# Assessing crossing degradation

J.J.J. Wegdam

TUDelft

ProRail

# Assessing crossing degradation

By

J.J.J. Wegdam

in partial fulfilment of the requirements for the degree of

**Professional Doctorate in Engineering**
in Civil Engineering

at the Delft University of Technology,
to be defended publicly on the 25th of May 2021 at 13:00.

| | | |
|---|---|---|
| Supervisor: | Dr. V.L. Markine | |
| Thesis committee: | Prof. dr. Z. Li, | TU Delft |
| | Prof.dr.ir. P.H.A.J.M. van Gelder, | TU Delft |
| | Dr. I.Y. Shevtsov, | ProRail |
| Client: | Ir. T.P.J.L. Kruse | ProRail |

# Contents

# Abstract

Crossings are important yet vulnerable parts of a railway network. This warrants that, in the Netherlands (a country with high traffic loads), crossing geometry is measured twice per year by dedicated measurement vehicles. This produces so much (point cloud) data that prioritizing and predicting is not possible by hand. It can be done by automating the assessments, by an automated process of four steps: cleaning the data, generating relevant performance indicators (features) per measurement, drawing conclusions from all features and visualizing/communicating the results. This project encompasses the first three steps for the most common type of crossing (the 1:9 fixed UIC54 common crossing). The results show that the features yield useful information and insights for both prioritizing and predicting.

## Acknowledgements

Ivan Shevtsov, ProRail

Theo Kruse, ProRail

Gerbert Brouwer, ProRail

Sander Buitendijk, ProRail

Arthur Kappers, ProRail

Gertjan van Rhee, ProRail

Bart Leistra, ProRail

Hanna Tijbosch, ProRail

Ingmar Bolderink, ProRail

Valeri Markine, TU Delft

Zili Li, TU Delft

Rolf Dollevoet, TU Delft

Pieter van Gelder, TU Delft

Ruud Kuyper, Vossloh Cogifer Kloos BV

Daniël Poll Jonker, Voestalpine Turnout Technology Netherlands BV

Andrew van Eck & Cor Scholtus, BAM Rail

Mart Folkerts & Ruud van Bezooijen, Asset Rail

Dick Terweij & Ronald Dros, Strukton

# 1 Introduction

Turnouts are important for railway networks. A single railway track can operate without turnouts. A railway network cannot do without however, to join multiple routes together and allow vehicles to transfer from one track to another. When turnouts fail, the railway network becomes less connected (less useful). Thus, providing stable and safe operations requires prevention of failures in turnouts. Doing so can be a challenge; turnouts feature varying rail cross sections (geometry), which makes its degradation mechanisms (both causes and consequences) complex.

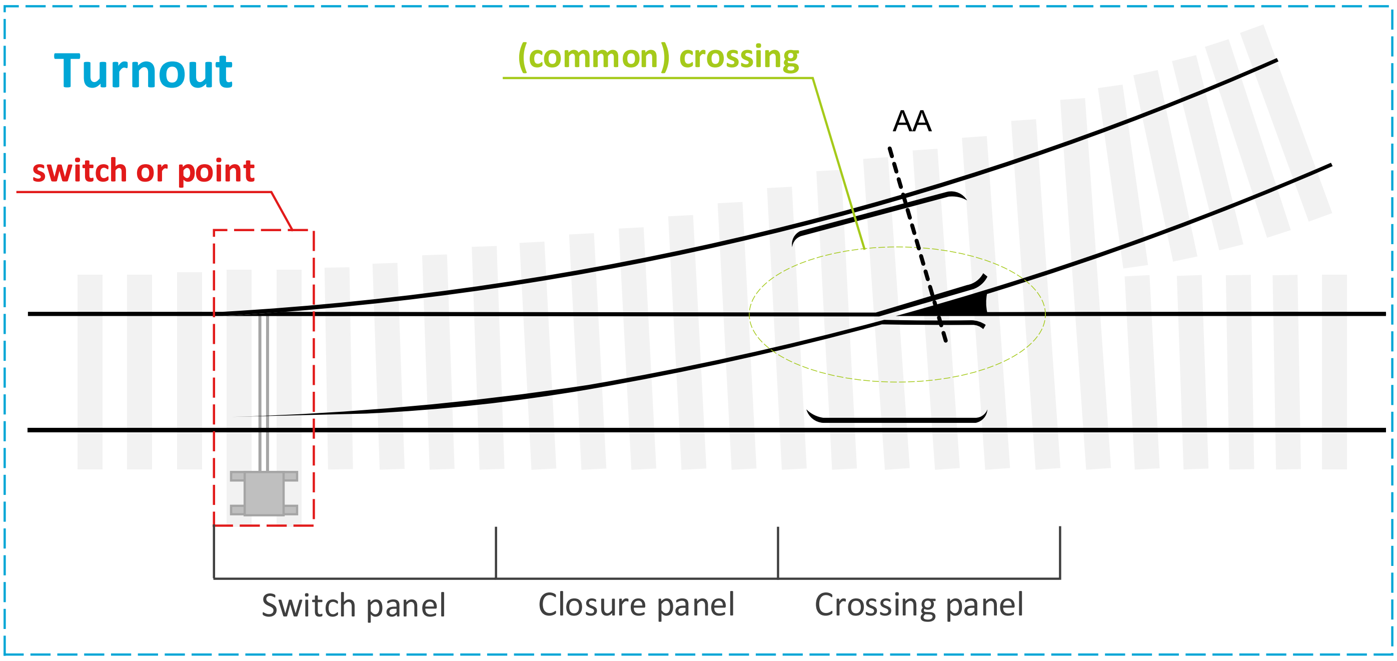


Figure 1 - Components of a turnout

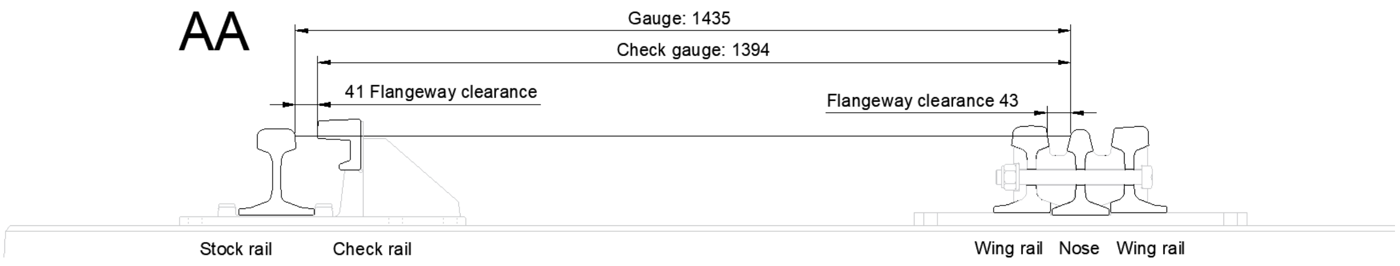


Figure 2 - Cross section AA, from Figure 1: some geometric relations in the crossing panel (regular Dutch situation)

The rails of a turnout can be divided in three areas (Figure 1): the switch panel, the closure panel and the crossing panel. The switch panel and crossing panel can't function without the use of complex geometry (like in Figure 2), because they are required to guide wheels safely in multiple directions. This complex geometry invokes dynamic amplification of vertical wheel-rail contact forces from passing wheels. The complexity of the geometry (and thus its consequences) is the worst at the crossing panel. In this area, the wheel-rail contact has to transfer from wing rail to nose or vice versa (Figure 3).

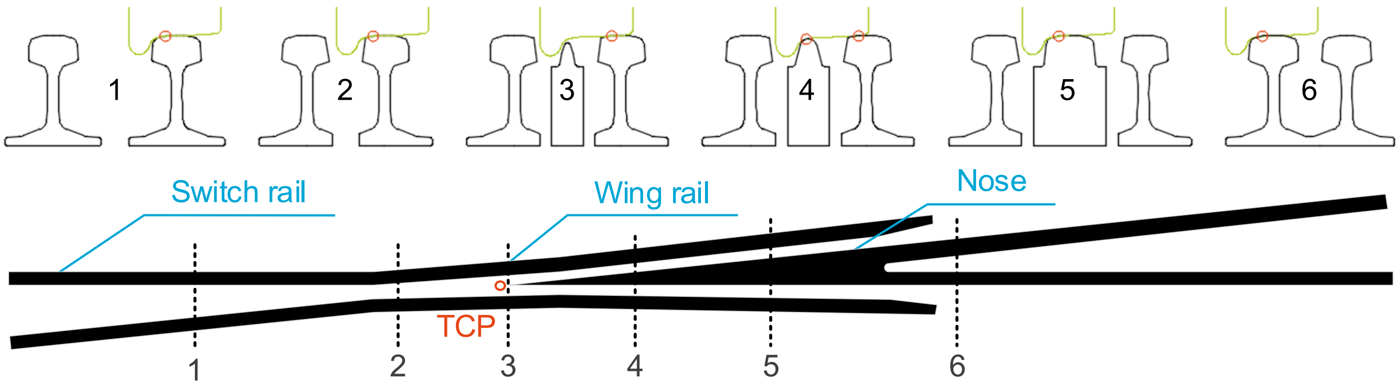


Figure 3 – Regular wheel (green) with wheel-rail contact locations (red) throughout a crossing

## 1.1 Motivation

### Problem definition: crossing degradation

Crossings must withstand severe wheel-rail contact conditions and preferably for a long time. Ideally, the crossing would wear evenly and predictably without any fatigue problems. In such a scenario, the crossings would eventually wear up to a point where either the geometry can't guarantee safe wheel guidance anymore or parts of the crossing have become so thin that there's a risk of breaking them.

