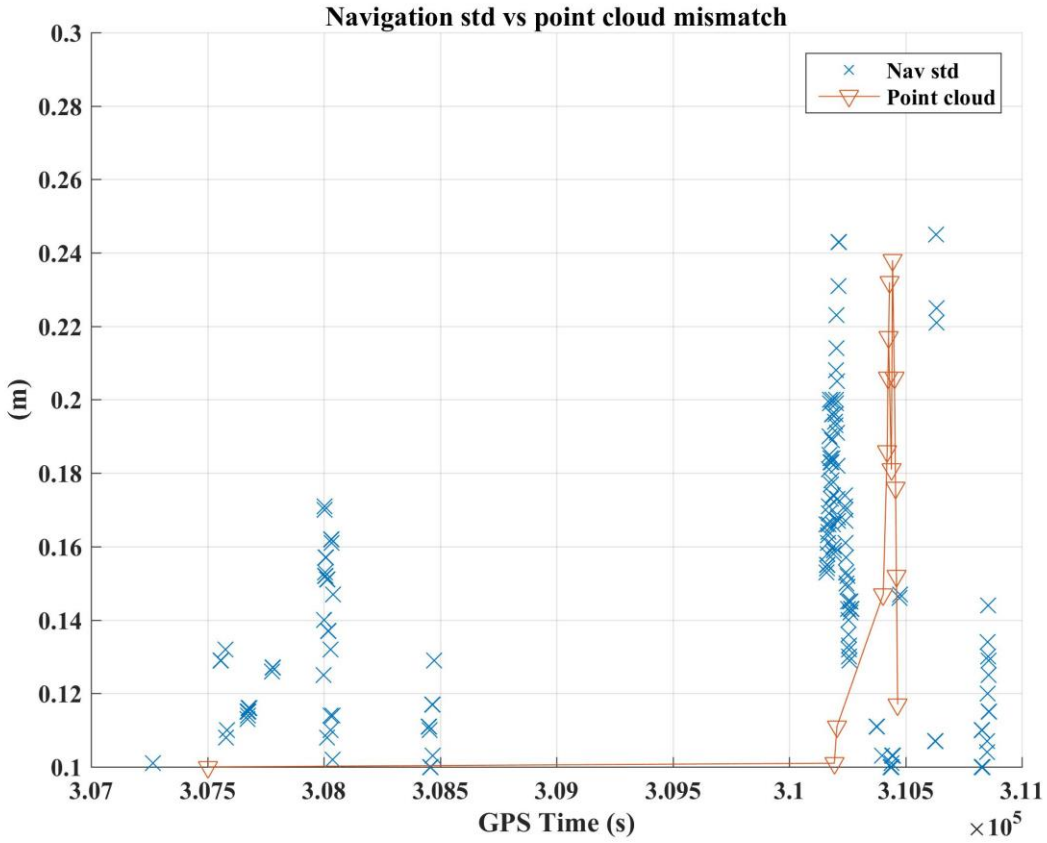



Figure 1.: Navigation performance effect on the point cloud (The Nav std line represents the estimated navigation error from the navigation filter; point cloud line represents the distance of the points between a MLS point cloud data compared to a reference dataset of the same location) .

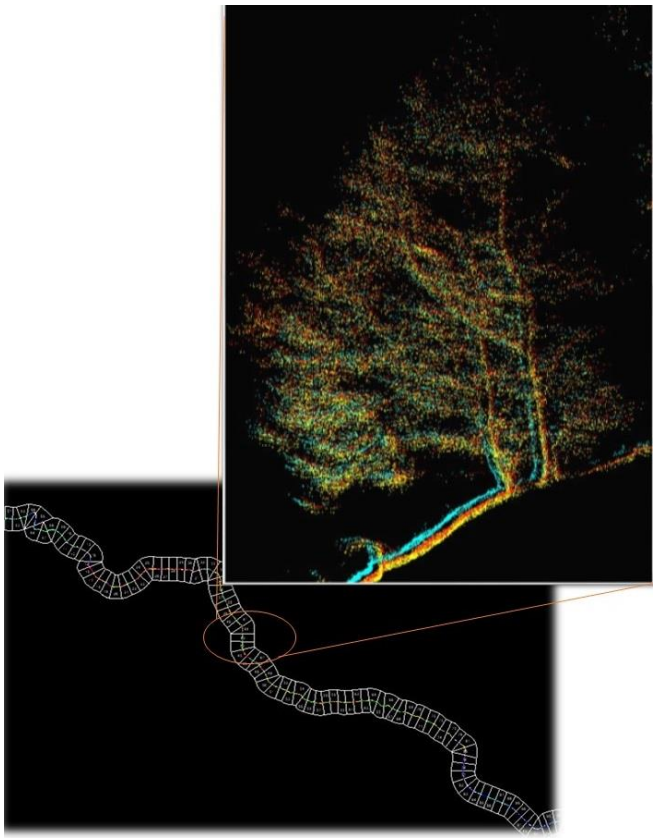


Figure 2.: Shadowing effects when accuracy is inconsistent between point cloud data

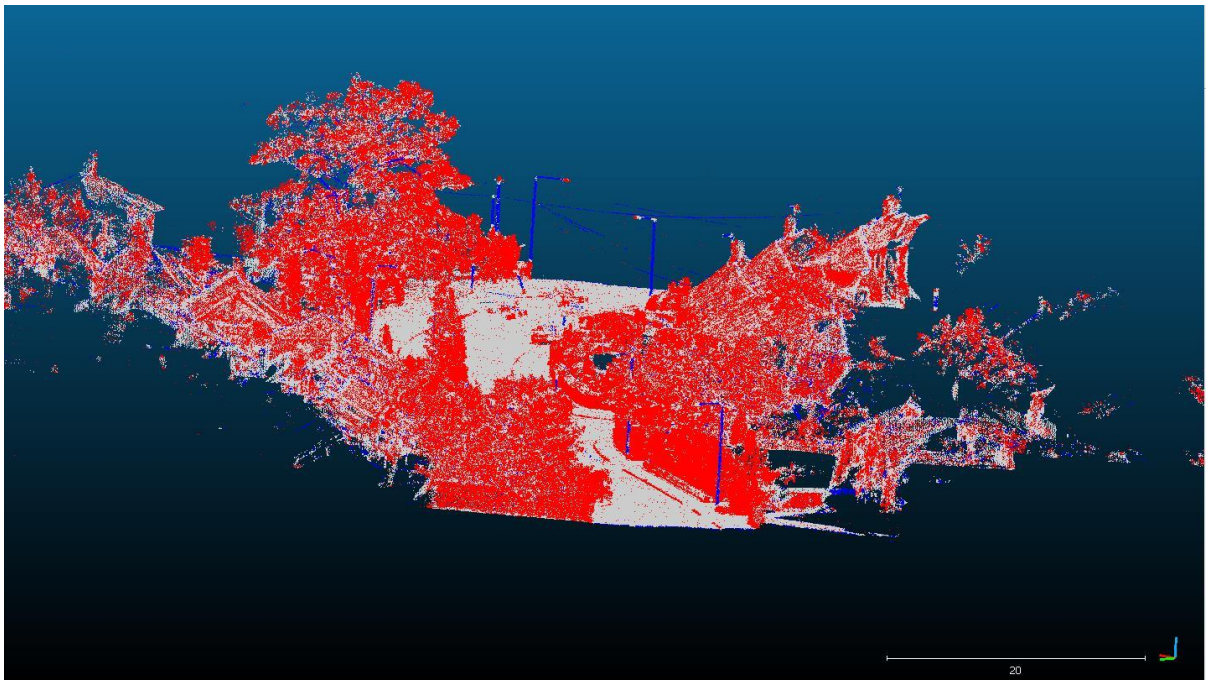


Figure 3.: Point cloud classification results using CANUPO (Blue: pole class (target object); Red: other points not representing the target object; Grey: points that cannot be classified by the tool)

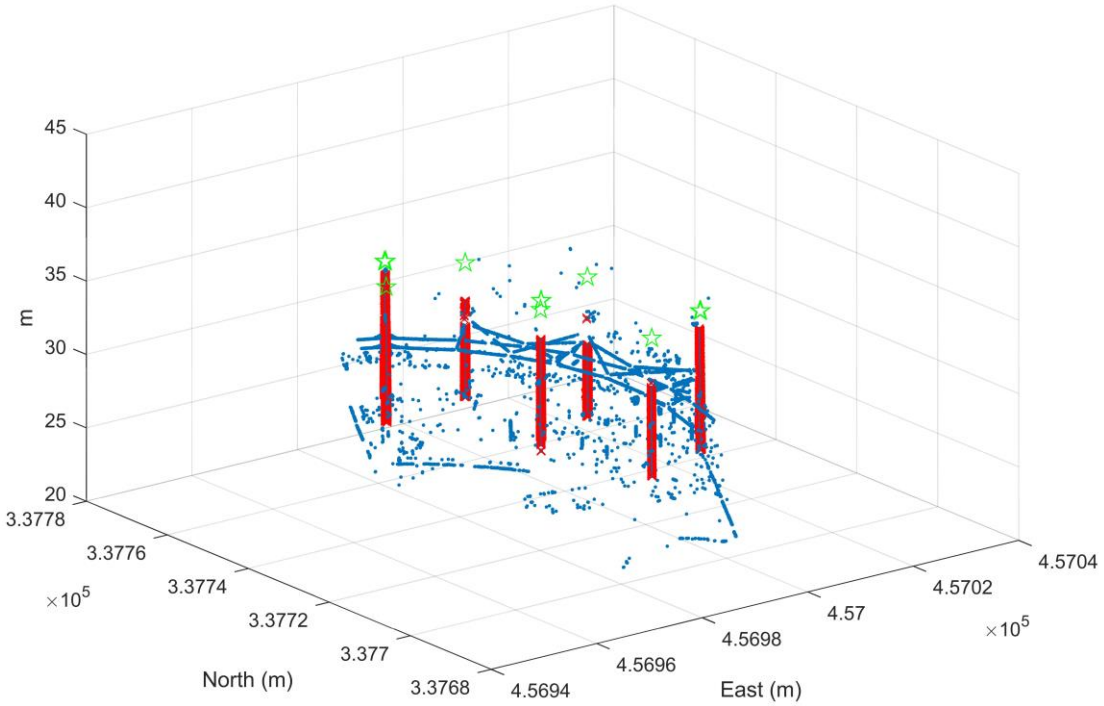



Figure 4.: Extracted features from classified points (red points indicate the final classified pole points, blue points are all the classified points from CANUPO (the blue points from Figure 3), green stars indicate the estimated 2D location of the pole object, plotted at the same height for visual indication)

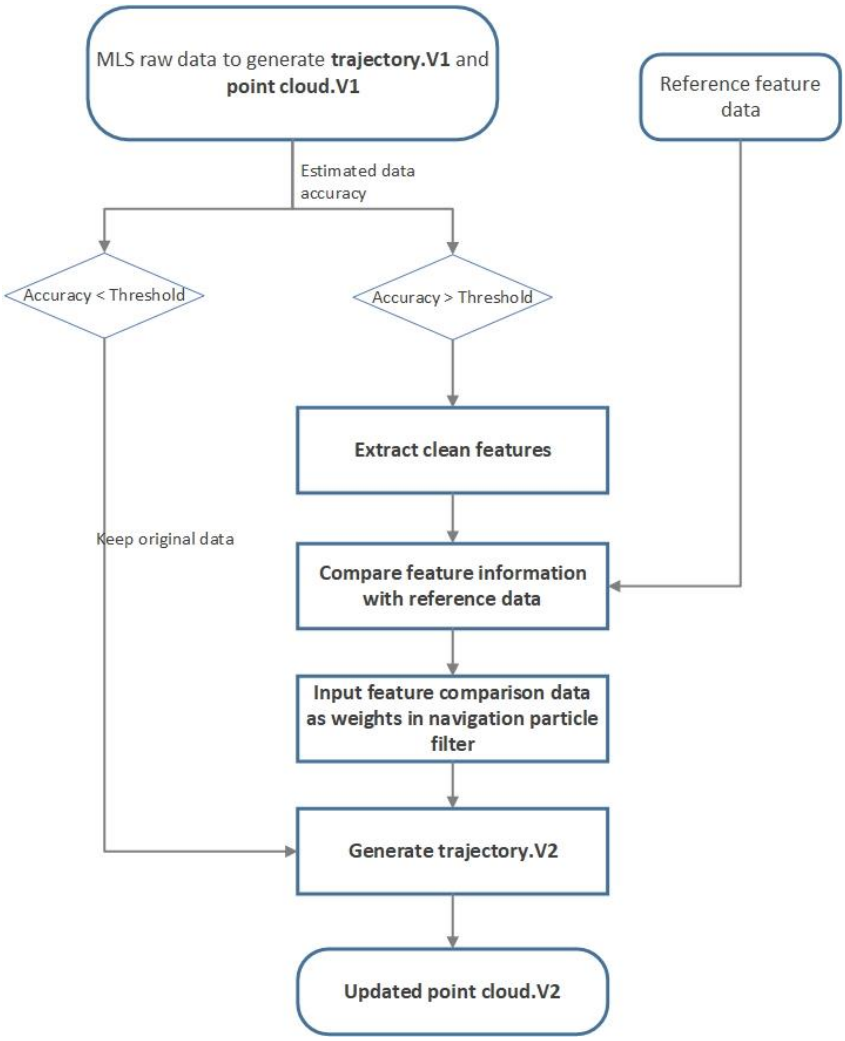



Figure 5.: Flowchart showing the basic steps of the FEPPA method

Figure 6 (a).: ROBIN MLS system

Figure 6(b): StreetMapper MLS system (the camera system in the picture was not used in this particular test)

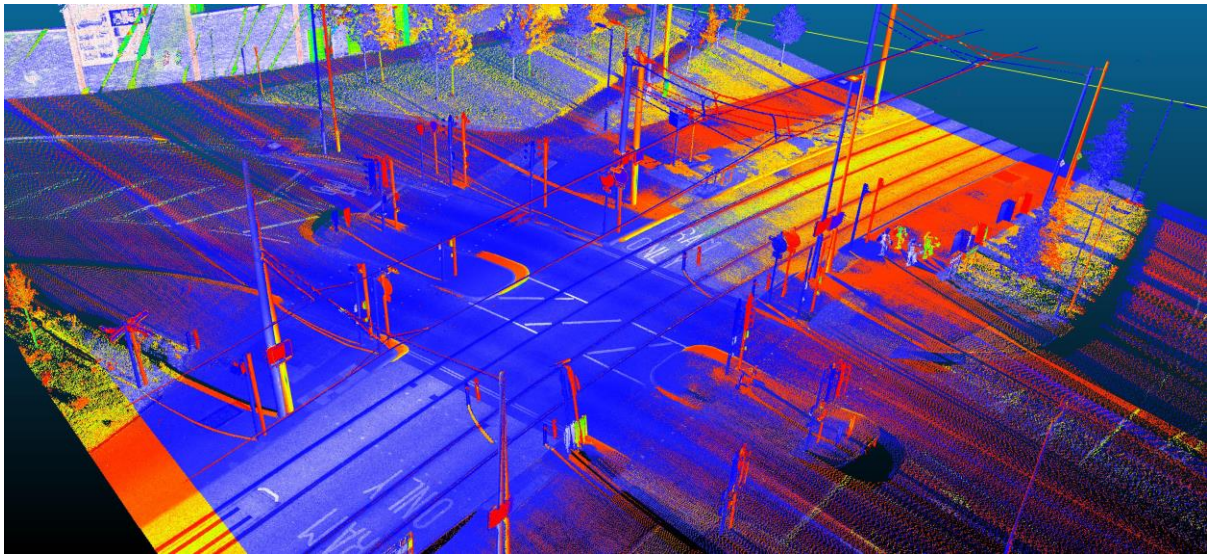


Figure 7.: ICP performance on the tram data (matched data compared to the reference data)

