an azimuth value spanning from 0° to 360°, and timestamps. This sensor can measure distance up to 170 m with a range resolution of 0.1 cm. Both of its vertical and horizontal angular sampling accuracy is 0.01°.

TABLE 8 Ouster OS-1 LiDAR key specifications.

Specifications	Values
Range	295 feet
Minimum Range	1.6 feet
Vertical Resolution	32, 64, or 128 points
Horizontal resolution	512, 1,024, or 2,048 points
Rotation rate	10 or 20 Hz
Vertical FOV	45°
Angular sampling accuracy	±0.01°
False positive rate	1/10,000
Range resolution	0.04 inches

The IMU (key specifications presented in TABLE 9) is attached on top of the LiDAR so that it can accurately capture the instant dynamic posture of the LiDAR via high-speed accelerometers and gyroscopes. The IMU accelerometers measure linear acceleration along three orthogonal axes (x_I, y_I, z_I), while the gyroscopes measure the rotational speed (i.e., angular velocity) around each of the three axes.

TABLE 9 3DM-GQ7 IMU key specifications.

Parameter	Accelerometer	Gyroscope
Range	±8 g	±300°/s
Random walk	20μg/√Hz	0.15°/√h
Bias instability	5 μg	1.5°/h
Noise density	20μg/√Hz	8.75°/h/√Hz

Note that these two sensors do not share the same coordinate axis (see Figure 34), where the LiDAR x+ is the IMU x-, the LiDAR z+ is the IMU z-, the LiDAR pitch+ is the IMU roll+, and the LiDAR roll+ is the IMU pitch-. Therefore, the IMU coordinate system is projected to the LiDAR coordinate system using equations:

$$\begin{bmatrix} x_L \\ y_L \\ z_L \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} x_I \\ y_I \\ z_I \end{bmatrix} \tag{14}$$

$$\begin{bmatrix} R_L \\ P_L \\ Y_L \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} R_I \\ P_I \\ Y_I \end{bmatrix} \tag{15}$$

where x, y, and z stands x, y, and z-axis of each sensor, while R stands for roll, P stands for pitch, and Y stands for yaw, respectively. The subscript L and I stands for LiDAR coordinate and IMU coordinate respectively.

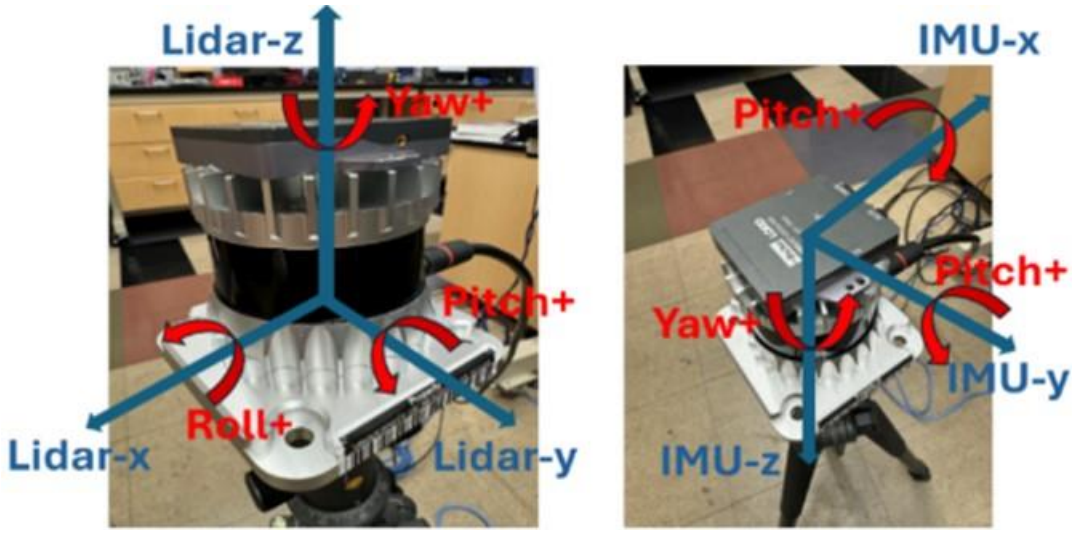

Figure 34 Coordinate axis difference between the LiDAR and the IMU.

In the end, all data from the LiDAR and the IMU are transmitted to the Jetson, which serves as the on-board data acquisition and processing unit because of its computational capacity for high-speed large data transmission, and its compatibility with ROS for automated data collection and processing.

During each data collection, the flight speed of the UAV was set at approximately 1 meter per second (m/s), and the altitude was set between 3-5 meters above the ground to ensure the point density on rails are acceptable. The UAV flew along three different flight paths: 1) in between the test tracks, 2) about 2 m north to the test track, and 3) about 2 m south to the test track. This ensured all three surfaces (i.e., top and sides) of the test tracks were scanned. The LiDAR scanned the rails at 10 Hz, with 1,024 vertical resolutions, and in single return mode. The IMU was configured to operate at 500 Hz. Both sensor data were streamed synchronously through ROS and stored on the Jetson.

3.2.2 Data Analysis/Software

Raw PCD frames include both rails (i.e., object of interests) and other surrounding objects that are of no interest. Therefore, to enable automated track geometry measurement, we need to semantically segment PCD frames to identify rail surfaces, and register small vertical FOV PCDs for large-scale curvature/profile measurements. Afterwards, outlier removal is performed to delete incorrectly predicted points from the segmentation before calculating geometries. General flow of the data processing procedure is displayed in Figure 35.

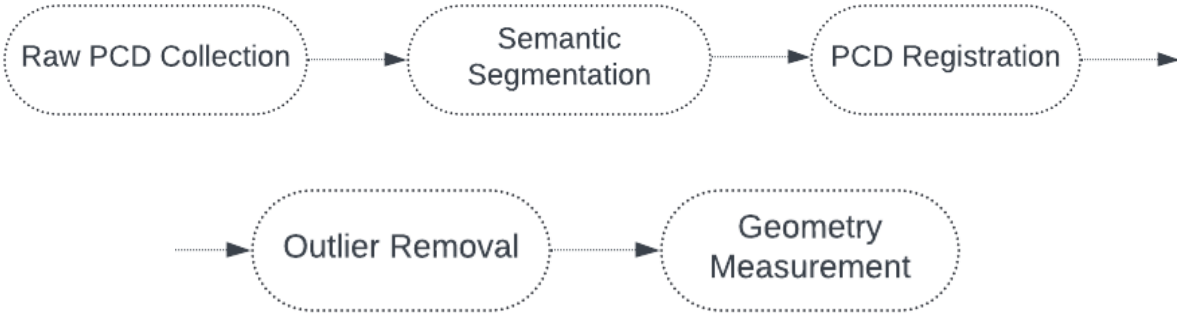

Figure 35 Flowchart of data processing.

3.2.2.1 PCD semantic segmentation

To realize automatic and accurate segmentation of railway point clouds for various numbers of scenarios, a supervised machine learning algorithm should be implemented. However, no labeled railway PCD is available for open access. Therefore, the first step is to annotate enough railway PCD to train a network. To do so, the original LiDAR data stream was decoded and saved as .pcd files for annotation. An open source PCD annotation platform, Supervisely was used to label rails in each frame (see Figure 36 (a)), where cuboids were positioned to minimally encapsulate points of interest. Since the rails are long and narrow, most of the points in a frame are background points. Therefore, each frame is cropped to include rails and nearby points to reduce the data size. Figure 36 (b) shows a labeled and cropped frame.

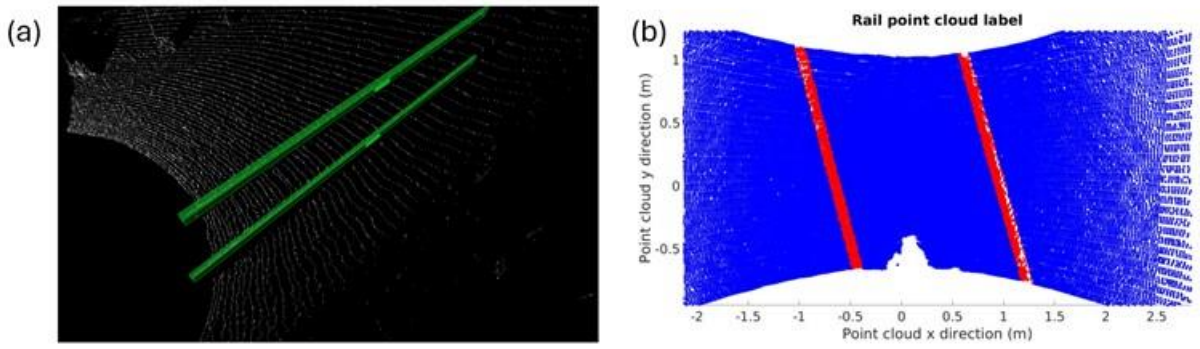

Figure 36 Supervisely annotation process. Cuboids are used to label rail points.

Machine learning requires enough training samples to obtain a good model. To increase the diversity of training samples, data augmentation techniques such as rotation, translation, and noises were applied to the original labeled frames. Eventually, 358 PCD frames were manually labeled, and augmented to a total of 1322 frames for the entire rail PCD dataset. The dataset was further split with a 0.7:0.2:0.1 ratio for training, validation, and testing set, respectively.

A 3D semantic segmentation neural network is adopted and optimized (see **Error! Reference source not found.**) for automated segmentation of railroad points. The network follows an

encoder-decoder structure. In the encoding stage, random sampling is used to down sample points. The local spatial encoding (LocSE) modules, and the attentive pooling (AP) modules are used to aggregate point features to maintain information of points sampled out.

The LocSE module selects one point p_i with feature f_i in the point cloud and calculate its K neighboring points $[p_i^1, p_i^2, \dots, p_i^K]$ in terms of their relative point position information r_i^k:

$$r_i^k = \mathrm{MLP}(p_i \oplus p_i^k \oplus (p_i - p_i^k) \oplus ||p_i - p_i^k||) \tag{16}$$

where $k = [1,2,\dots,K]$, MLP stands for multi-layer perceptron, $\oplus$ stands for concatenation, and $||\cdot||$ calculates the Euclidean distance. The information r_i^k are then concatenated to the feature of point p_i (i.e., $\hat{f}_i^k = f_i \oplus r_i^k$), introducing a new set of neighboring features $\hat{F}_i = [\hat{f}_i^1, \hat{f}_i^2, \dots, \hat{f}_i^K]$.

The AP module aggregates the set of neighboring point features $\hat{F}_i$. The attention score s_i^k is computed through a shared MLP layer followed by softmax function:

$$s_i^k = \mathrm{softmax}(\mathrm{MLP}(\hat{f}_i^k, W)) \tag{17}$$

where W is the learnable weights for the shared MLP layers. The weighted features are then calculated as:

$$\tilde{f}_i = \sum_{k=1}^{K} (\hat{f}_i^k \cdot s_i^k) \tag{18}$$

In the decoder stage, up-sampling modules and multi-layer perceptron (MLP) are used to predict the label for each point in the original PCD frame. Since the number of points in the two classes (i.e., "rail" and "background") are severely imbalanced, where only 1-2 % of the total points are rails, a more aggressive down sampling rate was adopted to decrease the effect of background points when training the network.

The loss function used in this network is cross entropy with logits. The loss is calculated as:

$$\mathrm{loss}_i = -\sum_{j=0}^{1} y_{i,j} \cdot \ln p_{i,j} \tag{19}$$

where $y_{i,j}$ stands for the ground truth for the j th point in the i th frame, and p_{ij} stands for the prediction for the j th point in the i th frame. The final result is achieved by minimizing all losses. To further address the data imbalance issue, a heavier penalty is put on the rail points predicted as background points. The overall structure of the network structure used is shown in **Error! Reference source not found.**.