In practice, crossings don't degrade evenly and predictable. Crossings degrade unevenly and predicting the degradation is hard. Moreover, wear is not the only degradation mechanism. Crossings suffer from three types of degradation: wear, deformation and rolling contact fatigue (RCF). Wear seems to worsen the consequences of wheel impacts. Deformation is problematic for safe wheel guidance (i.e. narrowing of the flangeway by lipping). RCF may be the worst of the three mechanisms; it seems to be the hardest mechanism to predict and the consequences can be both sudden and drastic (Figure 4).

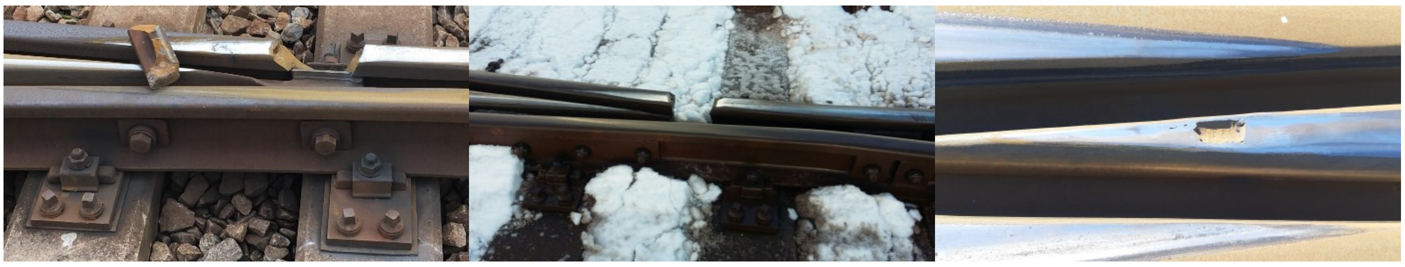


Figure 4 - Examples of damaged crossings [1]

The unpredictable behaviour is problematic for traffic operations, logistics, tendering and innovation. Sudden crossing failures disturb traffic (which is bad enough already) and cause the urgent need of replacement. Urgency is always a problem in terms of logistics; it is much more expensive to replace something in the upcoming night, than to replace it anywhere in the coming six months. This creates a lot of uncertainty for contractors. This uncertainty likely translates into extra costs for the infrastructure provider (in this case: the Dutch infra manager ProRail) when tendering maintenance of an area with quite some crossings in questionable conditions. Lastly, in terms of innovation it is hard to innovate with little knowledge of the consequences. For example: will explosion hardening of cast crossings, or introduction of bainitic crossings be worth the extra costs? In other words: making crossing degradation more predictable has the potential to yield lots of benefits.

## State-of-the-art: research

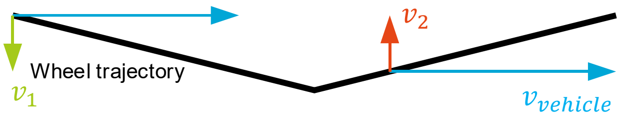


Figure 5 - Schematic overview of wheel dip

Figure 2 shows how the crossing features gaps in the rails: the so-called flangeways. The flangeways are required to allow the wheel flange to pass through the crossing in both directions, but is the main cause of problems around crossings. The gaps lead to a dip of the wheel; the sub-optimal support from the rails causes wheels to lower themselves and then raise back up again (dip). The transition between lowering and raising causes an impact (dynamically amplified vertical wheel-rail contact forces). A good way to start quantifying such an impact is by studying the wheel centre trajectory [2]. In Figure 5 the speeds of the wheel before and after the dip are shown; both with a lateral and vertical component. The impulse $J$ describes the amount of momentum change between $t_1$ and $t_2$, that is needed to change the direction of the wheel. It can be calculated by using the relevant mass $m$ (will be discussed later), the initial vertical speed $v_1$ and final vertical speed $v_2$ :

$$J = \int_{t_1}^{t_2} F\, dt = mv_2 - mv_1$$

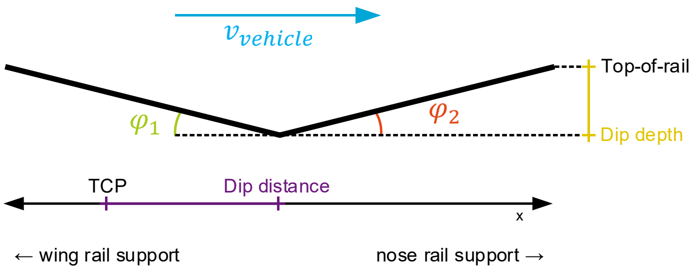


Figure 6 - Extended overview of wheel dip parameters [3]

The drawback of this formula is that $v_1$ and $v_2$ are hard to quantify. Instead, the dip angles $\varphi_1$ and $\varphi_2$ can be used, like in Figure 6:

$$J = m \cdot v_{vehicle} \cdot \tan(\varphi_1 + \varphi_2)$$

From this, it can be concluded that the severity of wheel impacts on crossings is caused by the combination of operating speeds and wheel dip angle. Measuring this dip angle (and being aware of the operating speeds) could be a good way to keep track of the root causes of crossing degradation. [3] proposes a measurement device to keep track of the wheel dip, by rolling a physical wheel over the crossing and keeping track of its vertical position.

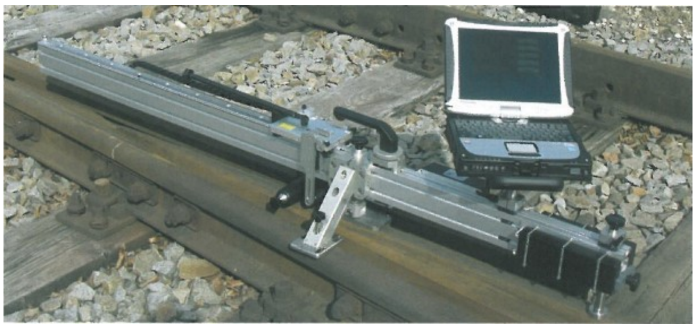


Figure 7 - Wheel dip measurement device [3]

Impulse is, however, something different than force; the same impulse distributed along a longer time yields less force and thus stress in the rail material. In practice, this time is determined by stiffness: vehicle suspensions, wheel material, rail material and the structure that supports the rail. In [4] a model of these effects is combined with the wheel dip parameter, in order to estimate the dynamically amplified vertical wheel-rail contact force (which is then used to determine the maximum allowable wear per operating speed). Interestingly enough, they use the difference between the designed dip depth and the worn dip depth as degradation parameter.

A second complication comes with differences in wheel profile wear and differences in the lateral wheel trajectory. To get rid of these complications, [5] suggests to estimate the forces in wheel and rail by measuring accelerations on wheel (from the axle box) and rail. Both approaches have their advantages and drawbacks. Track-mounted accelerometers can provide the impact-related track accelerations as well as the longitudinal impact locations (with some spread because of different vehicles / wheel trajectories) [6], but equipping and maintaining such sensors on all crossings would be too costly. Axle box accelerations have the potential of assessing lots of crossings with only one sensor setup [7], but measuring the exact longitudinal impact location is much harder and the measurement only accounts for one vehicle type / wheel trajectory.

While the dynamically amplified vertical wheel-rail contact force gives a good impression of impact related RCF damages, it doesn't give insights in RCF damage related to slip. The complex behaviour of the wheel-rail contact around flangeways can be studied with Multi Body Simulations (MBS), in packages like NUCARS [8], GENSYS [9], SIMPACK [10] and VI-Rail [11, 12, 13]. Various papers validate the results of these models, with track-mounted accelerometers [13, 6, 14], wheel-mounted strain gauges [15], axle box-mounted accelerometers [16] and track displacement measurements [6]. The MBS provide insights through output parameters like wheel trajectories (both vertical and lateral), vertical and lateral contact forces and wear number. Example of such wheel trajectories are shown in Figure 8. Moreover, some packages provide detailed contact-patch animations (like the one in Figure 43 on the right) and visualizations of stick and slip areas in the contact patch itself [17].

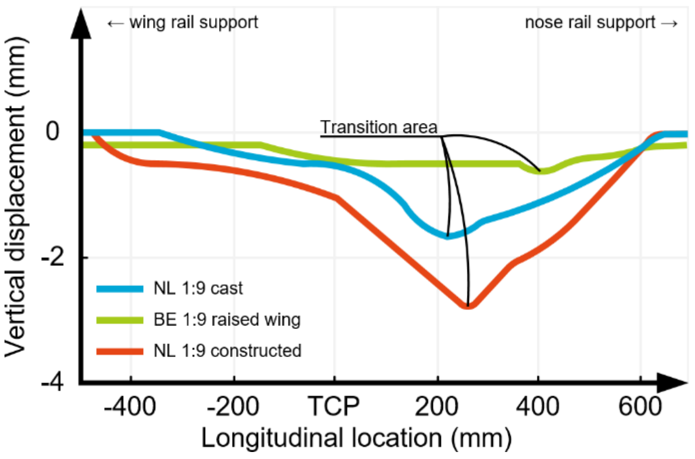


Figure 8 – Simulations of wheel dip trajectories [12]

The results of such MBS are used to better understand the interaction between crossings and passing vehicles and to predict consequences of changes to that system. Moreover, the outputs of MBS can be used to do a more precise analysis of wear and deformation by using Finite Element Models (FEM) [18]. The combination of MBS and FEM has been used to find a more optimal geometry and stiffness [19]. More recently, the combined model was extended to make predictions of wear and deformation [20].