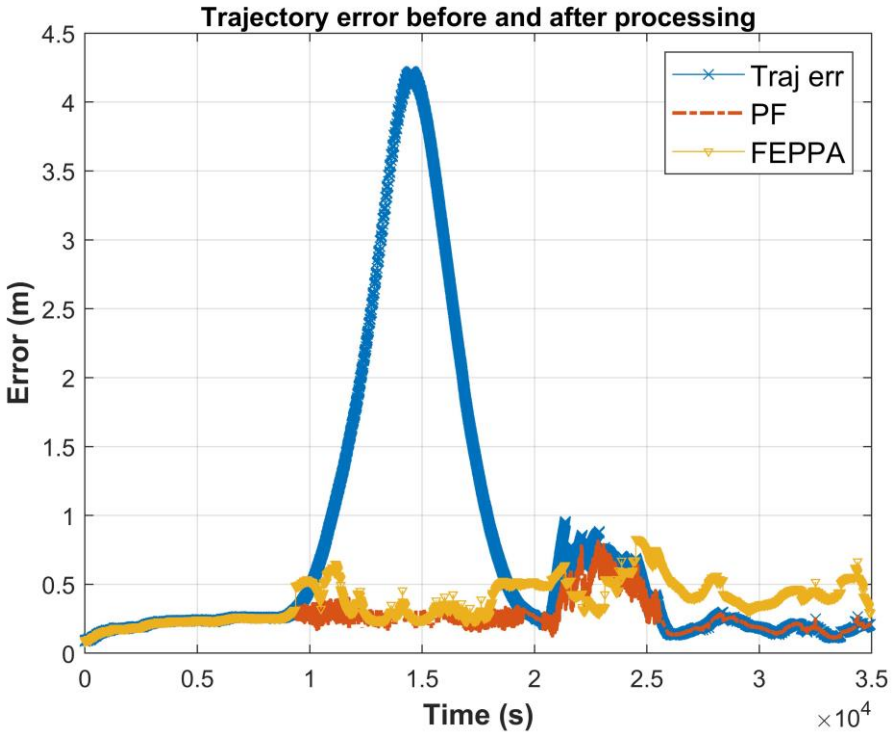


Figure 8.: Navigation correction results

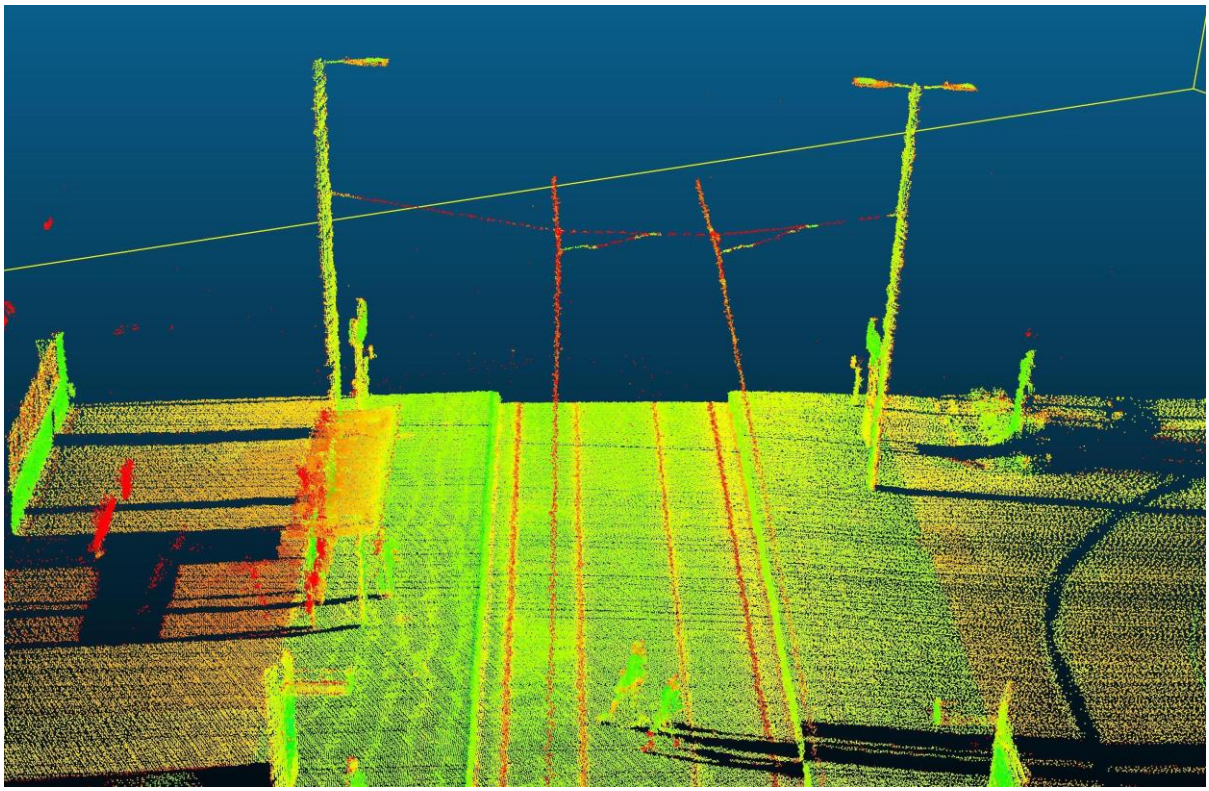


Figure 9.: PF point cloud

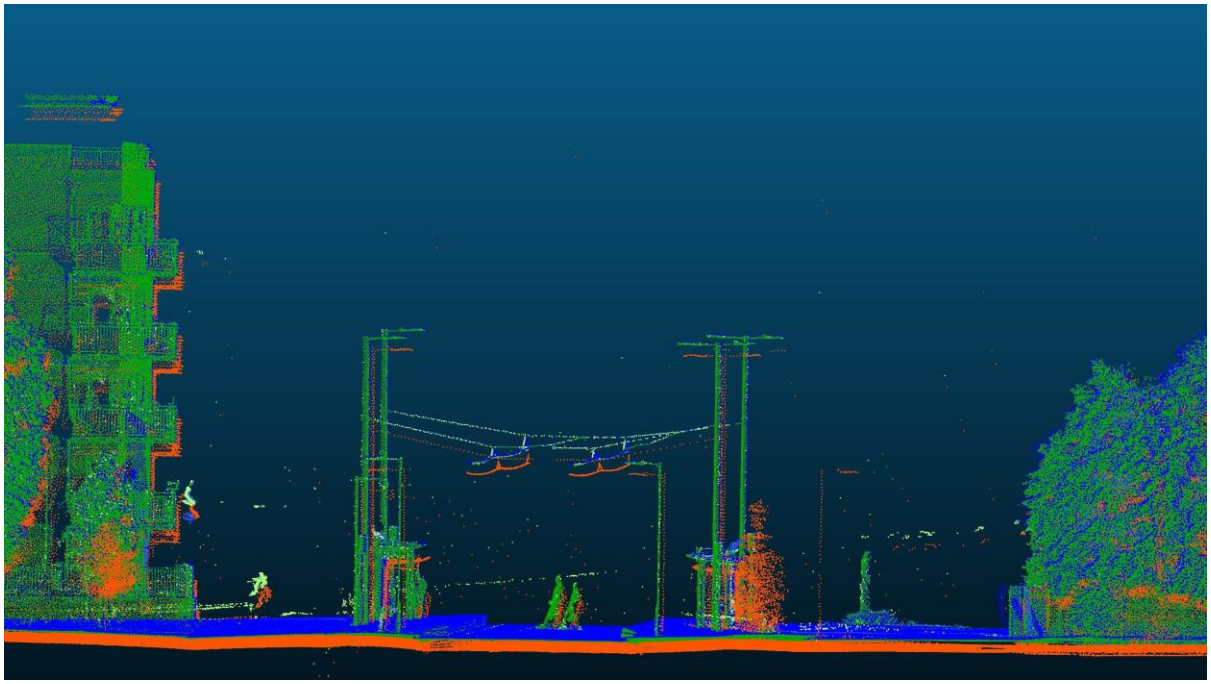


Fig 10 (a) FEPPA method

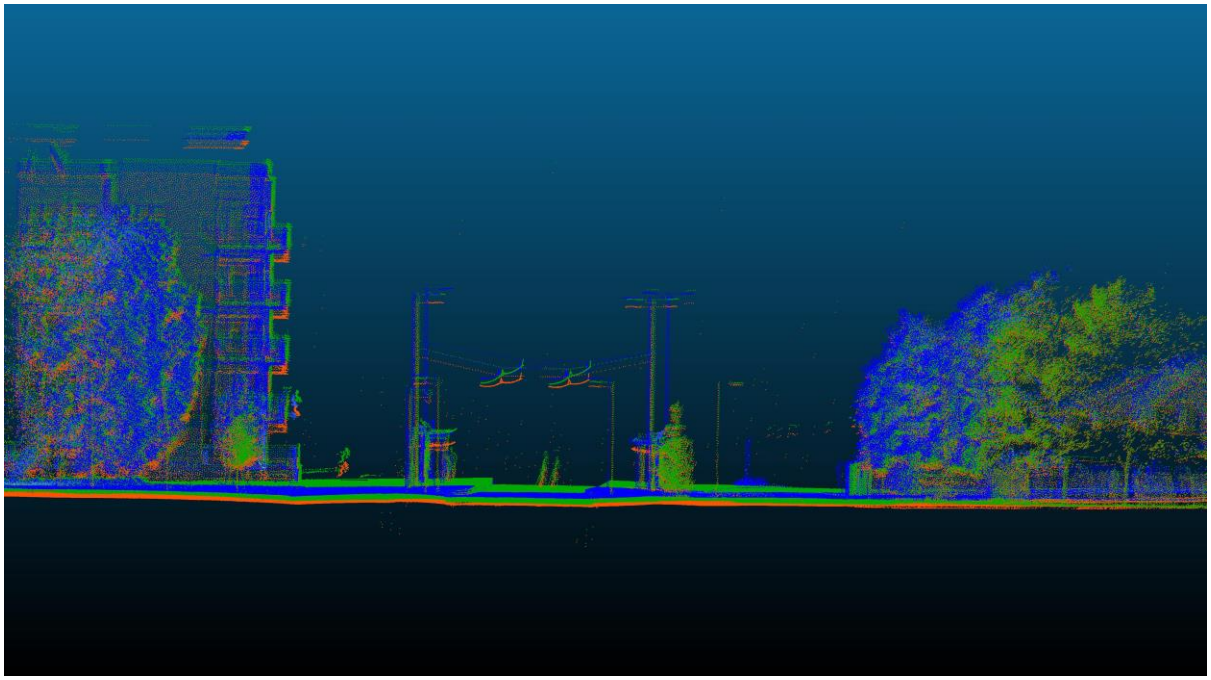


Figure 10 (b) ICP method

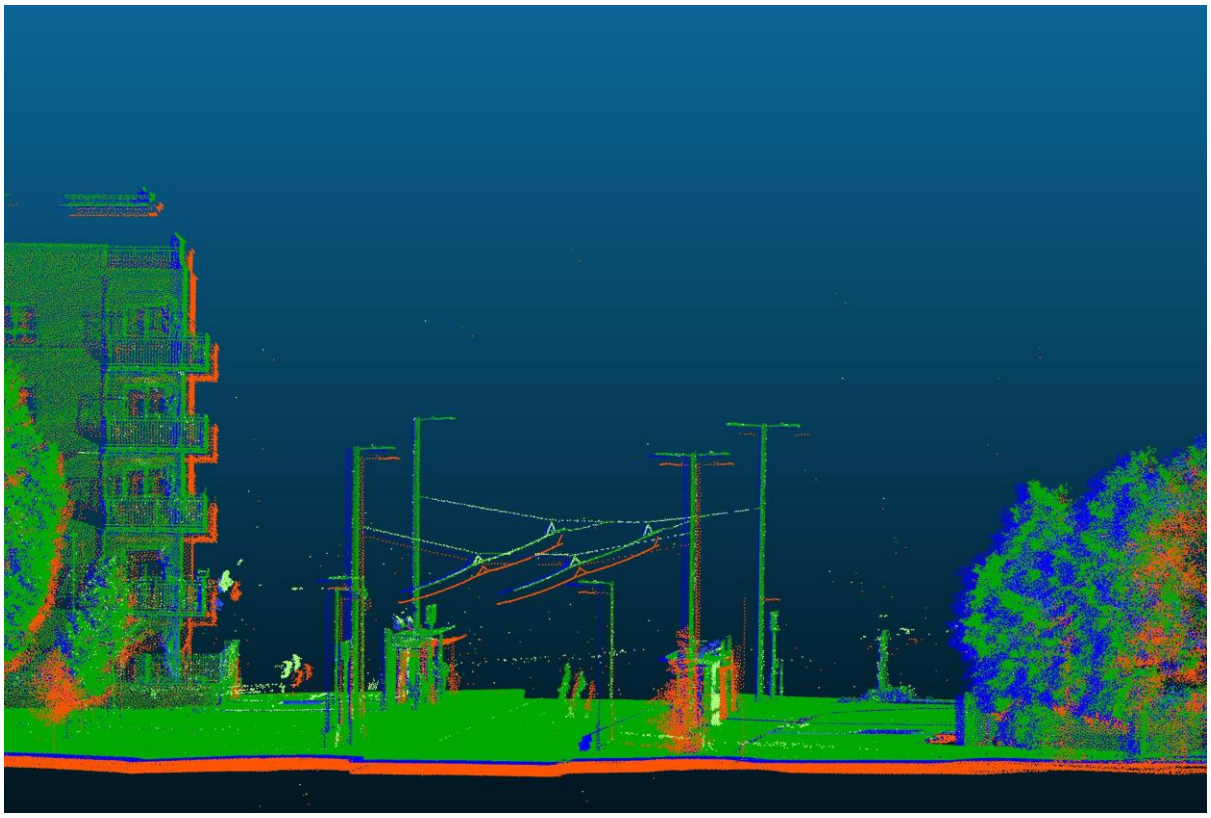


Figure 10 (c) Control point method

Figure 10.: Point cloud correction results after applying different correction methods (Green: the corrected point cloud; Orange: uncorrected point cloud; Blue: reference point cloud)

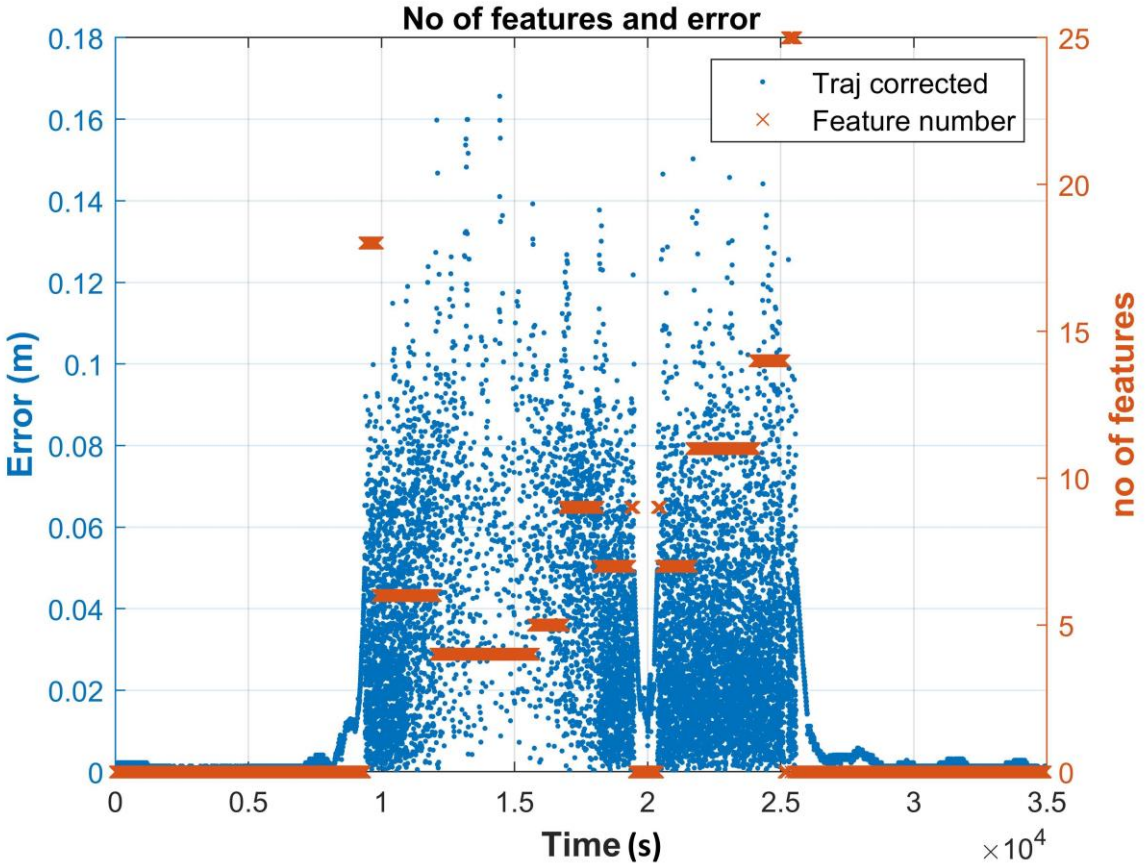



Figure 11.: Comparison between the number of features and the resulting navigation Accuracy

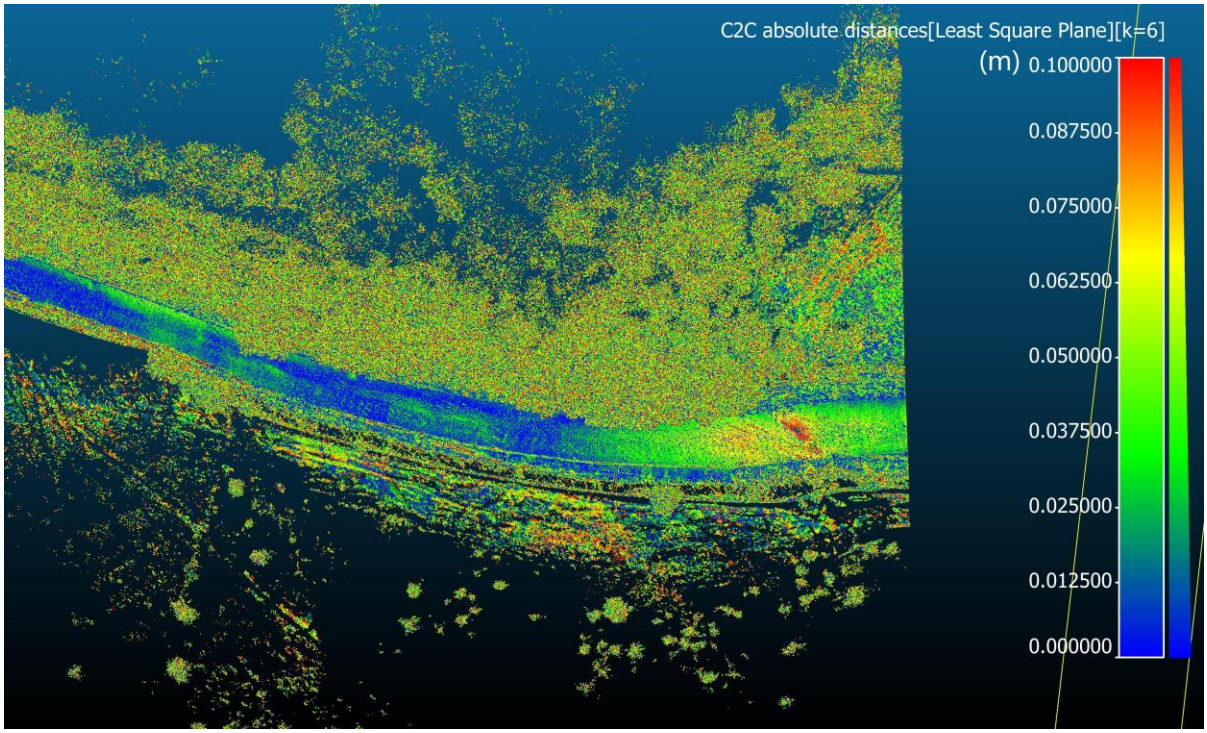



Figure 12 (a) Road change detection (comparing data corrected by FEPPA to a reference dataset)

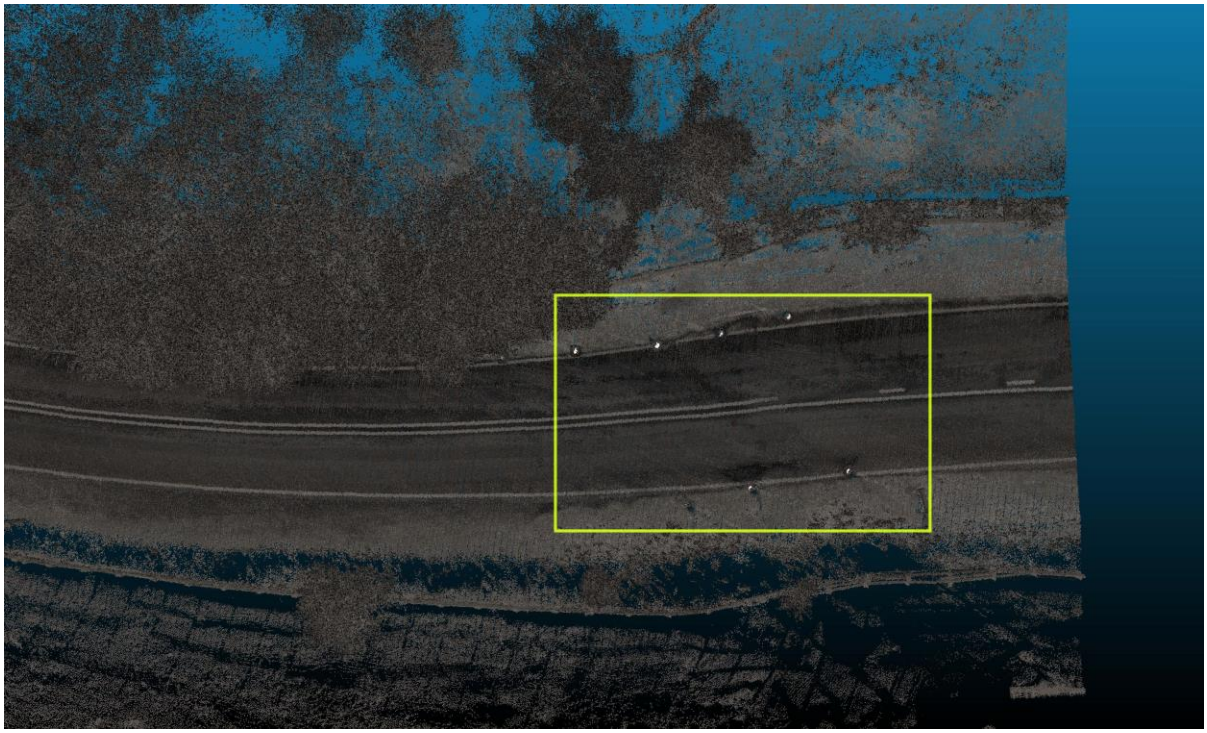


Figure 12 (b) Corrected point cloud of the road shown in intensity

Figure 12.: Change detection results after applying FEPPA (colour bar indicates difference in metres)

ARTICLE TEMPLATE

# Efficient point cloud corrections for mobile monitoring applications using road/rail-side infrastructure

**ARTICLE HISTORY**
Compiled January 2, 2020

**ABSTRACT**
LiDAR systems are known to capture high density and accuracy data much more efficiently than other surveying methods. Therefore they are used for many applications, e.g. mobile mapping and surveying, 3D modelling, hazard detection, etc. However, while the accuracy of the laser measurements is very high, the accuracy of the resulting 3D point cloud is greatly affected by the geo-referencing accuracy. This is especially problematic for mobile laser scanning systems (MLS), where the LiDAR is installed on a moving platform, e.g. a vehicle, and the point cloud is geo-referenced by the data provided by a navigation system.

Due to the complexity of the surrounding environments and external conditions, the accuracy of the navigation system varies and thereby changes the quality of the point cloud. Conventional methods for correcting the point cloud accuracy either rely heavily on manual work or semi-automatic registration methods. While they can provide geo-referencing under different conditions, each has their own problems. This paper presents a semi-automated geo-referencing trajectory correction method by extracting features from the pre-processed point cloud and integrating this information to reprocess the navigation trajectory which is then able to produce better quality point clouds. The method deals with the changing errors within a point cloud dataset, and reducing the trajectory error from metre level to decimetre level, improving the accuracy by at least 56%. The accuracy of the regenerated point cloud then becomes suitable for many accuracy-demanding monitoring and change detection applications.

**KEYWORDS**
navigation, particle filter, point cloud, mobile mapping

## 1. Introduction

Light Detection and Ranging (LiDAR), also known as laser scanning, surveys the surrounding environment by sending out pulsed laser light to target objects and measuring the reflected pulse return times, wavelengths and signal intensity, from which the distance between the scanner and the object could be obtained. Due to the high accuracy of laser measurements and the scanner's ability to scan objects with very high density in a short time (e.g. more than 1 million measurements per second), LiDARs can provide very high density and accurate measurements of the surveyed environment (Lato *et al.* 2009, Pu *et al.* 2011, Puente *et al.* 2013, Mukupa *et al.* 2017). Therefore, conventional surveying work are being replaced and improved by using LiDAR based terrestrial and airborne surveying, mapping and monitoring. It is also being adopted in more recent applications such as 3D High Definition mapping, autonomous navigation, etc.

The mobile laser scanning system (MLS) is a popular system which integrates Li-

DARs with a navigation system and installed onto a mobile platform, e.g. land vehicle, aeroplane or drone. While the LiDAR continuously collects the reflected laser measurement from the surrounding environment as the platform moves, the navigation system provides the absolute position to geo-reference the laser measurements (Hutton *et al.* 2016). 3D point clouds can be generated very quickly by integrating the measurements from the two systems, which can then be used for modelling, mapping, and asset management, etc. (Williams *et al.* 2013). Its versatility and efficiency suggests its potential application in geohazard change detection and monitoring for remote areas. These works are normally completed by traditional surveying methods or by using terrestrial LiDAR systems, which provide high accuracy data, but are labour intensive. A main challenge to using MLS for these applications is its point cloud accuracy, which is often reduced as the navigation accuracy varies during data collection. The work discussed here attempts to address this problem, i.e. improving the geo-referencing accuracy for MLS point cloud data in a more efficient way, so that it can be used for accuracy demanding applications.

The navigation system used for MLS usually consists of a Global Navigation Satellite System (GNSS) and an Inertial Measurement Unit (IMU), as on the StreetMapper system from 3D Laser Mapping Ltd (3DLM). Additional sensors may include odometer and barometer. Although the integrated navigation system are generally high performance systems, it is inevitable that its accuracy could be reduced at times as GNSS positioning relies on receiving satellite signals, which can easily be blocked or disturbed. IMUs provide relative attitude measurements and accelerations that can navigate in absence of GNSS. However, these measurement can only provide the navigation solution for a short period of time during GNSS absence as IMU errors tend to increase very quickly over time without external corrections. Therefore, achieving high quality navigation in environments such as under thick tree canopy and urban areas, is a challenging task. As a result, the navigation accuracy does not always meet the expected accuracy requirement to geo-reference the MLS data. In such cases, the produced point cloud need to be corrected.

In many MLS applications, the point cloud accuracy is ensured by installing physical targets in the scan area, which act as control points (Puente *et al.* 2013). However, as the targets may have to be installed in remote and difficult to access locations, this method is both inconvenient and costly due to the amount of human labour required. Other methods, such as relative registration, are sometimes applied. Yet the improved correction efficiency comes at a cost of reduced accuracy compared to using control points (Bitenc *et al.* 2011, Kukko *et al.* 2012, Lauterbach *et al.* 2015, Gézero *et al.* 2017, Toth *et al.* 2017).

To reduce the workload required for effective and accurate point cloud adjustments and corrections, this work presents a semi-automated point cloud correction method for terrestrial MLS by firstly improving the navigation trajectory accuracy. The improved trajectory is then used to geo-reference the LiDAR measurements, correcting the changing error within the dataset. This procedure makes use of the feature information extracted from the point cloud which is then integrated within the navigation processing algorithm, i.e. Feature Extraction based Particle filter Point cloud Aiding (FEPPA). While feature matching methods have been proposed for registration in previous research such as in (Jende *et al.* 2016), accurate aerial LiDAR data was applied which does not suffer from the bad navigation errors that land MLS does. The algorithm here is tested on various sets of real world LiDAR data under bad GNSS conditions to demonstrate its point cloud correction capabilities. The FEPPA workflow eliminates most of the tedious manual work, reducing the typical correction

workload from around a week to under a couple of hours with comparable accuracy.

Section 2 will present the current background in point cloud matching as well as common navigation algorithms. Section 3 discusses the FEPPA methods, which includes extracting features from MLS generated point clouds and its integration with the navigation algorithm. Section 4 will present the test results using the proposed method. Its capabilities as well as limitations will also be discussed here. The final section concludes the work and discusses developments to adapt to future applications.

## 2. Background on MLS data processing

One of the main motivations for this work was to improve the MLS data accuracy to a level that could be used for geo-hazard change detection, which looks for the difference between two datasets of the same area but captured at different times (Mukupa *et al.* 2017). Change detection techniques have evolved greatly over the past years, from conventional surveying to remote sensing techniques and the recent growing adaption of LiDAR systems (Maghiar *et al.* 2016, Xiao *et al.* 2013, Williams *et al.* 2013), following a trend of improved data acquisition with higher density and efficiency.

Point cloud change detection can be achieved by computing the distance of a point from the comparison point cloud data to its relevant position in the reference point cloud, i.e. cloud-to-cloud distance measurement. Any non-zero values reflects the position difference between the 3D points, revealing changes in the physical conditions of the target area, such as those caused by landslides, slope failure, earthquake, road damage etc (Lato *et al.* 2009, Jaboyedoff *et al.* 2010, Lindenbergh *et al.* 2015). Therefore, the level of accuracy required for change detection depends on the type of changed being assessed. Change detection in geo-hazards are mostly major or obvious changes in the environment. Therefore the desired point cloud accuracy is within the decimetre level to identify these changes.

Most monitoring applications nowadays rely on using static terrestrial LiDAR systems or airborne MLS. However, static systems have limited data coverage and it is not always convenient to find a suitable scanning location. Airborne systems, on the other hand, suffer from low data density due to their distance from the ground and limited views of ground objects. The proposed correction method aims to enhance the point cloud quality derived from terrestrial MLS so that a more efficient monitoring workflow could be used for time critical applications in less accessible areas, addressing issues in modern road and railway environment monitoring (Chen *et al.* 2015, Network Rail 2016).

Controlling the point cloud quality from MLS before change detection analysis is essential, as the inconsistency between compared data includes two aspects:

- the change in position or shape of the object that is represented by the point, i.e. detected change;
- the relative position error between the two point clouds, i.e. noise and error.

Removing, or reducing, the relative error to a minimum is vital before carrying out infrastructure or geotechnical monitoring analysis. The following subsections will introduce the common methods for processing MLS data, including navigation data and 3D point cloud processing. The major issues that affect the point cloud accuracy and some common methods to address them will also be highlighted.

### 2.1. *Point cloud processing*

The basic components of an MLS typically consists of one or two 2D laser scanner(s), a GNSS/IMU integrated navigation system, an on-board computer and storage, as well as power supply. 2D laser scanners capture relative range measurements from the scanner to surrounding features which reflect the laser beam. To produce a 3D point cloud, the 2D range measurements need to be time synchronised and integrated with the navigation data using the basic steps as outlined below:

(i) Process the navigation data using suitable integration algorithms, e.g. Kalman filter or particle filter, which produces trajectory data that includes both the position coordinates as well as the attitude at a high rate.

(ii) Integrate the LiDAR range measurements with a 3D position and attitude by matching their time stamps to the navigation data, thereby projecting the 2D range measurement into a 3D coordinate system, i.e. generating a 3D point cloud.

(iii) Merging multiple scans: to increase the density of the point cloud, the same route is sometimes scanned more than once by the MLS vehicle driving from different directions, where each scan is known as a flightline (adopted concept from airborne applications). Merging multiple flightlines into one point cloud increases data density, hence captures more information of the environment. However, prior to merging, the data must be corrected for mismatches, i.e. errors, between the different scans to ensure that all data overlay consistently.

(iv) Point cloud corrections: merging the flightlines ensures that relative errors of each scan are reduced. If control points were used, the merged point cloud needs to be matched to the scanned control points. This step reduces the global errors and ensures that the data fits with the global coordinate system used in the location of the data capture.

Further data analysis can be carried out once the high quality 3D point cloud is produced. Usually this procedure tries to imply some "meaning" to the point cloud, such as data classification or object recognition, etc.

Due to the data processing procedure, the accuracy of the 3D point cloud is affected by both the measurement accuracy of the laser scanner as well as the performance of the navigation system. The laser scanner range measurement accuracies are generally around 0.5 - 2 cm usually with millimetre level precision; whereas the navigation data accuracy can be affected by various external conditions, producing errors from a few centimetres up to tens of metres. Therefore, the accuracy of the point cloud is largely affected by the navigation system performance (Jing *et al.* 2016).

In ideal conditions, navigation systems can achieve decimetre or even centimetre accuracy by using methods such as Real-time Kinematic GNSS positioning (RTK) (Tang *et al.* 2015). However, GNSS based navigation is easily disturbed by the environment causing the accuracy to reduce to metre level or worse. This introduces errors in the MLS point cloud, and even worse, introduces inconsistent errors within a single point cloud dataset as the navigation error changes. Therefore, improving MLS geo-referencing accuracy is cruicial before carrying out data analysis on MLS data.

