From Point Cloud Data to IfcAlignment

July 27, 2018

Report
Advanced Topics in Building Information Modeling

Helge Hecht
Contents

1 Introduction 2

2 State of the Art 4
   2.1 Data Collection ................................................. 4
   2.2 Segmentation and Recognition ................................. 6
   2.3 Derivation of Alignment Parameters ......................... 9

3 Methodology 10
   3.1 Overview .......................................................... 10
   3.2 Preprocessing ..................................................... 11
   3.3 Rail Segmentation ................................................. 12
   3.4 Centerline Extraction ........................................... 14
   3.5 Alignment Parameter Estimation ............................... 14
   3.6 Software Implementation ....................................... 14

4 Results 17

5 Discussion 20

6 Conclusion 22
Chapter 1

Introduction

The application of Building Information Modeling (BIM) in infrastructure projects is slowly picking up pace but is still far behind compared to the more traditional building construction. Nevertheless, there is great potential in the application of BIM in those projects, since they involve intense collaboration, transparency and are usually associated with large costs (Ehrbar, 2017; Matějka, 2014). While the infrastructure required for naval or aerial transport consist of traditional building construction, roads and railways have special demands due to their vast expanse. Among those two, railways are more complicated since they have additional requirements and all norms and regulations have to be fulfilled in a strict way. A road being constructed a couple of centimetres too narrow or too wide does not pose a problem while railways for high speed trains have to be accurate at a millimetre level which makes them less fault tolerant. Classic building constructions use mostly rectangular and flat shapes and have fixed boundaries whereas infrastructure mostly revolves around an horizontal and vertical alignment and includes the environment which rarely features completely flat or rectangular shapes (Sheng et al., 2014). The requirements for a standardized representation of alignments used in infrastructure is described in Thomas Liebich (2015).

A BIM usually contains information about the whole life cycle of an object. This includes the planning phase, the construction process, the management cycle and the dismantling. Out of those, the management cycle covers the longest period of time - especially in transport infrastructure, which is basically built to last forever. This shows the vast amount of unused potential which are all existing transport routes that are not managed with BIM. Facility management of infrastructure revolves mainly about maintenance, as-planned vs. as-built comparison and possible alterations and extensions (Shr & Liu, 2016).

This project therefore proposes the recognition of an IfcAlignment from Point Cloud Data which allows a qualitative as-built analysis and is the basis of more complex procedures like maintenance. It replaces the manual modeling of existing railways in a CAD tool or
the extraction of all information required to create an IFC based description from existing documentation. The automation of this procedure is necessary since manual collection of all required data is very work intensive. It is unique in such a way that the complete toolchain starting with raw 3D point data to a parametrized IFC4X1 standard conform IfcAlignment representation is developed.

An overview about current developments, potentials and remaining challenges is given in 2, followed by the developed methods and approaches in 3. The algorithms and software implementations are also presented in the last section of the 3. chapter and are freely available as part of the TUM OpenInfraPlatform. The results presented in 4 are based on real data collected on more than 100km of railways in Germany and are achieved using the implemented software. Problems encountered during the implementation and the conceptual design of the algorithms and methods are discussed in 5 and future developments and improvements are proposed in 6.

Figure 1.1: Visualization of a point cloud, the corresponding octree and section bounding boxes in the TUM OpenInfraPlatform.
Chapter 2

State of the Art

2.1 Data Collection

A big portion of the work requirements in railway maintenance is the collection of data required to reconstruct a 3D digital model which can then be further processed. Ongoing work which involves the physical attendance of labourers at the rail track results in having to lock the track down during the work process (Marc Krückmann & Schneider, 2017). This again requires further preparation and detours have to be provided. It is therefore desirable to automate the data collection procedure in such a way that the regular operation is not affected. Current state of the art methods revolve around mobile or aerial laser scanning and camera based 3D reconstruction by means of photogrammetry. Detailed descriptions and comparisons of both procedures can be found in Lichti et al. (2002); Grussenmeyer et al. (2008).

**Laser Scanning**  So called *Mobile Laser Scanning* (MLS) systems are portable and smaller versions of classic terrestrial laser scanners and are usually mounted on vehicles to be able to perform scans from different angles and positions (Elberink et al., 2013; Arastounia, 2015; Arastounia & Elberink, 2016). In this special case of railway scanning, the MLS system is usually mounted directly on the vehicle to scan the currently taken route and its surroundings. An example is shown in figure 2.1. The MLS outputs a 3D point cloud with high uniform density and fairly high precision. Laser scanners are not capable of capturing sharp edges sind the points are created in a grid shape, not based on keypoints or distinct features.