## Current practices: norms and policy

ProRail has several standards for the maintenance of turnouts. Depending on the contract, either [21, 22, 23] or [24] prescribes the norms for alignment and geometry. Moreover, [25] prescribes the RCF inspection regime. An analysis of these norms for common crossings can be found in Appendix A. From this analysis it can be concluded that the geometry and alignment norms mainly ensure safety (with parameters like the ones in Figure 9) and focus less on durability related aspects that were discussed in the state-of-the-art research. For example, the norms for vertical wear in the transition zone of crossings (Figure 60) don't have a measurement procedure description and are not enforced, while [4] clearly relates this kind of wear to increased degradation. This focus on safety is consistent with the European counterpart of these norms: the Technical Specifications for Interoperability (TSI) Infra [26].

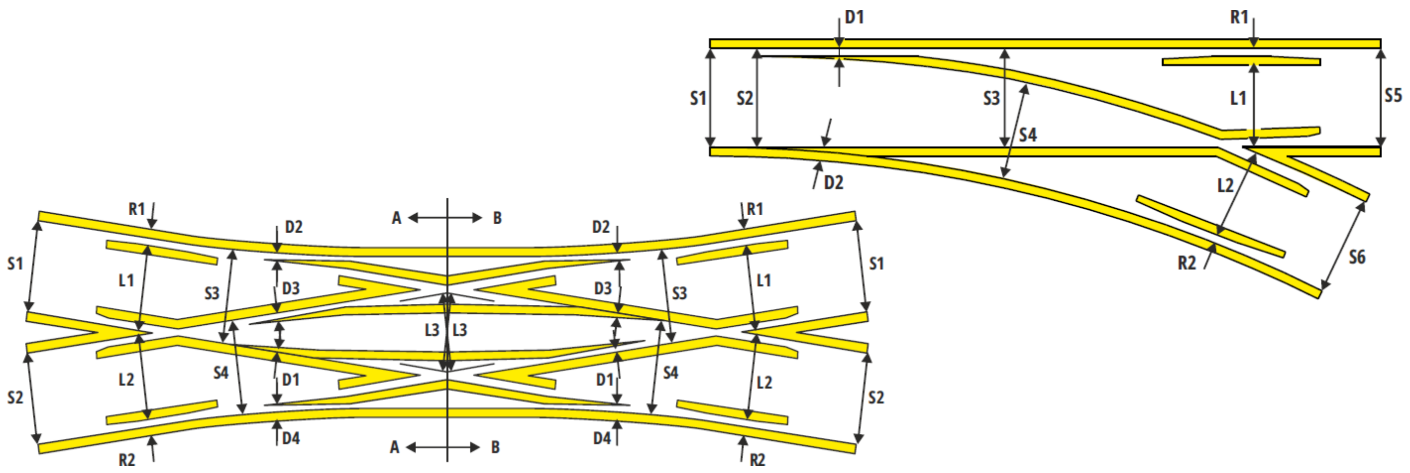


Figure 9 – Example of parameters measured by the Mermec on-board Turnout & Crossing Measurement System [27]

### Opportunity of ProRails measurements

Developments in geometry measurements in turnouts (an extensive overview can be found in [28]) have allowed to measure geometry without walking in the track, which is safer and cheaper. This is done by equipping railway vehicles with laser rangefinders (LIDAR), like in Figure 10. In February 2005 ProRail published standards [29] for the measurements of turnouts on dedicated inspection wagons. After a process of refinement and tendering ProRail contracted two parties to measure all 4643 [30] remotely operated turnouts in the Netherlands, from the 3rd of August 2016 up till today. Measurements were carried out approximately once per six months, which led to a dataset of over 35.000 measured turnouts at the start of this project. The measurements already feature detection of the safety related parameters, which is consistent with similar projects like the on-board Turnout & Crossing Measurement System from [27] and the trolley developed in [28].

Contractors can already choose to look at the measurements (online through BBMS [31]) to make additional (manual) assessments on durability related parameters. However, the amount of measurements is about 14.000 per year, while only ~20 experts assess them. Manually assessing 700 measurements per year per expert is inefficient and subjected to human error/bias.

Better defining the geometry parameters that determine crossing durability and automatically extracting them from the measurements could provide valuable insights in geometry degradation and the causes of RCF. This could enable better maintenance strategies, with the earlier mentioned possible benefits for traffic operations, logistics, tendering and innovation.

## 1.2 Project goals

The end goal is to prevent the described problem, crossing degradation, as much as possible. The turnout geometry data from ProRail and earlier research on crossing degradation are the means to get to that goal. The idea is that better insights in degradation could promote preventive maintenance and design innovations. For this reason, the project goals are defined as:

1. Studying existing crossing degradation related parameters
2. Coming up with and studying new additional parameters
3. Analysing the applicability of the parameters for degradation prevention and prediction

## 1.3 Scope

The scope of this project is to have the automated degradation assessments working for the most common type of crossing. The most common type of crossing is the straight 1:9 UIC54 fixed common crossing. With the help of [30] the impact of this scope was determined. The Netherlands has 6431 turnouts, of which 4643 are remotely operated (and thus measured). Within these remotely operated turnouts, 2947 turnouts have a 1:9 tangent. 2692 of these have a UIC54 rail profile.

During the project, it appeared that (single and double) slip turnouts and scissors crossovers cause trouble for the measurement vehicles. These situation leads both wheels through a crossing at the same time, which causes alignment problems that are hard to fix. This lead to a further narrowing of the scope, to turnouts that are not slip turnouts (2231) and not part of a scissors crossover (1840).

In other words: the scope of this project aims to cover 40% of all measured turnouts. This should already lead to a substantial improvement for the experts.

## 1.4 Project outline

The previous sections have introduced the topic and have shown the purpose of this project: automating crossing geometry assessments. The road to get there was split in four consecutive steps and will be discussed in the next four chapters. Chapter 2, will discuss the measurement issues and how they are repaired (step 1 of the pipeline). This will be referred to as the Repair step. The second step is discussed in Chapter 3 and is named Feature Generation. This is a term from the field of Machine Learning, where relevant properties (features) are derived from an observation (measurement) to capture the state of the asset in a concise way. After the generation of features is discussed, Chapter 4 discusses what to do with the features (step 3 and first bits of step 4). They are examined for the whole population of measurements, to find and prove hypotheses. Chapter 5 then closes off, with conclusions, recommendations and future work.

Figure 10 - A measurement train using LIDAR to measure rail geometry [32]

# 2 Automated measurement aligning

Structure and math of the feature generation script are discussed in Appendix B1.3

A measurement train is a railway vehicle. That means that the LIDAR scanners on the vehicle have six degrees of freedom with respect to the measured object (Figure 11). After measuring a cross section, the data points from the moving scanner should be transferred the fixed axes that infrastructure designers use. This leads to the question: where was my scanner with respect to the infrastructure, whilst measuring this data point? From studying the measurements, it became clear that not all six degrees of freedom cause substantial trouble. Only three of them do so: longitudinal, lateral, vertical and roll. Getting these right, is the first step in the project: measurement aligning.

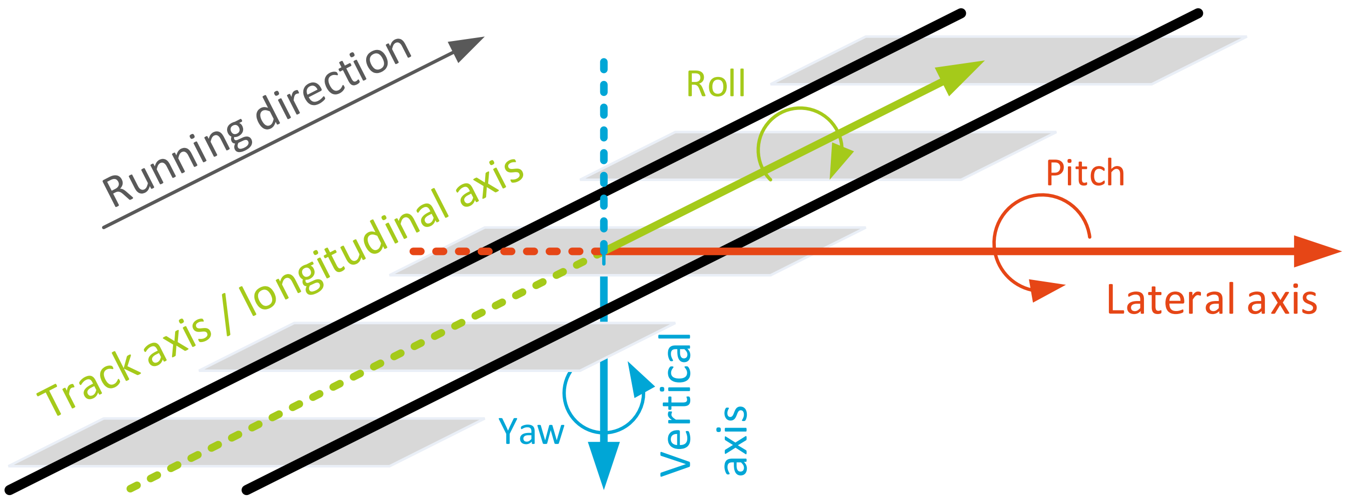


Figure 11 – Coordinate system for railway infrastructure, along with rotation naming conventions for vehicles

## 2.1 Longitudinal alignment

Additionally, the longitudinal location of the TCP needs to be determined very precisely. Determining it on the wrong location has consequences for the assessment. For this reason, the estimates from the measurement companies were not used. Instead, an algorithm was made that determines the location based on the green and blue areas in Figure 12. In these areas, the lateral coordinate at 14 mm below top of rail is determined for both wing rails. At these coordinates, a simple linear interpolation is used (per wing rail) to determine the exact orientations of the red areas in Figure 12 (the unguided opening). The longitudinal location of the intersection is used as the location of the crossing nose.