Two common methods to improve point cloud accuracy are relative registration and control point geo-referencing. Relative registration requires the availability of another dataset of the same scan location with higher accuracy, known as the reference data. The data of concern is then matched to the reference data using methods such as Iterative Closest Point (ICP), Local Descriptor Histograms or other statistics that can characterise the point cloud and find matching points between two data (Rusu *et al.*

2008, 2009, Marden *et al.* 2012).

Registration is relatively fast, but the data can only be matched to minimise the global differences between the two datasets. Therefore, it is unable to eliminate local distortion and errors, such as those introduced by the navigation system. Furthermore, the performance of registration is highly dependent on the characteristics of the point cloud data, such as the geometry of the features within the point cloud and the movements of features between scans.

Geo-referencing using control points achieves unrivalled accuracy and reliability. Control points have to be installed prior to data capture and surveyed using instruments such as total stations or survey-grade GNSS to provide millimetre or centimetre level position reference (Puente *et al.* 2013). The point cloud data is then corrected by manually finding the targets in the point cloud and shifting the position of the points representing the target to match the surveyed position of the targets. These shifts build up a regression function which also shifts the other points in the data accordingly (TerraScan 2016). This method produces better overall accuracy as it is able to correct local errors within a data. It is especially useful when errors are inconsistent within a dataset, which is generally the case for data produced by the MLS. However, this correction process is known to be labour intensive and must be done manually.

### 2.2. *Navigation data processing*

Due to its effect on the point cloud data accuracy, ensuring the navigation data accuracy is cruicial to reducing error in the point cloud. Many algorithms had been proposed to integrate GNSS and IMU measurements for continuous high performance navigation, including Kalman filter (KF), extended KF and Particle Filters (PF) etc. Since its introduction in 1960 (Kalman 1960), KF has become an efficient tool to solving linear prediction and estimations problems, including tracking and navigation. It continuously measures and estimates the navigation system state, i.e. position, velocity, attitude and biases, while estimated states are continuously updated by incoming new measurements. Integrated navigation methods provides continuous navigation even when GNSS measurement is not available for a period of time.

High performance IMUs will be able to produce acceptable navigation during a longer period of GNSS outage. However, IMU measurements always grow exponentially over time and eventually exceed the requirement levels if GNSS positioning or other sources of positioning measurements are continuously unavailable. If additional external measurements could be integrated when they become available, the navigation performance could become more reliable. Due to the non-linear characteristics of external measurements, PF is considered more suitable and achieves better performance.

Particle filtering is a recursive Bayesian filtering method that integrates measurements from different sources and predicts the system states, which are represented by a large set of particles with associated weights, through sequential Monte Carlo estimation (Gordon *et al.* 1993, Ristic *et al.* 2004) . The system state vector $X_k$ is a discrete time stochastic model expressed as below:

$$X_k = f_k(X_{k-1}, v_{k-1}) \tag{1}$$

where $f_k$ is the non-linear function of the state $X_{k-1}$ and process noise $v_{k-1}$ at time

$k$, . The state vector $X_k$ is recursively updated from observations $z_k$:

$$z_k = h_k(X_k, n_k) \tag{2}$$

where $h_k$ is usually a non-linear function with measurement noise $n_k$. PF estimates the state $X_k$ at time $k$, given observations $z_{1:k}$ up to time $k$. At each epoch, the predicted probability density function (pdf) is updated through measurements to represent the posterior pdf of the current state. As it is usually impossible to obtain the true posterior pdf, $N$ particles are generated to represent a discrete approximation $p(x)$,

$$p(x) \approx \sum_{i=1}^{N} w^i \delta(x - x^i) \tag{3}$$

where particles $x^i$ are drawn from the approximate density, $w^i$ is the normalised weight of the $i$th particle, and $\delta(\cdot)$ is the Dirac delta function. As $N \to \infty$, the approximation should approach the true posterior pdf. A brief summary to each iteration of a typical PF procedure for navigation based on (Gustafsson *et al.* 1993, Ristic *et al.* 2004) is an below.

- *Initialisation*: $N$ particles $x_0^i (i = 1, ..N)$ are created to represent an estimated probability distribution of the initial system state $p(X_0)$, i.e. for navigation systems, each particle includes information on location, historic location, its current state, and a weight. All particles usually start with equal weights, unless other prior information is available. The weighted average of the particles represents the initial system state estimation.
- *Prediction*: the particles propagate through a prediction model, as Eq.1, the prior pdf of the new state estimate $X_{k|k-1}$ at time $k$ is obtained,

$$p(x_k|Z_{1:k-1}) = \int p(x_k|x_{k-1}) p(x_{k-1}|Z_{1:k-1}) dx_{k-1} \tag{4}$$

  where $Z_{1:k-1}$ is a set of all available measurements $z$ up to time $k$, $p(x_k|x_{k-1})$ is the probabilistic model of the state propagation defined by the system equation and estimation of noise $v_{k-1}$.
- *Update*: a set of new measurement $z_k$ at time step $k$ is obtained to update the prior via Bayes rule and obtain the posterior of the system state

$$p(x_k|Z_{1:k}) = \frac{p(z_k|x_k) p(x_k|Z_{1:k-1})}{p(z_k|Z_{1:k-1})} \tag{5}$$

  where $p(z_k|x_k)$ is the conditional pdf of $z_k$ given $x_k$, $p(x_k|z_{1:k-1})$ is the prior pdf at time step $k$ and $p(z_k|Z_{1:k-1})$ is the probabilistic model of the state propagation defined by the system equation and estimation of noise $v_{k-1}$. The likelihood of each particle $x_k^i$ , i.e. the weight $w_k^i$, is computed based on $p(x_k|Z_{1:k-1})$.
- *Resample*: A weight threshold is defined based on the system requirements and any particle $i$ with a weight below the threshold is redefined as $w_k^i = 0$. If the number of valid particles falls below a threshold, the particle cluster is resampled to regenerate $N$ particles based on $p(x_k|Z_{1:k-1})$. The particle weights are then normalised.

- Return to step 2 (prediction) or end process: the weighted mean of the particles gives the state estimation at the end of each iteration.

With the flexibility to integrate any data source as they become available, PF is widely adopted for the integration of measurements from different sources. The work presented here introduces an MLS data correction method by integrating information extracted from the LiDAR measurements in the point cloud data and integrating them with other navigation measurements using PF. However, integrating LiDAR measurements can be highly computationally expensive due to its high data rate and density. FEPPA allows a "very loose-coupling" scheme that corrects navigation only when necessary and reduces the computational effort. The integration of LiDAR measurements with a GNSS/IMU system takes advantage of the high accuracy GNSS performance in open sky areas, high accuracy IMU attitude measurements and periodic corrections constrained by the feature landmarks in the point cloud data.

## 3. The FEPPA method

Unstable performance of the onboard sensors of an MLS leads to two main problems: 1) inaccuracy in the produced point cloud; 2) inconsistent accuracy in the point cloud; as shown in Figure 1, which plots the distance, i.e. mismatch, between the two point clouds representing the same location, one perfectly geo-referenced and another generated by uncorrected navigation data. The blue crosses are the estimated standard deviation from the navigation filter which gives an indication of the navigation accuracy; the red triangle line shows the average mismatch between the points within a point cloud dataset generated during those epoches. The mismatch between the point cloud data varies as the navigation accuracy changes, thus indicating that the accuracy of the point cloud generated by MLS can vary depending how the navigation error changes. When point cloud data of different accuracies are merged together, the resulting data can show a "shadowing" effect such as seen in Figure 2. This inaccuracy and inconsistency leads to further problems when analysing data, therefore needs to be minimised.

The proposed FEPPA method improves the navigation accuracy through integrating the measurements extracted from point cloud features which contains both absolute location information (position within the absolute global coordinate system, i.e. WGS84) and relative location information (distance between the system and other features). The sections below will explain the methods used to extract feature information and the PF based integration method to enhance navigation as well as point cloud accuracy and consistency.

### 3.1. *Point cloud feature extraction*

Feature extraction and object recognition have been one of the main research areas for LiDAR point clouds. Mainstream methods include key point detection, classification and object recognition (Bosch *et al.* 2007, Zhong 2009, Hansch *et al.* 2014). These include identifying unique points of interest within a point cloud even at different scales, such as corners and sides of objects (Assfalg *et al.* 2007, Knopp *et al.* 2010). Object recognition is the procedure to detect and extract points representing an entire object or class of features. These methods rely on using unique descriptors to describe the geometry features as well as other laser measurements, e.g. intensity, which help

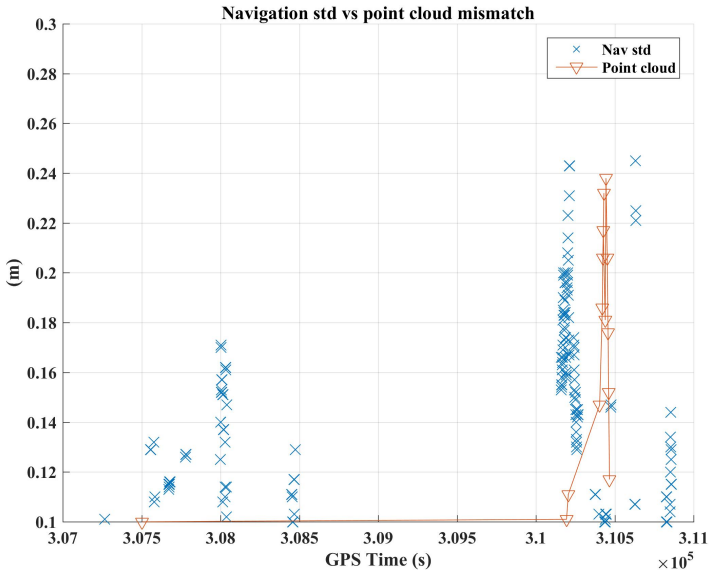



Figure 1.: Navigation performance effect on the point cloud (The Nav std line represents the estimated navigation error from the navigation filter; point cloud line represents the distance of the points between a MLS point cloud data compared to a reference dataset of the same location)

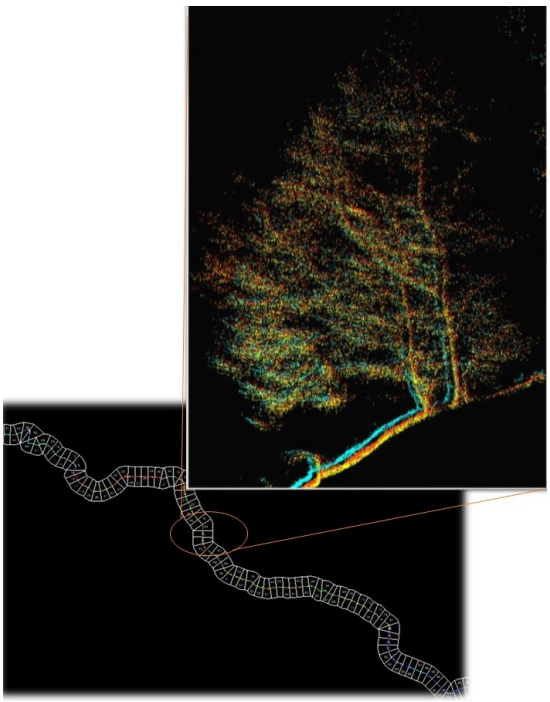


Figure 2.: Shadowing effects when accuracy is inconsistent between point cloud data

to recognise the object points among other data points in the point cloud (Lehtomaki *et al.* 2016, Yang *et al.* 2017). These descriptors specify each object with its unique identity which are pre-learned by the machine and features/objects are detected from

the point cloud using supervised or semi-supervised machine learning methods.

The first step of the FEPPA method relies on extracting measurements from stable features within the dataset. The object recognition/classification method used here is based on the CANUPO suite developed by Brodu et al (Brodu *et al.* 2012), which is available from the open source software CloudCompare. The CANUPO algorithm was developed especially to identify and classify complex scenes in the natural environment, such as vegetation, rock, gravel and water, etc. which is not addressed by many classification tools. The toolset handles large datasets and allows simple user input through semi-supervised machine learning procedures. The main idea of CANUPO is to characterise the local dimensionality properties of the scene at each point and different scales, i.e. how the point cloud objects look like at 1D, 2D or 3D scales. The learned characteristics help to identify objects or classify points at a later stage.

Within the context of this paper, the main application is the early detection and monitoring of geo-hazards. The datasets were collected along railways and roadsides. The CANUPO tool was tested to give relatively better performance for the types of datasets that were used, thus adopted as part of the workflow. It is used for the extraction of pole shaped infrastructure, including lamp posts and utility poles etc., among other similar natural environment features, e.g. trees.

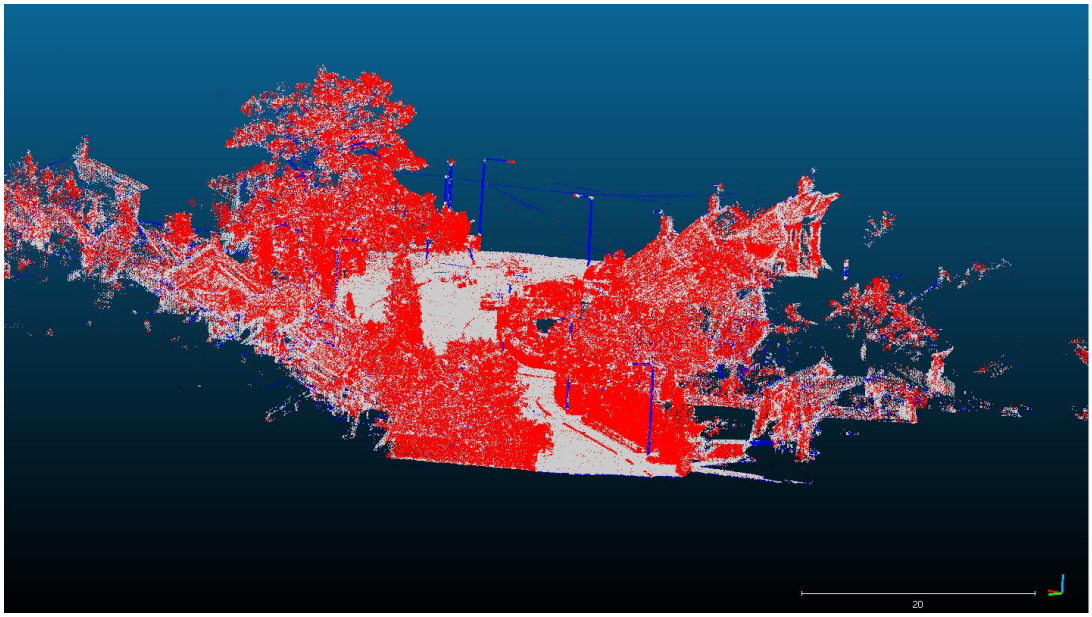


Figure 3.: Point cloud classification results using CANUPO
(Blue: pole class (target object); Red: other points not representing the target object; Grey: points that cannot be classified by the tool)

First of all, a descriptor is built in the CANUPO toolbox that can distinguish between the pole shaped infrastructures of interest (i.e. objects that will not change shape or position over time) and other objects in the point cloud. It is vital to exclude trees from the extracted feature dataset as they can introduce large ambiguities during processing. Once the descriptor is created, the point cloud datasets are run through the CANUPO toolbox, for pre-processing.

This step classifies the points in the point cloud into three classes: pole points, non-pole points and unidentified points, as shown in Figure 3. However, the classification results here are not perfect due to slight differences between the actual objects and

the descriptor. Therefore, the second step in the workflow, i.e. cleaning the classified points, is critical. The cleaning workflow was developed in Matlab by the author and deletes any false-positive points that have been classified as poles by the CANUPO tool, as shown in Figure 4. The cleaning workflow divides the classified target points (blue points from Figure 3) into $1m \times 1m$ grids, then filters out the data points if they were associated with low confidence by the CANUPO tool and the density of the points within the grid falls below a threshold, which is defined relative to the entire point cloud density.

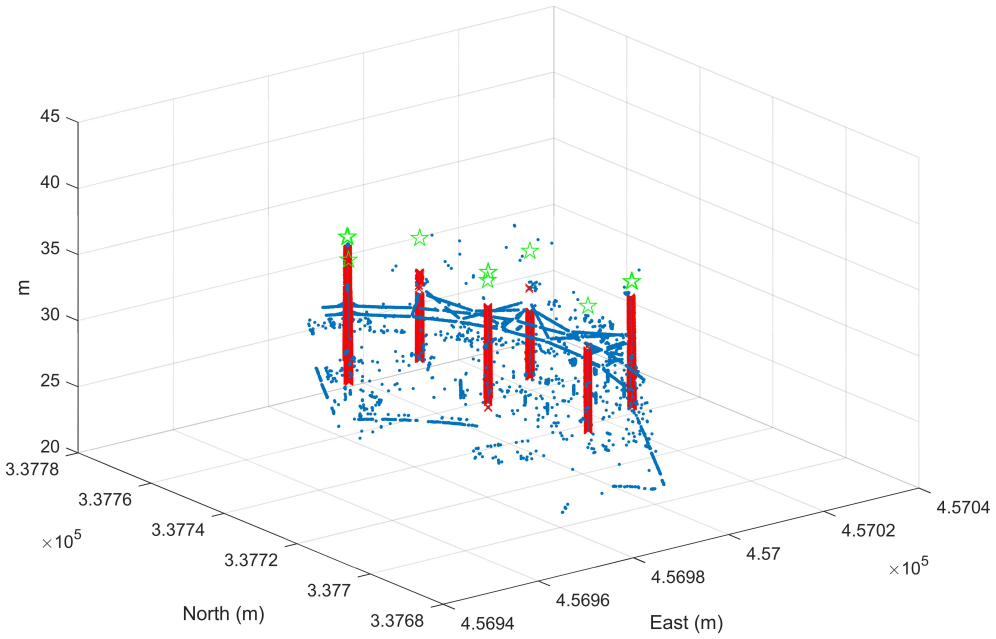



Figure 4.: Extracted features from classified points (red points indicate the final classified pole points, blue points are all the classified points from CANUPO (the blue points from Figure 3), green stars indicate the estimated 2D location of the pole object, plotted at the same height for visual indication)

Once the data has been filtered and the "clean" features have been identified, the Matlab script extracts further information from the points representing each object, i.e. $[t_{f_i}, x_{f_i}, y_{f_i}, \Delta h_{f_i}, intensity_{f_i}, \sigma_{f_i}]$, which represent the time of the data capture, Easting, Northing and height difference in OSGB36 grid coordinates (the geographic grid reference used in the UK), intensity of the points, and the confidence index of the CANUPO classification, respectively. The extracted information from a reference point cloud data is saved as the reference feature information; information extracted from the comparison dataset is saved as the comparison feature information. These information will be integrated into the navigation PF to reprocess the trajectory, which improves accuracy and consistency among the datasets captured at different times.

### 3.2. *Particle filtering based navigation aiding*

The navigation trajectory used for creating the original point clouds is generated from integrating GNSS positioning data and IMU measurements through loosely coupled KF, which is carried out by the software given by the navigation system provider. However, as discussed, navigation trajectory does not have consistent accuracy along the entire route. This is predominately due to changes and variations in the GNSS satellite condition and environment when datasets are captured at different times of the day or year. Hence changing the accuracy of the point cloud data. Therefore, it is important to ensure that different point cloud datasets are corrected to the same accuracy level before they can be used to carry out change detection or other similar analysis. The FEPPA algorithm proposed here reprocesses this trajectory by integrating the feature information extracted in Section 3.1.

The algorithm follows the basic PF procedures described in Section 2.2 where the system state model $X_k$ represents the navigation system position information, the propagation vector follows the measurements obtained from the IMU and the extracted feature information is integrated to provide measurement updates as explained in more detail below. The process noise $v_k$ is normally distributed with a mean of 0 and standard deviation given based on the system specifications for position and heading errors, the measurement noise $n_k$ is normally distributed with a mean of 0 and a standard deviation following the current estimated navigation error which is derived from comparing the feature locations and the current system location. A basic workflow chart is given in Figure 5.

As outlined in Section 3.1, the extracted features of interest should remain stable over time and thus retain the same location information. Inconsistent navigation accuracy is detected when the same features are geo-referenced at slightly different locations. Therefore, the position difference of the reference features $f_{ref}$ and the comparison features $f_i$, i.e. $\Delta dis_{f_i}^{f_{ref}}$, will be used as an observation measurement integrated into the PF to constrain the navigation error in post-processing. The steps of the FEPPA workflow is described as below:

1) Estimate the centre position of each of the $N$ features $\{x_{f_i}, y_{f_i}, \Delta h_{f_i}\}_{i=1,...,N}$ from the data points,

$$x_{f_i} = mean(x_1, x_2, ..., x_q); \tag{6}$$

$$y_{f_i} = mean(y_1, y_2, ..., y_q); \tag{7}$$

$$\Delta h_{f_i} = z_i^{max} - z_i^{min}; \tag{8}$$

where $f_i$ is the feature number, and $q$ is the number of points representing feature $f_i$. The time stamp of each point representing each feature $t_q^i$ is saved as a separate array for reference. The time reference is essential as the navigation error and point cloud error are correlated by time.

2) Compute the 2-D geometry index of the group of features extracted at the current epoch, i.e. $FDOP$ (Feature Dilution of Precision), adopted from the GNSS

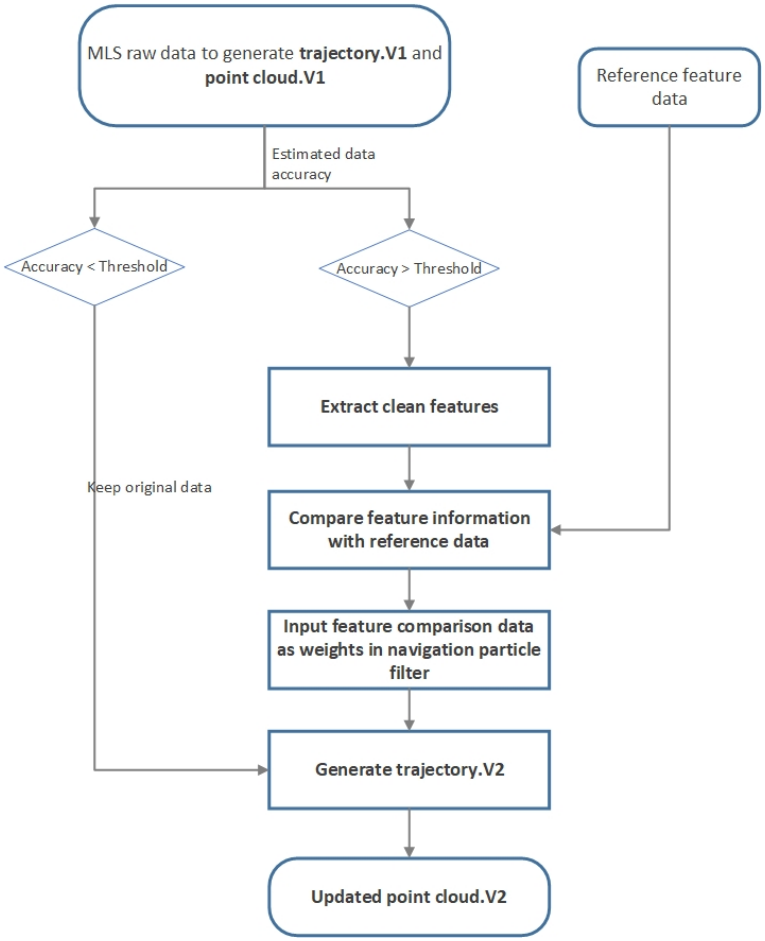



Figure 5.: Flowchart showing the basic steps of the FEPPA method

positioning DOP concept:

$$FDOP = \sqrt{\sigma_x^2 + \sigma_y^2} \tag{9}$$

where $\sigma_x^2 = \frac{x_i - x}{R_i}$, $\sigma_y^2 = \frac{y_i - y}{R_i}$, $R_i = \sqrt{(x_i - x)^2 + (y_i - y)^2}$, $x_i$ and $y_i$ are the 2D coordinates of the features, $x$ and $y$ are the 2D coordinates of the system at the current epoch. At each epoch, the features selected for computation are features whose data points were captured within a time threshold of the current epoch, denoted as "visible" features in the text below. The selection of "visible" features is to ensure that the navigation accuracy representated by the features are similar to the accuracy of the current epoch. $FDOP$ provides an estimated indication of whether the extracted feature information could improve the trajectory accuracy at the current epoch. Small $FDOP$ indicates good geometry layout surrounding the MLS system, i.e. evenly spread out features along the travelling path on both sides, therefore these features can be used to give a better accuracy when post-processing the trajectory; larger $FDOP$s indicate bad geometry, either due to low number of available features, or that features are clustered close together, thus less capable of giving a better accuracy estimate.