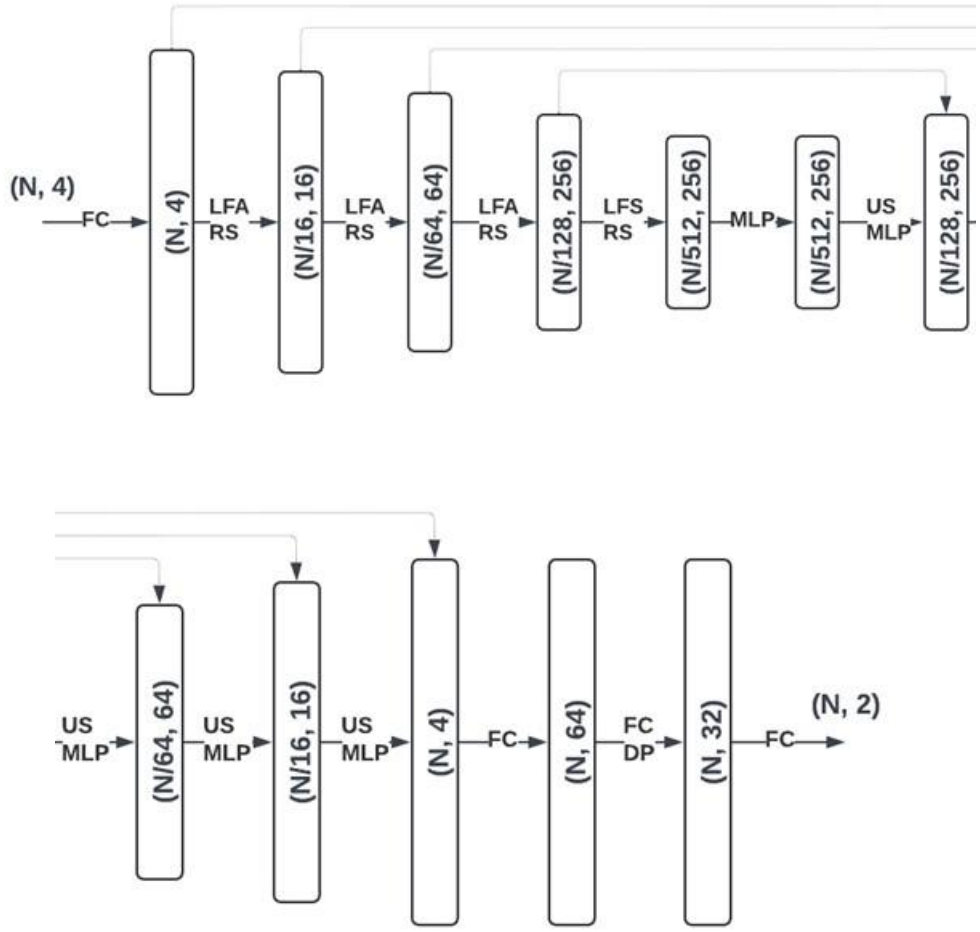

Figure 37 Network structure. (N, F) stands for N points and F dimension of features. FC: fully connected layer. LFA: local feature aggregation. LFA consists of multiple LocSE and AP modules. RS: random sampling. MLP: multi-layer perceptron. US: up-sampling. DP: dropout layer.

3.2.2.2 Point cloud registration

For rail curvature and profile that require the rail length no less than 62ft, the FOV of a single PCD frame does not suffice. Therefore, multiple PCD frames need to be aligned into a single coordinate system to generate a larger scene. To achieve this goal, a SLAM algorithm () is exploited to register rail PCD frames with the fusion of IMU kinematic readings and engineered point cloud features.

3.2.2.2.1 IMU kinematics

Raw IMU readings are presented as follows:

$$\widehat{w}_t = w_t + b_t^w + n_t^w \tag{20}$$

$$\hat{a}_t = R_t^{BW}(a_t - g) + b_t^a + n_t^a \tag{21}$$

where $\hat{w}_t$ and $\hat{a}_t$ represent the readings from the IMU in world frame at time t, and w_t and a_t are the true readings at time t. These readings are interfered by a slowly varying bias b_t and white noise n_t. R_t stands for the rotation matrix from world coordinate system to body coordinate system, and g stands for the constant gravity vector in world coordinate system.

The motion of the LiDAR can now be inferred using these measurements. The velocity, position, and rotation of the LiDAR at time $t + \Delta t$ can be computed as follows:

$$v_{t+\Delta t} = v_t + g\Delta t + R_t(\hat{a}_t - b_t^a + n_t^a)\Delta t \tag{22}$$

$$p_{t+\Delta t} = p_t + v_t\Delta t + \frac{1}{2}g\Delta t^2 + \frac{1}{2}R_t(\hat{a}_t - b_t^a + n_t^a)\Delta t^2 \tag{23}$$

$$R_{t+\Delta t} = R_t^{WB}\exp\left((\hat{w}_t - b_t^w - n_t^w)\Delta t\right) \tag{24}$$

Using the preintegrated IMU method (), the relative speed Δv_{ij}, relative position Δp_{ij}, and relative rotation ΔR_{ij} between time i and j can be computed as:

$$\Delta v_{i,j} = R_i^T(v_j - v_i - g\Delta t_{i,j}) \tag{25}$$

$$\Delta p_{i,j} = R_i^T(p_j - p_i - v_i\Delta t_{i,j} - \frac{1}{2}g\Delta t_{i,j}^2) \tag{26}$$

$$\Delta R_{i,j} = R_i^T R_j \tag{27}$$

these values are jointly updated with point cloud feature matching.

3.2.2.2.2 Point cloud feature

To find point cloud features, the roughness $c_{i,k}$ of a point $p_{i,k}$ is calculated:

$$c_{i,k} = \frac{1}{|S|\cdot||p_{i,k}||}\left|\left|\sum_{j\in S, j\neq k}(p_{i,k} - p_{i,j})\right|\right| \tag{28}$$

where S stands for the set of points with half of them on one side of point $p_{i,k}$. Points with high roughness are considered as edge features, and points with low roughness are considered as planar features. A voxel map containing features from previous n frames is used for feature matching. A new scan F_{i+1} with edge feature F_{i+1}^e and planar feature F_{i+1}^p is matched to the voxel map through planar to planar, edge to edge matching. This matching is calculated through equations:

$$d_{e_k} = \frac{|(p_{i+1,k}^e - p_{i,u}^e)\times(p_{i+1,k}^e - p_{i,v}^e)|}{|p_{i,u}^e - p_{i,v}^e|} \tag{29}$$

$$d_{p_k} = \frac{\left(p^p_{i+1,k} - p^p_{i,u}\right)}{\frac{\left|\left(p^p_{i,u} - p^p_{i,v}\right) \times \left(p^p_{i,u} - p^p_{i,w}\right)\right|}{\left|(p^p_{i,u} - p^p_{i,v}) \times (p^p_{i,u} - p^p_{i,w})\right|}} \tag{30}$$

where d_{e_k} represents the distance between the edge point $p^e_{i+1,k}$ to a line formed by the corresponding edge points $p^e_{i,u}$ and $p^e_{i,v}$, d_{p_k} represents the distance between the planar point $p^p_{i+1,k}$ and the plane formed by corresponding planar points $p^p_{i,u}$, $p^p_{i,v}$, and $p^p_{i,w}$. GaussNewton method is used to solve for the optimal transformation through:

$$\min_{T_{i+1}} \{\sum_{p^e_{i+1,k} \in \acute{F}^e_{i+1}} d_{e_k} + \sum_{p^p_{i+1,k} \in \acute{F}^p_{i+1}} d_{p_k}\} \tag{31}$$

where $\acute{F}_{i+1}$ represents features transformed with preintegrated kinematic values. This optimization procedure calculates the best estimate of the actual pose difference between two specific poses. This estimated pose difference is used to align two frames in the world coordinate.

3.2.2.3 Outlier removal

First, all rail points in the registered point cloud map are rotated using principal component analysis (PCA) so that the longitudinal direction of the rail points align with the x-axis, the lateral width of the track aligns with the y-axis, and the height of the track aligns with the z-axis. The track points are then divided into 11 segments (each segment is approximately 62 ft) along the x-axis. Each segment is then rotated using PCA again so that they all fall around the origin with same coordinate layout as mentioned above. Two quadratic regressions $y = f(x)$ is applied onto the points in the $x - y$ plane, and another quadratic regression $z = g(x)$ for the $x - z$ plane to approximate each track, respectively. The distance of each point to these two regressions in each plane is then calculated. Points with distance larger than a determined threshold are removed.

3.2.2.4 Geometry measurement

3.2.2.4.1 Gauge

To simulate the standard method of measuring rail gauge, we need to first identify the ball of the rail head, then locate the rail inner surface 5/8 inch below the ball of the rail head. To approximate the ball of the rail head, we select the top 10% of the points each segment, and calculate their average value in the z-axis to represent the ball rail head. Then, the innermost 10% of the rail points that are 15.5 – 16.25 mm beneath this average value in the z-axis are selected as the inner rail surface points for gauge measurement. Because PCD are innately sparse, direct measurement would be difficult. Therefore, two linear regressions are fitted to interpolate these surface points. A 5 m long sliding window is then used to select critical points, where $(x_{1,1}, y_{1,1})$, $(x_{1,2}, y_{1,2})$ are two points on one linear regression line that are 5 m apart, and $(x_{2,1}, y_{2,1})$, $(x_{2,2}, y_{2,2})$ are the ones 5 m apart on the other linear regression line. The sliding gauge value is then be calculated as the distance between the midpoint $(x_{1,m}, y_{1,m}) = \left((x_{1,1} + x_{1,2})/2\,, (y_{1,1}, y_{1,2})/2\right)$ to the other gauge line at $(x_{2,1}, y_{2,1})$ and $(x_{2,2}, y_{2,2})$ ():

$$d_g = \frac{N_g}{\sqrt{(y_{2,2}-y_{2,1})^2+(x_{2,2}-x_{2,1})^2}} \quad (32)$$

The window shifts 0.5 m at a time.

3.2.2.4.2 Curvature measurement

To determine rail curvature, we employ an 18 m sliding window and focus on identifying the gauge side of the reference rail. Ideally, points on the side of the rail head would be used to identify the gauge side, but due to the sparse nature of the PCD, this approach can lead to inaccurate regression fitting. Instead, all rail head points within the window are used for regression fitting in the horizontal plane.

The top 10% of the rail points within each 18 m window are selected and fitted with a quadratic regression in the x-y plane that represents the gauge side of the reference rail. The points at both ends of the window are fitted with a linear regression to approximate the 62-foot chord in the x-y plane. The rail curvature for that section is then calculated as the lateral distance between the quadratic and the linear regressions at the window's midpoint. This sliding window moves 3 m every time along the rail for each subsequent calculation, providing a continuous measurement of curvature along the track.

3.2.2.4.3 Profile calculation

Similar to calculating curvature, an 18 m sliding window is employed to analyze the rail profile. Within each window, the top 10% of points are selected to represent the rail head. In the x-z plane, these points are fitted with a quadratic curve, while the points at both ends are fitted with straight lines. The rail profile measurement for that section is determined by calculating the vertical distance between the quadratic curve and the linear fits at the window's midpoint. This process is repeated as the window moves along the rail at 3 m each time, providing a continuous profile measurement.