**Photogrammetry**  Photogrammetry revolves around the reconstruction of 3D data from sets of 2D images or videos. The resulting point cloud depends on the quality of images, the difference between consecutive images of the same object, the shape of the object itself and
Figure 2.1: A MLS mounted onto a train equipped with additional measurement tools. Image taken from Arastounia (2015).

the reconstruction method. The most prominent algorithms for the reconstruction of point clouds from images are Structure from Motion (SfM) and Scale-Invariant Feature Transform (SIFT).

**SfM** Structure from Motion tries to detect identical keypoints in a series of images and incorporates the information about the change of viewpoint between the images. It is therefore necessary to keep track of these properties.

**SIFT** The SIFT algorithm detects keypoints which are invariant to transformations as well as illumination to align the images and extract common features.

A detailed description of SfM and SIFT can be found in Westoby et al. (2012) and Sun et al. (2014) respectively. Teza et al. (2016) gives a detailed comparison of SfM and terrestrial laser scanning.
2.2 Segmentation and Recognition

The key aim of most state of the art methods is the segmentation of the rail tracks to compute a centreline. These are mostly based on either neighbourhood analysis (Arastounia, 2015) or template/pattern matching (Arastounia & Elberink, 2016) or a combination of both for coarse and fine segmentation (Elberink et al., 2013).

Neighbourhood Analysis
The neighbourhood of a point, meaning the point set \( N = \{n_0, n_1, \ldots, n_i\} \) lying within a sphere of radius \( r \) centred on point \( p \), is analysed to classify the point as one of the considered classes. For \( k \) classes, this usually means implementing \( k \) binary classification algorithms.

Elberink et al. (2013) implement separate algorithms for the detection of wire points and rail points. They rasterize the entire cloud in \( 1 \times 1 \) meter grid cells in \( x \) (east) and \( y \) (north) dimension and process the points inside these grid cells. If 90% of the points inside a grid cell are less than half a meter above the Digital Terrain Model (DTM) and the remaining 10% reside between 0.5 and 4.5 meters above the DTM, the cell is considered for further processing. Otherwise it is labelled as not containing rail or wire points. Given the case that the cell is considered as containing rail and wire points, the lowest 5cm of all points between 5.5m and 6.5m are segmented as wire points while all other points between the wires and half a meter above DTM height are considered for rail detection. In case that the difference between the highest 2% and the lowest 10% of the points is larger than 10cm, the lowest one of the points forming the upper bound and all points in a 10cm sphere around it are marked as potential rail points.

Arastounia (2015) implements algorithms based on neighbourhood analysis for the classification of the track bed, rail tracks, overhead power cables and masts and cantilevers.

Track Bed The Histogram of the standard height deviation within \( N \) of radius 1m is used to detect the track bed. The decision boundary is constructed by taking the first bin further to the right that has a high histogram gradient. This is depicted in figure 2.2. Since this might still include non track bed points, a region growing approach with a search radius of 20cm is used to generate clusters. The largest cluster is then considered as the track bed. This also includes the rail track points, so that only the track bed points are considered for rail detection.

Rail Tracks Another standard height deviation Histogram using a 20cm spherical neighbourhood is computed and used for rail track detection. The Histogram gradient is used to detect the two primary peaks, where the larger one represents the track bed and the smaller bin representing a larger height deviation the rail tracks. To get a
connected rail component, a two stage region growing approach is used. The first step is similar to the one used for track bed segmentation. The second one starts with a random cluster and searches for candidate clusters within 1m proximity. If the angle between the vector connecting the candidate and the original cluster and the initial direction of the original cluster is less than 20°, the two clusters are connected.

**Power Cables** The power cables are recognized by computing the vector for each point to the respective closest point on the trackbed. If the projection of this vector along the z (height) axis is larger than 0.8. The power cables consist of three different types: Contact cables, which are linear, catenary and return cables which are curvilinear-shaped. To check a neighbourhood for linearity, Principal Component Analysis (PCA) is used. Region growing using a 20cm threshold is then performed. The growing thresholds are a 20° angle difference and 20cm height difference, to distinguish the contact cables form the other types. Afterwards, the projection of the vector from potential cable points to the track bed on the z axis is repeated, but now for all points above the contact cables and the closest contact cable point instead of the track bed respectively. This group forms the catenary cables. They are then again clustered with 20cm search range region growing. The last remaining cable type is segmented as all points above the catenary points which have a linear neighbourhood.