3) Initialise the PF: initialise a set of particles $p_m$ around the initial position with a level of uncertainty $\sigma_{pt}$ based on the trajectory data, where $m = 1, ..., M$ , i.e. $M = 1000$, $w_k^m = 1/M$ is used here.
4) Propagate the particles based on the measurements from the navigation system, i.e.

$$p_x^{m_t} = p_x^{m_{t-1}} + \Delta_x; \tag{10}$$

$$p_y^{m_t} = p_y^{m_{t-1}} + \Delta_y; \tag{11}$$

$\Delta_x$ and $\Delta_y$ should be derived from the navigation system measurements. However, in this case, the commericial navigation system used did not provide raw IMU measurements. Therefore, $\Delta_x$ and $\Delta_y$ are derived from the produced navigation trajectory, i.e. the position and heading change between the current and previous epoch with added process noise $v_k$ to allow for errors in the produced trajectory.
5) The weight of each particle $w_k^m$ is reassigned in this step after comparing the distance between each particle and the location of "visible" features (i.e. $R_{p_m}^{f_i}$) and the distance to the same features in a reference dataset (i.e. $R_{p_m}^{f_{ref}}$) :

$$R_{p_m}^{f_i} = \sqrt{(p_x^m - f_{i_x})^2 + (p_y^m - f_{i_y})^2} \tag{12}$$

$$R_{p_m}^{f_{ref}} = \sqrt{(p_x^m - f_{ref_x})^2 + (p_y^m - f_{ref_y})^2} \tag{13}$$

$$\Delta R_{p_m} = R_{p_m}^{f_{ref}} - R_{p_m}^{f_i} \tag{14}$$

If the difference $\Delta R_{p_m}$ is over a predefined distance threshold, depending on the estimated navigation error, the particle weight $w_k^m$ is simply give $w_k^m = 0$; otherwise, $w_k^m$ is the inverse of the difference in the distance, i.e.

$$w_k^m = \frac{1}{\Delta R_{p_m} \cdot a} \tag{15}$$

where $a$ defines how quickly the difference in the distance changes the weight. The position and height information of the points representing each feature is used to justify the matching between the current feature $f_i$ and reference feature $f_{ref}$. If there should be a mismatch due to several features being too close together, the range difference will appear to be sufficiently larger than expected, thus giving a lower weight or 0.
6) Resampling: to avoid the particles clustering towards a wrong location in the process or particle impoverishemnt, both of which are common problems in PF if the wrong weighting and resampling threshold is used, the particles are checked

for resampling after every iteration. If there are any invalid particles, i.e. any particles with $w_k^m = 0$, particles with a higher weight $w_k^m$ is given higher probability to regenerate new particles, while low-weighted particles will have less opportunity to generate new particles, to replace these invalid particles. A system noise is added to the regenerated particle cluster. If there are no remaining valid particles, a new particle cluster is generated based on the current estimated position. As the feature range measurements can constrain the particles to quite a small area, resampling will ensure there are always valid particles to continue the iteration.

7) Positioning estimation: the updated position estimation of the navigation system, i.e. the MLS position, is computed from the weighted average of all particle positions, i.e.

$$\{x_{update}, y_{update}\} = \{\frac{\sum_{m=1}^{M}(p_x^m \cdot w_m)}{\sum_{m=1}^{M}(w_m)}, \frac{\sum_{m=1}^{M}(p_y^m \cdot w_m)}{\sum_{m=1}^{M}(w_m)}\}; \tag{16}$$

The PF is continued until the whole trajectory has been reprocessed using the feature information; generating a new trajectory for geo-referencing, i.e. the PF results.

8) The PF trajectory is used to update the GNSS measurement data, which is then reprocessed by GNSS/IMU integration software provided by the navigation provider, producing the final FEPPA trajectory. The point cloud is re-generated using the FEPPA trajectory.

The final step is required here as we did not have access to the raw measurements produced from the navigation system, which was a limitation from the commerial hardware and software package. The PF algorithm could only update the position estimations, thus need to be reprocessed in the integration software to provide improved position and attitude outputs before the data can be used to geo-reference the point cloud. The PF integration method was chosen here due to that the integrated feature measurements cannot be easily modelled for KF due to their changing and unpredictable nature. Yet integration of these measurements in PF is a straightforward procedure.

The FEPPA process is designed based on the assumption that the difference in the navigation accuracy is reflected by the relative distance shift between the same features extracted from the corresponding point cloud dataset. The distance shift is estimated and used to reprocess the trajectory, aiming to reduce the error. However, it is recognised that the distance shift between the features does not fully represent the actual navigation error, due to errors being either enlarged or reduced by the changing attitude measurements of the navigation system. Therefore, the coefficient $a$ and cut-off threshold are introduced to provide an adaptive weighting scheme. This reduces the constraint of the measurement update and allows for the slight difference between the actual navigation error and the distance shift between features.

Analysis on several datasets are given below to discuss the performance of the FEPPA workflow and some considerations when using the method.

## 4. Results and analysis

This section will analyse the performance of the FEPPA workflow on two datasets captured in different environments and using navigation systems of different performance levels. The point cloud correction performance of the FEPPA method will be compared to registration methods and also standard MLS data processing workflows using the commercial software suite, TerraScan/TerraMatch. The results of the correction is compared to the reference point cloud using cloud-to-cloud distance computation to see how well it matches the desired data. The efficiency of the workflow and also minor problems that need to be considered are also discussed.

### 4.1. *Simulation data and processing*

Two sets of MLS data were captured for the analysis of the proposed workflow. The first dataset was captured along the Nottingham tram line, which simulates railway and roadside scenarios in terms of infrastructure, covering both suburban and urban environments. This data simulates the environment which requires infrastructure and asset monitoring. The second dataset was captured along the A52 highway road near Bingham, England, which has a mostly rural environment and simulates the a rural site which requires geohazard monitoring.

The MLS system used for the tram data capture was the ROBIN system from 3D Laser Mapping Ltd and the A52 road data is captured using the StreetMapper system also from 3D Laser Mapping Ltd, as shown in Figure 6. The StreetMapper IV system consists of a Riegl VUX-1 laser scanner and an IGI TerraControl navigation system, which consists of a NovAtel GNSS receiver and the IGI Fibre optic gyros (FOG) IMU-IIe. The ROBIN system consists of a Riegl VUX-1HA scanner and the same GNSS receiver but integrated with a MEMS (Micro-Electro-Mechanical System) IMU, which is a lower grade IMU compared to FOG IMUs. Specifications for the VUX-1 scanner is listed in Table 1. The scanner is installed facing backwards of the vehicle with a 50° angle to the horizontal plane, allowing the scanner to capture details of the ground and along both sides of the road. Specifications for the two navigation systems are listed in Table 2.

(a) ROBIN MLS system

(b) StreetMapper MLS system (the camera system in the picture was not used in this particular test)

Figure 6.: Systems used for data capture

Table 1.: RIEGL VUX-1 laser scanner performance

| | **Max range** | **FOV** | **Meas. rate** | **Resolution** | **Accuracy** | **Precision** |
|---|---|---|---|---|---|---|
| **VUX-1** | up to 920m | 330° | up to 550kHz | 0.001° | 10mm | 5mm |

Table 2.: IGI TerraControl navigation system

| | **Pos (m)** | **Vel (m/s)** | **Heading (°)** | **Pitch/Roll (°)** |
|---|---|---|---|---|
| **FOG-IIe** | 0.02 | 0.005 | 0.03 | 0.015 |
| **MEMS** | 0.02 | 0.005 | 0.01 | 0.004 |
| | Gyro bias ($°/hr$) | Gyro random walk ($°/hr^2$) | Acc bias (mg) | |
| **FOG-IIe** | 1 | 0.07 | 0.3 | |
| **MEMS** | 0.03 | 0.0005 | 0.1 | |

For testing and evaluation, a good quality reference dataset is generated. The data was captured in an environment without any long tunnels or thick foliage which may potentially block the sky view and reduce navigation accuracy significantly. For both datasets, the navigation and point cloud data were processed to its best possible solution using the standard workflow that would be used to deliver data to clients, i.e. the final navigation accuracy is centimetre level for over 95% of the entire trajectory and the point cloud data was manually corrected and processed using the TerraSolid software suite with high quality calibration. These results were saved as the reference data.

To introduce errors into the dataset, sections of GNSS measurements are deleted

from the navigation data to simulate loss of GNSS signals. The period of deleted measurements varies for the different navigation system used, but both produced navigation errors of over a metre. For the tram data, a period of 90 seconds was deleted resulting in around 4m error; 4.5 minutes of GNSS measurements were deleted from the A52 highway data producing a maximum error of around 2m. The resulting trajectory and point cloud data were saved as the comparison data.

Pole features were extracted from the point cloud using the feature extraction workflow described in Section 3.1 for both the reference and comparison data. For both datasets, the most common features used were the lamp posts and utility poles. The trajectories of the comparison data was reprocessed using the FEPPA workflow to reduce the trajectory errors, which can then be used to regenerate the comparison point cloud so that it matches the reference dataset. Navigation error here is referred to as the difference between the comparison or reproduced trajectory and the reference trajectory. The point cloud difference were measured by the Cloud-to-cloud distance (C2C) tool provided in CloudCompare and results are displayed in colour scale.

### 4.2. *FEPPA performance and comparison*

To analyse the correction performance of the FEPPA workflow, registration methods were also used to match the comparison and reference data and the results were compared to the FEPPA outputs. A series of tools, including Spin image registration developed by 3DLM, registration tools available in Point Cloud Library, and registration tools in CloudCompare, were tested for comparison. Although with slightly different accuracies, all tools presented similar pros and cons for the data that were tested on. The results from the ICP tool in CloudCompare is given here as an example for typical problems when using registration for large MLS datasets. The CloudCompare ICP tool usually gives quick and efficient results with reasonable accuracy in general registration problems (Wu *et al.* 2018). However, as with all registration methods, it is unable to deal with changing accuracy within the same point cloud dataset.

The ICP matching results for a small section of the tram data is shown in Figure 7. Here the point cloud data registered using ICP, shown in orange-yellow points, is overlaid on top of the reference data, shown in blue. The change in error levels of

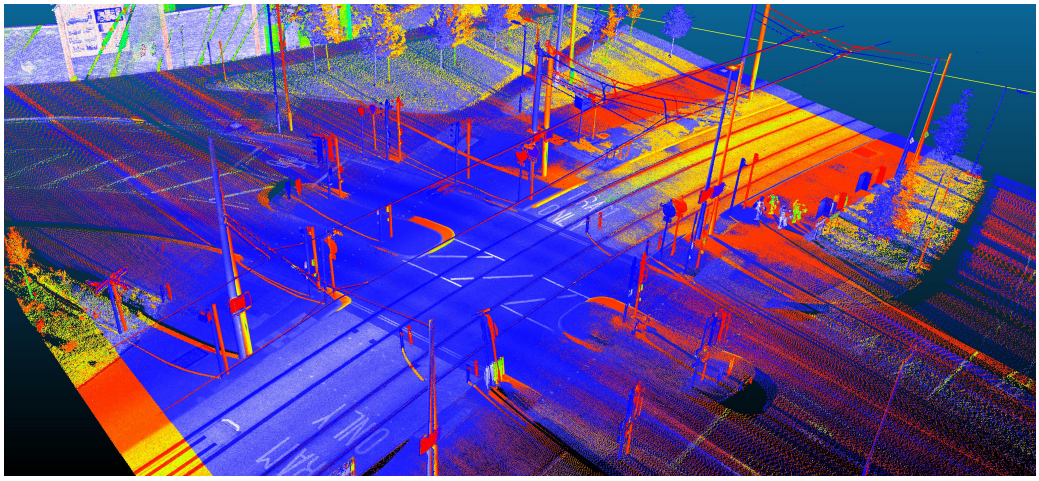


Figure 7.: ICP performance on the tram data (matched data compared to the reference data)

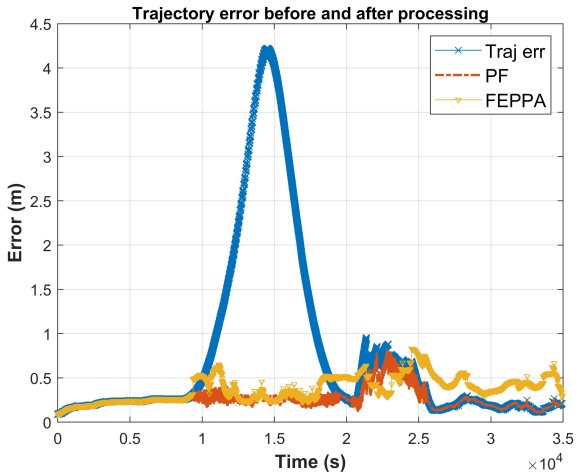



Figure 8.: Navigation correction results

the MLS data could be clearly seen, where the features between the two data on the left side of the data are much more closer than the features on the right-hand side of the data. This reflects one of the main problems of using rigid registration methods for MLS point cloud geo-referencing, where the errors are inconsistent throughout the data.

The tram data will be discussed in more detail below, due to its better visualisation effects. Therefore, enabling clearer comparison.

The FEPPA workflow corrects the inconsistent error within the same point cloud dataset by reprocessing the navigation data using error estimation extracted from features in the point cloud. Figure 8 shows the error of the original comparison navigation data compared to the navigation error after being processed by two workflows, i.e. PF and FEPPA. The PF method is actually a part of the FEPPA workflow but only reprocessing the data up to Step (7) outlined in the workflow in Section 3.2. Results from using the PF method only is shown in the red dotted line.

The FEPPA results, plotted in yellow lines, shows the results processed by the complete FEPPA workflow, i.e. outputs after Step (8). As mentioned previously, Step (8) is required here due to not being able to access the sensor raw measurements. This step reprocesses the data to update both the position and attitude outputs. Although it seems that the position error of the trajectory is lower by using PF only, the attitude measurements are not updated according to the new updated positions, the generated point cloud data can appear to be "messy", as seen in Figure 9.

Table 3.: FEPPA processing results

| | | **Before correction** | **PF only** | **FEPPA** |
|---|---|---|---|---|
| Tram data | **RMS (m)** | 1.051 | 0.022 | 0.119 |
| | **Max (m)** | 4.224 | 0.183 | 0.261 |
| Highway data | **RMS (m)** | 0.919 | 0.145 | 0.441 |
| | **Max (m)** | 1.880 | 1.669 | 1.314 |

Table 3 shows the trajectory error before corrections, after being processed using PF only and after being processed using the FEPPA workflow. Although the position

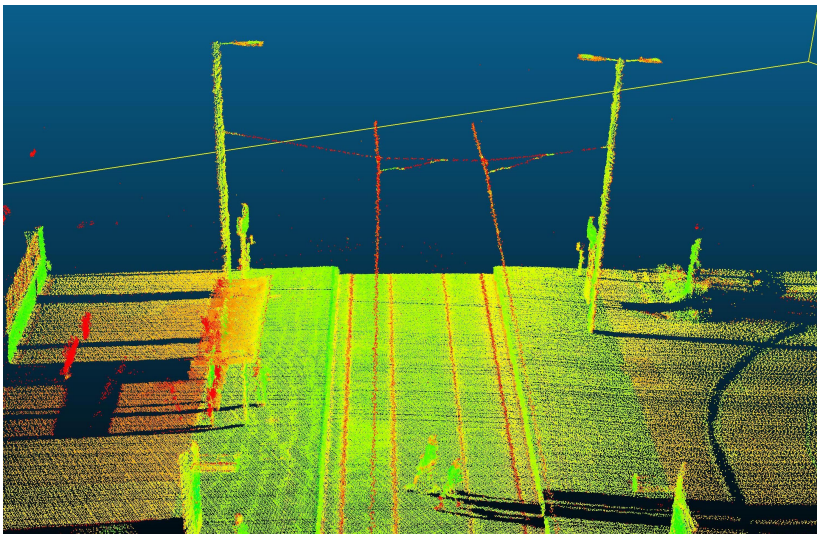
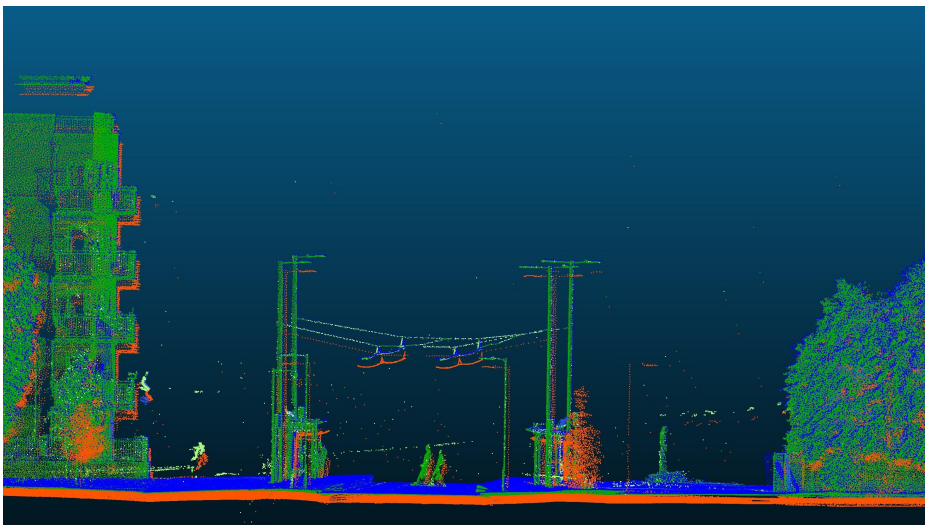


Figure 9.: PF point cloud

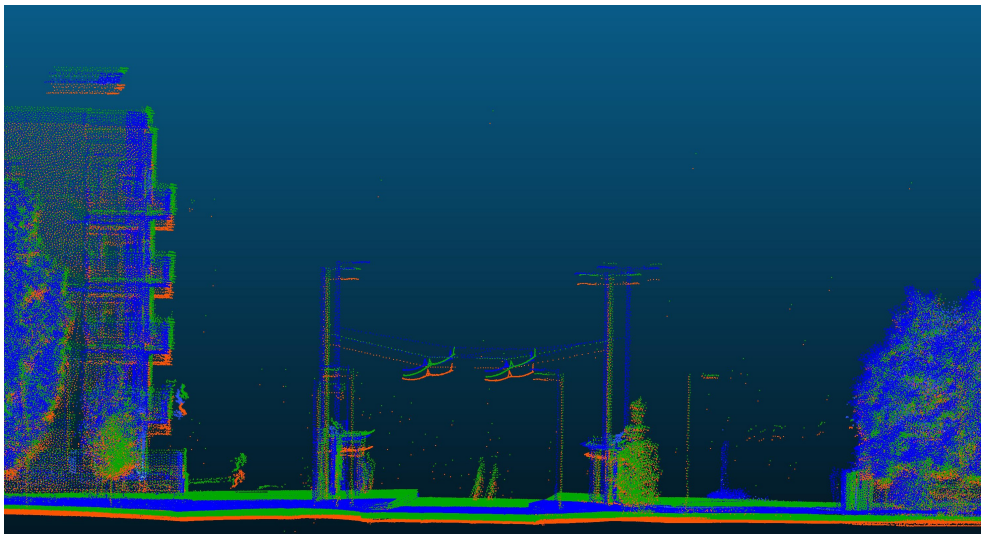
errors from the FEPPA results are higher compared to the PF results, this result is processed by the navigation system's software, therefore provides an updated position and attitude result for generating point clouds. RMS indicates the Root Mean Square Error, calculated by,

$$RMS = \sqrt{\frac{\sum_{n=1}^{N}(\hat{x}_n) - x_n)^2}{N}} \tag{17}$$

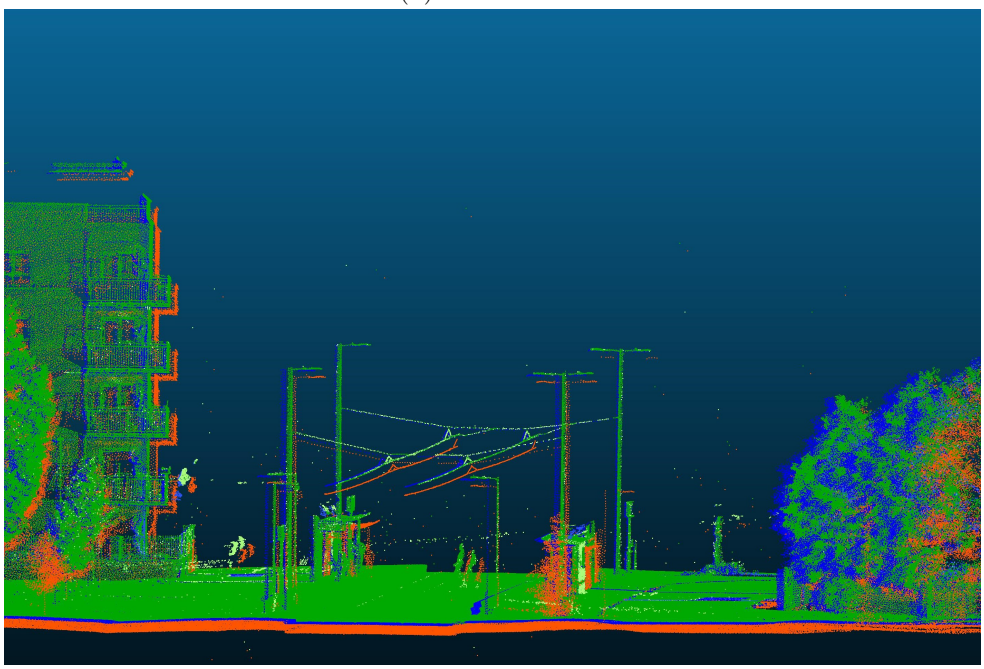
To compare the point cloud correction results, Figure 10 shows the a section of the final point cloud after being processed by three different methods, i.e. the proposed FEPPA workflow, ICP registration method and a control point method. Each corrected point cloud (shown in green) is overlaid on top of the reference point cloud (shown in blue) and the original comparison point cloud (shown in orange).



(a) FEPPA method



(b) ICP method



(c) Control point method

Figure 10.: Point cloud correction results after applying different correction methods (Green: the corrected point cloud; Orange: uncorrected point cloud; Blue: reference point cloud)

Figure 10(a) shows the reproduced point cloud using the FEPPA method, which gives an alignment difference of around 2cm overall. Figure 10(b) shows the results given by the CloudCompare ICP registration tool as described above. Results show that the error in the comparison data has reduced, but the performance decreases towards the edge of the data.

Figure 10(c) shows the results of the control point correction method, which is a more or less standardised manual process used to correct MLS datasets, discussed in Section 2.1. It usually gives the best results, but require long manual work to

survey control points and manually matching the point cloud to the control points. The standard workflow uses the TerraScan/TerraMatch commercial software (from the Terrasolid software suite (TerraScan 2016)) to input surveyed control point positions and manually find those control points in the point cloud, the user must tell the software the exact points representing the control point. The TerraMatch software stores this information for each control point and works out a regression model to correct the point cloud so that the position of the points matches the surveyed control points. To save the surveying time, here the extracted features used in the FEPPA process are used as control point inputs. The point cloud is then matched using the standard workflow described above.

Note that the results from the control point matching method does not seem to give better results than the FEPPA method. This is due to that in the actual standard workflow, the control point positions need to be surveyed to the millimetre accuracy level before being input into the software. Here, the position of the features are extracted and estimated from the reference dataset using the feature extraction process described in Section 3.1. Therefore introducing inconsistent errors exist during extraction and estimation. The quality of the regression model produced by the software could be reduced by such inputs, thus ouputing lower quality correction results. The comparison of the results in (a) and (c) simply shows that the FEPPA process is able to deal with measurement errors in the extracted features better than the TerraMatch software, which do not expect bad measurements as control inputs. It is not to state that the TerraMatch will not give good results when the standard procedure and good quality control points are used.

### 4.3. *FEPPA accuracy analysis*

The quality of the feature extraction results and the quantity of the features extracted are crucial to the performance of the FEPPA method. The effects of feature quantity on the processed navigation accuracy can be seen in Figure 11. The navigation data error after processing (in blue '·') is plotted against the number of features used during each epoch to correct the data (in red 'x'). The level of error is reflected by the density of the blue dots, i.e. darker regions indicate a high density of dots clustered in the low error section, lighter regions indicate a more spread-out error distribution, which implies a higher proportions of large errors. The red 'x' shows the number of features used in the FEPPA workflow for reprocessing the data. The plots shows that the resulting navigation error was relatively larger when less features were used in the correction workflow, whereas the navigation error reduced when more features were used.