3.3 Results

This section presents the platform measurement results using a custom collected. The data collection was carried out at the Nevada State Railroad Museum located in Boulder City, Nevada, USA. An almost 700 ft test section of mostly straight and slightly curved rail track was selected for this experiment (see Figure 37). This was the only available track section at the museum.

Figure 37 Test site track section for data collection.

3.3.1 Semantic segmentation result

The precision of the track geometry calculations depends heavily on the accuracy of the semantic segmentation process. During the training process, the batch size is set to 8, and the learning rate is set to 0.01. Given the significant imbalance between the two classes of points in rail PCD, the Intersection over Union (IoU) metric is employed to assess the model's performance. During testing, the model demonstrated a highest IoU of 82.4%, which is illustrated in Figure 38.

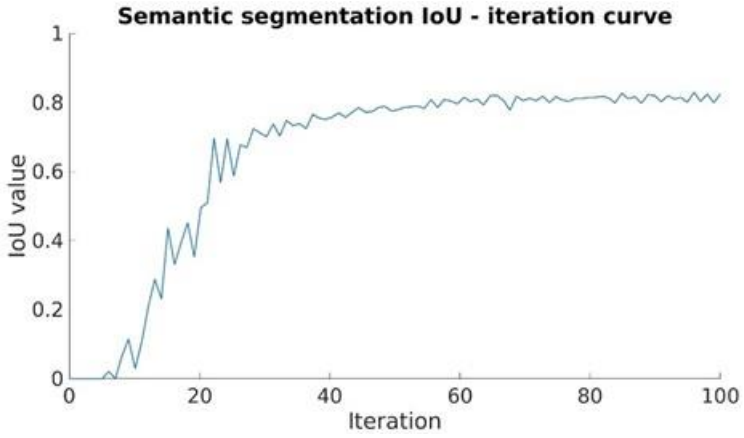

Figure 38 Testing IoU curve.

3.3.2 Point cloud registration result

The process of point cloud registration involves aligning and merging successive LiDAR frames to create a comprehensive, detailed composite representation of the test site. Figure 39 presents the registered point cloud data of the test site. This figure showcases the effectiveness of the registration process, with the rails prominently highlighted in red. This coloration is based on the predicted labels, clearly distinguishing the rails from other elements within the scene.

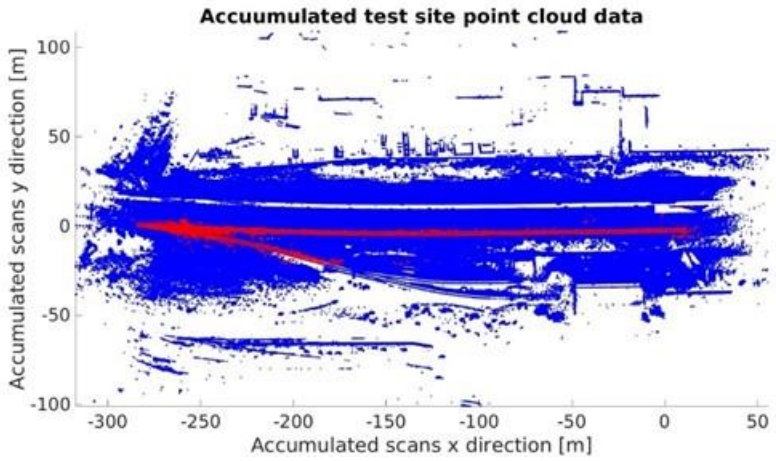

Figure 39 Registered test site point cloud map

3.3.3 Geometry calculation result

First, the outliers are removed using the threshold method. The thresholds are set to 7 cm in the x-y plane, and 8 cm in the x-z plane. These numbers are selected based on the visual results after removing outliers. In this dataset, 10,511 outliers are removed from 50,503 rail points.

All calculated results are compared to the field measurement results using specialized rail tools (see Figure 40). Gauge values were measured every 5 meters. Curvature and profile values were measured at 62 ft apart, with 11 measurements in total. All measurements started at the landmark position.

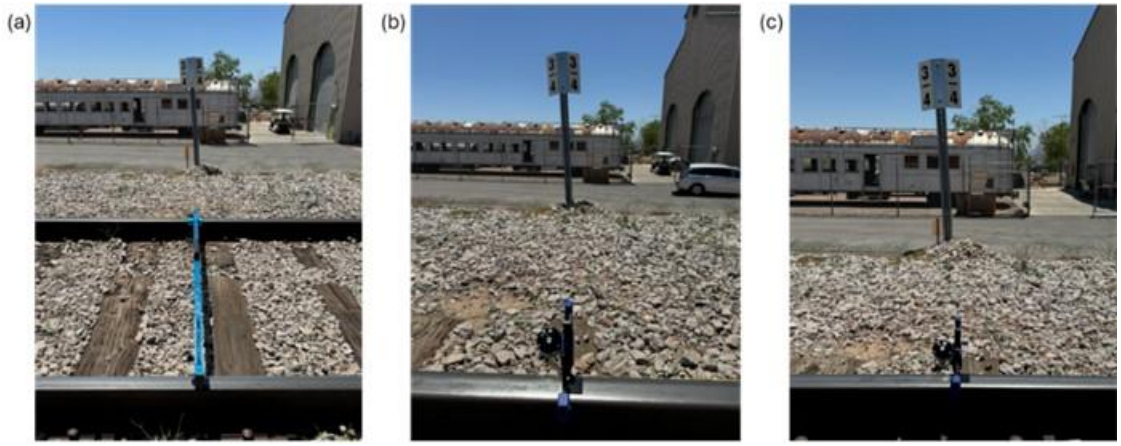

Figure 40 Using specialized rail tools to measure gauge, curvature, and profile at the test site.

3.3.3.1 Gauge

Gauge is measured at 5/8 of an inch below the top of the ball of the rail. Figure 41 illustrates the gauge calculation process in the PCD that simulates this measurement method. The inner surface is identified by locating it approximately 15.5 - 16.25 mm below the rail head. Figure 44 (a)

presents a histogram of deviations between these two sets of gauge measurements. Notably, around 78.57% of these deviations are within 2cm.

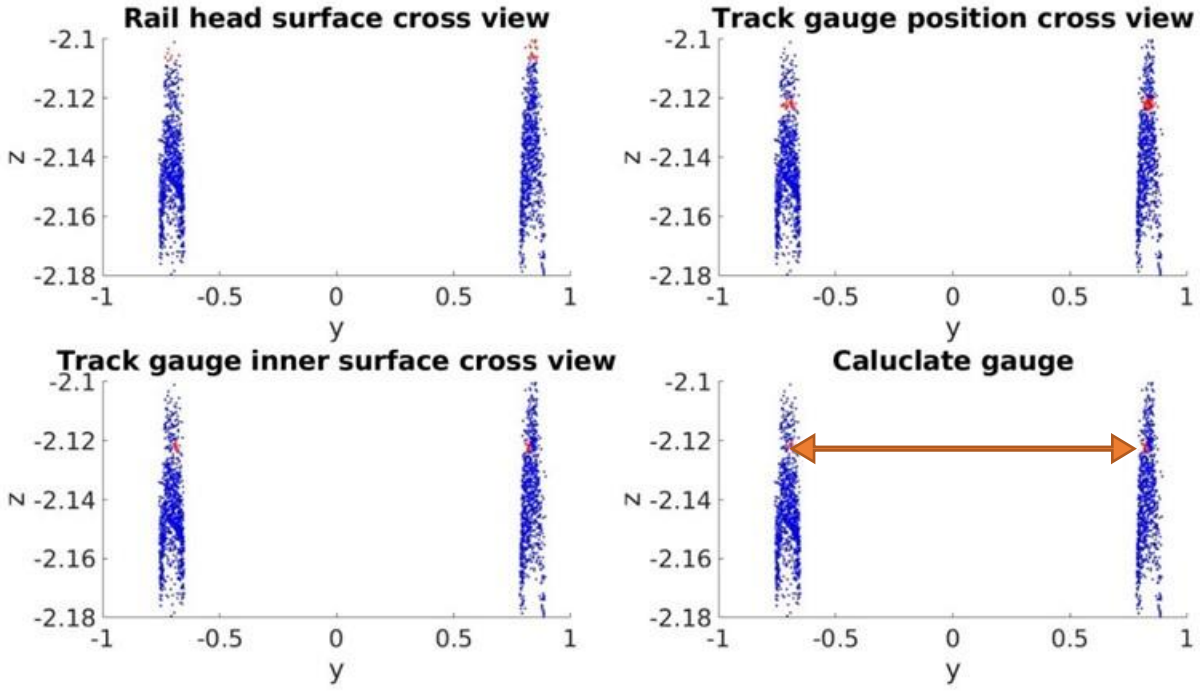

Figure 41 Gauge calculation process. (TL) Selecting rail head. (TR) Finding 5/8 inch below rail head. (BL) Finding inner surface. (BR) Calculating gauge.

3.3.3.2 Curvature

Curvature is assessed by measuring the gap between the 62 feet chord and the gauge side of the reference rail. Figure 42 illustrates the curvature calculation process that simulates this method by fitting a linear regression as the chord and a quadratic regression at the gauge side of the reference rail in the $x - y$ plane. Figure 44 (b) shows the histogram depicting the deviations between the measured curvature and the calculated curvature values. Overall, 77.27% of the deviations fall within 1 cm.

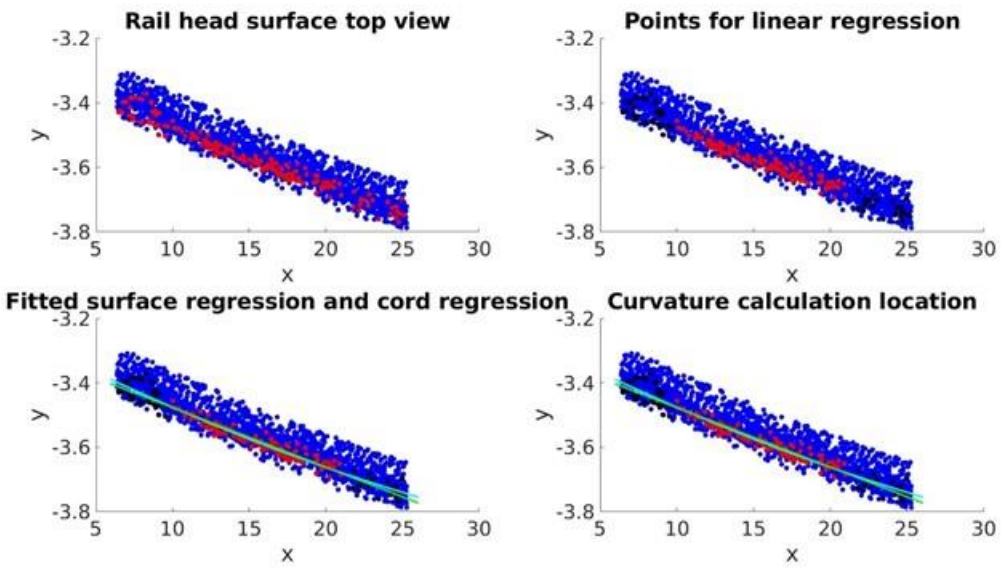

Figure 42 Curvature calculation process. (TL) Finding rail head points. (TR) Finding the points on both ends for chord simulation. (BL) Fitting surface regression and cord regression. (BR) Calculating the gap at the mid-ordinate.