**Masts and Cantilevers** All points remaining above the track bed are now considered to be either masts or cantilevers. Region growing from a random seed point is used to segment everything until the track bed. To distinguish between the two remaining classes, the 1m spherical neighbourhood of these points is inspected. If the distribution in the horizontal is less than 20cm and the projection of the vector connecting the two points with the largest distance in the neighbourhood along the z axis is larger than 0.8, the point is classified as mast point, otherwise as cantilever.

**Template Matching**
Matching predefined structures to the data to find occurrences of these patterns is called template matching. These structures can be based on the point cloud representation of the data or follow different metrics (Elberink et al., 2013; Arastounia & Elberink, 2016). Same as for neighbourhood analysis, this technique can be used to detect various objects like rail tracks and cables in the point cloud. The template matching performed in Elberink et al. (2013) uses a parametrized rail track model which is fitted to the previously coarsely segmented rail track points. The only free parameters in the fitting of the model are the position and orientation combined in Θ. These parameters are estimated using a Markov Chain Monte Carlo algorithm. The aim is to find argmax \( P(M_i | D) \), the configuration that maximizes the probability of being generated by the given dataset. Using Bayes’ rule, \( P(M_i | D) \) can
be calculated as \( P(M_i \mid D) = \eta P(D \mid M_i) P(M_i) \) where \( P(M_i) \sim \mathcal{N}(\Theta, \mu, \Sigma) \) is the prior distribution. The mean vector is initialized with the position and orientation of the previous segment and the covariance matrix \( \Sigma \) is chosen to assume no correlation between the parameters. The actual Markov Chain Monte Carlo sampling is then performed using the Metropolis-Hastings algorithm.

While the previously described template matching algorithm tries to fit a realistic model of the object under inspection to the actual data, Arastounia & Elberink (2016) developed a different approach based on small grey scale images. After a coarse, grid based segmentation, a top view image of the point cloud where each pixel represents a 0.1m \( \times \) 0.1m area of the railroad environment. The pixel color is determined based on the projected points: rail track points are coloured white, contact cables grey and everything else black. This template is then matched to all pixels of the point cloud image in a sliding window procedure. For each pixel, the template is used in all possible rotations, from \(-90^\circ\) to \(+89^\circ\). The correlation of the template at location \( x \) is computed as

\[
\text{Correlation}(x) = \frac{\sum_{i=-N}^{N} (T(i) \times I(x+i))}{\sqrt{\sum_{i=-N}^{N} (T(i))^2} \times \sqrt{\sum_{i=-N}^{N} (I(x+i))^2}} \tag{2.1}
\]

\( T(i) \) denotes the value of the template at position \( i \), given that the template is of size \( 2N \), and \( I(x+i) \) the value of the projected cloud at position \( x+i \), given that the template is centered on pixel \( x \). If this correlation value exceeds a certain threshold for a rotation,
the rail pixels under the template are considered for further calculations, otherwise not. To improve computation performance, segments after the first one only use the first determined rotation $\pm 20^\circ$.

### 2.3 Derivation of Alignment Parameters

The estimation of parameters for horizontal alignment segments is based on the bearing diagram analysis (Gikas & Stratakos, 2012). The Azimuth, the angle between the alignment direction and the true north vector, is computed for each point along the alignment. This leads to a function mapping from chainage to bearing.

\[
a = \arctan \left( \frac{dx}{dy} \right) \frac{180^\circ}{\pi} \quad (\text{Bearing})
\]

\[
a' = \frac{a_{i+1} - a_i}{s_{i+1} - s_i} \quad (\text{Curvature})
\]

\[
a'' = \frac{a'_{i+1} - a'_i}{s_{i+1} - s_i} \quad (\text{Change of Curvature})
\]