Although the FDOP value is affected by both the number of features used and the geometry of the feature locations, for the analysed datasets here, it is mostly affected by the number of features, as the extracted features were mostly spread out on either side of the navigation system in a similar pattern. The produced FDOP value at each epoch played a role in changing the weighting scheme in the update step. Low FDOP values indicated more "visible" features, thus more measurements integrated into the workflow, which means that the particles' weights would change more severely if the measured range difference was high compared to when large FDOP values were computed.

Further to the number of features, two thresholds also changed the processing results, i.e. the time difference and distance difference threshold, as described in Step 5 in

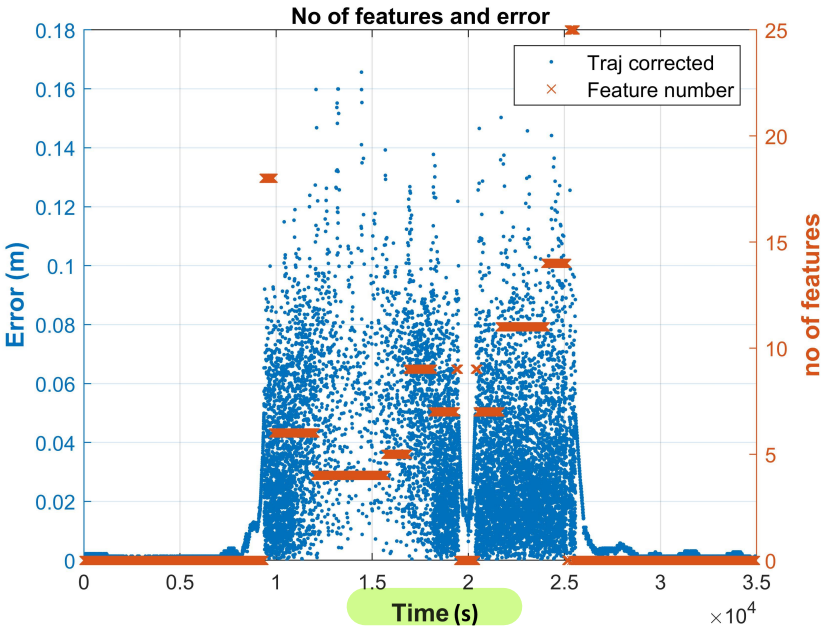



Figure 11.: Comparison between the number of features and the resulting navigation accuracy

Section 3.2. . Table 4 lists the navigation processing results for three test datasets after applying the FEPPA workflow. "Threshold $n$" indicates a different pair of thresholds selected.

The time threshold, which is used to select "visible" features at each epoch, was $8s$ for Threshold 1, 2 and 3, $5s$ for Threshold 4 and 5, and $10s$ for Threshold 6. The distance threshold, which is the predefined threshold for particle weighting, is set at $0.5m$ for Threshold 1 and 6, $0.3m$ for Threshold 2 and 4, and $0.1m$ for Threshold 3 and 5. The "% processed" column indicates the percentage of epochs that produced valid outputs out of the total epochs that required processing. Invalid outputs are regarded as epochs where the produced results were unreasonable or could not be processed due to insufficient feature information available.

Table 4.: PF processing results with different parameters

| Data | | Mean (m) | Max (m) | % processed |
|---|---|---|---|---|
| **Data 1** | Before corrections | 0.919 | 1.880 | |
| | Threshold 1 | 0.232 | 1.646 | 99.9% |
| | Threshold 2 | 0.145 | 1.669 | 99.8% |
| | Threshold 3 | 0.107 | 1.715 | 97% |
| **Data 2** | Before corrections | 1.967 | 9.678 | |
| | Threshold 4 | 0.856 | 9.678 | 60.9% |
| | Threshold 3 | 0.473 | 9.678 | 60.4% |
| | Threshold 5 | 0.021 | 0.164 | 31.1% |
| **Data 3** | Before corrections | 0.543 | 4.038 | |
| | Threshold 1 | 0.032 | 0.648 | 100% |
| | Threshold 2 | 0.021 | 0.386 | 84.8 % |
| | Threshold 6 | 0.033 | 0.678 | 100 % |

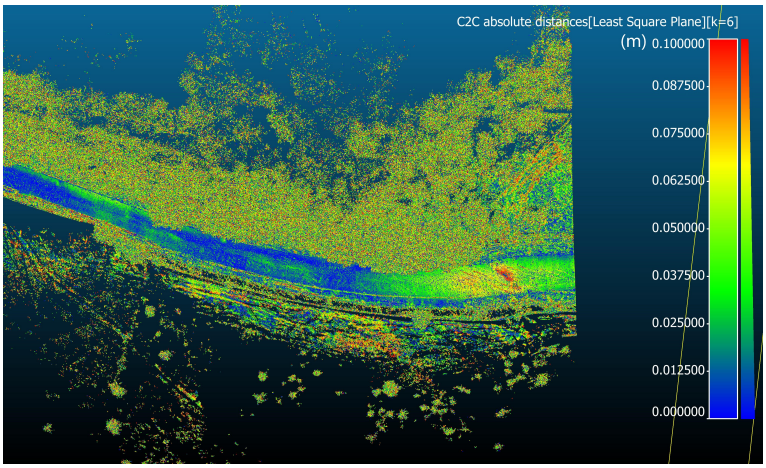
Both the time and distance threshold presents a trade-off between accuracy and percentage of valid outputs. As mentioned Section 3.2, the "visible" features are selected by the time difference between the current epoch and the time when the data representing the feature were captured. Due to that the navigation errors continously change over time, ideally only features captured in the same epoch can reflect the same level of error. However, this limits the number of features that can be used and therefore reduces the update measurement input to help correct the trajectory. Increasing the time threshold would mean that more "visible" features could be extracted, i.e. reducing the $FDOP$ value, integrating more measurements in the filter and potentially producing better accuracy. However, if the time difference continues to grow, the additional included features may no longer reflect the navigation error of the current epoch. In the results for Data 2 and Data 3, a decrease in the accuracy starts to show when the time threshold increases. Overall for all tested data, $8s$ is a tolerable threshold. However, this also depends on the speed of the navigation system and the density of features in the environment.

For the distance threshold, a tighter threshold, i.e. short distance, should be applied to improve accuracy, which gives significantly higher weights to particles that have better range estimations to the features and lowers weights to others. However, introducing a tighter threshold could reduce the number of valid particles left after each updating step. Therefore, less capable of dealing with situations when errors are high in the measurement input. This could reduce the percentage of valid output and cause the problem of particle impoverishment more frequently. This could be seen in the comparison of results for Data 1, Threshold 3 and 4 for Data 2, and Threshold 1 and 2 for Data 3, where tighter distance thresholds produced better average results but also reduced the total number of valid epochs. As such, the actual threshold should be tuned based on the estimated data accuracy, desired accuracy and feature extraction quality to maximise the percentage of valid processed data while also reducing the navigation error.

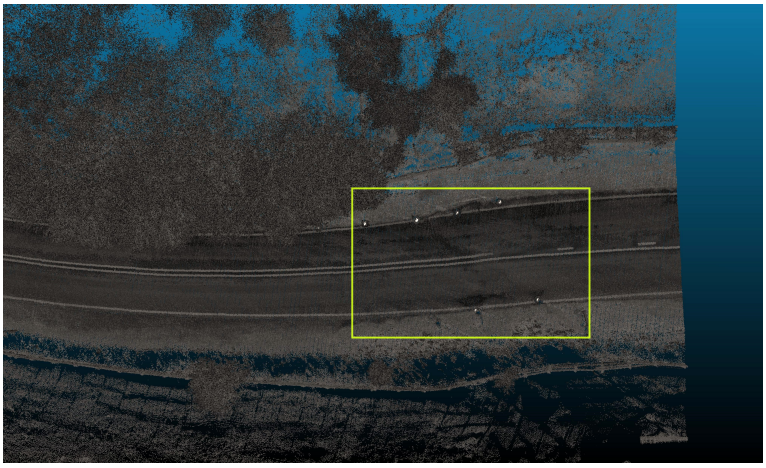
Finally, an example of change detection result is given below based on the FEPPA corrected data. Change detection is performed by carrying out C2C computation between the processed comparison data and the reference dataset. Figure 12a shows a section of the highway data where a change on the ground (hightlighted in red) had been detected on a section of the road. Figure 12b shows the comparison point cloud data coloured by intensity. Here we can see the cause of the change, i.e. a crack on the road showing up as a diagonal marking along the road. Theses results indicate the potential of using the FEPPA workflow to reduce the workload for processing MLS point cloud data and being able to carry out data analysis such as these more quickly.

## 5. Conclusions and future work

LiDAR systems, especially mobile LiDAR, have become increasingly popular for a range of applications, including surveying, monitoring and modelling. However, the methods to correct and control the 3D point cloud accuracy and quality from a mobile laser scanning system has been challenging due to demanding manual workloads. This paper introduces an improved method to correct the MLS navigation accuracy as well as the point cloud quality by using the feature information extracted from the point cloud data. It is demonstrated through two different types of data: 1) a suburban/urban area with a variety of infrastructure and clear objects; 2) a highway in the rural environment with overgrowing trees on either side, which causes prob-




(a) Road change detection (comparing data corrected by FEPPA to a reference dataset)



(b) Corrected point cloud of the road shown in intensity

Figure 12.: Change detection results after applying FEPPA (colour bar indicates difference in metres)

lems in navigation and conventional point cloud registration. The proposed FEPPA method first extracts useful features from the point cloud data and integrates their location information into the navigation PF algorithm to update the trajectory. The final step reproduces the point cloud using the updated trajectory. Results show that the workflow is able to reduce the mean error by around 56% - 88% and the maximum error by around 93%. One of its main advantages compared to conventional MLS data processing is the huge reduction in processing time and effort. As most steps here are automated, less skills and time effort is required to produce a reliable point cloud data.

However, a few issues were briefly mentioned in the paper but not address as part of this work. First of all, the FDOP was measured as part of the FEPPA workflow. Yet, in the analysed data, it did not reflect much information on the geometry and was more of an indication of the number of features available. This was limited to the environment of the data and the nature of the features were placed along the road/rail side. The algorithm was also used only to correct the 2D position of the trajectory due to that thee features were mostly spreadly spread out in the horizontal plane relative to the navigation location. As there was insufficient height change in

the features, it was not used to correct the height information in this case. In future simulations, analysis should be carried out on how the 3D geometry of the features affect the accuracy. Secondly, the reprocessing procedure in the final step of FEPPA allows the software to produce a smoothed data and therefore generate clear point clouds. However, as seen in the results' plot, the smoothed data introduces some errors. This is due to the way the software integrated GNSS and IMU data. However, in this case the IMU provider did not provide an open source IMU data, therefore limited the possibility of integrating the raw measurements in the PF algorithm. Such effects should be investigated in future work with open source IMU measurements. Finally, the feature errors are not linearly correlated to the navigation error due to the changing attitude measurements of the IMU. How the attitude affects the error and how the integration should tune parameters based on the attitude measurements should also be investigated in future work.

## References


Assfalg, J., Bertini, M., Del Bimbo, A. and Pala, P., 2007. Content-based retrieval of 3-D objects using spin image signatures. *IEEE Transactions on Multimedia*. 9(3), 589–599.

Bitenc, M., Lindenbergh, R., Khoshelham, K., Waarden, V. and Pieter, A., 2011. Evaluation of a LiDAR land-based mobile mapping system for monitoring sandy coasts, *Remote Sensing*. 3(7), 1472–1491.

Bosch, A., Zisserman, A. and Munoz, X., 2007. Image classification using random forests and ferns. *2007 IEEE 11th international conference on computer vision,IEEE*. 2007, 1–8.

Brodu, N. and Lague, D., 2012. 3D terrestrial lidar data classification of complex natural scenes using a multi-scale dimensionality criterion: Applications in geomorphology. *ISPRS Journal of Photogrammetry and Remote Sensing*. 68, 121–134.

Chen, H.M., Ulianov, C. and Shaltout, R., 2015. 3D laser scanning technique for the inspection and monitoring of railway tunnels. *Transport problems*. 10(SE).

Gézero, L. and Antunes, C., 2017. A registration method of point clouds collected by mobile LIDAR using solely standard LAS files information. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*. Vol. XLII-1/W1, 2017.

Gordon, N.J., Salmond D.J. and Smith, A.F., 1993. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEE proceedings F (radar and signal processing)*. Vol. 140, IET, 1993, 107–113.

Gustafsson, F., Gunnarsson, F., Bergman, N., Forssell, U., Jansson, J., Karlsson,.R. and Nordlund, P.-J., 2002. Particle filters for positioning, navigation, and tracking. *IEEE Transactions on Signal Processing*, 50(2), 425 - 437.

Hänsch, R., Weber, T. and Hellwich, O., 2014. Comparison of 3D interest point detectors and descriptors for point cloud fusion. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*. Vol. II, Copernicus GmbH, 2014, 57.

Hutton, J., Gopaul, N., Zhang, X., Wang, Menon, V., Rieck, D., Kipka, A. and Pastor, F., 2016. Centimeter-level, robust GNSS-aided inertial post-processing for mobile mapping without local reference stations, *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*. Vol.XLI-B3, 2016, 819–826.

Jaboyedoff, M., Oppiker, T., Abellan, A., Derron, M., Loye, A., Metzger, R. and Pedrazzini, A., 2010. Use of LIDAR in landslide investigations: a review. *Natural Hazards*. 61 (2012) 5–28.

Jende, P., Peter, M., Gerke, M. and Vosselman, G., 2016. Advanced tie feature matching for the registration of mobile mapping imaging data and aerial imagery. *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences*. Vol. XLI-B1, 2016.

Jing, H., Slatcher, N., Meng, X. and Hunter, G., 2016. Monitoring capabilities of a mobile mapping system based on navigation qualities. *ISPRS-International Archives of the Pho-*

*togrammetry, Remote Sensing and Spatial Information Sciences*. Vol. XLI-B1, 2016, 625–631.

Kalman, R.E., 1960. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*. 82(1), 35–45.

Knopp, J., Prasad, M., Willems, G., Timofte, R. and Van Gool, L., 2010. Hough transform and 3D SURF for robust three dimensional classification. *European Conference on Computer Vision*. Springer, 2010, 589–602.

Kukko, A., Kaartinen, H., Hyyppä, J. and Chen, Y., 2012. Multiplatform mobile laser scanning: Usability and performance, *Sensors*. 12(9), 11712–11733.

Lato, M., Hutchinson, D., Harrap, R., Diederichs, M. and Ball, D., 2009. Engineering monitoring of rockfall hazards along transportation corridors: Using mobile terrestrial LiDAR. *Natural Hazards and Earth System Sciences*, 9, 935–946.

Lauterbach, H., Borrmann, D., Heß, R., Eck, D., Schilling, K. and Nüchter, A., 2015. Evaluation of a backpack-mounted 3D mobile scanning system. *Remote Sensing*.7(10), 13753–13781.

Lehtomäki, M., Jaakkola, A., Hyyppä, J., Lampinen, J., Kaartinen, H., Kukko, A., Puttonen, E. and Hyyppä, H., 2016. Object classification and recognition from mobile laser scanning point clouds in a road environment. *IEEETransactions on Geoscience and Remote Sensing*. 54(2), 1226–1239.

Lindenbergh, R. and Pietrzyk, P., 2015. Change detection and deformation analysis using static and mobile laser scanning. Applied Geomatics. 7(2), 65–74.

Maghiar, M., Maldonado, G., Newsome, S., Clendenen, J. and Jackson, M., 2016. Accuracy comparison of 3D structural models produced via close-range photogrammetry and laser scanning. *Construction Research Congress*. 2016.

Marden, S. and Guivant J., 2012. Improving the performance of ICP for real-time applications using an approximate nearest neighbour search. *Proceedings of the Australasian Conference on Robotics and Automation*. Wellington, New Zealand, 2012, 3–5.

Mukupa, W., Roberts, G.W., Hancock, C.M. and Al-Manasir, K., 2017. A review of the use of terrestrial laser scanning application for change detection and deformation monitoring of structures. *Survey Review*. 49(353), 99–116.

Network Rail, 2016. `https://media.sa.catapult.org.uk/wp-content/uploads/2016/10/10154511/Network-Rail-Industry-Challenge-Statement-101016.pdf` *Challenge statement: Earthwork monitoring from above*.

Pu, S., Rutzinger, M., Vosselman, G. and Elberink, S.O., 2011. Recognizing basic structures from mobile laser scanning data for road inventory studies, *ISPRS Journal of Photogrammetry and Remote Sensing: Advances in LiDAR data processing and applications*. 66(6), S28–S39.

Puente, I., González-Jorge H., Martínez-Sánchez, J. and Arias, P., 2013. Review of mobile mapping and surveying technologies. *Measurement*. 46(7), 2127–2145.

Puente, I., Solla, M., González-Jorge, H. and Arias, P., 2013. Validation of mobile LiDAR surveying for measuring pavement layer thicknesses and volumes. *NDT & E International: Independent Nondestructive Testing and Evaluation*. 60, 70–76.

Ristic, B., Arulampalam, S. and Gordon, N. *Beyond the Kalman Filter: Particle filters for tracking applications*. Artech House, Boston, London, 2004.

Rusu, R.B., Blodow, N., Marton, Z.C. and Beetz, M., 2008. Aligning point cloud views using persistent feature histograms. *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE*. 3384–3391.

Rusu, R.B., Blodow, N. and Beetz, M., 2009. Fast point feature histograms (FPFH) for 3D registration. *IEEE International Conference on Robotics and Automation, IEEE*. 3212–3217.

Tang, J., Chen, Y., Niu, X., Wang, L., Chen, L., Liu, J., Shi, C. and Hyyppä, J., 2015. LiDAR scan matching aided inertial navigation system in GNSS-denied environments. *Sensors*. 15(7), 16710–16728.

TerraSolid, 2016. `https://www.terrasolid.com/download/tscan.pdf` *TerraScan User's Guide*.

Toth, C., Koppanyi, Z., Navratil, V. and Grejner-Brzezinska, D., 2017. Georeferencing in GNSS-challenged environment: Integrating UWB and IMU technologies. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*. Vol. XLII-1/W1, 2017.

Williams, K., Olsen, M., Roe, G. and Glennie, C., 2013. Synthesis of transportation applications of mobile LiDAR. *Remote Sensing*. 5(9), 4652–4692.

Wu, M.L., Chien, J.C., Wu, C.T. and Lee, J.D., 2018. An augmented reality system using improved-iterative closest point algorithm for on-patient medical image visualization. *Sensors*, 18(8), 2505.

Xiao, W., Vallet, B. and Paparoditis, N., 2013. Change detection in 3D point clouds acquired by a mobile mapping system. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*. Vol. II-5/W2, 2013, 331–336.

Yang, B., Dong, Z., Liu, Y., Liang, F. and Wang, Y., 2017. Computing multiple aggregation levels and contextual features for road facilities recognition using mobile laser scanning data. *ISPRS Journal of Photogrammetry and Remote Sensing*. 126, 180–194.

Zhong, Y., 2009. Intrinsic shape signatures: A shape descriptor for 3D object recognition. *2009 IEEE 12th international conference on computer vision workshops*. ICCV Workshops, IEEE, 2009, 689–696.

```
This is pdfTeX, Version 3.14159265-2.6-1.40.19 (TeX Live 2018/W32TeX)
(preloaded format=pdflatex 2018.7.12)  20 JAN 2020 08:12
entering extended mode
 restricted \write18 enabled.
 %&-line parsing enabled.
**sr_manuscript.tex
(./SR_manuscript.tex
LaTeX2e <2018-04-01> patch level 5

! LaTeX Error: File `interact.cls' not found.

Type X to quit or <RETURN> to proceed,
or enter new name. (Default extension: cls)

Enter file name:
! Emergency stop.
<read *>

l.5 ^^M

*** (cannot \read from terminal in nonstop modes)


Here is how much of TeX's memory you used:
 11 strings out of 492646
 232 string characters out of 6133325
 56384 words of memory out of 5000000
 3994 multiletter control sequences out of 15000+600000
 3640 words of font info for 14 fonts, out of 8000000 for 9000
 1141 hyphenation exceptions out of 8191
 10i,0n,8p,46b,8s stack positions out of 5000i,500n,10000p,200000b,80000s
!  ==> Fatal error occurred, no output PDF file produced!
```

ARTICLE TEMPLATE

# Efficient point cloud corrections for mobile monitoring applications using road/rail-side infrastructure

**ARTICLE HISTORY**
Compiled January 2, 2020

**ABSTRACT**
LiDAR systems are known to capture high density and accuracy data much more efficiently than other surveying methods. Therefore they are used for many applications, e.g. mobile mapping and surveying, 3D modelling, hazard detection, etc. However, while the accuracy of the laser measurements is very high, the accuracy of the resulting 3D point cloud is greatly affected by the geo-referencing accuracy. This is especially problematic for mobile laser scanning systems (MLS), where the LiDAR is installed on a moving platform, e.g. a vehicle, and the point cloud is geo-referenced by the data provided by a navigation system.

Due to the complexity of the surrounding environments and external conditions, the accuracy of the navigation system varies and thereby changes the quality of the point cloud. Conventional methods for correcting the point cloud accuracy either rely heavily on manual work or semi-automatic registration methods. While they can provide geo-referencing under different conditions, each has their own problems. This paper presents a semi-automated geo-referencing trajectory correction method by extracting features from the pre-processed point cloud and integrating this information to reprocess the navigation trajectory which is then able to produce better quality point clouds. The method deals with the changing errors within a point cloud dataset, and reducing the trajectory error from metre level to decimetre level, improving the accuracy by at least 56%. The accuracy of the regenerated point cloud then becomes suitable for many accuracy-demanding monitoring and change detection applications.

**KEYWORDS**
navigation, particle filter, point cloud, mobile mapping

## 1. Introduction

Light Detection and Ranging (LiDAR), also known as laser scanning, surveys the surrounding environment by sending out pulsed laser light to target objects and measuring the reflected pulse return times, wavelengths and signal intensity, from which the distance between the scanner and the object could be obtained. Due to the high accuracy of laser measurements and the scanner's ability to scan objects with very high density in a short time (e.g. more than 1 million measurements per second), LiDARs can provide very high density and accurate measurements of the surveyed environment (Lato *et al.* 2009, Pu *et al.* 2011, Puente *et al.* 2013, Mukupa *et al.* 2017). Therefore, conventional surveying work are being replaced and improved by using LiDAR based terrestrial and airborne surveying, mapping and monitoring. It is also being adopted in more recent applications such as 3D High Definition mapping, autonomous navigation, etc.

The mobile laser scanning system (MLS) is a popular system which integrates Li-

DARs with a navigation system and installed onto a mobile platform, e.g. land vehicle, aeroplane or drone. While the LiDAR continuously collects the reflected laser measurement from the surrounding environment as the platform moves, the navigation system provides the absolute position to geo-reference the laser measurements (Hutton *et al.* 2016). 3D point clouds can be generated very quickly by integrating the measurements from the two systems, which can then be used for modelling, mapping, and asset management, etc. (Williams *et al.* 2013). Its versatility and efficiency suggests its potential application in geohazard change detection and monitoring for remote areas. These works are normally completed by traditional surveying methods or by using terrestrial LiDAR systems, which provide high accuracy data, but are labour intensive. A main challenge to using MLS for these applications is its point cloud accuracy, which is often reduced as the navigation accuracy varies during data collection. The work discussed here attempts to address this problem, i.e. improving the geo-referencing accuracy for MLS point cloud data in a more efficient way, so that it can be used for accuracy demanding applications.

The navigation system used for MLS usually consists of a Global Navigation Satellite System (GNSS) and an Inertial Measurement Unit (IMU), as on the StreetMapper system from 3D Laser Mapping Ltd (3DLM). Additional sensors may include odometer and barometer. Although the integrated navigation system are generally high performance systems, it is inevitable that its accuracy could be reduced at times as GNSS positioning relies on receiving satellite signals, which can easily be blocked or disturbed. IMUs provide relative attitude measurements and accelerations that can navigate in absence of GNSS. However, these measurement can only provide the navigation solution for a short period of time during GNSS absence as IMU errors tend to increase very quickly over time without external corrections. Therefore, achieving high quality navigation in environments such as under thick tree canopy and urban areas, is a challenging task. As a result, the navigation accuracy does not always meet the expected accuracy requirement to geo-reference the MLS data. In such cases, the produced point cloud need to be corrected.

In many MLS applications, the point cloud accuracy is ensured by installing physical targets in the scan area, which act as control points (Puente *et al.* 2013). However, as the targets may have to be installed in remote and difficult to access locations, this method is both inconvenient and costly due to the amount of human labour required. Other methods, such as relative registration, are sometimes applied. Yet the improved correction efficiency comes at a cost of reduced accuracy compared to using control points (Bitenc *et al.* 2011, Kukko *et al.* 2012, Lauterbach *et al.* 2015, Gézero *et al.* 2017, Toth *et al.* 2017).