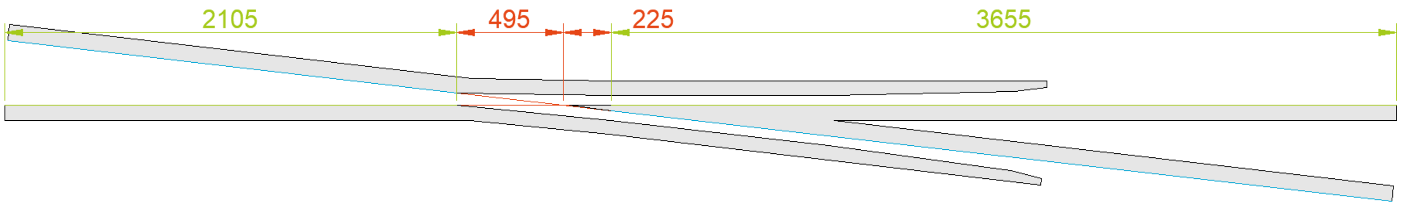


Figure 12 - Areas that are important (blue and green), when determining the nose tip location

## 2.2 Lateral alignment

Finding the lateral position of the LIDAR scanners with respect to the railway track is usually done by estimating the location of the rail profiles. Based on looking through the unedited measurements, it appears that the standard positioning algorithm tries to fit two UIC54 (or UIC60 or NP46) rail profiles through a cross section. In the top half of Figure 14, the track axis (green) is determined by creating a point half way both rail top locations.

This approach works sufficient for standard track, but fails dramatically in turnouts. Finding a best fit for a standard rail in the midst of a crossing doesn't result in the true location (like Figure 14 bottom right). In practise it produces results like Figure 13, leading to wrong alignment.

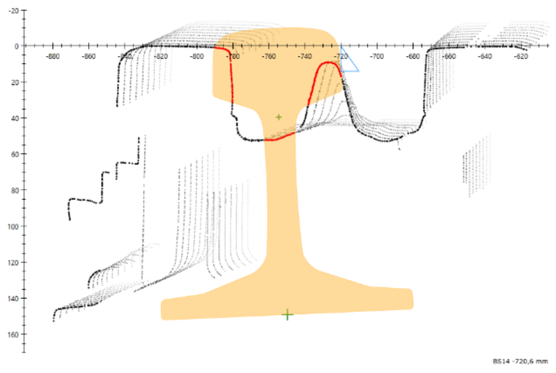


Figure 13 – Best UIC54 fit in a crossing [31]

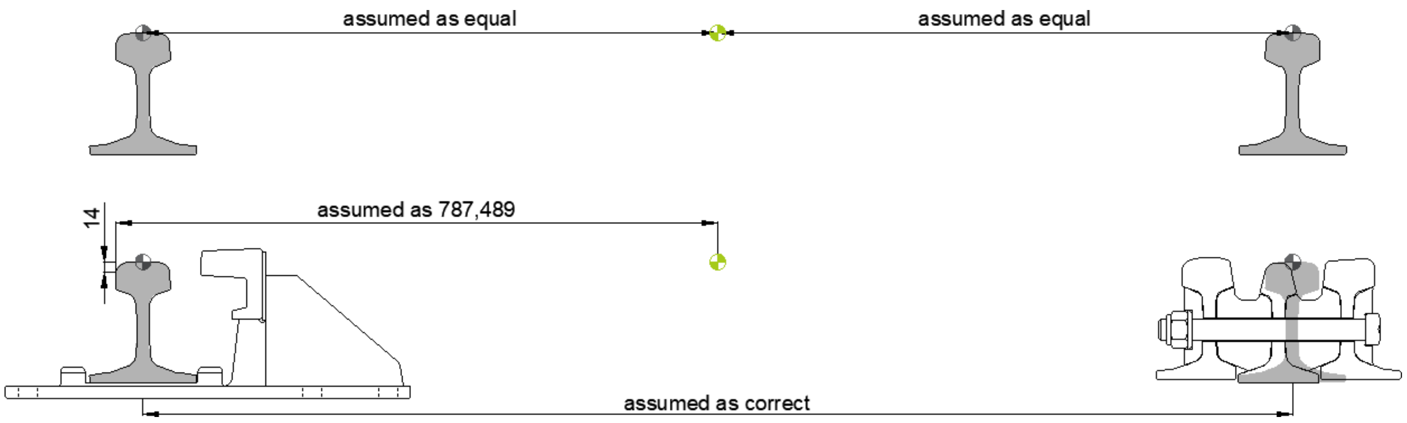


Figure 14 - Horizontal reference system: standard algorithm (top half) and new algorithm (bottom half)

Turnouts call for a different approach. Due to the varying geometry, wear and plastic deformation, it was key to find an area that is always the same (with respect to the track axis). Moreover, that area had to be covered by LIDAR as much as possible; note that some parts (like check rails) cast shadows. This area was found at the field side of the stock rail (the back of the rail opposing the crossing). This area is always visible on LIDAR, doesn't vary in terms of geometry, can't be subjected to wear and can't be subjected to plastic deformation. For this reason, the horizontal alignment algorithm was designed like in the bottom half of Figure 14. The distance from the track centre line to the field side (at rail top -14 mm) was assumed to be standard. Moreover, the distance between both rails was also assumed as correct (while both rails are measured by different scanners).

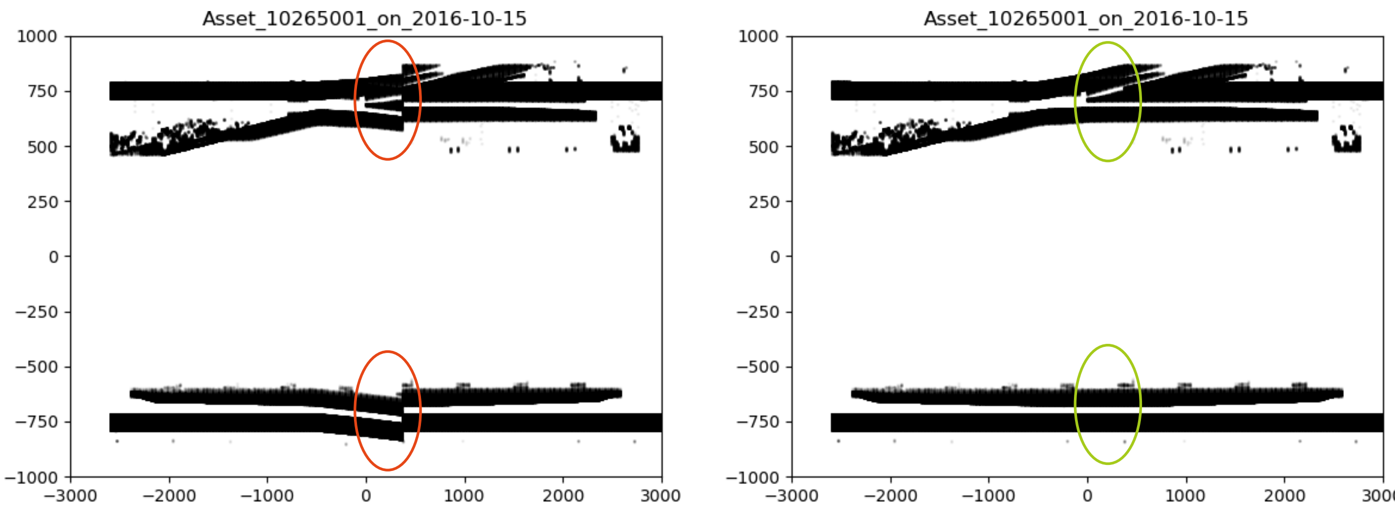


Figure 15 – Comparison of a measurement from the original dataset (red) and its laterally aligned version (green)

In Figure 15 it can be seen how the new algorithm repairs the most common horizontal alignment error; the open-door like shape (indicated by the red ovals) between the throat and the nose tip of the crossing. The open door shape is caused by the alignment algorithm of the measurement companies, that doesn't take the unguided opening (Figure 12, in red) of the crossing in to account.