The derivatives of the bearing function with respect to the chainage are the curvature and the rate of change of curvature. These are used to identify linear, circular arc and clothoidal sections and to estimate their parameters. A curvature value $a'_i < a'_{\min} = 0.002^\circ/m$ is considered as belonging to a linear segment, while a constant non-zero curvature represents a circular arc. Clothoidal segments have a changing curvature value, but the rate of change of curvature $a''$ is constant nonzero. To estimate the radius of a circular arc segment, the frequency histogram of the curvature $P(a'_i) = \frac{n_i}{N}$ where $n_i$ is the histogram value for the bin holding $a'_i$ and $N$ the total number of samples in the circular arc segment. An initial guess for the radius can then be computed as

\[
R = \frac{180^\circ}{\pi \cdot \arg\max_{a'} P(a')},
\]

the inverse of the maximum frequency curvature. Since this initial guess depends on the bin width and other parameters, a simple least squares regression solving the equation for points on a circle leads to better results. Once the start and end points for the linear segments and the circular arc segments are determined, the remaining sections are transition curves, out of which the clothoid is the most prominent one. Given the length of the segment $L$ and the radius of the adjacent circular arc $R$, the clothoid parameter $A$ can be computed as

\[
A = \sqrt{L \cdot R}.
\]

Other transition curve segments like Bloss curves are not considered by Gikas & Stratakos (2012).
Chapter 3

Methodology

3.1 Overview

The proposed algorithms are based on 3D neighbourhood analysis of the point cloud data. The whole procedure can be split into several stages: (1) preprocessing, (2) rail segmentation, (3) centerline extraction, and (4) alignment parameter estimation. The whole pipeline is depicted in figure 3.1.

Figure 3.1: Proposed pipeline for automatic parametrized railroad detection.
The proposed algorithms are evaluated on ~5km of real data collected using a MLS system mounted on a train. The point cloud contains two rail tracks inside a gallery. No color, surface normals or intensity information is used.

### 3.2 Preprocessing

The point cloud data is preprocessed to reduce the point cloud size and extract the points which are potential rail track points. The first step is to discard points depending on their height value. A $10 \times 10$ meter grid is computed and only points $\pm 0.5m$ around the median value are kept. Afterwards, the number of points in a spherical neighbourhood with radius 10cm is computed for each point and all points having less than 20 neighbours are discarded. This removes outliers and rather sparse areas such as the track bed surroundings. The parameters for the minimum threshold as well as the kernel radius are determined empirically as having the best performance in the inspected datasets. These values can of course differ for point clouds generated with different devices or technologies.

The next step is to perform spherical neighbourhood analysis for all points to detect potential rail track points. The algorithm is based on the method presented in Elberink et al. (2013) but differs in a way such that no $1 \times 1$ meter grid cells are used, but the 15cm spherical neighbourhood of each point. The exact algorithm is described in algorithm 1.

#### Algorithm 1: Rail point segmentation

**Data:** Point cloud $P = \{p_0, p_1, \ldots, p_n\} \in \mathbb{R}^3$

**Result:** Point cloud $P' = \emptyset$

1. begin
2.     for $p \in P$ do
3.         $N = \{n \in P \mid \|n - p\| \leq 0.15\}$
4.         sort($N$) $\rightarrow \forall n \in N : n^{i-1} < n^i$
5.         $u = \lfloor 0.98 \cdot |N| \rfloor$
6.         $l = \lfloor 0.1 \cdot |N| \rfloor$
7.         if $\|n^u_z - n^l_z\| \geq 0.15$ and $\|n_z^{\lfloor |N| - 1 \rfloor} - n_0\| \leq 0.25$ then
8.             for $i = u$ to $|N| - 1$ do
9.                 $P' = P' \cup n^i$
10.            end
11.        end
12.    end
13. end

The last step in the preprocessing stage is to compute the average difference in height in a 5cm radius spherical neighbourhood around each point and only points with an average difference of 2mm are kept. For a detailed description see algorithm 2. The purpose of this
step is to reduce the points on the rail track to the top side points of both - left and right - rail track to also allow for the derivation of a vertical alignment and for measuring the height difference between the left and the right rail track. The underlying assumption is that the top side of the rail tracks is a flat surface with high point density, since it is a homogeneous surface. Areas with high deviations in height distribution will be filtered out, such as the edges of the track bed.

Algorithm 2: Height difference based segmentation

Data: Point cloud $P = \{p_0, p_1, \ldots, p_n\} \in \mathbb{R}^3$

Result: Point cloud $P' = \emptyset$

1 begin
2  for $p \in P$ do
3     $N = \{n \in P \mid \|n - p\| \leq 0.05\}$
4     $diff = \frac{1}{|N|} \sum_{n \in N} \|p_z - n_z\|$
5     if $diff \leq 0.002$ then
6         $P' = P' \cup p$
7     end
8  end
9 end

The resulting point cloud contains far less points and primarily rail points are kept. This reduces the number of computations for the next steps and also improves visualization performance.