To reduce the workload required for effective and accurate point cloud adjustments and corrections, this work presents a semi-automated point cloud correction method for terrestrial MLS by firstly improving the navigation trajectory accuracy. The improved trajectory is then used to geo-reference the LiDAR measurements, correcting the changing error within the dataset. This procedure makes use of the feature information extracted from the point cloud which is then integrated within the navigation processing algorithm, i.e. Feature Extraction based Particle filter Point cloud Aiding (FEPPA). While feature matching methods have been proposed for registration in previous research such as in (Jende *et al.* 2016), accurate aerial LiDAR data was applied which does not suffer from the bad navigation errors that land MLS does. The algorithm here is tested on various sets of real world LiDAR data under bad GNSS conditions to demonstrate its point cloud correction capabilities. The FEPPA workflow eliminates most of the tedious manual work, reducing the typical correction

workload from around a week to under a couple of hours with comparable accuracy.

Section 2 will present the current background in point cloud matching as well as common navigation algorithms. Section 3 discusses the FEPPA methods, which includes extracting features from MLS generated point clouds and its integration with the navigation algorithm. Section 4 will present the test results using the proposed method. Its capabilities as well as limitations will also be discussed here. The final section concludes the work and discusses developments to adapt to future applications.

## 2. Background on MLS data processing

One of the main motivations for this work was to improve the MLS data accuracy to a level that could be used for geo-hazard change detection, which looks for the difference between two datasets of the same area but captured at different times (Mukupa *et al.* 2017). Change detection techniques have evolved greatly over the past years, from conventional surveying to remote sensing techniques and the recent growing adaption of LiDAR systems (Maghiar *et al.* 2016, Xiao *et al.* 2013, Williams *et al.* 2013), following a trend of improved data acquisition with higher density and efficiency.

Point cloud change detection can be achieved by computing the distance of a point from the comparison point cloud data to its relevant position in the reference point cloud, i.e. cloud-to-cloud distance measurement. Any non-zero values reflects the position difference between the 3D points, revealing changes in the physical conditions of the target area, such as those caused by landslides, slope failure, earthquake, road damage etc (Lato *et al.* 2009, Jaboyedoff *et al.* 2010, Lindenbergh *et al.* 2015). Therefore, the level of accuracy required for change detection depends on the type of changed being assessed. Change detection in geo-hazards are mostly major or obvious changes in the environment. Therefore the desired point cloud accuracy is within the decimetre level to identify these changes.

Most monitoring applications nowadays rely on using static terrestrial LiDAR systems or airborne MLS. However, static systems have limited data coverage and it is not always convenient to find a suitable scanning location. Airborne systems, on the other hand, suffer from low data density due to their distance from the ground and limited views of ground objects. The proposed correction method aims to enhance the point cloud quality derived from terrestrial MLS so that a more efficient monitoring workflow could be used for time critical applications in less accessible areas, addressing issues in modern road and railway environment monitoring (Chen *et al.* 2015, Network Rail 2016).

Controlling the point cloud quality from MLS before change detection analysis is essential, as the inconsistency between compared data includes two aspects:

- the change in position or shape of the object that is represented by the point, i.e. detected change;
- the relative position error between the two point clouds, i.e. noise and error.

Removing, or reducing, the relative error to a minimum is vital before carrying out infrastructure or geotechnical monitoring analysis. The following subsections will introduce the common methods for processing MLS data, including navigation data and 3D point cloud processing. The major issues that affect the point cloud accuracy and some common methods to address them will also be highlighted.

### 2.1. *Point cloud processing*

The basic components of an MLS typically consists of one or two 2D laser scanner(s), a GNSS/IMU integrated navigation system, an on-board computer and storage, as well as power supply. 2D laser scanners capture relative range measurements from the scanner to surrounding features which reflect the laser beam. To produce a 3D point cloud, the 2D range measurements need to be time synchronised and integrated with the navigation data using the basic steps as outlined below:

(i) Process the navigation data using suitable integration algorithms, e.g. Kalman filter or particle filter, which produces trajectory data that includes both the position coordinates as well as the attitude at a high rate.
(ii) Integrate the LiDAR range measurements with a 3D position and attitude by matching their time stamps to the navigation data, thereby projecting the 2D range measurement into a 3D coordinate system, i.e. generating a 3D point cloud.
(iii) Merging multiple scans: to increase the density of the point cloud, the same route is sometimes scanned more than once by the MLS vehicle driving from different directions, where each scan is known as a flightline (adopted concept from airborne applications). Merging multiple flightlines into one point cloud increases data density, hence captures more information of the environment. However, prior to merging, the data must be corrected for mismatches, i.e. errors, between the different scans to ensure that all data overlay consistently.
(iv) Point cloud corrections: merging the flightlines ensures that relative errors of each scan are reduced. If control points were used, the merged point cloud needs to be matched to the scanned control points. This step reduces the global errors and ensures that the data fits with the global coordinate system used in the location of the data capture.

Further data analysis can be carried out once the high quality 3D point cloud is produced. Usually this procedure tries to imply some "meaning" to the point cloud, such as data classification or object recognition, etc.

Due to the data processing procedure, the accuracy of the 3D point cloud is affected by both the measurement accuracy of the laser scanner as well as the performance of the navigation system. The laser scanner range measurement accuracies are generally around 0.5 - 2 cm usually with millimetre level precision; whereas the navigation data accuracy can be affected by various external conditions, producing errors from a few centimetres up to tens of metres. Therefore, the accuracy of the point cloud is largely affected by the navigation system performance (Jing *et al.* 2016).

In ideal conditions, navigation systems can achieve decimetre or even centimetre accuracy by using methods such as Real-time Kinematic GNSS positioning (RTK) (Tang *et al.* 2015). However, GNSS based navigation is easily disturbed by the environment causing the accuracy to reduce to metre level or worse. This introduces errors in the MLS point cloud, and even worse, introduces inconsistent errors within a single point cloud dataset as the navigation error changes. Therefore, improving MLS geo-referencing accuracy is cruicial before carrying out data analysis on MLS data.

Two common methods to improve point cloud accuracy are relative registration and control point geo-referencing. Relative registration requires the availability of another dataset of the same scan location with higher accuracy, known as the reference data. The data of concern is then matched to the reference data using methods such as Iterative Closest Point (ICP), Local Descriptor Histograms or other statistics that can characterise the point cloud and find matching points between two data (Rusu *et al.*

2008, 2009, Marden *et al.* 2012).

Registration is relatively fast, but the data can only be matched to minimise the global differences between the two datasets. Therefore, it is unable to eliminate local distortion and errors, such as those introduced by the navigation system. Furthermore, the performance of registration is highly dependent on the characteristics of the point cloud data, such as the geometry of the features within the point cloud and the movements of features between scans.

Geo-referencing using control points achieves unrivalled accuracy and reliability. Control points have to be installed prior to data capture and surveyed using instruments such as total stations or survey-grade GNSS to provide millimetre or centimetre level position reference (Puente *et al.* 2013). The point cloud data is then corrected by manually finding the targets in the point cloud and shifting the position of the points representing the target to match the surveyed position of the targets. These shifts build up a regression function which also shifts the other points in the data accordingly (TerraScan 2016). This method produces better overall accuracy as it is able to correct local errors within a data. It is especially useful when errors are inconsistent within a dataset, which is generally the case for data produced by the MLS. However, this correction process is known to be labour intensive and must be done manually.

### 2.2. *Navigation data processing*

Due to its effect on the point cloud data accuracy, ensuring the navigation data accuracy is cruicial to reducing error in the point cloud. Many algorithms had been proposed to integrate GNSS and IMU measurements for continuous high performance navigation, including Kalman filter (KF), extended KF and Particle Filters (PF) etc. Since its introduction in 1960 (Kalman 1960), KF has become an efficient tool to solving linear prediction and estimations problems, including tracking and navigation. It continuously measures and estimates the navigation system state, i.e. position, velocity, attitude and biases, while estimated states are continuously updated by incoming new measurements. Integrated navigation methods provides continuous navigation even when GNSS measurement is not available for a period of time.

High performance IMUs will be able to produce acceptable navigation during a longer period of GNSS outage. However, IMU measurements always grow exponentially over time and eventually exceed the requirement levels if GNSS positioning or other sources of positioning measurements are continuously unavailable. If additional external measurements could be integrated when they become available, the navigation performance could become more reliable. Due to the non-linear characteristics of external measurements, PF is considered more suitable and achieves better performance.

Particle filtering is a recursive Bayesian filtering method that integrates measurements from different sources and predicts the system states, which are represented by a large set of particles with associated weights, through sequential Monte Carlo estimation (Gordon *et al.* 1993, Ristic *et al.* 2004) . The system state vector $X_k$ is a discrete time stochastic model expressed as below:

$$X_k = f_k(X_{k-1}, v_{k-1}) \tag{1}$$

where $f_k$ is the non-linear function of the state $X_{k-1}$ and process noise $v_{k-1}$ at time

$k$, . The state vector $X_k$ is recursively updated from observations $z_k$:

$$z_k = h_k(X_k, n_k) \tag{2}$$

where $h_k$ is usually a non-linear function with measurement noise $n_k$. PF estimates the state $X_k$ at time $k$, given observations $z_{1:k}$ up to time $k$. At each epoch, the predicted probability density function (pdf) is updated through measurements to represent the posterior pdf of the current state. As it is usually impossible to obtain the true posterior pdf, $N$ particles are generated to represent a discrete approximation $p(x)$,

$$p(x) \approx \sum_{i=1}^{N} w^i \delta(x - x^i) \tag{3}$$

where particles $x^i$ are drawn from the approximate density, $w^i$ is the normalised weight of the $i$th particle, and $\delta(\cdot)$ is the Dirac delta function. As $N \to \infty$, the approximation should approach the true posterior pdf. A brief summary to each iteration of a typical PF procedure for navigation based on (Gustafsson *et al.* 1993, Ristic *et al.* 2004) is an below.

- *Initialisation*: $N$ particles $x_0^i (i = 1, ..N)$ are created to represent an estimated probability distribution of the initial system state $p(X_0)$, i.e. for navigation systems, each particle includes information on location, historic location, its current state, and a weight. All particles usually start with equal weights, unless other prior information is available. The weighted average of the particles represents the initial system state estimation.
- *Prediction*: the particles propagate through a prediction model, as Eq.1, the prior pdf of the new state estimate $X_{k|k-1}$ at time $k$ is obtained,

$$p(x_k|Z_{1:k-1}) = \int p(x_k|x_{k-1}) p(x_{k-1}|Z_{1:k-1}) dx_{k-1} \tag{4}$$

  where $Z_{1:k-1}$ is a set of all available measurements $z$ up to time $k$, $p(x_k|x_{k-1})$ is the probabilistic model of the state propagation defined by the system equation and estimation of noise $v_{k-1}$.
- *Update*: a set of new measurement $z_k$ at time step $k$ is obtained to update the prior via Bayes rule and obtain the posterior of the system state

$$p(x_k|Z_{1:k}) = \frac{p(z_k|x_k) p(x_k|Z_{1:k-1})}{p(z_k|Z_{1:k-1})} \tag{5}$$

  where $p(z_k|x_k)$ is the conditional pdf of $z_k$ given $x_k$, $p(x_k|z_{1:k-1})$ is the prior pdf at time step $k$ and $p(z_k|Z_{1:k-1})$ is the probabilistic model of the state propagation defined by the system equation and estimation of noise $v_{k-1}$. The likelihood of each particle $x_k^i$ , i.e. the weight $w_k^i$, is computed based on $p(x_k|Z_{1:k-1})$.
- *Resample*: A weight threshold is defined based on the system requirements and any particle $i$ with a weight below the threshold is redefined as $w_k^i = 0$. If the number of valid particles falls below a threshold, the particle cluster is resampled to regenerate $N$ particles based on $p(x_k|Z_{1:k-1})$. The particle weights are then normalised.

- Return to step 2 (prediction) or end process: the weighted mean of the particles gives the state estimation at the end of each iteration.

With the flexibility to integrate any data source as they become available, PF is widely adopted for the integration of measurements from different sources. The work presented here introduces an MLS data correction method by integrating information extracted from the LiDAR measurements in the point cloud data and integrating them with other navigation measurements using PF. However, integrating LiDAR measurements can be highly computationally expensive due to its high data rate and density. FEPPA allows a "very loose-coupling" scheme that corrects navigation only when necessary and reduces the computational effort. The integration of LiDAR measurements with a GNSS/IMU system takes advantage of the high accuracy GNSS performance in open sky areas, high accuracy IMU attitude measurements and periodic corrections constrained by the feature landmarks in the point cloud data.

## 3. The FEPPA method

Unstable performance of the onboard sensors of an MLS leads to two main problems: 1) inaccuracy in the produced point cloud; 2) inconsistent accuracy in the point cloud; as shown in Figure 1, which plots the distance, i.e. mismatch, between the two point clouds representing the same location, one perfectly geo-referenced and another generated by uncorrected navigation data. The blue crosses are the estimated standard deviation from the navigation filter which gives an indication of the navigation accuracy; the red triangle line shows the average mismatch between the points within a point cloud dataset generated during those epoches. The mismatch between the point cloud data varies as the navigation accuracy changes, thus indicating that the accuracy of the point cloud generated by MLS can vary depending how the navigation error changes. When point cloud data of different accuracies are merged together, the resulting data can show a "shadowing" effect such as seen in Figure 2. This inaccuracy and inconsistency leads to further problems when analysing data, therefore needs to be minimised.

The proposed FEPPA method improves the navigation accuracy through integrating the measurements extracted from point cloud features which contains both absolute location information (position within the absolute global coordinate system, i.e. WGS84) and relative location information (distance between the system and other features). The sections below will explain the methods used to extract feature information and the PF based integration method to enhance navigation as well as point cloud accuracy and consistency.

### 3.1. *Point cloud feature extraction*

Feature extraction and object recognition have been one of the main research areas for LiDAR point clouds. Mainstream methods include key point detection, classification and object recognition (Bosch *et al.* 2007, Zhong 2009, Hansch *et al.* 2014). These include identifying unique points of interest within a point cloud even at different scales, such as corners and sides of objects (Assfalg *et al.* 2007, Knopp *et al.* 2010). Object recognition is the procedure to detect and extract points representing an entire object or class of features. These methods rely on using unique descriptors to describe the geometry features as well as other laser measurements, e.g. intensity, which help

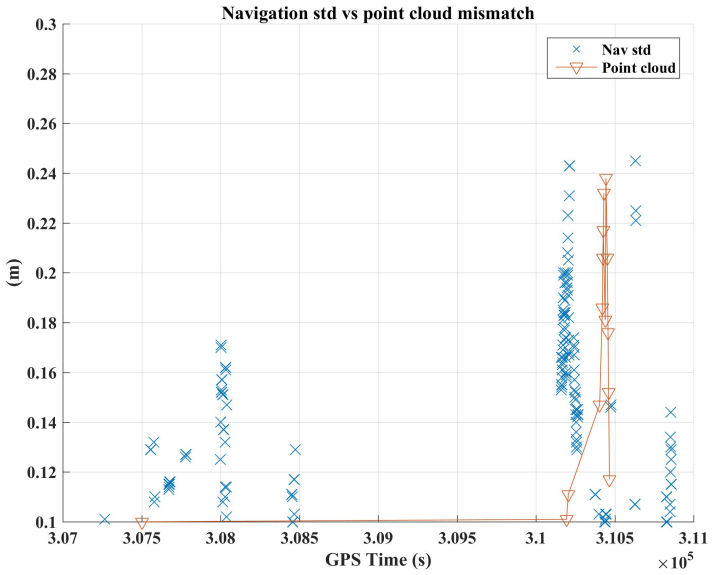



Figure 1.: Navigation performance effect on the point cloud (The Nav std line represents the estimated navigation error from the navigation filter; point cloud line represents the distance of the points between a MLS point cloud data compared to a reference dataset of the same location)

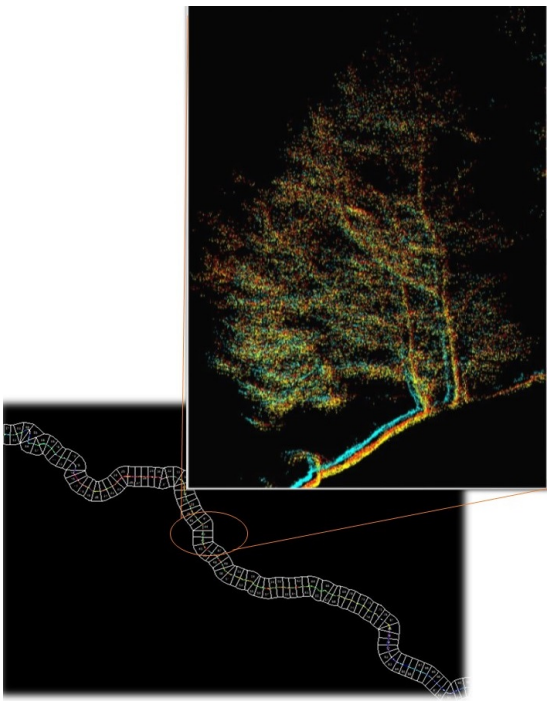


Figure 2.: Shadowing effects when accuracy is inconsistent between point cloud data

to recognise the object points among other data points in the point cloud (Lehtomaki *et al.* 2016, Yang *et al.* 2017). These descriptors specify each object with its unique identity which are pre-learned by the machine and features/objects are detected from

the point cloud using supervised or semi-supervised machine learning methods.

The first step of the FEPPA method relies on extracting measurements from stable features within the dataset. The object recognition/classification method used here is based on the CANUPO suite developed by Brodu et al (Brodu *et al.* 2012), which is available from the open source software CloudCompare. The CANUPO algorithm was developed especially to identify and classify complex scenes in the natural environment, such as vegetation, rock, gravel and water, etc. which is not addressed by many classification tools. The toolset handles large datasets and allows simple user input through semi-supervised machine learning procedures. The main idea of CANUPO is to characterise the local dimensionality properties of the scene at each point and different scales, i.e. how the point cloud objects look like at 1D, 2D or 3D scales. The learned characteristics help to identify objects or classify points at a later stage.

Within the context of this paper, the main application is the early detection and monitoring of geo-hazards. The datasets were collected along railways and roadsides. The CANUPO tool was tested to give relatively better performance for the types of datasets that were used, thus adopted as part of the workflow. It is used for the extraction of pole shaped infrastructure, including lamp posts and utility poles etc., among other similar natural environment features, e.g. trees.

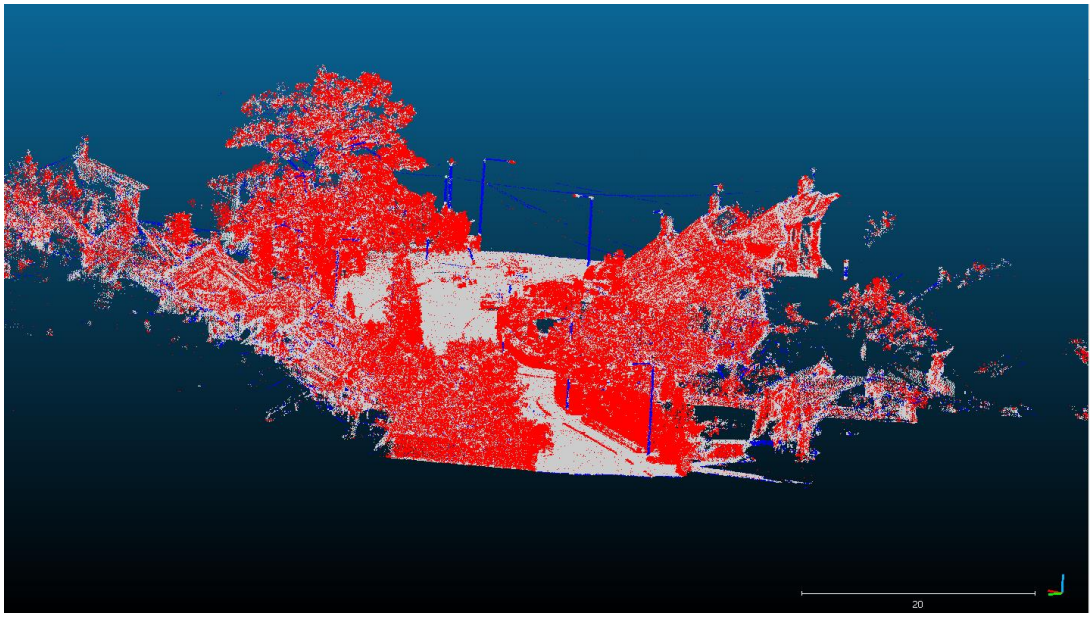



Figure 3.: Point cloud classification results using CANUPO
(Blue: pole class (target object); Red: other points not representing the target object; Grey: points that cannot be classified by the tool)

First of all, a descriptor is built in the CANUPO toolbox that can distinguish between the pole shaped infrastructures of interest (i.e. objects that will not change shape or position over time) and other objects in the point cloud. It is vital to exclude trees from the extracted feature dataset as they can introduce large ambiguities during processing. Once the descriptor is created, the point cloud datasets are run through the CANUPO toolbox, for pre-processing.

This step classifies the points in the point cloud into three classes: pole points, non-pole points and unidentified points, as shown in Figure 3. However, the classification results here are not perfect due to slight differences between the actual objects and

the descriptor. Therefore, the second step in the workflow, i.e. cleaning the classified points, is critical. The cleaning workflow was developed in Matlab by the author and deletes any false-positive points that have been classified as poles by the CANUPO tool, as shown in Figure 4. The cleaning workflow divides the classified target points (blue points from Figure 3) into $1m \times 1m$ grids, then filters out the data points if they were associated with low confidence by the CANUPO tool and the density of the points within the grid falls below a threshold, which is defined relative to the entire point cloud density.

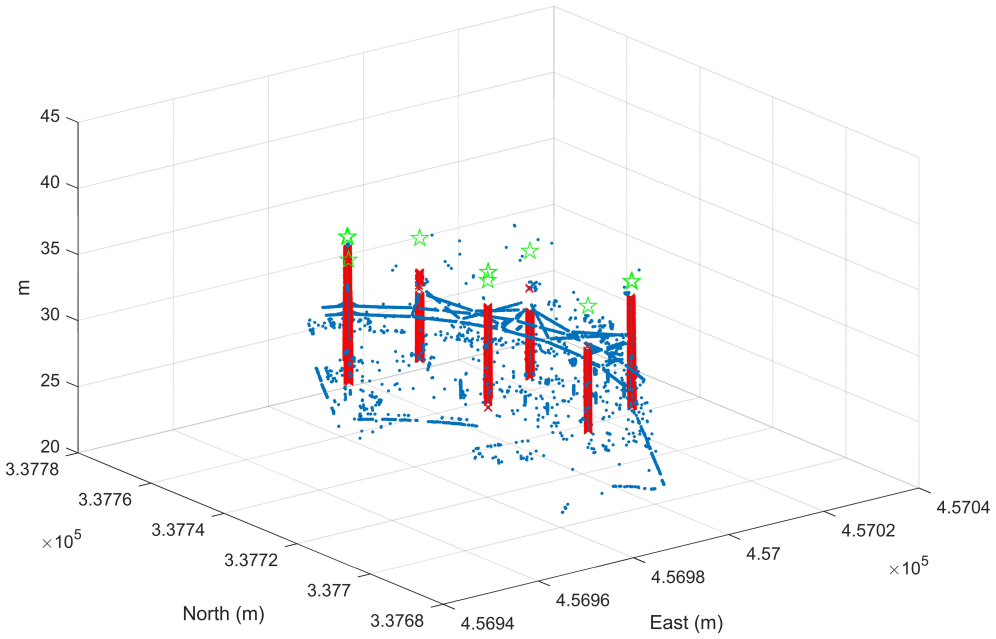



Figure 4.: Extracted features from classified points (red points indicate the final classified pole points, blue points are all the classified points from CANUPO (the blue points from Figure 3), green stars indicate the estimated 2D location of the pole object, plotted at the same height for visual indication)

Once the data has been filtered and the "clean" features have been identified, the Matlab script extracts further information from the points representing each object, i.e. $[t_{f_i}, x_{f_i}, y_{f_i}, \Delta h_{f_i}, intensity_{f_i}, \sigma_{f_i}]$, which represent the time of the data capture, Easting, Northing and height difference in OSGB36 grid coordinates (the geographic grid reference used in the UK), intensity of the points, and the confidence index of the CANUPO classification, respectively. The extracted information from a reference point cloud data is saved as the reference feature information; information extracted from the comparison dataset is saved as the comparison feature information. These information will be integrated into the navigation PF to reprocess the trajectory, which improves accuracy and consistency among the datasets captured at different times.