3.3.3.3 Profile

Profile is measured by placing a 62 feet string line along the top of the reference rail and measuring the distance from the midpoint of the string line to the top of the reference rail. Figure 43 demonstrates the profile calculation in the rail points through fitting a linear regression as the string line and a quadratic regression as the top of the reference rail in the $x - z$ plane. Figure 44 (c) depicts a histogram of the deviations between the measured profile values and the calculated profile values, where 77.27% of the deviations fall within 1 cm.

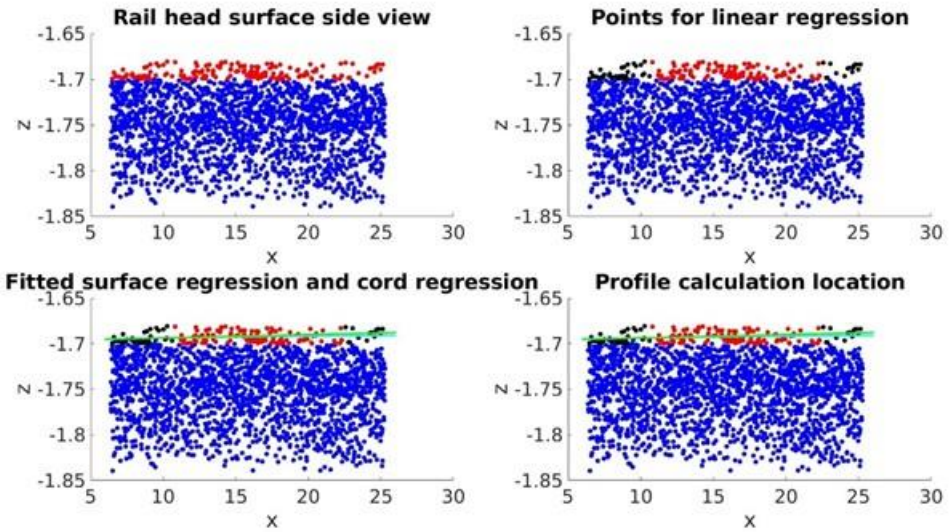

Figure 43 Profile calculation. (TL) Finding rail head. (TR) Finding chord location. (BL) Fitting rail head and chord. (BR) Calculating profile gap.

Table 10 shows the mean, the standard deviation (Std), the root mean square error (RMSE), and the average relative error percentage (AREP) of all three calculations. Although the RMSE values for all three calculations are similar, the gauge measurement demonstrates a notably lower AREP value. Given that the measured gauge values are around 1.435 m, the relative error percentage for the gauge is only approximately 0.77%. In contrast, for curvature and profile, the relative error percentages are notably higher. Therefore, this system shows particularly good performance in measuring rail gauge.

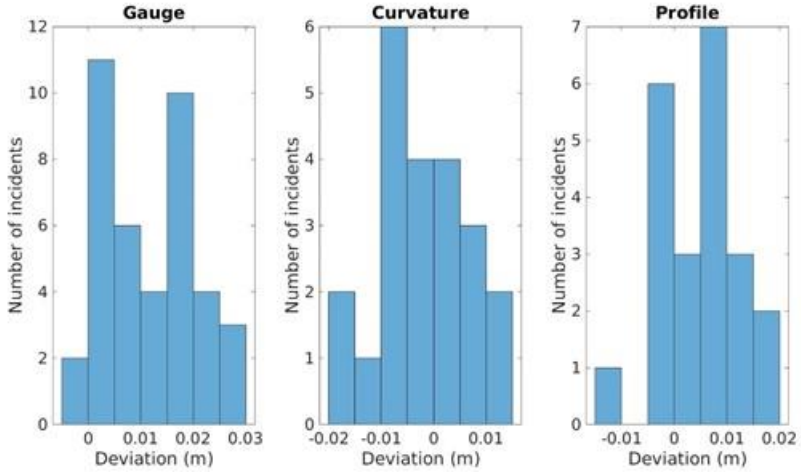

Figure 44 Histogram of deviations of calculated values from measured values.

Table 10 Platform calculation result summary using RMSE

Parameter	Mean (cm)	Std (cm)	RMSE (cm)	AREP
Gauge	144.88	1.68	1	0.77%
Curvature	-0.19	0.84	0.84	143.77%
Profile	0.124	0.35	0.87	111.20%

3.4 Conclusion

In stage two, the authors developed a UAV-LiDAR based rail track geometry measurement platform capable of conducting inspections alongside normal rail operations. Built upon a UAV platform equipped with a LiDAR sensor, the platform leverages ML for rail point segmentation, LiDAR SLAM for expanding the point cloud FOV, regressions for outlier removal, and regressions for geometry calculations. Compared to field measurements using specialized tools, the platform demonstrates high accuracy in gauge measurement but relatively poor performance in curvature and profile measurements. Future work will focus on improving measurement accuracy in curvature and profile, as well as incorporating cross level and warp assessments.

REFERENCES

1. Cannon DF, Edel KO, Grassie SL, Sawley K. Rail defects: an overview. Fatigue & Fracture of Engineering Materials & Structures. 2003 Oct;26(10):865-86.
2. Iwnicki S. Handbook of railway vehicle dynamics. CRC press; 2006 May 22.
3. Sadeghi J, Khajehdezfuly A, Heydari H, Askarinejad H. Development of railway ride comfort prediction model: Incorporating track geometry and rolling stock conditions. Journal of Transportation Engineering, Part A: Systems. 2020 Mar 1;146(3):04020006.
4. Zhang Y, Han J, Song H, Liu Y. Subway embedded track geometric irregularity safety limits. Chinese Journal of Mechanical Engineering. 2021 Dec;34:1-0.
5. Federal railroad administration track safety standards fact sheet. https://railroads.dot.gov/divisions/railroad-safety/track-safety-standards, 2013.
6. Federal Railroad Administration. Track Safety Standards Compliance Manual. Federal Railroad Administration, 1 April 2007, https://www.fra.dot.gov. Archived from the original (PDF) on 28 May 2008. Retrieved 13 November 2012.
7. Tsunashima H, Naganuma Y, Kobayashi T. Track geometry estimation from car-body vibration. Vehicle System Dynamics. 2014 May 30;52(sup1):207-19.
8. Farkas A. Measurement of railway track geometry: A state-of-the-art review. Periodica Polytechnica Transportation Engineering. 2020;48(1):76-88.
9. Chen Q, Niu X, Zuo L, Zhang T, Xiao F, Liu Y, Liu J. A railway track geometry measuring trolley system based on aided INS. Sensors. 2018 Feb 10;18(2):538.
10. Escalona JL, Urda P, Muñoz S. A track geometry measuring system based on multibody kinematics, inertial sensors and computer vision. Sensors. 2021 Jan 20;21(3):683.
11. Wang X, Pan H, Guo K, Yang X, Luo S. The evolution of LiDAR and its application in high precision measurement. InIOP Conference Series: Earth and Environmental Science 2020 May 1 (Vol. 502, No. 1, p. 012008). IOP Publishing.
12. Arastounia M. Automated recognition of railroad infrastructure in rural areas from LiDAR data. Remote Sensing. 2015 Nov 6;7(11):14916-38.
13. Arastounia M. An enhanced algorithm for concurrent recognition of rail tracks and power cables from terrestrial and airborne lidar point clouds. Infrastructures. 2017 Jun 2;2(2):8.
14. Sahebdivani S, Arefi H, Maboudi M. Rail track detection and projection-based 3D modeling from UAV point cloud. Sensors. 2020 Sep 13;20(18):5220.
15. Geng Y, Pan F, Jia L, Wang Z, Qin Y, Tong L, Li S. UAV-LiDAR-based measuring framework for height and stagger of high-speed railway contact wire. IEEE Transactions on Intelligent Transportation Systems. 2021 May 24;23(7):7587-600.
16. Zhang L, Wang J, Shen Y, Liang J, Chen Y, Chen L, Zhou M. A deep learning based method for railway overhead wire reconstruction from airborne LiDAR data. Remote Sensing. 2022 Oct 21;14(20):5272.
17. Manier A, Moras J, Michelin JC, Piet-Lahanier H. Railway lidar semantic segmentation with axially symmetrical convolutional learning. ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences. 2022 May 17;2:135-42.
18. Lin S, Xu C, Chen L, Li S, Tu X. LiDAR point cloud recognition of overhead catenary system with deep learning. Sensors. 2020 Apr 14;20(8):2212.
19. Wang Y, Song W, Lou Y, Zhang Y, Huang F, Tu Z, Liang Q. Rail vehicle localization and mapping with LiDAR-vision-inertial-GNSS fusion. IEEE Robotics and Automation Letters. 2022 Jul 12;7(4):9818-25.

20. Dai X, Song W, Wang Y, Xu Y, Lou Y, Tang W. LiDAR-Inertial Integration for Rail Vehicle Localization and Mapping in Tunnels. IEEE Sensors Journal. 2023 Jun 20.
21. Supervisely. Supervisely Computer Vision platform. Supervisely Ecosystem [Internet]. Supervisely; 2023 Jul [cited 2023 Jul 20]. Available from: https://supervisely.com
22. Maharana K, Mondal S, Nemade B. A review: Data pre-processing and data augmentation techniques. Global Transitions Proceedings. 2022 Jun 1;3(1):91-9.
23. Hu Q, Yang B, Xie L, Rosa S, Guo Y, Wang Z, Trigoni N, Markham A. Randla-net: Efficient semantic segmentation of large-scale point clouds. InProceedings of the IEEE/CVF conference on computer vision and pattern recognition 2020 (pp. 11108-11117).
24. Shan T, Englot B, Meyers D, Wang W, Ratti C, Rus D. Lio-sam: Tightly-coupled lidar inertial odometry via smoothing and mapping. In2020 IEEE/RSJ international conference on intelligent robots and systems (IROS) 2020 Oct 24 (pp. 5135-5142). IEEE.
25. Forster C, Carlone L, Dellaert F, Scaramuzza D. On-manifold preintegration for real-time visual--inertial odometry. IEEE Transactions on Robotics. 2016 Aug 31;33(1):1-21.
26. Alexander DC, Koeberlein GM. Elementary geometry for college students. Houghton Mifflin; 1999 Jan.
27. Soilán M, Sánchez-Rodríguez A, del Río-Barral P, Perez-Collazo C, Arias P, Riveiro B. Review of laser scanning technologies and their applications for road and railway infrastructure monitoring. Infrastructures. 2019 Sep 20;4(4):58.
28. Jixian ZH, Fei LI. Review of visual SLAM environment perception technology and intelligent surveying and mapping application. Acta Geodaetica et Cartographica Sinica.;52(10):1617.
29. Saadat S, Stuart C, Carr G, Payne J. FRA autonomous track geometry measurement system technology development: past, present, and future. InASME/IEEE Joint Rail Conference 2014 Apr 2 (Vol. 45356, p. V001T05A002). American Society of Mechanical Engineers.
30. Keylin A. Measurement and Characterization of Track Geometry Data: Literature Review and Recommendations for Processing FRA ATIP Program Data.
31. Tratman EE. Railway track and track work. Engineering news publishing Company; 1908.
32. Khan MU, Zaidi SA, Ishtiaq A, Bukhari SU, Samer S, Farman A. A comparative survey of lidar-slam and lidar based sensor technologies. In2021 Mohammad Ali Jinnah University International Conference on Computing (MAJICC) 2021 Jul 15 (pp. 1-8). IEEE.
33. Escalona JL, Urda P, Muñoz S. A track geometry measuring system based on multibody kinematics, inertial sensors and computer vision. Sensors. 2021 Jan 20;21(3):683.
34. Alpaydin E. Machine learning. MIT press; 2021 Aug 17.
35. Naganuma Y, Yada T, Uematsu T. Development of an inertial track geometry measuring trolley and utilization of its high-precision data. International Journal of Transport Development and Integration. 2019 Aug 14;3(3):271-85.
36. Chen Q, Niu X, Zuo L, Zhang T, Xiao F, Liu Y, Liu J. A railway track geometry measuring trolley system based on aided INS. Sensors. 2018 Feb 10;18(2):538.
37. Lim KG, Siruno D, Tan MK, Liau CF, Huang S, Teo KT. Mobile machine vision for railway surveillance system using deep learning algorithm. In2021 IEEE International Conference on Artificial Intelligence in Engineering and Technology (IICAIET) 2021 Sep 13 (pp. 1-6). IEEE.