3.3 Rail Segmentation

To detect the rail points after preprocessing, the first step is to compute the chainage for each point, which represents its position along the axis or alignment under examination. This is done in an octree cell wise manner. The data is split in octree cells with $\sim 100\text{m}$ size in each dimension. For each cell, the center of mass is computed and all points in a $100\text{m}$ sphere radius are collected. The main axis of all these points and the projection length of the center of mass along this axis - the chainage - is computed and stored. Afterwards, the chainage values for all points are interpolated using the main axes of the two closest cell centers-of-mass. Due to irregularities in the distribution of cells, only cells having an above half times average population are considered.

This chainage is then used to split the dataset into sections whose main extend is orthogonal to the alignment axis. This step is done in order to reduce the search space for possible rail point pairs since possibly matching points are only sought after in the same section. Each
section can cover an adjustable amount of distance along the main axis. A rendered image displaying the bounding boxes around the points belonging to a section is shown in 3.2.

![Figure 3.2: The computed sections of points along the main axis. The bounding boxes around each axis are drawn in blue. Sections each cover 40cm along the main axis.](image)

The last step is to find point pairs of points belonging to the left and right rail track respectively. The standard European gauge width is 1.435m and the width of rail track heads is 0.067m. The distance of paired points should be very close to this value, allowing a 1cm error. Instead of inspecting all points in a 1.5m spherical neighbourhood, points pairs are only sought after inside a section.

**Algorithm 3: Pair detection**

**Data:** Point cloud section \( S = \{p_0, p_1, \ldots, p_n\} \in \mathbb{R}^3 \)

**Result:** Point pairs \( P = \emptyset \)

\[
\begin{align*}
\text{begin} & \quad \text{gauge} \leftarrow 1.435 \\
& \quad \text{head} \leftarrow 0.067 \\
& \quad \epsilon \leftarrow 0.01 \\
\text{for} \quad \forall (p_i, p_{ii}) \in S \times S : i < ii \quad \text{do} & \\
& \quad \text{error} = ||p_i - p_{ii}|| - \text{gauge} - \text{head} \\
& \quad \text{if} \quad \text{error} < \epsilon \quad \text{and} \quad |p_{iz} - p_{iz'}| < 0.2 \quad \text{then} \\
& \quad \quad \text{if} \quad \forall (p_i, p_{ii'}) \in P : ||p_{ii'} - p_{ii}|| > 0.1 \quad \text{then} \\
& \quad \quad \quad \quad P = P \cup (p_i, p_{ii}) \\
& \quad \quad \text{end} \\
\text{end} & \\
\text{end}
\end{align*}
\]

The algorithm for point pair detection as described in algorithm 3 does not allow for pairs having partly identical points within close proximity. This is in order to reduce the number of detected pairs and to establish a one-to-one correspondence. After the pair detection,
the pairs are filtered according to the density in a small spherical neighbourhood, to remove outliers and false pairs.

### 3.4 Centerline Extraction

The centerline is extracted out of the centerpoints of the pairs computed in the previous step. For each pair, the centerpoint is added to the cloud together with the chainage value. These centerpoints are then filtered for duplicates, which means the minimum point distance is 2mm. This is necessary to make the density of the centerlines more uniform since PCA is sensitive to mass concentrations. Since a point cloud can contain several tracks, these centerpoints have to be sorted into connected line segments. The procedure is described in algorithm 4.

After sorting the points into lines, these lines are smoothed by splitting it into 1cm segments and replacing all points in each segment by the segment’s center of mass.

### 3.5 Alignment Parameter Estimation

The first parameter which has to be computed is the bearing. To determine the direction at a single point, all points 10m before and after the point under inspection are used to compute the main axis in \( x \) and \( y \) dimension. This leads to the following equation for the bearing:

\[
\text{acos} \left( \text{axis} \cdot \begin{pmatrix} 0 \\ 1 \end{pmatrix} \right) \cdot \frac{180}{\pi}
\]

(3.1)

The first and second derivatives of the bearing function, the curvature and the change of curvature are computed as described in Gikas & Stratakos (2012), besides introducing an averaging step between the first and the second derivative, to reduce the noise, and allowing for a variable curvature step size instead of using fixed 5cm. Computing the linear, circular arc and clothoidal section parameters as well as their start- and endpoints can be implemented in the manner as described in chapter 2 section 2.3 using the respective equations.