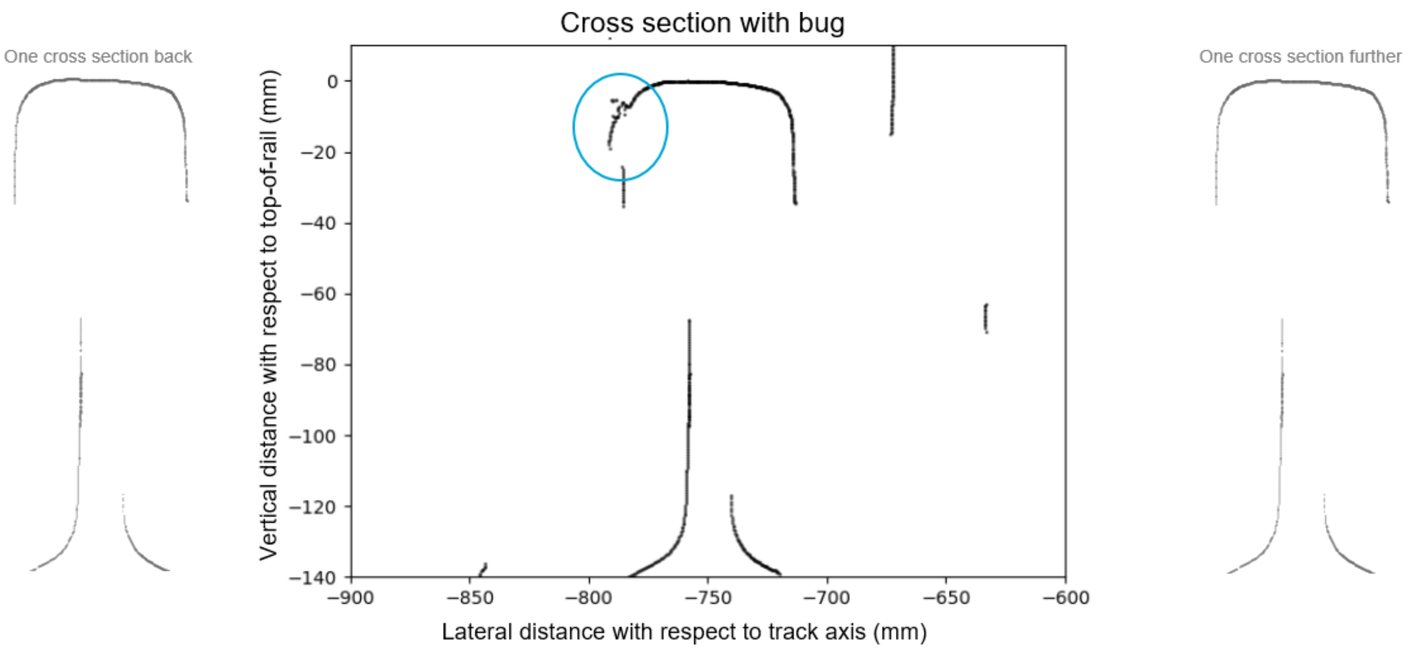


Figure 16 - Example of a lateral alignment problem: unexplainable noise on the stock rail field side

Aside from fixing the most common error (the one in Figure 16), the alignment algorithm has been equipped to also handle more rare errors. The algorithm has been tested on over one thousand random measurements; do the alignment, create a plot and manually check whether the result makes sense. This uncovered some more errors like the one in Figure 16. Finding and solving these bugs was necessary to achieve a success percentage of 99,8% (based on 932 measurements), because this alignment step is the first and most simple step.

## 2.3 Vertical alignment

During the development of the measurement repair procedure, the vertical coordinates of the two rails seemed to differ. The top of a worn rail had a vertical coordinate of zero, while the original rail top would be a more suitable reference point in later steps. The origin of this problem lies in the definition of rail top. For practical purposes, rail top is generally defined as the highest point of the rail (even when the rail is worn 10 mm). For simulation purposes, it is more convenient to refer to the original rail top. The difference is shown in Figure 17. Other examples are possible; think of the consequences for a crossing with elevated wing rail.

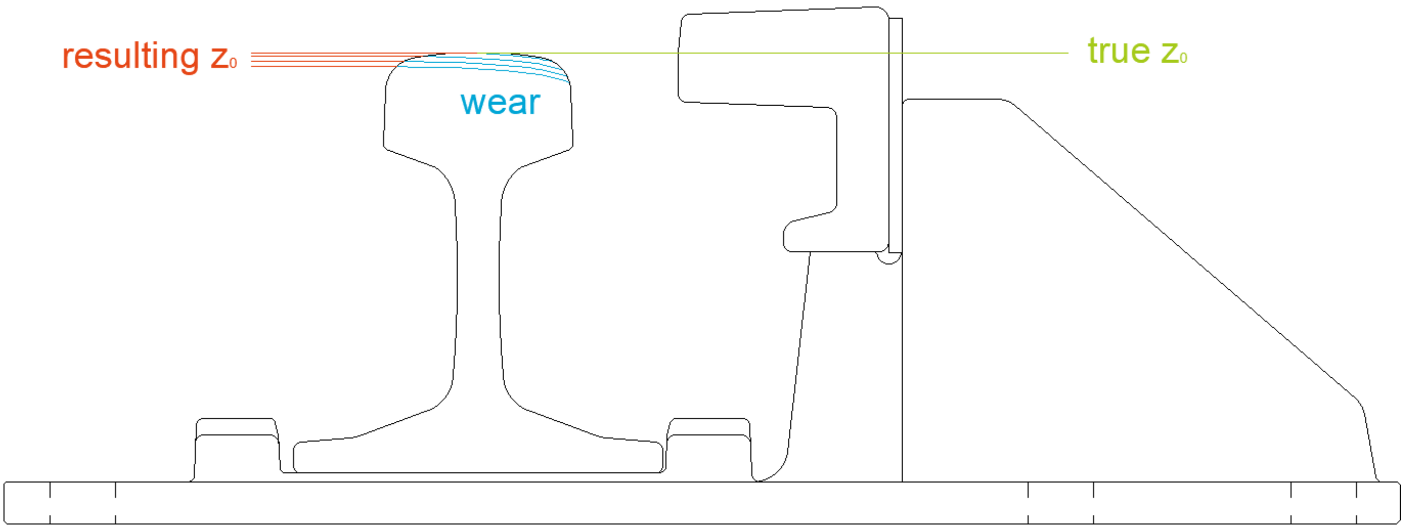


Figure 17 – Vertical alignment of the standard algorithm (red) versus the ideal situation (green)

During the development of the repair procedures, algorithms were created to reconstruct the original rail top based on the rail head. Measurement companies already do this, based on the location of the rail foot, but this is insufficient in a crossing where the rail foot is not visible or not even present. Figure 18 shows the theory behind the algorithm: take the highest point on the rail head, go down 14 mm and measure the distance $dy$ to the outer side of the rail head (green). Based on this known distance, the rail top shift is reconstructed. This is done by a lookup table, that contains 36 combinations of green and blue values. In between those values, linear interpolation is applied. Using a lookup table of 72 values would increase the precision with about 0.01 mm (as measured in AutoCAD), for wear values up to 6.4 mm. Higher wear values are rarely seen. Note that 14 mm was chosen arbitraty. Other distances (and rail profiles) can also be used, but need a new lookup table.

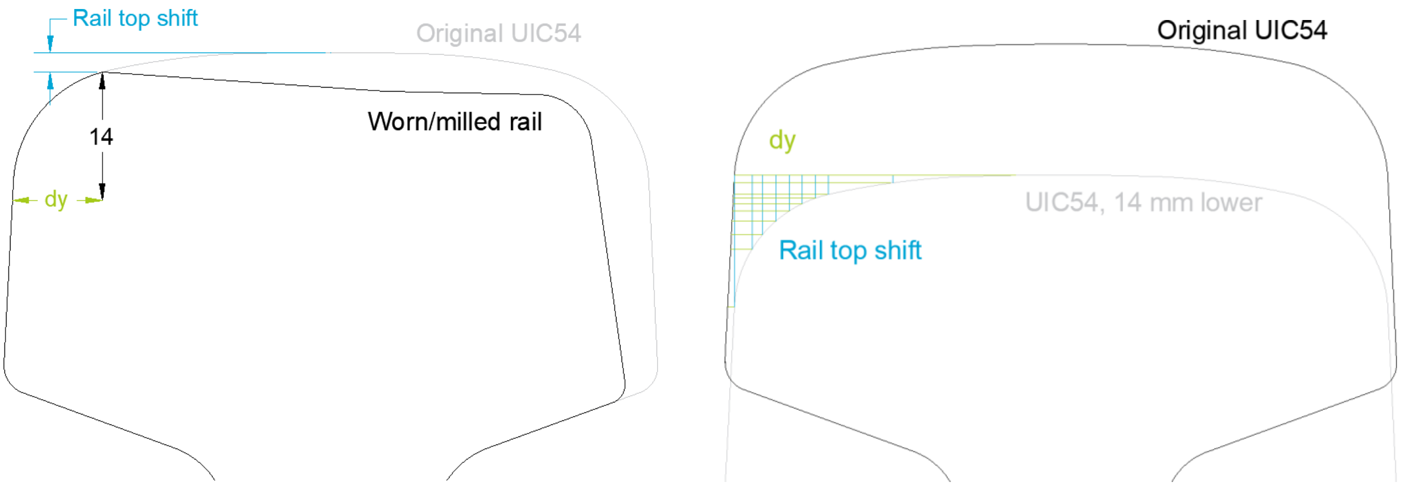


Figure 18 – The definition of dy and rail top shift (left) and the relation between the two (right)