4 LIDAR CAMERA DATA FUSION

4.1 Overview

Given the results in Chapter 3, the authors proceeded to work on integrating multi-modal (i.e., multiple sensors) data with the aim of improving measurement accuracy. Naturally, the authors decided to include a camera as the second data source. The inclusion of a camera sensor brings several benefits that cannot be achieved with LiDAR alone. These benefits include:

1. Enhanced Detail Capture: Camera sensors can capture fine details and textures that LiDAR might miss, providing a richer dataset.
2. Color Information: Unlike LiDAR, which typically provides only distance, intensity, reflectivity data, etc., cameras can capture color information, aiding in more comprehensive scene understanding.
3. Improved Object Identification: The visual data from cameras can assist in identifying and classifying objects more accurately, which is particularly useful in complex environments.

To integrate camera data effectively, several critical tasks must be performed.

1. Image data semantic segmentation: Rail segmentation in image data, preferably through unsupervised methods, is necessary to help achieve high accuracy in rail localization. Without knowing the rail location in images, fusing image data to point cloud data is meaningless.
2. Camera intrinsic calibration: The intrinsic parameters of a camera depend on how it captures the images. It is necessary to determine these parameters to ensure accurate geometric measurements.
3. LiDAR camera extrinsic calibration: Precise calibration between the LiDAR and camera systems is essential to align their data streams into one coordinate.

Consequently, this phase of the project encompasses unsupervised rail image semantic segmentation, camera intrinsic parameter calibration, and LiDAR-camera extrinsic parameter calibration.

4.2 Literature Review

In this section, techniques used in the experiment are reviewed, including unsupervised semantic segmentation in image, camera intrinsic parameter estimation, and LiDAR camera extrinsic parameter estimation.

4.2.1 Unsupervised semantic segmentation in image

Unsupervised image segmentation is a crucial task in computer vision, aiming to partition an image into meaningful regions without prior labeling or training data. Over the years, various approaches have been proposed to tackle this challenge.

One study (Felzenszwalb and Huttenlocher, 2004) introduced an efficient graph-based method for image segmentation. The algorithm adapts to local image characteristics, producing segmentations that respect global properties of the image while being computationally efficient. This work has been widely cited and implemented in various applications.

Another study (Shi and Malik, 2000) proposed the normalized cuts algorithm, which treats image segmentation as a graph partitioning problem. This method considers both the total dissimilarity between different groups and the total similarity within groups, providing a balanced approach to segmentation. This paper has been influential in both computer vision and spectral clustering research.

Achanta et al. (Achanta et al., 2012) introduced the Simple Linear Iterative Clustering (SLIC) algorithm for generating superpixels. Although primarily designed for superpixel generation, SLIC has been widely used as a preprocessing step in many unsupervised segmentation pipelines due to its efficiency and ability to adhere to image boundaries.

Kim et al. (Kim et al., 2013) proposed a nonparametric higher-order learning approach for unsupervised image segmentation. Their method learns higher-order relations between superpixels using a nonparametric Bayesian framework, allowing for more flexible and accurate segmentations.

4.2.2 Camera intrinsic parameters estimation

Camera intrinsic calibration is a fundamental task in computer vision, crucial for applications requiring accurate 3D measurements and corrections for lens distortions. Over the years, various methods and algorithms have been proposed to enhance the accuracy and efficiency of camera calibration. Zhang (Zhang, 2000) introduced a flexible new technique for camera calibration that uses a planar pattern observed at a few different orientations. This method, known as Zhang's calibration, became widely adopted due to its simplicity and effectiveness, requiring only a printed checkerboard pattern and a set of images taken from different angles. A four-step calibration procedure (Heikkilä and Silvén, 1997) involving image acquisition, corner extraction, parameter estimation, and re-projection error minimization. This method also addressed the issue of radial distortion and offered a comprehensive solution for camera parameter estimation. Tsai (Tsai, 1986) developed a two-stage camera calibration approach that combined 3D reference points with a coplanar grid pattern. This method significantly improved the accuracy of intrinsic and extrinsic parameter estimation and has been influential in both academic research and industrial applications. Sturm and Maybank (Sturm and Maybank, 1999) explored a method for plane-based camera calibration that employed vanishing points and lines for parameter estimation. Their approach demonstrated robustness in handling various distortion models and provided a versatile framework for different camera setups. Bouguet (Bouguet, 2004) developed a comprehensive camera calibration toolbox for MATLAB, which became a standard tool for researchers and practitioners. This toolbox implements Zhang's method and provides an easy-to-use interface for both intrinsic and extrinsic camera calibration.

4.2.3 Camera LiDAR extrinsic parameters estimation

Extrinsic calibration between cameras and LiDAR sensors is crucial for multi-sensor fusion in autonomous vehicles and robotics (6). Many studies have been conducted regarding this topic. One method (Zhang et al., 2004) uses a checkerboard for 2D laser rangefinder and camera calibration, achieving automated calibration with minimal human intervention. This pioneering work laid the foundation for subsequent research. Geiger et al. (Geiger et al., 2012) extended this approach to automatically calibrate multiple cameras and LiDAR sensors using a single shot of multiple checkerboards, improving efficiency in multi-sensor setups.

As research progressed, limitations of planar targets became apparent. To address issues with horizontal edges, one study (Tóth et al. 2020) introduced a method using spherical targets, achieving more accurate and viewpoint-invariant calibration. In a similar vein, another study (Park et al., 2014) employed a polygonal planar board, demonstrating improved feature extraction in LiDAR scans.

While target-based methods offer high accuracy, they can be impractical in some real-world scenarios. This realization led to the development of targetless calibration techniques. Researchers (Moghadam et al., 2013) developed a method extracting line segments from natural scenes, eliminating the need for special calibration objects. Their approach showed robust performance in urban environments. Leveraging advances in computer vision, one study (Zhu et al. 2020) utilized semantic segmentation of camera images for calibration, demonstrating increased robustness to noise and low-resolution LiDAR data compared to traditional edge-based methods. This approach opened new possibilities for calibration in complex, real-world settings.

4.3 Experiment

This section outlines the experimental process for data fusion. The authors employed several techniques for image segmentation, including canny edge detector-based method, Segment-Anything model, and unsupervised image segmentation.

4.3.1 Unsupervised image segmentation

Various techniques were experimented by the authors to extract rails from the image using unsupervised image segmentation, including canny edge detection coupled with Hough transform and a threshold-based region growing algorithm, Segment-Anything, and a semi-supervised image segmentation method.

4.3.1.1 Canny edge detection-based method

Rail tracks can be characterized as approximate straight lines with distinct coloration compared to their surroundings, forming continuous connected as show in Figure 45 (a). Therefore, the task becomes identifying long tangent lines with continuity. To do so, a combination of inverted binary transformation (Figure 45 (b)), edge detection algorithms (i.e., canny edge detection), line detection algorithms (i.e., Hough transform)), and a threshold-based region algorithm (i.e., binary threshold) is implemented for rail labeling.

Figure 45 Rail images. (a) Rails in original condition. (b) Inverted binary image.

4.3.1.1.1 Canny edge detection

To detect edges, noise reduction using a 5×5 Gaussian filter is first applied to the image. Then, the smoothened image is filtered with a Sobel kernel in both horizontal and vertical direction to get first derivative in horizontal direction (G_x) and vertical direction (G_y). The edge gradient and direction of each pixel is than calculated as:

$$G = \tan^{-1}\left(\frac{G_y}{G_x}\right) \tag{33}$$

After getting gradient magnitude and direction, a full scan of images is done to remove any unwanted pixels which may not constitute the edge. For this, at every pixel, pixel is checked if it is local maximum in its neighborhood in the direction of gradient (
Figure *46*).

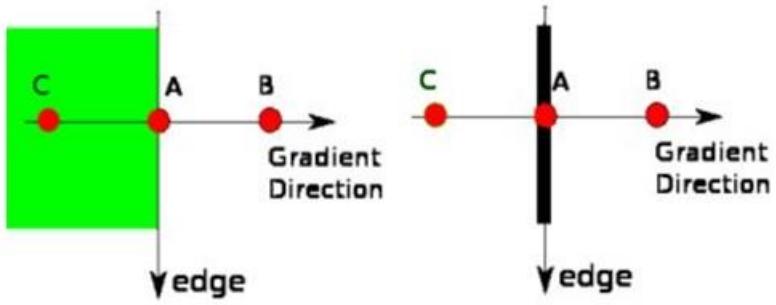

Figure 46 Non-maximum suppression. Point A is on the edge in the vertical direction. Gradient direction is normal to the edge. Point B and C are in gradient directions. Point A is checked with point B and C to see if it forms a local maximum. If so, it is considered for next stage, otherwise, it is suppressed by setting to zero. Image taken from https://docs.opencv.org/4.x/da/d22/tutorial_py_canny.html

Furthermore, to get thin edges, hysteresis thresholding is applied to suppress edges that are not connected to edge greater than maxVal (
Figure *47*).

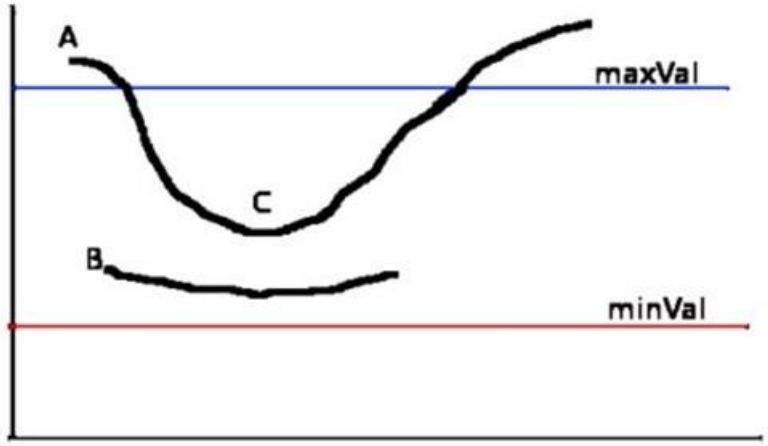

Figure 47 Hysteresis thresholding. Edge B is removed becasue it is not connencted to any edge larger thant maxVal. Edge C is preserved because it connects to edge A. Image taken from https://docs.opencv.org/4.x/da/d22/tutorial_py_canny.html

4.3.1.1.2 Hough Line Detection

Canny edge detection would detect all edges. To detect the straight edges for the rails, Hough Line detection is applied. The equation used for detecting straight lines in the Hough Transform is given by:

$$\rho = x\cos\theta + y\sin\theta \tag{34}$$

where ρ is the perpendicular distance from the origin to the line, θ is the angle between the x-axis and the line perpendicular to the detected line, and (x, y) are the coordinates of any point on the line. For each edge point (x, y) detected by the Canny edge detector, the Hough transform considers all possible lines that could pass through that point. Each of these possible lines is represented by a pair of (ρ, θ) pairs are plotted in the Hough space, creating sinusoidal curves. When multiple sinusoidal curves intersect at a point in the Hough space, it indicates that the corresponding edge points in the original image lie on the same straight line.