### 3.2. *Particle filtering based navigation aiding*

The navigation trajectory used for creating the original point clouds is generated from integrating GNSS positioning data and IMU measurements through loosely coupled KF, which is carried out by the software given by the navigation system provider. However, as discussed, navigation trajectory does not have consistent accuracy along the entire route. This is predominately due to changes and variations in the GNSS satellite condition and environment when datasets are captured at different times of the day or year. Hence changing the accuracy of the point cloud data. Therefore, it is important to ensure that different point cloud datasets are corrected to the same accuracy level before they can be used to carry out change detection or other similar analysis. The FEPPA algorithm proposed here reprocesses this trajectory by integrating the feature information extracted in Section 3.1.

The algorithm follows the basic PF procedures described in Section 2.2 where the system state model $X_k$ represents the navigation system position information, the propagation vector follows the measurements obtained from the IMU and the extracted feature information is integrated to provide measurement updates as explained in more detail below. The process noise $v_k$ is normally distributed with a mean of 0 and standard deviation given based on the system specifications for position and heading errors, the measurement noise $n_k$ is normally distributed with a mean of 0 and a standard deviation following the current estimated navigation error which is derived from comparing the feature locations and the current system location. A basic workflow chart is given in Figure 5.

As outlined in Section 3.1, the extracted features of interest should remain stable over time and thus retain the same location information. Inconsistent navigation accuracy is detected when the same features are geo-referenced at slightly different locations. Therefore, the position difference of the reference features $f_{ref}$ and the comparison features $f_i$, i.e. $\Delta dis_{f_i}^{f_{ref}}$, will be used as an observation measurement integrated into the PF to constrain the navigation error in post-processing. The steps of the FEPPA workflow is described as below:

1) Estimate the centre position of each of the $N$ features $\{x_{f_i}, y_{f_i}, \Delta h_{f_i}\}_{i=1,...,N}$ from the data points,

$$x_{f_i} = mean(x_1, x_2, ..., x_q); \tag{6}$$

$$y_{f_i} = mean(y_1, y_2, ..., y_q); \tag{7}$$

$$\Delta h_{f_i} = z_i^{max} - z_i^{min}; \tag{8}$$

   where $f_i$ is the feature number, and $q$ is the number of points representing feature $f_i$. The time stamp of each point representing each feature $t_q^i$ is saved as a separate array for reference. The time reference is essential as the navigation error and point cloud error are correlated by time.
2) Compute the 2-D geometry index of the group of features extracted at the current epoch, i.e. $FDOP$ (Feature Dilution of Precision), adopted from the GNSS

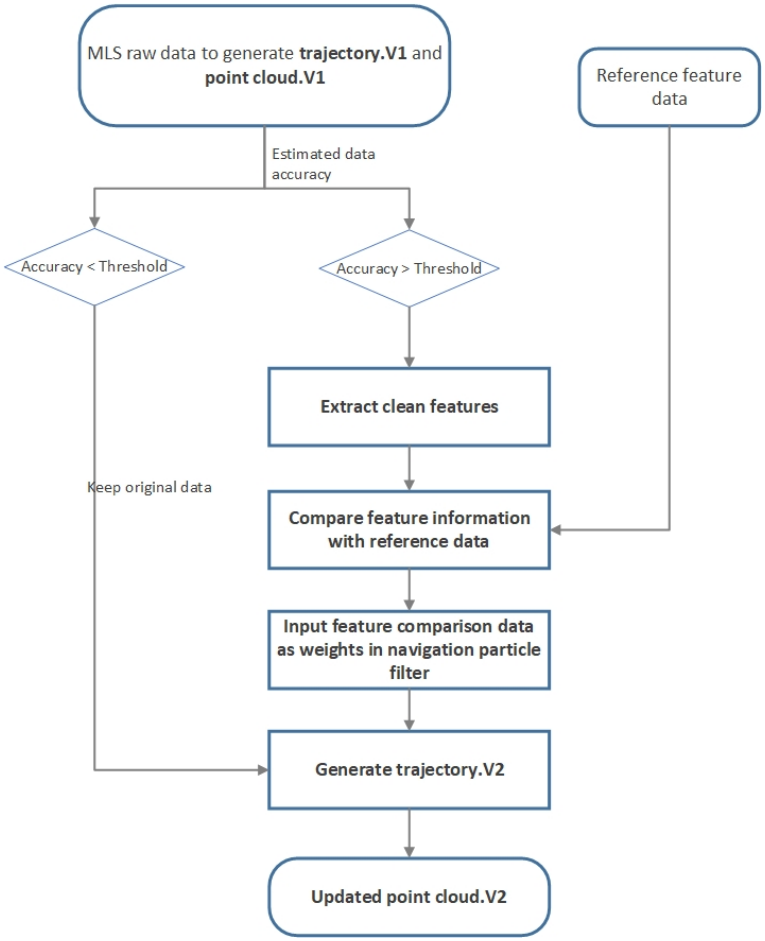



Figure 5.: Flowchart showing the basic steps of the FEPPA method

positioning DOP concept:

$$FDOP = \sqrt{\sigma_x^2 + \sigma_y^2} \tag{9}$$

where $\sigma_x^2 = \frac{x_i - x}{R_i}$, $\sigma_y^2 = \frac{y_i - y}{R_i}$, $R_i = \sqrt{(x_i - x)^2 + (y_i - y)^2}$, $x_i$ and $y_i$ are the 2D coordinates of the features, $x$ and $y$ are the 2D coordinates of the system at the current epoch. At each epoch, the features selected for computation are features whose data points were captured within a time threshold of the current epoch, denoted as "visible" features in the text below. The selection of "visible" features is to ensure that the navigation accuracy representated by the features are similar to the accuracy of the current epoch. $FDOP$ provides an estimated indication of whether the extracted feature information could improve the trajectory accuracy at the current epoch. Small $FDOP$ indicates good geometry layout surrounding the MLS system, i.e. evenly spread out features along the travelling path on both sides, therefore these features can be used to give a better accuracy when post-processing the trajectory; larger $FDOP$s indicate bad geometry, either due to low number of available features, or that features are clustered close together, thus less capable of giving a better accuracy estimate.

3) Initialise the PF: initialise a set of particles $p_m$ around the initial position with a level of uncertainty $\sigma_{pt}$ based on the trajectory data, where $m = 1, ..., M$ , i.e. $M = 1000$, $w_k^m = 1/M$ is used here.
4) Propagate the particles based on the measurements from the navigation system, i.e.

$$p_x^{m_t} = p_x^{m_{t-1}} + \Delta_x; \tag{10}$$

$$p_y^{m_t} = p_y^{m_{t-1}} + \Delta_y; \tag{11}$$

$\Delta_x$ and $\Delta_y$ should be derived from the navigation system measurements. However, in this case, the commericial navigation system used did not provide raw IMU measurements. Therefore, $\Delta_x$ and $\Delta_y$ are derived from the produced navigation trajectory, i.e. the position and heading change between the current and previous epoch with added process noise $v_k$ to allow for errors in the produced trajectory.
5) The weight of each particle $w_k^m$ is reassigned in this step after comparing the distance between each particle and the location of "visible" features (i.e. $R_{p_m}^{f_i}$) and the distance to the same features in a reference dataset (i.e. $R_{p_m}^{f_{ref}}$) :

$$R_{p_m}^{f_i} = \sqrt{(p_x^m - f_{i_x})^2 + (p_y^m - f_{i_y})^2} \tag{12}$$

$$R_{p_m}^{f_{ref}} = \sqrt{(p_x^m - f_{ref_x})^2 + (p_y^m - f_{ref_y})^2} \tag{13}$$

$$\Delta R_{p_m} = R_{p_m}^{f_{ref}} - R_{p_m}^{f_i} \tag{14}$$

If the difference $\Delta R_{p_m}$ is over a predefined distance threshold, depending on the estimated navigation error, the particle weight $w_k^m$ is simply give $w_k^m = 0$; otherwise, $w_k^m$ is the inverse of the difference in the distance, i.e.

$$w_k^m = \frac{1}{\Delta R_{p_m} \cdot a} \tag{15}$$

where $a$ defines how quickly the difference in the distance changes the weight. The position and height information of the points representing each feature is used to justify the matching between the current feature $f_i$ and reference feature $f_{ref}$. If there should be a mismatch due to several features being too close together, the range difference will appear to be sufficiently larger than expected, thus giving a lower weight or 0.
6) Resampling: to avoid the particles clustering towards a wrong location in the process or particle impoverishemnt, both of which are common problems in PF if the wrong weighting and resampling threshold is used, the particles are checked

for resampling after every iteration. If there are any invalid particles, i.e. any particles with $w_k^m = 0$, particles with a higher weight $w_k^m$ is given higher probability to regenerate new particles, while low-weighted particles will have less opportunity to generate new particles, to replace these invalid particles. A system noise is added to the regenerated particle cluster. If there are no remaining valid particles, a new particle cluster is generated based on the current estimated position. As the feature range measurements can constrain the particles to quite a small area, resampling will ensure there are always valid particles to continue the iteration.

7) Positioning estimation: the updated position estimation of the navigation system, i.e. the MLS position, is computed from the weighted average of all particle positions, i.e.

$$\{x_{update}, y_{update}\} = \{\frac{\sum_{m=1}^{M}(p_x^m \cdot w_m)}{\sum_{m=1}^{M}(w_m)}, \frac{\sum_{m=1}^{M}(p_y^m \cdot w_m)}{\sum_{m=1}^{M}(w_m)}\}; \tag{16}$$

The PF is continued until the whole trajectory has been reprocessed using the feature information; generating a new trajectory for geo-referencing, i.e. the PF results.

8) The PF trajectory is used to update the GNSS measurement data, which is then reprocessed by GNSS/IMU integration software provided by the navigation provider, producing the final FEPPA trajectory. The point cloud is re-generated using the FEPPA trajectory.

The final step is required here as we did not have access to the raw measurements produced from the navigation system, which was a limitation from the commerial hardware and software package. The PF algorithm could only update the position estimations, thus need to be reprocessed in the integration software to provide improved position and attitude outputs before the data can be used to geo-reference the point cloud. The PF integration method was chosen here due to that the integrated feature measurements cannot be easily modelled for KF due to their changing and unpredictable nature. Yet integration of these measurements in PF is a straightforward procedure.

The FEPPA process is designed based on the assumption that the difference in the navigation accuracy is reflected by the relative distance shift between the same features extracted from the corresponding point cloud dataset. The distance shift is estimated and used to reprocess the trajectory, aiming to reduce the error. However, it is recognised that the distance shift between the features does not fully represent the actual navigation error, due to errors being either enlarged or reduced by the changing attitude measurements of the navigation system. Therefore, the coefficient $a$ and cut-off threshold are introduced to provide an adaptive weighting scheme. This reduces the constraint of the measurement update and allows for the slight difference between the actual navigation error and the distance shift between features.

Analysis on several datasets are given below to discuss the performance of the FEPPA workflow and some considerations when using the method.

## 4. Results and analysis

This section will analyse the performance of the FEPPA workflow on two datasets captured in different environments and using navigation systems of different performance levels. The point cloud correction performance of the FEPPA method will be compared to registration methods and also standard MLS data processing workflows using the commercial software suite, TerraScan/TerraMatch. The results of the correction is compared to the reference point cloud using cloud-to-cloud distance computation to see how well it matches the desired data. The efficiency of the workflow and also minor problems that need to be considered are also discussed.

### 4.1. *Simulation data and processing*

Two sets of MLS data were captured for the analysis of the proposed workflow. The first dataset was captured along the Nottingham tram line, which simulates railway and roadside scenarios in terms of infrastructure, covering both suburban and urban environments. This data simulates the environment which requires infrastructure and asset monitoring. The second dataset was captured along the A52 highway road near Bingham, England, which has a mostly rural environment and simulates the a rural site which requires geohazard monitoring.

The MLS system used for the tram data capture was the ROBIN system from 3D Laser Mapping Ltd and the A52 road data is captured using the StreetMapper system also from 3D Laser Mapping Ltd, as shown in Figure 6. The StreetMapper IV system consists of a Riegl VUX-1 laser scanner and an IGI TerraControl navigation system, which consists of a NovAtel GNSS receiver and the IGI Fibre optic gyros (FOG) IMU-IIe. The ROBIN system consists of a Riegl VUX-1HA scanner and the same GNSS receiver but integrated with a MEMS (Micro-Electro-Mechanical System) IMU, which is a lower grade IMU compared to FOG IMUs. Specifications for the VUX-1 scanner is listed in Table 1. The scanner is installed facing backwards of the vehicle with a 50° angle to the horizontal plane, allowing the scanner to capture details of the ground and along both sides of the road. Specifications for the two navigation systems are listed in Table 2.

(a) ROBIN MLS system

(b) StreetMapper MLS system (the camera system in the picture was not used in this particular test)

Figure 6.: Systems used for data capture

Table 1.: RIEGL VUX-1 laser scanner performance

| | **Max range** | **FOV** | **Meas. rate** | **Resolution** | **Accuracy** | **Precision** |
|---|---|---|---|---|---|---|
| **VUX-1** | up to 920m | 330° | up to 550kHz | 0.001° | 10mm | 5mm |

Table 2.: IGI TerraControl navigation system

| | **Pos (m)** | **Vel (m/s)** | **Heading (°)** | **Pitch/Roll (°)** |
|---|---|---|---|---|
| **FOG-IIe** | 0.02 | 0.005 | 0.03 | 0.015 |
| **MEMS** | 0.02 | 0.005 | 0.01 | 0.004 |
| | Gyro bias ($°/hr$) | Gyro random walk ($°/hr^2$) | Acc bias (mg) | |
| **FOG-IIe** | 1 | 0.07 | 0.3 | |
| **MEMS** | 0.03 | 0.0005 | 0.1 | |

For testing and evaluation, a good quality reference dataset is generated. The data was captured in an environment without any long tunnels or thick foliage which may potentially block the sky view and reduce navigation accuracy significantly. For both datasets, the navigation and point cloud data were processed to its best possible solution using the standard workflow that would be used to deliver data to clients, i.e. the final navigation accuracy is centimetre level for over 95% of the entire trajectory and the point cloud data was manually corrected and processed using the TerraSolid software suite with high quality calibration. These results were saved as the reference data.

To introduce errors into the dataset, sections of GNSS measurements are deleted

from the navigation data to simulate loss of GNSS signals. The period of deleted measurements varies for the different navigation system used, but both produced navigation errors of over a metre. For the tram data, a period of 90 seconds was deleted resulting in around 4m error; 4.5 minutes of GNSS measurements were deleted from the A52 highway data producing a maximum error of around 2m. The resulting trajectory and point cloud data were saved as the comparison data.

Pole features were extracted from the point cloud using the feature extraction workflow described in Section 3.1 for both the reference and comparison data. For both datasets, the most common features used were the lamp posts and utility poles. The trajectories of the comparison data was reprocessed using the FEPPA workflow to reduce the trajectory errors, which can then be used to regenerate the comparison point cloud so that it matches the reference dataset. Navigation error here is referred to as the difference between the comparison or reproduced trajectory and the reference trajectory. The point cloud difference were measured by the Cloud-to-cloud distance (C2C) tool provided in CloudCompare and results are displayed in colour scale.

### 4.2. *FEPPA performance and comparison*

To analyse the correction performance of the FEPPA workflow, registration methods were also used to match the comparison and reference data and the results were compared to the FEPPA outputs. A series of tools, including Spin image registration developed by 3DLM, registration tools available in Point Cloud Library, and registration tools in CloudCompare, were tested for comparison. Although with slightly different accuracies, all tools presented similar pros and cons for the data that were tested on. The results from the ICP tool in CloudCompare is given here as an example for typical problems when using registration for large MLS datasets. The CloudCompare ICP tool usually gives quick and efficient results with reasonable accuracy in general registration problems (Wu *et al.* 2018). However, as with all registration methods, it is unable to deal with changing accuracy within the same point cloud dataset.

The ICP matching results for a small section of the tram data is shown in Figure 7. Here the point cloud data registered using ICP, shown in orange-yellow points, is overlaid on top of the reference data, shown in blue. The change in error levels of

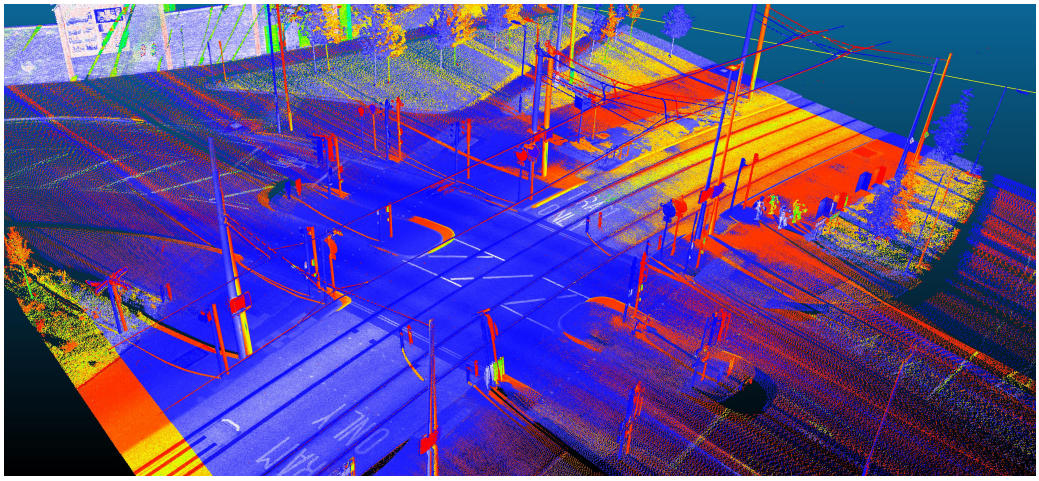


Figure 7.: ICP performance on the tram data (matched data compared to the reference data)

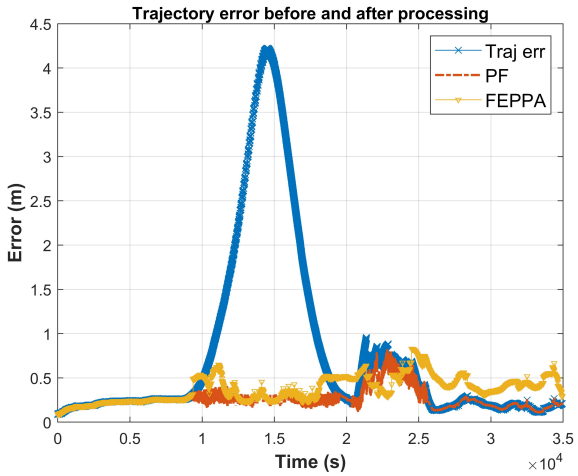



Figure 8.: Navigation correction results

the MLS data could be clearly seen, where the features between the two data on the left side of the data are much more closer than the features on the right-hand side of the data. This reflects one of the main problems of using rigid registration methods for MLS point cloud geo-referencing, where the errors are inconsistent throughout the data.

The tram data will be discussed in more detail below, due to its better visualisation effects. Therefore, enabling clearer comparison.

The FEPPA workflow corrects the inconsistent error within the same point cloud dataset by reprocessing the navigation data using error estimation extracted from features in the point cloud. Figure 8 shows the error of the original comparison navigation data compared to the navigation error after being processed by two workflows, i.e. PF and FEPPA. The PF method is actually a part of the FEPPA workflow but only reprocessing the data up to Step (7) outlined in the workflow in Section 3.2. Results from using the PF method only is shown in the red dotted line.

The FEPPA results, plotted in yellow lines, shows the results processed by the complete FEPPA workflow, i.e. outputs after Step (8). As mentioned previously, Step (8) is required here due to not being able to access the sensor raw measurements. This step reprocesses the data to update both the position and attitude outputs. Although it seems that the position error of the trajectory is lower by using PF only, the attitude measurements are not updated according to the new updated positions, the generated point cloud data can appear to be "messy", as seen in Figure 9.

Table 3.: FEPPA processing results

| | | **Before correction** | **PF only** | **FEPPA** |
|---|---|---|---|---|
| Tram data | **RMS (m)** | 1.051 | 0.022 | 0.119 |
| | **Max (m)** | 4.224 | 0.183 | 0.261 |
| Highway data | **RMS (m)** | 0.919 | 0.145 | 0.441 |
| | **Max (m)** | 1.880 | 1.669 | 1.314 |

Table 3 shows the trajectory error before corrections, after being processed using PF only and after being processed using the FEPPA workflow. Although the position

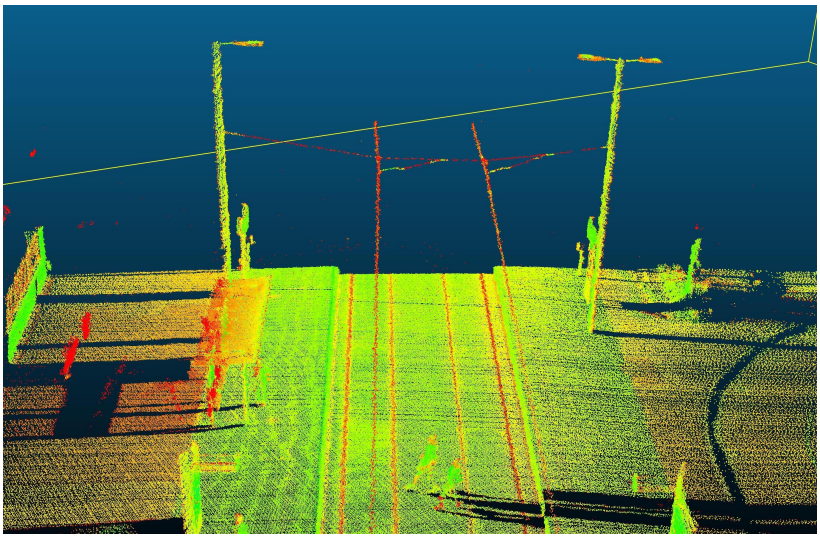
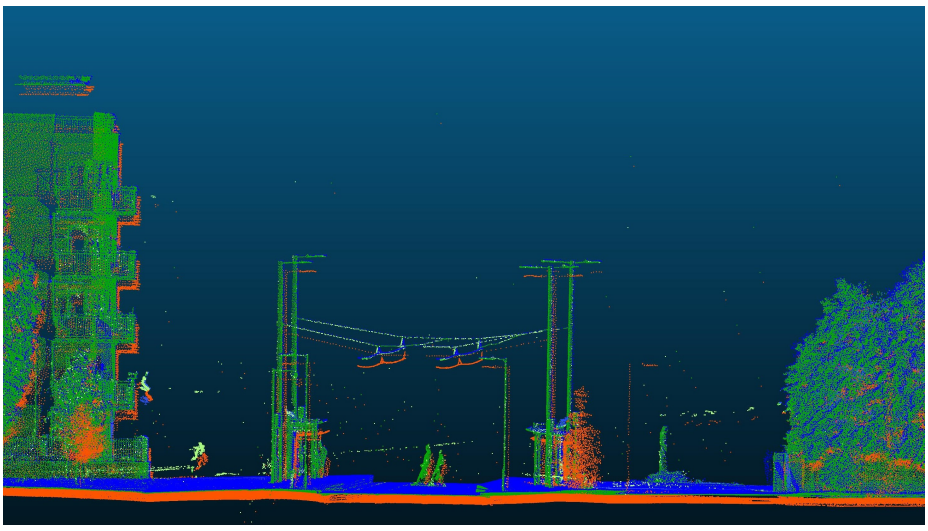


Figure 9.: PF point cloud

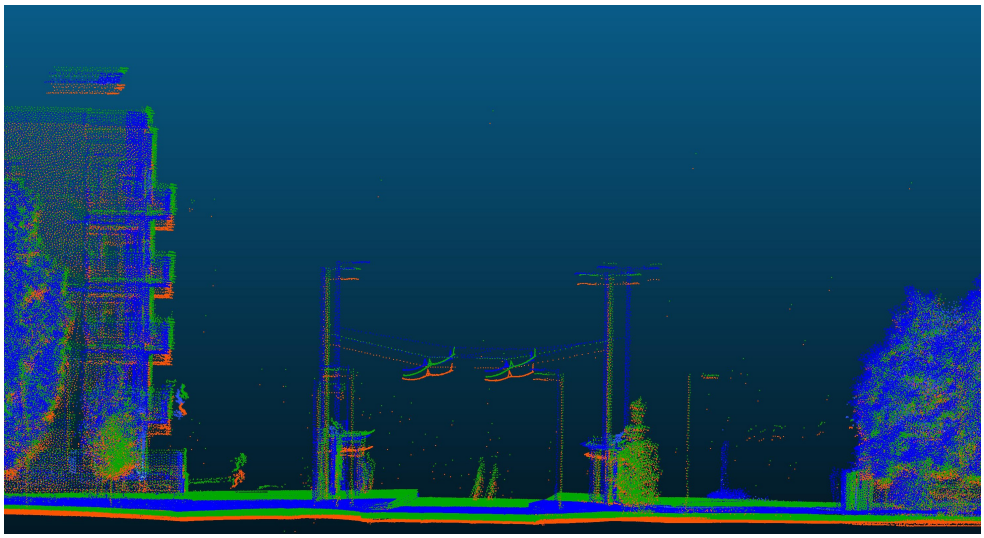
errors from the FEPPA results are higher compared to the PF results, this result is processed by the navigation system's software, therefore provides an updated position and attitude result for generating point clouds. RMS indicates the Root Mean Square Error, calculated by,

$$RMS = \sqrt{\frac{\sum_{n=1}^{N}(\hat{x}_n) - x_n)^2}{N}} \tag{17}$$

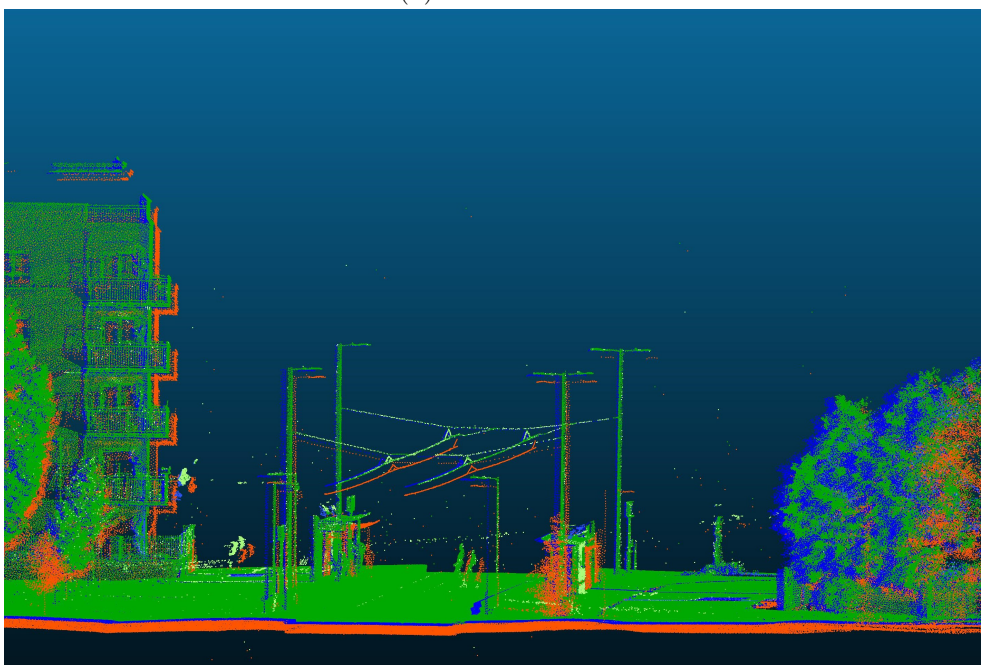
To compare the point cloud correction results, Figure 10 shows the a section of the final point cloud after being processed by three different methods, i.e. the proposed FEPPA workflow, ICP registration method and a control point method. Each corrected point cloud (shown in green) is overlaid on top of the reference point cloud (shown in blue) and the original comparison point cloud (shown in orange).