The derivation of the height of a worn UIC54 profile is enough for one of the possible wing rail designs. The Netherlands has two other wing rail designs: flat wing rail and the raised wing rail. The flat wing rail has a horizontal top (at the height of rail top) and can be found in cast crossings. Extending the height calculation algorithm for these crossings was easy: simply take the highest point and call it rail top. The height calculation for raised wing rails was much harder though. The highest point on a raised wing rail is higher than rail top and thus should be treated as such. To find out how high the highest point on the raised wing rails is throughout the whole crossing, 86 cross sections (from [12], based on [33]) were analysed every (longitudinal) 22,5 mm. In AutoCAD, the highest point on each cross section was found and noted down in a csv (along with the longitudinal coordinate). The result was a lookup table, which shows the algorithm the height of the highest point it finds on a certain longitudinal location (as determined in section 2.1). Keep in mind that this lookup table shows results from a crossing that is manufactured and measured perfectly, which is often not the case in practice. This introduces some noise in the vertical positioning of cross sections on raised wing crossings. This has to be taken in to account when interpreting vertical parameters that span multiple cross (sections 3.5 and 3.6) on those crossings.

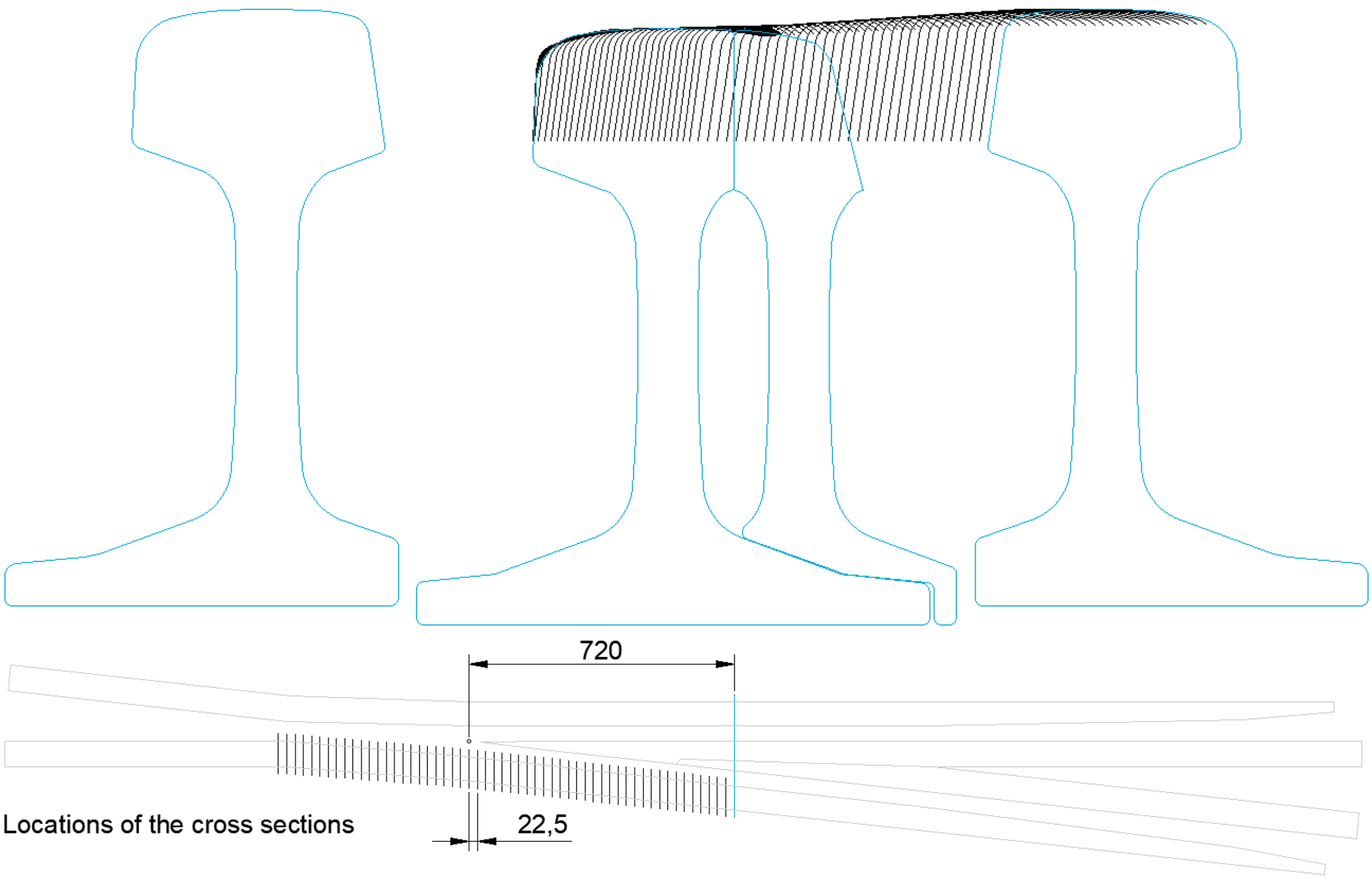


Figure 19 - Cross sections of a raised wing rail, from which the highest points were derived

With a good estimation of the original rail top on both the stock rail and the crossing, it was possible to align both sides vertically with a good precision. It worked well, except for the assumption that the vertical axes for both rails are independent. Later on in the project, this assumption proved to be wrong. The algorithm from the measurement companies didn't correct left and right vertically to make the rail top $z_0$, but apparently rotated the entire cross section to achieve that. This leads to the next section, about roll.

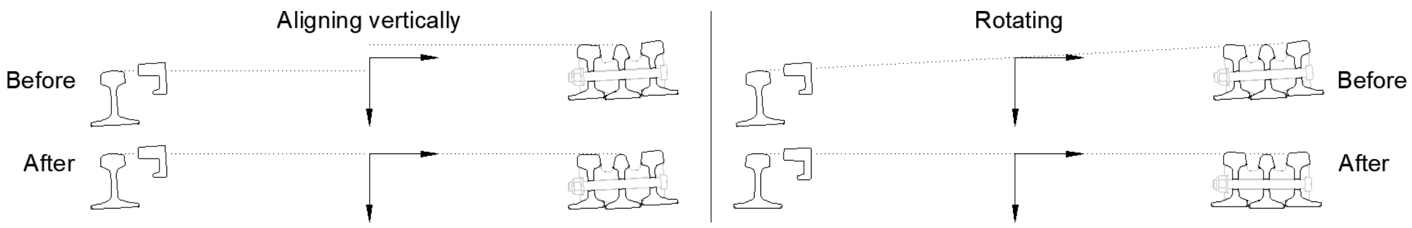


Figure 20 - The difference between aligning and rotating the cross section: the theory

## 2.4 Rotation (roll)

Like described in Figure 11, roll is a rotation in the yz-plane. Because it's the only relevant rotation in this project, roll will further be referred to as 'rotation'. Emulating rotation with two vertical translations proved to cause too much inaccuracy (the difference is shown in Figure 20). Noise from the (unknown) algorithm of the measurement companies causes distortions like the one in Figure 21; all cross sections of the left wing rail are aligned with $z_0$, while cross sections on the right are not (while symmetry is to be expected, if no rotation noise were present).

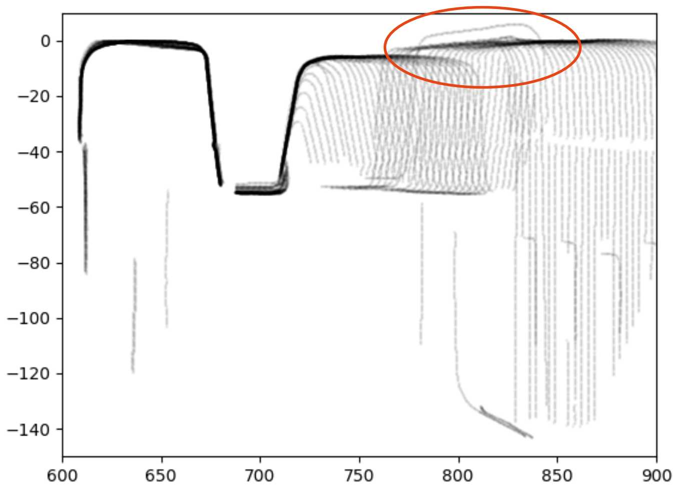


Figure 21 – Practical consequences of rotation noise, when simply aligning vertically on the left wing rail

For this reason, the choice was made to deactivate the earlier developed (section 2.3) vertical alignment algorithms. The new (and final) approach is to 1) repair lateral alignment and 2) repair the rotation.