4.3.1.1.3 Threshold-based Region Algorithm

Hough line detection might pick up other straight lines in the images that are not part of the track. Therefore, a threshold-based region algorithm is applied to help filter out false positive edges. A binary threshold method is used to filter out weak edges, where:

$$\mathrm{dst}(x,y) = \begin{cases} \mathrm{maxval} & \text{if } (x,y) > \mathrm{thresh} \\ 0 & \text{otherwise} \end{cases} \tag{35}$$

where maxval is the max value for each pixel. In this experiment, the maxval is set to 255, and the threshold value is set to 100.

4.3.1.2 Segment-anything

The Segment Anything Model (SAM) (Kirillov, 2023), developed by Meta AI, is a versatile tool designed to segment virtually any object in an image, even those it hasn't encountered before. This characteristic aligns perfectly with the requirements of this experiment, given the lack of open-source labeled rail images. SAM's ability to generalize to novel objects makes it an ideal choice for the task of rail image segmentation in diverse environments.

SAM operates based on user-provided prompts, which can be points, boxes, guiding the model to segment specific objects of interest.

4.3.1.3 Semi-supervised segmentation

The authors also explored a semi-supervised segmentation approach. Instead of labeling the entire rail using polygon, the contours of each rail are determined using assisted labels and then used to infer rail labels. The flowchart of the entire process is depicted in
Figure *48*.

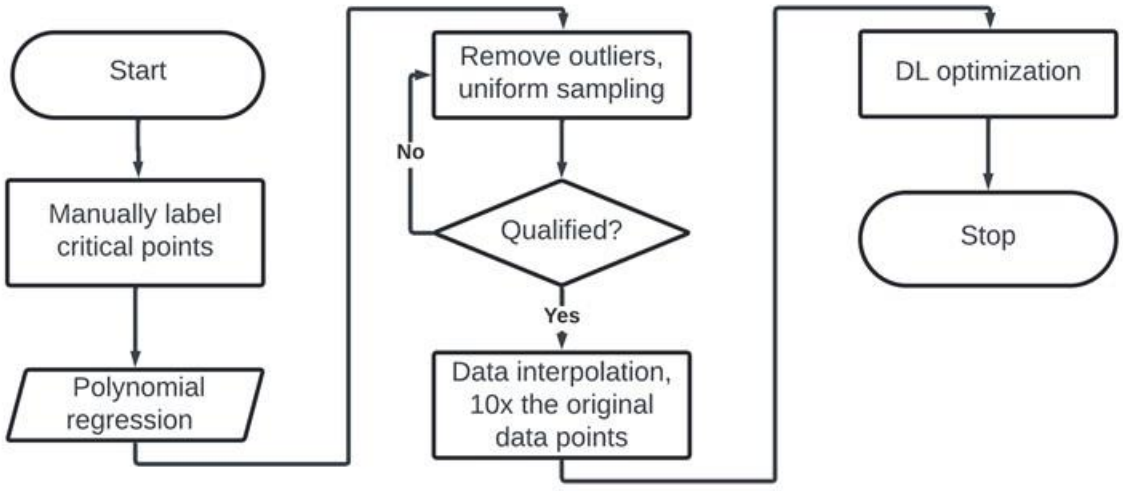

Figure 48 Flowchart of unsupervised image segmentation

This method initiates with manual identification of critical points along the rails in the images (Figure *49* left). Subsequently, a polynomial regression is applied to these key points to determine if any outlier exists in these points. The outliers are determined if the number of pixels between the point and the regression line is more than a preset threshold. These critical points are uniformly sampled to generate a better line. The uniformly distributed critical points are then visually checked to see if they align with rails in each image. If not, the manual labels are modified to reduce the difference between the two. For the ones that pass visual check, the points undergo

tenfold interpolation to create a more detailed contour. Areas between each set of contour lines are determined as rail points (
Figure *50*). Then, a U-net like network, shown in

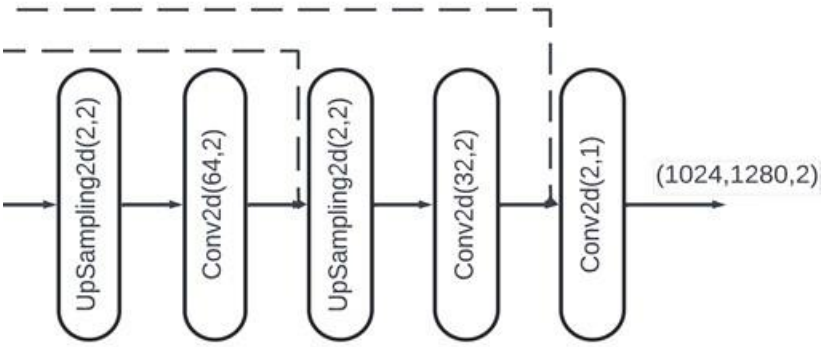

Figure *51*, is employed to predict this label.

Afterwards, a U-net like network, shown in

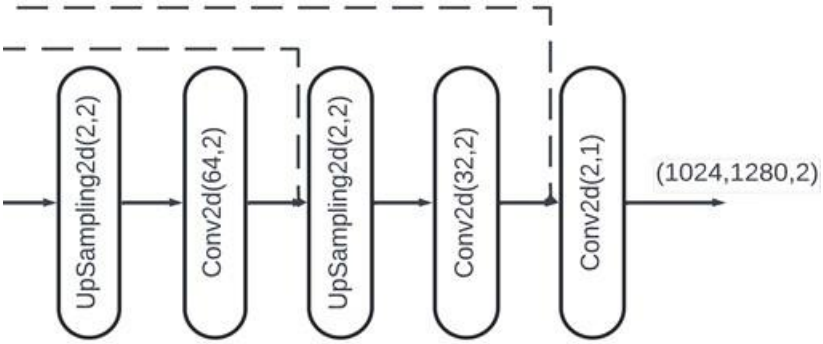

Figure *51*, is employed to predict this label. The model is trained on 46 labeled samples, and tested in a one-shot testing setup. The input of this network are images of size (1024, 1280, 3), while the output consists of masks of the same resolution as the input images, with two channels indicating "rail" or "background".

Figure 49 Key points. Left: Manually selected key points. Right: Uniformly sampled key points (red).

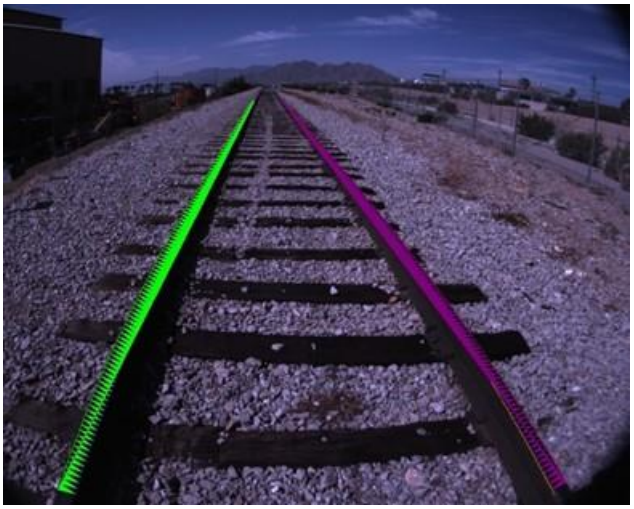

Figure 50 Cover ROI

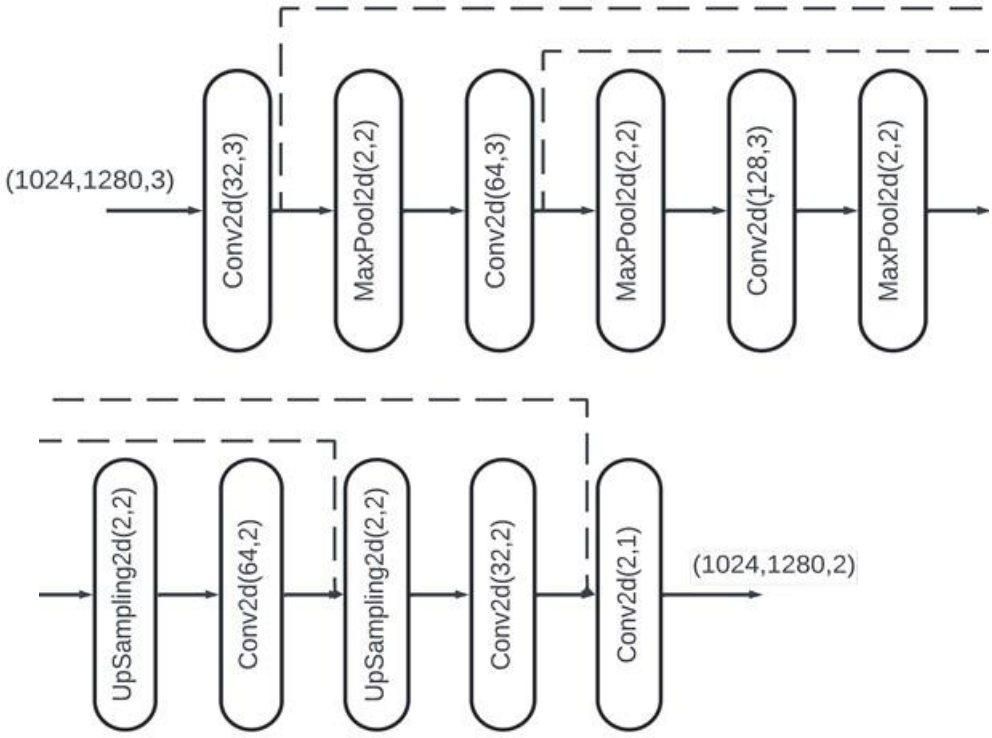

Figure 51 U-net like model to predict point wise labels. Conv2d: 2D convolution. MaxPool2D: 2D max pooling. UpSampling: up sampling. Dashed line: skip connection.

4.3.2 Camera intrinsic calibration

Calibrating the intrinsic parameters of a camera is essential to understand the internal characteristics of the camera, such as the focal length (f_x, f_y), optical center (c_x, c_y), and lens distortion. The camera matrix K is unique to a specific camera, and can be expressed by focal length and optical length through a 3×3 matrix:

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{36}$$

Two major kinds of distortion are radial distortion and tangential distortion. Radial distortion causes straight lines to appear curved. Radial distortion becomes larger the farther points are from the center of the image. Radial distortion can be represented as follows:

$$x_{rd} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \tag{37}$$

$$y_{rd} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \tag{38}$$

where (x, y) is the undistorted coordinate of a point, (x_{rd}, y_{rd}) is the radially distorted coordinate, p_1 and p_2 are the tangential distortion coefficients, $r^2 = x^2 + y^2$ is the squared radius from the optical center.

Similarly, tangential distortion occurs because the image-taking lens is not aligned perfectly parallel to the imaging plane. Therefore, some areas in the image may appear nearer than expected. The amount of tangential distortion can be represented as:

$$x_{td} = x + [2p_1 xy + p_2(r^2 + 2x^2)] \tag{39}$$

$$y_{td} = y + [p_1(r^2 + 2y^2) + 2p_2 xy] \tag{40}$$

where (x_{td}, y_{td}) is the tangentially distorted coordinate. Therefore, the distortion coefficients need to be determined include $(k_1, k_2, p_1, p_2, k_3)$.