### 3.6 Software Implementation

The proposed algorithms and procedures are implemented in the TUM OpenInfraPlatform, an open source software project. It is published under the GPL license. The library used for basic point cloud functionality is the CloudCompare library. This includes file I/O, adding attributes to a point cloud - so called Scalar Fields - , building an octree on the cloud as well as
Algorithm 4: Sorting centerpoints into centerlines

**Data:** Centerpoints $P_C = \{p_0, p_1, \ldots, p_n\} \in \mathbb{R}^3$

**Result:** Centerlines $L = \{l_0\}$, $l_0 = \{p_0\}$

```
begin
for $p \in P_C$ do
    inserted ← false
    for $l \in L$ do
        $N_l ← |l|$
        for $n = 1$ to $n < \min(100, N_l)$ do
            $p_{end} = p_0^{N_l-n} \in l$
            $dist = \|p - p_{end}\|$
            $\delta_{Chainage} = |\text{chainage}(p) - \text{chainage}(p_{end})|$
            $ratio = \frac{\delta_{Chainage}}{dist}$
            if $distance < 5$ then
                if $ratio > 0.9$ or $distance < 0.25$ then
                    $l = l \cup p$
                    inserted ← true
                    break
                else
                    continue
                end
            else
                break
            end
        end
        if inserted then
            break
        end
    end
    if inserted = false then
        $l' = \{p\}$
        $L = L \cup l'$
    end
end
```

performing nearest neighbour search. The visualization of the point cloud, the corresponding octree as well as the sections etc. is implemented using the BlueFramework rendering library. This library is used for all data visualization inside TUM OpenInfraPlatform.

The filters and segmentation methods described are applicable via the applications GUI and the parameters are manually tunable. This allows for testing of different parameter sets without having to change the source code and recompile the application. The GUI is designed and connected to the application using the Qt framework. This framework includes a visual
GUI design application and a *Signal* and *Slot* system which makes connecting GUI elements to their functionality very comfortable. Using specific naming conventions, UI elements can also automatically be connected to functions.

To balance between programming effort, maintainability and runtime performance, most algorithms are implemented using *OpenMP*, a framework for basic thread level parallelism. Using OpenMP does not provide a speedup comparable to executing the algorithms on a GPU, but code using OpenMP works independent from the underlying hardware, operating system or compiler, while most *General Purpose Graphics Processing Unit* (GPGPU) programming languages are more hardware specific. Additionally, using OpenMP can be turned on and off using preprocessor directives.
Chapter 4

Results

The developed algorithms are evaluated on a real dataset obtained using a MLS system mounted onto a train. The track is about 6km long and includes only 2 tracks without switches or objects along the railroad corridor. This isolated environment is chosen to avoid having to deal with connections between rail tracks and branching alignments, which have to be handled in a special way. The data is recorded and stored in 1km sections which are preprocessed individually, as described in the Preprocessing section in 3, and then fused together. This is necessary since the developed application is not capable of loading an the entire point cloud which has a size of above 4GB. In order to be able to process such datasets, managing out-of-core memory accesses is necessary as well as dynamic loading and storing of data. In this preprocessed and fused dataset, a centerline covering nearly the full extent can be extracted without gaps or manual stitching or intervention. The same algorithm was tested using different smoothing parameters and identical centerline computation parameters. The centerlines are generated to consist of one point every 5cm and by using a maximum point distance threshold of 25cm, as described in 4. To compute the bearing, points in $\pm 10 m$ difference in chainage are used to compute the direction using PCA. The chainage is computed as described in chapter 3.

The results presented in fig. 4.1 are obtained by computing the curvature value for each point and applying no additional smoothing. This is similar to the baseline presented in Gikas & Stratakos (2012) without the fractal behaviour based filtering described in the paper. The curvature values are in a range of $0.05 \pm 0.1/\text{m}$ while the linear sections oscillate with an amplitude of approximately $\pm 0.025$. This amplitude is higher for clothoidal and circular arc segments. There is no visible regularity in the noise besides having a slightly stronger amplitude in the outer direction, the sign of the curvature value.