To rotate, $z_0$ had to be found on the crossing and on the stock rail. To find $z_0$ on the crossing, the earlier developed script from section 2.3 was used. On the stock rail side, the idea was to combine the (highest point on the rail and the vertical wear parameter from the measurement companies (based on the rail foot). The wear parameter is an estimate of the wear (delivered with the point cloud), derived from the difference between the best-fit rail profile and the measurements (Figure 13). This proved to be too inaccurate.

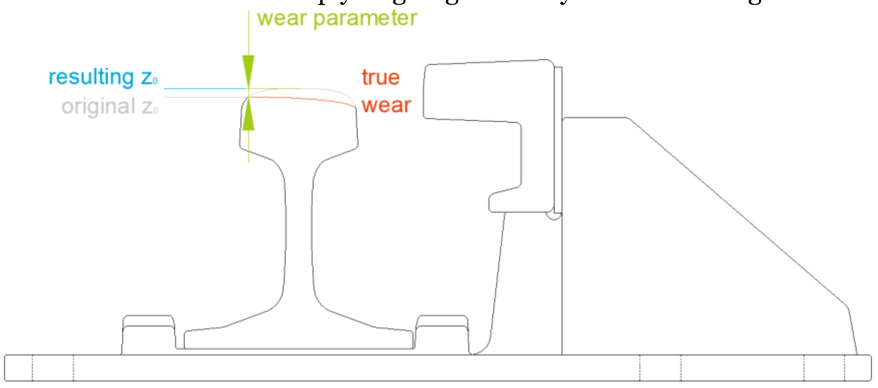


Figure 22 - Combining the wear parameter and (worn) rail top

The second attempt to make a rotation algorithm, was based on check rail rotation. The top of the check rail is a location that doesn't wear. Its supports are well equipped to prevent deformation (have a look at Figure 22 on the right). This makes the check rail a suitable reference point to determine the rotation of the cross section. The height of check rails may vary, but its top face (Figure 23 in red) is always in range of the LIDAR scan and in the same orientation/slope. The idea was to use the slope of this face to determine the rotation of the cross section. The algorithm worked in three steps: 1) cluster the front, top and back faces of the check rail by using local slopes (explanation in Appendix B1.7 and results in Figure 23) and 2) determine the orientation of the front, top and back faces by using least-squares and 3) determine the rotation of the check rail with respect to its normal appearance (and thus the cross section) by using the top face (or front/back if the top is invisible). The method worked, but didn't provide results that were exact enough to remove rotations. Despite that two methods proved to be not exact enough, the problem of rotation really had to be solved. So, a third approach was developed.

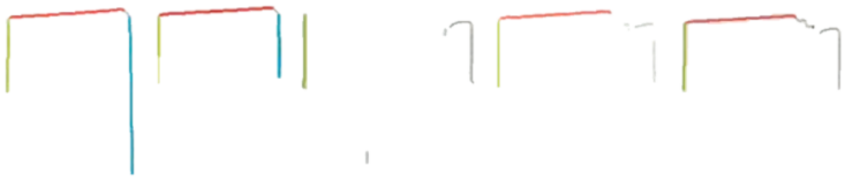


Figure 23 – Results for automated detection of check rail front (green), top (red) and back (blue)

The third and final approach was to take the highest points on both wing rails as reference points (Figure 24); take the highest point on both wing rails and rotate the cross section (with a rotation matrix) such, that both highest points end up on $z = 0$. This made the rotation fully independent from the original alignment algorithm and gave by far the best results for most of the measurements. The only downside is that this method doesn't work for crossings with thin wing rails (specific type of cast 1:9 crossing) with severe height loss that spans the full width of the wing rail, as explained in Figure 25. Luckily, this is quite a specific situation and the benefits are greater than this drawback. Summarizing: the final method is to first align the measurement laterally and then rotate the measurement based on the highest point on both wing rails.

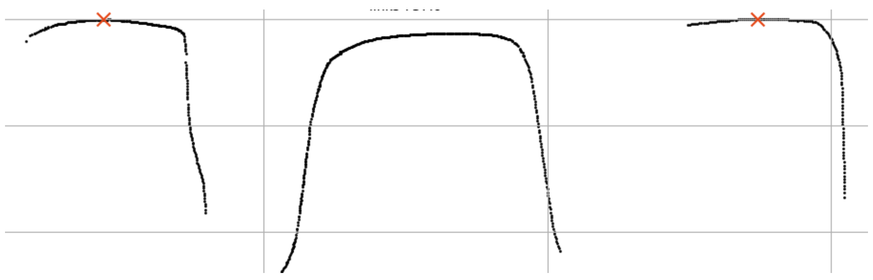


Figure 24 - Rotating based on the highest point on the wing rail

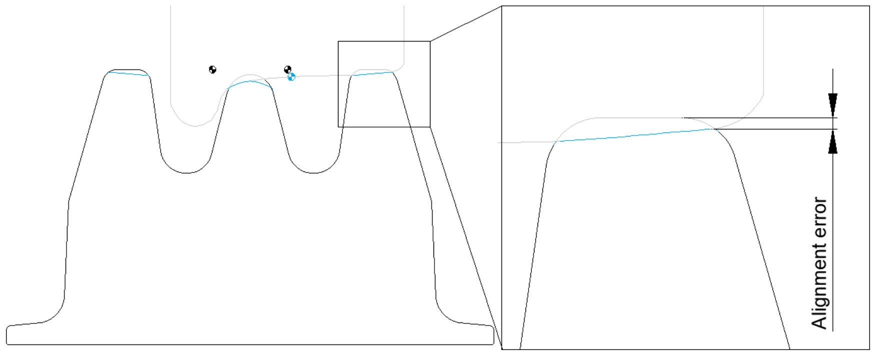


Figure 25 - For severe height loss and narrow wing rails, the vertical position of the original rail top cannot be determined

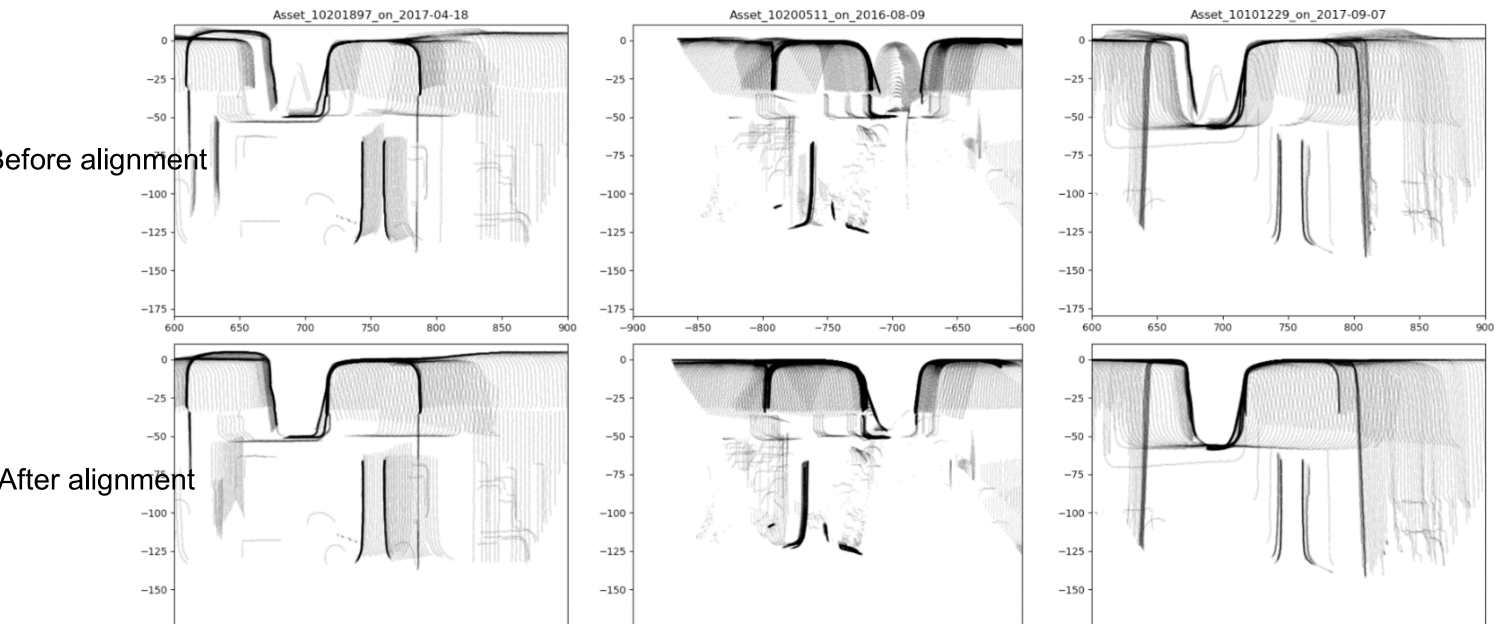


Figure 26 - Alignment algorithm results for high wing rail (left), regular constructed (mid) and cast (right) 1:9 crossings

## 2.5 Discussion

In data analysis projects, a lot of time is often spent on the first step: preprocessing / data preparation. In this project, **it took 75% of the time** to get to know the data and to create an algorithm that solves all of its issues automatically. A big investment, but definitely necessary for everything that comes after this chapter. The data cannot be assessed (is not usable) if all kinds of repairable errors are still in there.