The calibration mechanism involves using the identified corner points to solve a series of equations that relate the 2D image points to the 3D points on the calibration pattern. The transformation from 3D world coordinates to 2D image coordinates can be expressed as:

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K[R \quad t] \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{41}$$

where $[u \quad v \quad 1]^T$ are the homogenous coordinates of the 2D image points, $[X \quad Y \quad Z \quad 1]^T$ represents the coordinates of the 3D world points, R and t are the rotation and translation matrices, and s is a scaling factor.

The calibration process aims to determine the camera intrinsic matrix K. Additionally, lens distortions are calculated to correct image distortions caused by the camera lens. By solving equation 41 using a sufficient number of calibration images, the intrinsic parameters and distortion coefficients are optimized to minimize the re-projection error, resulting in an accurate camera model.

The authors selected OpenCV camera intrinsic calibration library for camera intrinsic calibration. This calibration requires at least 10 images with a certain test pattern. A CM3-U3-13Y3C-CS FLIR camera (Figure 52) is used to capture images of a 6×9 checkerboard with 1.1cm × 1.1cm squares, as shown in Figure 53. 88 images are captured in total for intrinsic calibration.

Figure 52 FLIR camera setup for data fusion

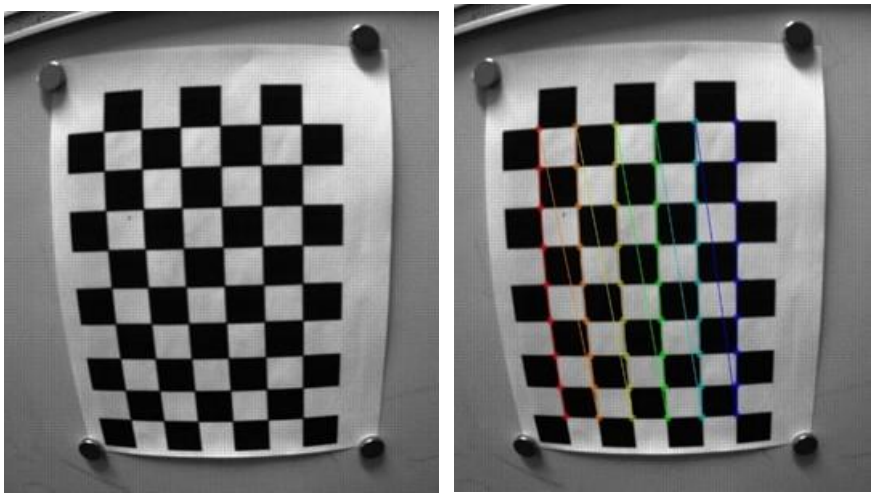

Figure 53 Checkerboard for camera intrinsic calibration. Left: original image. Right: corner feature identified.

In the data, corner features of the chessboard is extracted using cv.findChessboardCorners function. This function extracts all the corner points on the chessboard given the pattern size (Figure 53). Then, cv.calibrateCamera function is used to calculate camera matrix, distortion coefficients, rotation and translation vectors etc. These parameters are used in the extrinsic parameter calibration.

4.3.3 LiDAR camera extrinsic calibration

LiDAR sensors and cameras are commonly used together because a LiDAR sensor collects 3D spatial information while a camera captures the appearance and texture of that space in 2D images. Fusing the data from these sensors correctly can potentially improve object detection and classification. The process of LiDAR-camera calibration estimates a 4×4 homogeneous transformation matrix that gives the relative rotation and translation between the two sensors. The matrix is represented as follows:

$$H = \begin{bmatrix} R & T \\ 0^T & 1 \end{bmatrix} \tag{42}$$

where R is a 3×3 rotation matrix, T is a 3×1 translation vector, and 0^T is a row vector of three zeros. The rotation matrix R describes the orientation of the LiDAR sensor relative to the camera. It has a 3×3 orthogonal matrix with the following properties:

$$RR^T = R^T R = I \tag{43}$$

$$\det(R) = 1 \tag{44}$$

where I is the identity matrix. The translation vector T represents the position of the LiDAR sensor relative to the camera. It is expressed as:

$$T = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \tag{45}$$

where t_x, t_y, and t_z are the translation components along the x, y, and z axes, respectively.
Given a 3D point P_{LiDAR} in the LiDAR coordinate system, the corresponding point P_{cam} in the camera coordinate system can be obtained using the transformation matrix H:

$$P_{cam} = H \times P_{LiDAR} \tag{46}$$

Expressed in homogeneous coordinates, this transformation can be expressed as:

$$\begin{bmatrix} x_{cam} \\ y_{cam} \\ z_{cam} \\ 1 \end{bmatrix} = \begin{bmatrix} R & T \\ 0^T & 1 \end{bmatrix} \begin{bmatrix} x_{LiDAR} \\ y_{LiDAR} \\ z_{LiDAR} \\ 1 \end{bmatrix} \quad (47)$$

With the camera intrinsic matrix K, the 2D image coordinates (u, v) of LiDAR points in a image can be expressed as:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K \begin{bmatrix} \frac{x_{cam}}{z_{cam}} \\ \frac{y_{cam}}{z_{cam}} \\ 1 \end{bmatrix} \quad (48)$$

The MATLAB LiDAR and camera calibration toolbox (Zhou, 2018) is used to estimate the extrinsic parameters. This calibration requires a series of test patterns in terms of image and point cloud (
Figure *54*). This multi-modal data was collected using the FLIR camera and an Ouster OS1-128 LiDAR. The pattern used is a chessboard of size 5×7 with 9.5cm × 9.5cm squares. The corners and planes of the checkerboard are extracted from both LiDAR and camera data to establish a geometrical relationship between their coordinate systems to perform calibration. In image data, the checkerboard corner features are identified through estimateCheckerboardCorners3d function. In LiDAR data, checkerboard plane features are identified through detecRectangularPlanePoints function. Then, function estimateLidarCameraTransform is applied to estimate the rigid transformation matrix (Equation 48) between the LiDAR sensor and the camera.

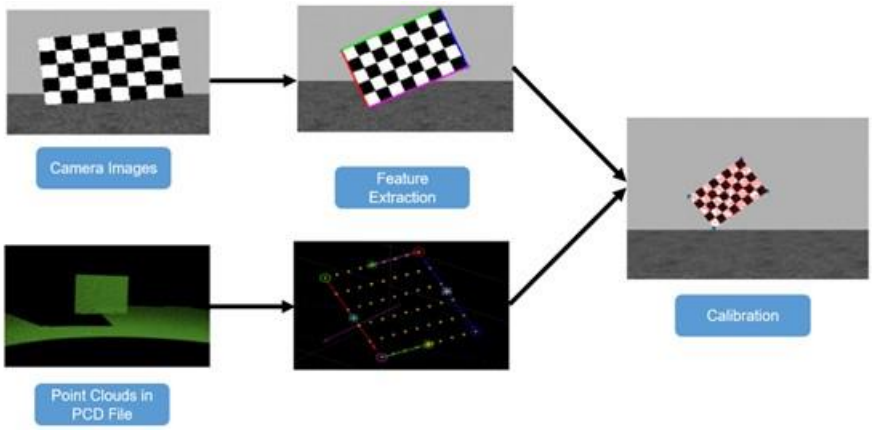

Figure 54 LiDAR camera calibration process. Image taken from https://www.mathworks.com/help/lidar/ug/lidar-and-camera-calibration.html

4.4 Results and analysis

4.4.1 Unsupervised image segmentation result

4.4.1.1 Canny edge detection-based method

This method can successfully detect the rails in the image as shown in Figure *55*, where rails in Figure 45 (a) are detected. The detection process involves identifying linear line features in the image that correspond to the rails. However, since the surrounding environment includes ties, warehouses, etc., those edges would be detected as well, affecting the segmentation result. Therefore, this method is not suitable for the current dataset. With dataset void of significant background objects, this method may be effective.

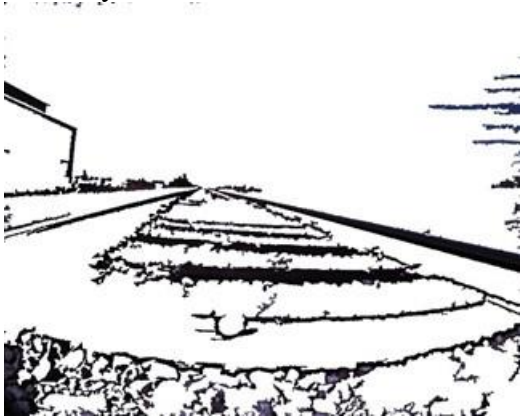

Figure 55 Canny edge detection-based method result.

4.4.1.2 Segment-Anything result

This method is capable of segmenting the rails out of the background with a high accuracy (Figure *56*), where rails are clearly separated from the rest of the areas. Nevertheless, running this algorithm does not generate the same label for rail every time, meaning the rails cannot be segmented even though there is a label for them. In addition, each rail is given an individual label, making it difficult to find all rails. Therefore, this method is not suitable for adding label to image data.

Figure 56 Segment-Anything result. Left: result one. Right: result two.

4.4.1.3 Semi-supervised segmentation

This method uses semi-automated method to find segments. As shown in Figure *57*, this method is capable of segmenting rails when the camera is placed along the rail. However, the semantic segmentation is severely affected by the view angle (Figure *58*). This is likely caused by a lack of data, where most of the rails are placed vertically in the collected images.

Another drawback of this method is the cost of time. To achieve a high accuracy, critical points need to be selected carefully to ensure a higher quality label. An incorrectly selected critical points at the beginning would affect the quality of regressions, and in turns affect the accuracy of contour labels.

Figure 57 Segmentation result when camera is placed along the rail.

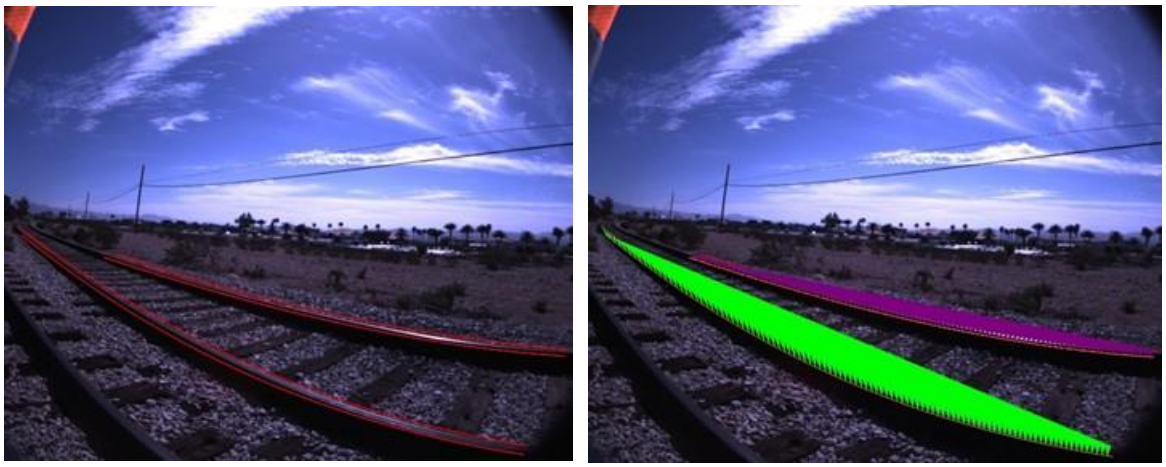

Figure 58 Segmentation result from sideview

4.4.2 Calibration result

The calibration results are evaluated using both checkerboard data and field data. Figure 59 presents the lab calibration result with the checkerboard data, where the error appears negligible.