(a) FEPPA method



(b) ICP method



(c) Control point method

Figure 10.: Point cloud correction results after applying different correction methods (Green: the corrected point cloud; Orange: uncorrected point cloud; Blue: reference point cloud)

Figure 10(a) shows the reproduced point cloud using the FEPPA method, which gives an alignment difference of around 2cm overall. Figure 10(b) shows the results given by the CloudCompare ICP registration tool as described above. Results show that the error in the comparison data has reduced, but the performance decreases towards the edge of the data.

Figure 10(c) shows the results of the control point correction method, which is a more or less standardised manual process used to correct MLS datasets, discussed in Section 2.1. It usually gives the best results, but require long manual work to

survey control points and manually matching the point cloud to the control points. The standard workflow uses the TerraScan/TerraMatch commercial software (from the Terrasolid software suite (TerraScan 2016)) to input surveyed control point positions and manually find those control points in the point cloud, the user must tell the software the exact points representing the control point. The TerraMatch software stores this information for each control point and works out a regression model to correct the point cloud so that the position of the points matches the surveyed control points. To save the surveying time, here the extracted features used in the FEPPA process are used as control point inputs. The point cloud is then matched using the standard workflow described above.

Note that the results from the control point matching method does not seem to give better results than the FEPPA method. This is due to that in the actual standard workflow, the control point positions need to be surveyed to the millimetre accuracy level before being input into the software. Here, the position of the features are extracted and estimated from the reference dataset using the feature extraction process described in Section 3.1. Therefore introducing inconsistent errors exist during extraction and estimation. The quality of the regression model produced by the software could be reduced by such inputs, thus ouputing lower quality correction results. The comparison of the results in (a) and (c) simply shows that the FEPPA process is able to deal with measurement errors in the extracted features better than the TerraMatch software, which do not expect bad measurements as control inputs. It is not to state that the TerraMatch will not give good results when the standard procedure and good quality control points are used.

### 4.3. *FEPPA accuracy analysis*

The quality of the feature extraction results and the quantity of the features extracted are crucial to the performance of the FEPPA method. The effects of feature quantity on the processed navigation accuracy can be seen in Figure 11. The navigation data error after processing (in blue '·') is plotted against the number of features used during each epoch to correct the data (in red 'x'). The level of error is reflected by the density of the blue dots, i.e. darker regions indicate a high density of dots clustered in the low error section, lighter regions indicate a more spread-out error distribution, which implies a higher proportions of large errors. The red 'x' shows the number of features used in the FEPPA workflow for reprocessing the data. The plots shows that the resulting navigation error was relatively larger when less features were used in the correction workflow, whereas the navigation error reduced when more features were used.

Although the FDOP value is affected by both the number of features used and the geometry of the feature locations, for the analysed datasets here, it is mostly affected by the number of features, as the extracted features were mostly spread out on either side of the navigation system in a similar pattern. The produced FDOP value at each epoch played a role in changing the weighting scheme in the update step. Low FDOP values indicated more "visible" features, thus more measurements integrated into the workflow, which means that the particles' weights would change more severely if the measured range difference was high compared to when large FDOP values were computed.

Further to the number of features, two thresholds also changed the processing results, i.e. the time difference and distance difference threshold, as described in Step 5 in

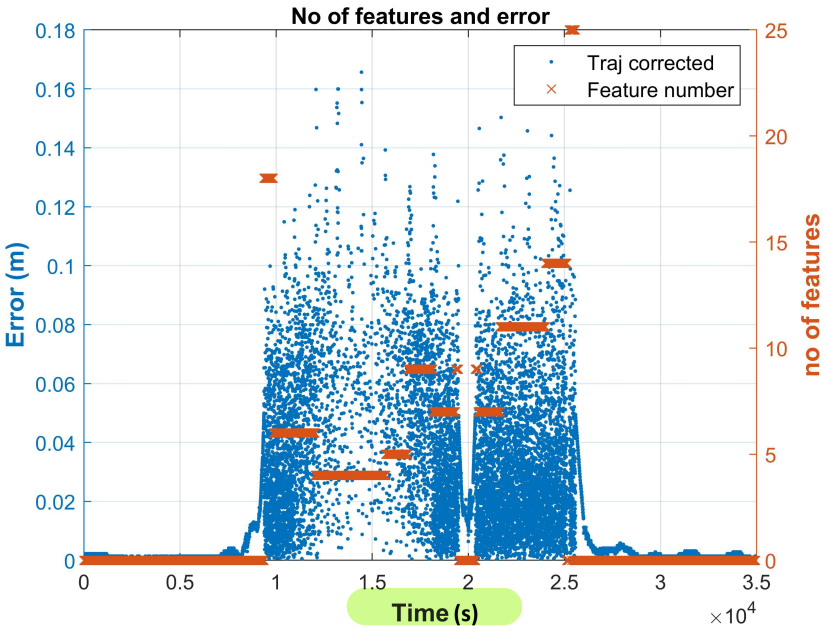



Figure 11.: Comparison between the number of features and the resulting navigation accuracy

Section 3.2. . Table 4 lists the navigation processing results for three test datasets after applying the FEPPA workflow. "Threshold $n$" indicates a different pair of thresholds selected.

The time threshold, which is used to select "visible" features at each epoch, was $8s$ for Threshold 1, 2 and 3, $5s$ for Threshold 4 and 5, and $10s$ for Threshold 6. The distance threshold, which is the predefined threshold for particle weighting, is set at $0.5m$ for Threshold 1 and 6, $0.3m$ for Threshold 2 and 4, and $0.1m$ for Threshold 3 and 5. The "% processed" column indicates the percentage of epochs that produced valid outputs out of the total epochs that required processing. Invalid outputs are regarded as epochs where the produced results were unreasonable or could not be processed due to insufficient feature information available.

Table 4.: PF processing results with different parameters

| Data | | Mean (m) | Max (m) | % processed |
|---|---|---|---|---|
| **Data 1** | Before corrections | 0.919 | 1.880 | |
| | Threshold 1 | 0.232 | 1.646 | 99.9% |
| | Threshold 2 | 0.145 | 1.669 | 99.8% |
| | Threshold 3 | 0.107 | 1.715 | 97% |
| **Data 2** | Before corrections | 1.967 | 9.678 | |
| | Threshold 4 | 0.856 | 9.678 | 60.9% |
| | Threshold 3 | 0.473 | 9.678 | 60.4% |
| | Threshold 5 | 0.021 | 0.164 | 31.1% |
| **Data 3** | Before corrections | 0.543 | 4.038 | |
| | Threshold 1 | 0.032 | 0.648 | 100% |
| | Threshold 2 | 0.021 | 0.386 | 84.8 % |
| | Threshold 6 | 0.033 | 0.678 | 100 % |

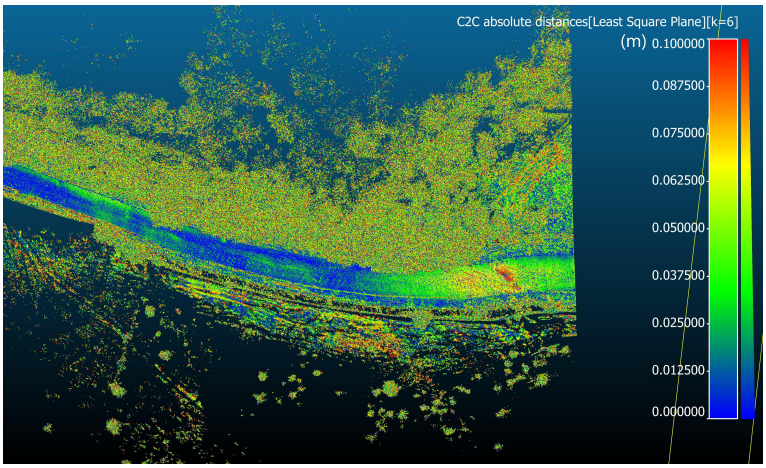
Both the time and distance threshold presents a trade-off between accuracy and percentage of valid outputs. As mentioned Section 3.2, the "visible" features are selected by the time difference between the current epoch and the time when the data representing the feature were captured. Due to that the navigation errors continously change over time, ideally only features captured in the same epoch can reflect the same level of error. However, this limits the number of features that can be used and therefore reduces the update measurement input to help correct the trajectory. Increasing the time threshold would mean that more "visible" features could be extracted, i.e. reducing the $FDOP$ value, integrating more measurements in the filter and potentially producing better accuracy. However, if the time difference continues to grow, the additional included features may no longer reflect the navigation error of the current epoch. In the results for Data 2 and Data 3, a decrease in the accuracy starts to show when the time threshold increases. Overall for all tested data, $8s$ is a tolerable threshold. However, this also depends on the speed of the navigation system and the density of features in the environment.

For the distance threshold, a tighter threshold, i.e. short distance, should be applied to improve accuracy, which gives significantly higher weights to particles that have better range estimations to the features and lowers weights to others. However, introducing a tighter threshold could reduce the number of valid particles left after each updating step. Therefore, less capable of dealing with situations when errors are high in the measurement input. This could reduce the percentage of valid output and cause the problem of particle impoverishment more frequently. This could be seen in the comparison of results for Data 1, Threshold 3 and 4 for Data 2, and Threshold 1 and 2 for Data 3, where tighter distance thresholds produced better average results but also reduced the total number of valid epochs. As such, the actual threshold should be tuned based on the estimated data accuracy, desired accuracy and feature extraction quality to maximise the percentage of valid processed data while also reducing the navigation error.

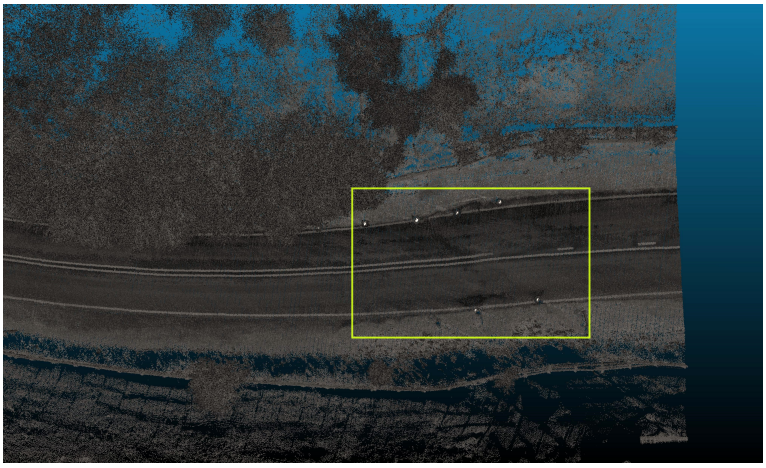
Finally, an example of change detection result is given below based on the FEPPA corrected data. Change detection is performed by carrying out C2C computation between the processed comparison data and the reference dataset. Figure 12a shows a section of the highway data where a change on the ground (hightlighted in red) had been detected on a section of the road. Figure 12b shows the comparison point cloud data coloured by intensity. Here we can see the cause of the change, i.e. a crack on the road showing up as a diagonal marking along the road. Theses results indicate the potential of using the FEPPA workflow to reduce the workload for processing MLS point cloud data and being able to carry out data analysis such as these more quickly.

## 5. Conclusions and future work

LiDAR systems, especially mobile LiDAR, have become increasingly popular for a range of applications, including surveying, monitoring and modelling. However, the methods to correct and control the 3D point cloud accuracy and quality from a mobile laser scanning system has been challenging due to demanding manual workloads. This paper introduces an improved method to correct the MLS navigation accuracy as well as the point cloud quality by using the feature information extracted from the point cloud data. It is demonstrated through two different types of data: 1) a suburban/urban area with a variety of infrastructure and clear objects; 2) a highway in the rural environment with overgrowing trees on either side, which causes prob-




(a) Road change detection (comparing data corrected by FEPPA to a reference dataset)



(b) Corrected point cloud of the road shown in intensity

Figure 12.: Change detection results after applying FEPPA (colour bar indicates difference in metres)

lems in navigation and conventional point cloud registration. The proposed FEPPA method first extracts useful features from the point cloud data and integrates their location information into the navigation PF algorithm to update the trajectory. The final step reproduces the point cloud using the updated trajectory. Results show that the workflow is able to reduce the mean error by around 56% - 88% and the maximum error by around 93%. One of its main advantages compared to conventional MLS data processing is the huge reduction in processing time and effort. As most steps here are automated, less skills and time effort is required to produce a reliable point cloud data.

However, a few issues were briefly mentioned in the paper but not address as part of this work. First of all, the FDOP was measured as part of the FEPPA workflow. Yet, in the analysed data, it did not reflect much information on the geometry and was more of an indication of the number of features available. This was limited to the environment of the data and the nature of the features were placed along the road/rail side. The algorithm was also used only to correct the 2D position of the trajectory due to that thee features were mostly spreadly spread out in the horizontal plane relative to the navigation location. As there was insufficient height change in

the features, it was not used to correct the height information in this case. In future simulations, analysis should be carried out on how the 3D geometry of the features affect the accuracy. Secondly, the reprocessing procedure in the final step of FEPPA allows the software to produce a smoothed data and therefore generate clear point clouds. However, as seen in the results' plot, the smoothed data introduces some errors. This is due to the way the software integrated GNSS and IMU data. However, in this case the IMU provider did not provide an open source IMU data, therefore limited the possibility of integrating the raw measurements in the PF algorithm. Such effects should be investigated in future work with open source IMU measurements. Finally, the feature errors are not linearly correlated to the navigation error due to the changing attitude measurements of the IMU. How the attitude affects the error and how the integration should tune parameters based on the attitude measurements should also be investigated in future work.

## References


Assfalg, J., Bertini, M., Del Bimbo, A. and Pala, P., 2007. Content-based retrieval of 3-D objects using spin image signatures. *IEEE Transactions on Multimedia*. 9(3), 589–599.

Bitenc, M., Lindenbergh, R., Khoshelham, K., Waarden, V. and Pieter, A., 2011. Evaluation of a LiDAR land-based mobile mapping system for monitoring sandy coasts, *Remote Sensing*. 3(7), 1472–1491.

Bosch, A., Zisserman, A. and Munoz, X., 2007. Image classification using random forests and ferns. *2007 IEEE 11th international conference on computer vision,IEEE*. 2007, 1–8.

Brodu, N. and Lague, D., 2012. 3D terrestrial lidar data classification of complex natural scenes using a multi-scale dimensionality criterion: Applications in geomorphology. *ISPRS Journal of Photogrammetry and Remote Sensing*. 68, 121–134.

Chen, H.M., Ulianov, C. and Shaltout, R., 2015. 3D laser scanning technique for the inspection and monitoring of railway tunnels. *Transport problems*. 10(SE).

Gézero, L. and Antunes, C., 2017. A registration method of point clouds collected by mobile LIDAR using solely standard LAS files information. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*. Vol. XLII-1/W1, 2017.

Gordon, N.J., Salmond D.J. and Smith, A.F., 1993. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEE proceedings F (radar and signal processing)*. Vol. 140, IET, 1993, 107–113.

Gustafsson, F., Gunnarsson, F., Bergman, N., Forssell, U., Jansson, J., Karlsson,.R. and Nordlund, P.-J., 2002. Particle filters for positioning, navigation, and tracking. *IEEE Transactions on Signal Processing*, 50(2), 425 - 437.

Hänsch, R., Weber, T. and Hellwich, O., 2014. Comparison of 3D interest point detectors and descriptors for point cloud fusion. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*. Vol. II, Copernicus GmbH, 2014, 57.

Hutton, J., Gopaul, N., Zhang, X., Wang, Menon, V., Rieck, D., Kipka, A. and Pastor, F., 2016. Centimeter-level, robust GNSS-aided inertial post-processing for mobile mapping without local reference stations, *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*. Vol.XLI-B3, 2016, 819–826.

Jaboyedoff, M., Oppiker, T., Abellan, A., Derron, M., Loye, A., Metzger, R. and Pedrazzini, A., 2010. Use of LIDAR in landslide investigations: a review. *Natural Hazards*. 61 (2012) 5–28.

Jende, P., Peter, M., Gerke, M. and Vosselman, G., 2016. Advanced tie feature matching for the registration of mobile mapping imaging data and aerial imagery. *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences*. Vol. XLI-B1, 2016.

Jing, H., Slatcher, N., Meng, X. and Hunter, G., 2016. Monitoring capabilities of a mobile mapping system based on navigation qualities. *ISPRS-International Archives of the Pho-*

*togrammetry, Remote Sensing and Spatial Information Sciences*. Vol. XLI-B1, 2016, 625–631.

Kalman, R.E., 1960. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*. 82(1), 35–45.

Knopp, J., Prasad, M., Willems, G., Timofte, R. and Van Gool, L., 2010. Hough transform and 3D SURF for robust three dimensional classification. *European Conference on Computer Vision*. Springer, 2010, 589–602.

Kukko, A., Kaartinen, H., Hyyppä, J. and Chen, Y., 2012. Multiplatform mobile laser scanning: Usability and performance, *Sensors*. 12(9), 11712–11733.

Lato, M., Hutchinson, D., Harrap, R., Diederichs, M. and Ball, D., 2009. Engineering monitoring of rockfall hazards along transportation corridors: Using mobile terrestrial LiDAR. *Natural Hazards and Earth System Sciences*, 9, 935–946.

Lauterbach, H., Borrmann, D., Heß, R., Eck, D., Schilling, K. and Nüchter, A., 2015. Evaluation of a backpack-mounted 3D mobile scanning system. *Remote Sensing*.7(10), 13753–13781.

Lehtomäki, M., Jaakkola, A., Hyyppä, J., Lampinen, J., Kaartinen, H., Kukko, A., Puttonen, E. and Hyyppä, H., 2016. Object classification and recognition from mobile laser scanning point clouds in a road environment. *IEEETransactions on Geoscience and Remote Sensing*. 54(2), 1226–1239.

Lindenbergh, R. and Pietrzyk, P., 2015. Change detection and deformation analysis using static and mobile laser scanning. Applied Geomatics. 7(2), 65–74.

Maghiar, M., Maldonado, G., Newsome, S., Clendenen, J. and Jackson, M., 2016. Accuracy comparison of 3D structural models produced via close-range photogrammetry and laser scanning. *Construction Research Congress*. 2016.

Marden, S. and Guivant J., 2012. Improving the performance of ICP for real-time applications using an approximate nearest neighbour search. *Proceedings of the Australasian Conference on Robotics and Automation*. Wellington, New Zealand, 2012, 3–5.

Mukupa, W., Roberts, G.W., Hancock, C.M. and Al-Manasir, K., 2017. A review of the use of terrestrial laser scanning application for change detection and deformation monitoring of structures. *Survey Review*. 49(353), 99–116.

Network Rail, 2016. `https://media.sa.catapult.org.uk/wp-content/uploads/2016/10/10154511/Network-Rail-Industry-Challenge-Statement-101016.pdf` *Challenge statement: Earthwork monitoring from above*.

Pu, S., Rutzinger, M., Vosselman, G. and Elberink, S.O., 2011. Recognizing basic structures from mobile laser scanning data for road inventory studies, *ISPRS Journal of Photogrammetry and Remote Sensing: Advances in LiDAR data processing and applications*. 66(6), S28–S39.

Puente, I., González-Jorge H., Martínez-Sánchez, J. and Arias, P., 2013. Review of mobile mapping and surveying technologies. *Measurement*. 46(7), 2127–2145.

Puente, I., Solla, M., González-Jorge, H. and Arias, P., 2013. Validation of mobile LiDAR surveying for measuring pavement layer thicknesses and volumes. *NDT & E International: Independent Nondestructive Testing and Evaluation*. 60, 70–76.

Ristic, B., Arulampalam, S. and Gordon, N. *Beyond the Kalman Filter: Particle filters for tracking applications*. Artech House, Boston, London, 2004.

Rusu, R.B., Blodow, N., Marton, Z.C. and Beetz, M., 2008. Aligning point cloud views using persistent feature histograms. *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE*. 3384–3391.

Rusu, R.B., Blodow, N. and Beetz, M., 2009. Fast point feature histograms (FPFH) for 3D registration. *IEEE International Conference on Robotics and Automation, IEEE*. 3212–3217.

Tang, J., Chen, Y., Niu, X., Wang, L., Chen, L., Liu, J., Shi, C. and Hyyppä, J., 2015. LiDAR scan matching aided inertial navigation system in GNSS-denied environments. *Sensors*. 15(7), 16710–16728.

TerraSolid, 2016. `https://www.terrasolid.com/download/tscan.pdf` *TerraScan User's Guide*.

Toth, C., Koppanyi, Z., Navratil, V. and Grejner-Brzezinska, D., 2017. Georeferencing in GNSS-challenged environment: Integrating UWB and IMU technologies. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*. Vol. XLII-1/W1, 2017.

Williams, K., Olsen, M., Roe, G. and Glennie, C., 2013. Synthesis of transportation applications of mobile LiDAR. *Remote Sensing*. 5(9), 4652–4692.

Wu, M.L., Chien, J.C., Wu, C.T. and Lee, J.D., 2018. An augmented reality system using improved-iterative closest point algorithm for on-patient medical image visualization. *Sensors*, 18(8), 2505.

Xiao, W., Vallet, B. and Paparoditis, N., 2013. Change detection in 3D point clouds acquired by a mobile mapping system. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*. Vol. II-5/W2, 2013, 331–336.

Yang, B., Dong, Z., Liu, Y., Liang, F. and Wang, Y., 2017. Computing multiple aggregation levels and contextual features for road facilities recognition using mobile laser scanning data. *ISPRS Journal of Photogrammetry and Remote Sensing*. 126, 180–194.

Zhong, Y., 2009. Intrinsic shape signatures: A shape descriptor for 3D object recognition. *2009 IEEE 12th international conference on computer vision workshops*. ICCV Workshops, IEEE, 2009, 689–696.