By applying a moderately strong smoothing by computing the average over 10 curvature values, which means smoothing over 5m of total rail track length, sinusoidal patterns in the
noise become visible in fig. 4.2. There is no constant and persisting frequency visible, but a mixture of oscillations which appear to be modulated onto each other. There is a component having a period length of \(\sim 200m\) starting at a chainage of 1500m to 3000m, as well as several other components with higher frequencies.
This noise can be oppressed by applying very strong smoothing and by computing the curvature only for points with larger distance. This approach is infeasible, since it greatly falsifies the results. By smoothing over longer distances, the start- and end points of the segments under inspection start to blur and are no longer detectable with sufficient accuracy. In fig. 4.3, only very small oscillations and general noise is visible, but the linear segment from 1350m to 1650m is barely noticeable upon inspecting the curvature diagram.
Chapter 5

Discussion

The bearing and curvature values obtained using the proposed and implemented algorithms contain a lot of noise which appears to follow an unknown regularity. There are several possible reasons for errors which lead to the noisy results.

Chainage The chainage computation does not provide the true stationing value along the axis, but an approximation. The accuracy depends on the extent of the alignment. The distortion appears in curved segments since the chainage is obtained by projecting a point along its main axis. This distortion could be diminished by choosing a fixed start point and then iteratively shifting the next point to this fixed point and then computing the chainage as the sum of the start point’s chainage and currently implemented method. This point then becomes the new start point.

Pair Detection Outliers in the pair detection shift the centerline by a small amount into their direction, resulting in small curvature changes. These outliers could possibly be excluded by using a median value based centerline smoothing instead of a center-of-mass based smoothing approach.

Machine Precision Since the overall pipeline involves a lot of computations performed sequentially on the same values, every computation leads to a small error. Additionally, the curvature values are already small numbers in the range of $10^{-3}$. An error that results in a 1mm difference from the original value is visible in the derivative of the bearing function.

The visible sinusoidal noise could also not be related to the performed computations, but exist in the recorded data. A possible explanation is the presence of a so called Hunting Oscillation. This oscillation is immanent to all railway vehicles and depends on the movement speed, the difference in elevation between the left and right rail track, and the diameter of
the wheel. Given these parameters, the wavelength of the motion can be computed to remove this signal.

Another consideration is the integration of more data representations like images or video footage recorded in addition to the laser scan to have more information available. This can vastly improve the preprocessing as well as the verification of segmentations and classifications. If a segment or point is classified identically using different metrics, the confidence of having correctly classified the point increases.
Chapter 6

Conclusion

Implementing the entire pipeline to automatically extract an IfcAlignment from point cloud data is possible using current state of the art techniques and individual research groups have solved sub-parts of the problem with high accuracy. Developing a complete software solution which is applicable for real world datasets on the other hand requires a lot of additional considerations.

Data Management The large amounts of data which are collected in real world scenarios require special treatment. Required data has to be loaded only on demand and attributes which are computed point wise, like the chainage, have to be stored externally to have all RAM available for computations. This makes the entire application very slow and cumbersome to use. The best solution is to develop a dedicated out-of-core data usage framework, which is an entirely different topic on its own.

Switches Real point clouds of railroads usually contain switches which lead to several problems. First of all, it makes finding corresponding rail pairs harder since it means the presence of an additional rail track inside or outside the expected range. Additionally, centerpoints of pairs can no longer be assigned strictly to one distinct centerline, but could be part of two centerlines after or before a switch.

Boundaries To use the proposed techniques on entire railway networks instead of single routes, the connections between multiple datasets and the possibly emerging routes from varying switch constellations have to be considered and addressed in a more abstract representation than via raw points or centerlines. A framework which is capable of describing connections between tracks, different routes from one point to another, etc. has to be developed. The representation of rail tracks as IfcAlignment either has to be integrated into this framework, or a new representation has to be developed.
The overall constraints and results can be summarized by saying that being able to correctly classify a single rail track of 500m length into linear, circular arc, and clothoidal segments is not sufficient to process real world datasets. Even though the theoretical foundations in the form of centerline extraction from point cloud data, bearing diagram analysis and specifications of horizontal and vertical alignments in the Ifc4x1 standard are available, merging these approaches requires more than combining the individual parts on one isolated dataset due to the different scopes and levels of abstraction of the technologies.
Bibliography