Figure 60 displays a frame of the calibrated result from the field data. In this image, where the red labels represent the output of point cloud segmentation described in Section 3.2.2.1, while the

green labels are generated from camera data segmentation detailed in Section 4.3.1.3. The discrepancy between these two label sets is readily apparent.

Based on this visual comparison, it is evident that the current calibration is not sufficiently accurate to perform reliable data fusion, regardless of the segmentation accuracy. One potential factor for misalignment in rail data but not the checkerboard data is because rail is significantly smaller than checkerboard, where negligible error cannot be omitted. The noticeable misalignment between the red and the green labels indicates that further refinement of the calibration process is necessary to achieve the precision for effective sensor fusion in this application.

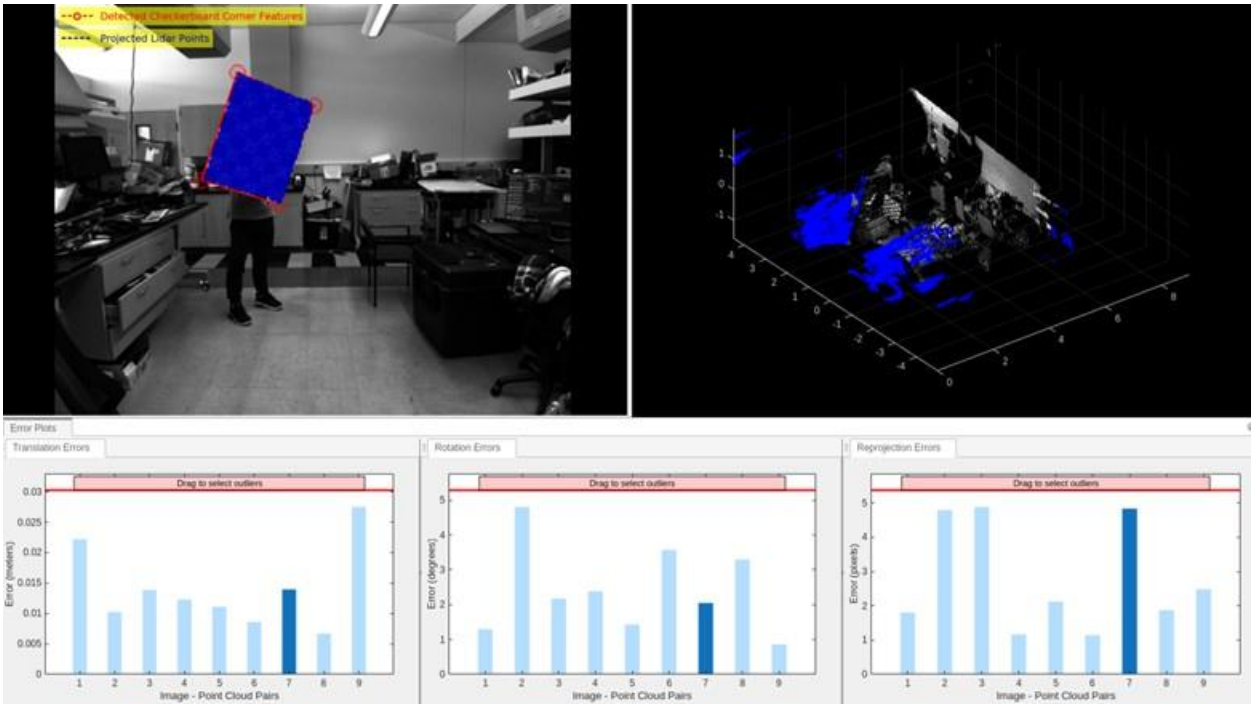

Figure 59 Lab calibration result. Top left: Calibrated result of point cloud overlapping with image. Top right: Grey scene represents calibrated area

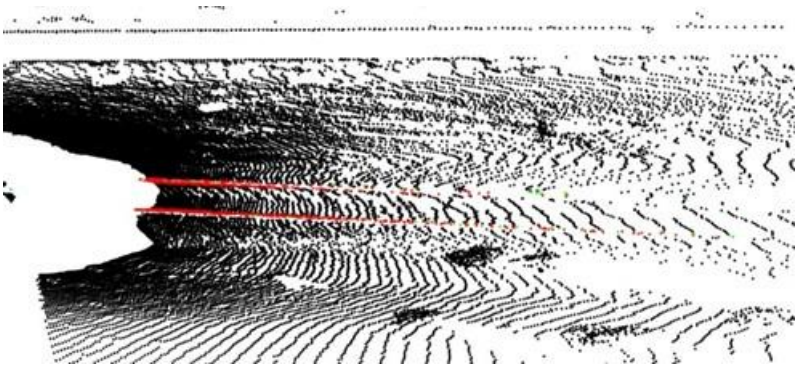

Figure 60 Field data calibration result

4.5 Conclusion

In this stage, the authors explored the possibility of integrating camera data to LiDAR to enhance track geometry measurement. Due to time constraints, semi-assisted supervised image semantic segmentation was utilized, achieving high segmentation accuracy when the rails were vertically aligned in the images. The calibration results were highly accurate with checkerboard data but showed significant errors when applied to rail data. Consequently, the current data fusion approach is not suitable for the geometry measurement platform discussed in Section 3. Future research aims to achieve more comprehensive image segmentation and improve LiDAR camera calibration accuracy.

4.6 References

1. Zhang, Z. (2000). A flexible new technique for camera calibration. IEEE Transactions on pattern analysis and machine intelligence, 22(11), 1330-1334.
2. Heikkila, J., & Silvén, O. (1997, June). A four-step camera calibration procedure with implicit image correction. In Proceedings of IEEE computer society conference on computer vision and pattern recognition (pp. 1106-1112). IEEE.
3. Tsai, R. Y. (1986). An efficient and accurate camera calibration technique for 3D machine vision. In Proc. IEEE Conf. Computer Vision and Pattern Recognition, 1986 (pp. 364-374).
4. Sturm, P. F., & Maybank, S. J. (1999, June). On plane-based camera calibration: A general algorithm, singularities, applications. In Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149) (Vol. 1, pp. 432-437). IEEE.
5. Bouguet, J. Y. (2004). Camera calibration toolbox for matlab. http://www. vision. caltech. edu/bouguetj/calib_doc/.
6. Wang, Y., Li, J., Sun, Y., & Shi, M. (2021, September). A survey of extrinsic calibration of LiDAR and camera. In International Conference on Autonomous Unmanned Systems (pp. 933-944). Singapore: Springer Singapore.
7. Zhang, Q., & Pless, R. (2004, September). Extrinsic calibration of a camera and laser range finder (improves camera calibration). In 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566) (Vol. 3, pp. 2301-2306). IEEE.
8. Geiger, A., Moosmann, F., Car, Ö., & Schuster, B. (2012, May). Automatic camera and range sensor calibration using a single shot. In 2012 IEEE international conference on robotics and automation (pp. 3936-3943). IEEE.
9. Tóth, T., Pusztai, Z., & Hajder, L. (2020, May). Automatic LiDAR-camera calibration of extrinsic parameters using a spherical target. In 2020 IEEE International Conference on Robotics and Automation (ICRA) (pp. 8580-8586). IEEE.
10. Park, Y., Yun, S., Won, C. S., Cho, K., Um, K., & Sim, S. (2014). Calibration between color camera and 3D LIDAR instruments with a polygonal planar board. Sensors, 14(3), 5333-5353.
11. Moghadam, P., Bosse, M., & Zlot, R. (2013, May). Line-based extrinsic calibration of range and image sensors. In 2013 IEEE International Conference on Robotics and Automation (pp. 3685-3691). IEEE.
12. Zhu, Y., Li, C., & Zhang, Y. (2020, May). Online camera-lidar calibration with sensor semantic information. In 2020 IEEE International Conference on Robotics and Automation (ICRA) (pp. 4970-4976). IEEE.

13. Felzenszwalb, P. F., & Huttenlocher, D. P. (2004). Efficient graph-based image segmentation. International journal of computer vision, 59, 167-181.
14. Shi, J., & Malik, J. (2000). Normalized cuts and image segmentation. IEEE Transactions on pattern analysis and machine intelligence, 22(8), 888-905.
15. Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., & Süsstrunk, S. (2012). SLIC superpixels compared to state-of-the-art superpixel methods. IEEE transactions on pattern analysis and machine intelligence, 34(11), 2274-2282.
16. Kim, S., Nowozin, S., Kohli, P., & Yoo, C. (2011). Higher-order correlation clustering for image segmentation. Advances in neural information processing systems, 24.
17. Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., ... & Girshick, R. (2023). Segment anything. In Proceedings of the IEEE/CVF International Conference on Computer Vision (pp. 4015-4026).
18. Zhou, L., Li, Z., & Kaess, M. (2018, October). Automatic extrinsic calibration of a camera and a 3d lidar using line and plane correspondences. In 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (pp. 5562-5569). IEEE.

5 ACKNOWLEDGEMENTS

This study was conducted with the support from the USDOT Tier 1 University Transportation Center on Railroad Sustainability and Durability.

6 ABOUT THE AUTHOR

Lihao Qiu, M.S.

Mr. Lihao Qiu was a MS student and is a Ph.D. student in Electrical and Computer Engineering while he worked on this research project. He received his B.S. degree from Shanghai Maritime University.

Ming Zhu, Ph.D.

Dr. Ming Zhu is the Electrical Engineering Laboratory Director of the Department of Electrical and Computer Engineering at the University of Nevada Las Vegas. His research interests include circuits and VLSI design, robotics and automation, AI/machine learning algorithms and applications, computer and network architectures, computation algorithms and system design. He has a Ph.D. in Electrical Engineering from the University of Nevada Las Vegas.

Jee Woong Park, Ph.D.

Dr. Jee Woong Park is an Associate Professor of Civil and Environment Engineering and Construction. He is specialized in construction management, asset management, construction automation, and informatics. He has his MS degree from Stanford University and his Ph.D. from Georgia Tech.

Yingtao Jiang, Ph.D.

Dr. Yingtao Jiang is a professor in the Department of Electrical and Computer Engineering at the University of Nevada, Las Vegas. His research interests include semiconductors, microelectronics, computer-aided design, unmanned vehicle systems and applications, AI/machine learning algorithms and applications, sensors and instrumentations, computer architectures, wireless communications and STEM education. He has a Ph.D. in Computer Science from the University of Texas at Dallas.

Tianding Chen, Ph.D.

Dr. Tianding Chen is a professor at College of Physics and Information Engineering at Minnan Normal University. He received his Ph.D. in Communication Engineering from Zhejiang University. His research interests include robot vision-based navigation, intelligent image and scene understanding, and machine learning and deep learning.

Han Li, Ph.D.

Dr. Han Li is a professor at College of Electrical and Electronic Engineering at Wenzhou University. He received his Ph.D. degree in Communication from Zhejiang University. His research interests include artificial intelligence, including machine learning, deep learning, and computer vision.

Dr. Haijian Shao, Ph.D.

Dr. Haijian Shao was an Associate Professor at the Jiangsu University of Science and Technology, Zhenjiang, Jiangsu, China when he worked on this research project. He specializes in computer science and image processing.

Hualiang (Harry) Teng, Ph.D.

Dr. Hualiang (Harry) Teng is a Professor of Civil and Environmental Engineering and Construction. He is specialized in railroad system, intelligent transportation system, and highway safety. He received his Ph.D. from Purdue University.