The results seem **worth the time investment**. Figure 26 compares the three types of 1:9 crossing before and after the automatic alignment. Lateral issues are solved and rotation is removed. For high wing rail crossings the height profile is accounted for, like illustrated in Figure 27.

**Now that the measurements are well-aligned, the measurements are usable for feature generation**; the derivation of interesting indicators of the status of the geometry degradation. These indicators can be used to predict geometry degradation, but other mechanisms as well. Think of head checks for example. The next chapter will show how to get these

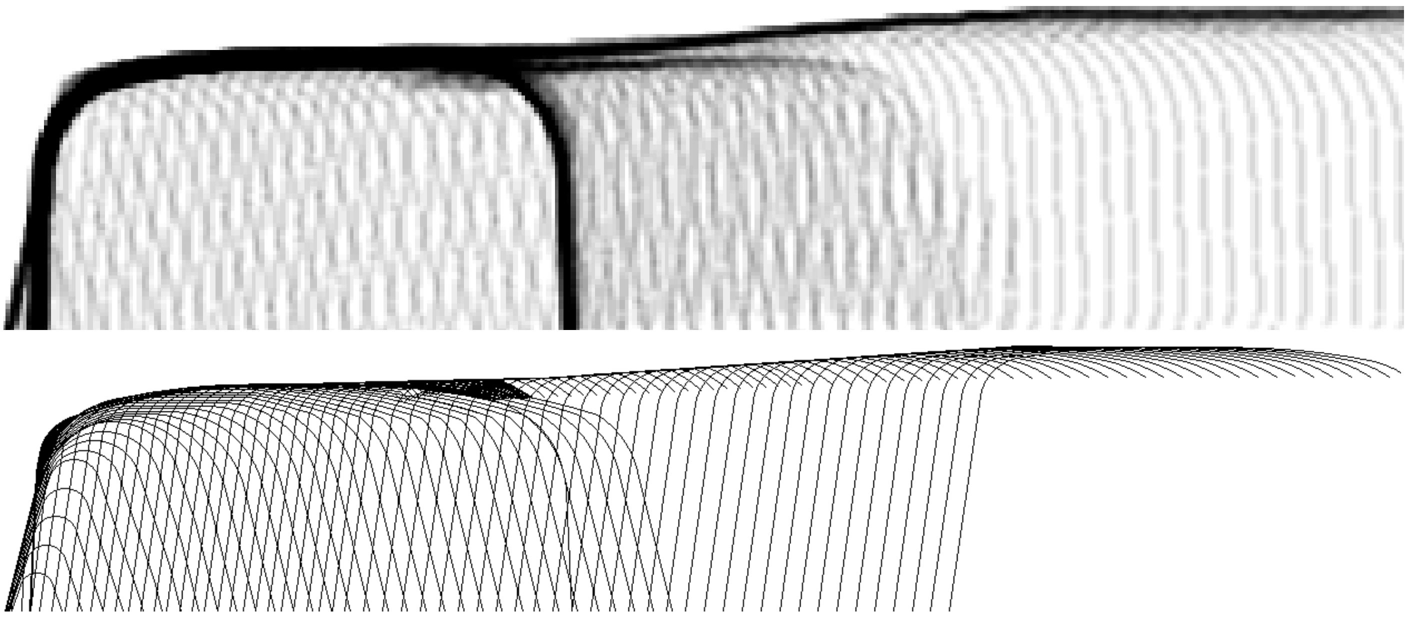


Figure 27 - Comparison of aligned measurement and design drawings, for a raised wing rail crossing

# 3 Feature generation

Structure and math of the feature generation script are discussed in Appendix B1.4

Degradation of a crossing can be described as a combination of various mechanisms. For example: vertical wing wear has causes/consequences different from horizontal nose deformation. The first step in feature generation is to make a clear distinction between these interconnected yet different mechanisms. This is done with a two-step clustering algorithm. It first divides the crossing in to inner[1] wing rail, nose and outer wing rail (Figure 28 left). This cuts the cross section in to three cross sections. The wings and nose are then cut in to front, top and back (Figure 28 right). This pre-selection allows easier (faster) calculation of the various degradation features.

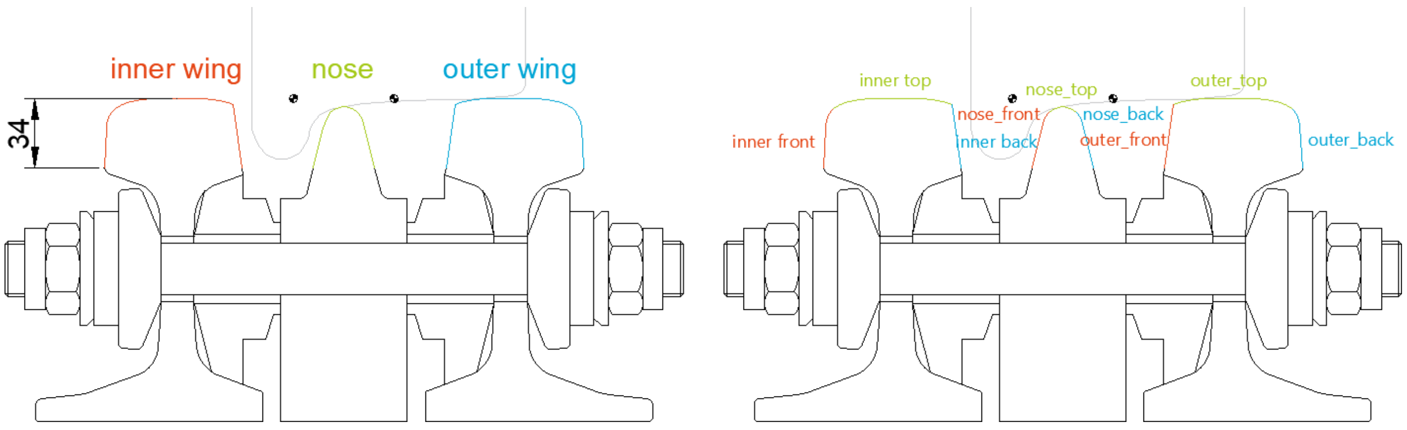


Figure 28 – Example of how a cross section would be divided in nine selections

## 3.1 Lateral wing rail wear

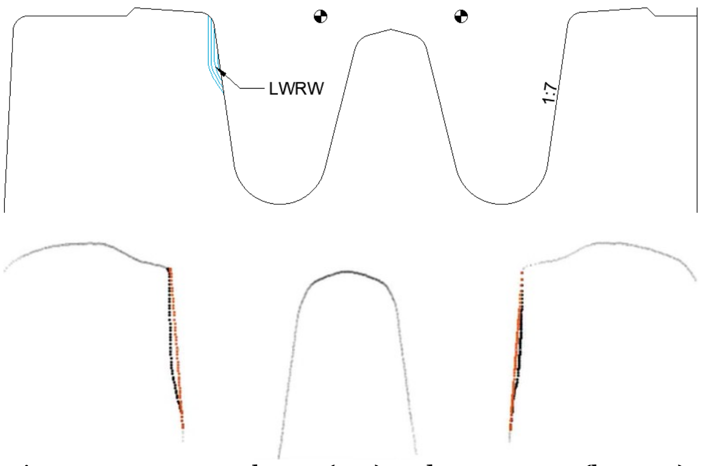


Figure 29 – LWRW theory (top) and assessment (bottom)

The first feature is lateral wing rail wear (LWRW). The top half of Figure 29 shows how this mechanism looks in theory. Flange-back contact (caused by unfavourable wheel trajectories) scrapes off material from the 1:7 slope at the side of the wing rail. The amount of lateral wing rail wear can be expressed either as maximum depth or as volume. The maximum depth is a practical metric, that is most visible/measurable on-site. The volume is harder to assess on-site, but seems like a better metric for degradation tracking (chapter 4).

The theory has been implemented in the algorithm by taking the inner back and outer front of the cross section (like described in Figure 28 on the right) and taking their vertical coordinates. Another script then discretises a 1:7 slope (by adding lateral coordinates to

[1] The names inner and outer match with the inner and outer side of the wheel of the measurement vehicle (like indicated with the grey wheel). If the measurement vehicle would pass the crossing on the other track (which would flip the orientation of the wheel), the names outer and inner would be flipped.

the vertical coordinates) a couple mm away from where the true 1:7 is expected. The last step is to move that slope laterally, on to the cross section, with the minimum possible distance. The whole process can be regarded as creating a 1:7 slope ruler and moving it laterally till it hits the wing rail. The result is can be seen as the red dots in Figure 29.

Extracting both metrics - maximum wear distance and wear volume - is easy now. It's simply a matter of subtracting the red and black line. This is done respectively as $\max(vector - vector)$ and $sum(vector - vector)$, which computes very fast compared to looping over all values.

## 3.2 Lateral wing rail deformation

While the wing rail can wear laterally, the opposite can also happen (Figure 30). Too much vertical pressure can cause the wing rail to plastically deform laterally (known as lipping). This deformation is traceable by simply upgrading the algorithm of the previous section to handle 'negative' wear.

To do so, the 1:7 slope couldn't take all of the relevant (black) data points in to account anymore. Doing so led to errors (Figure 31, in red). Therefore, the algorithm was restricted to only looking at data points with $20 < z < 34\,mm$ below rail top. This led to good results for both wear and deformation (Figure 31 lower row).

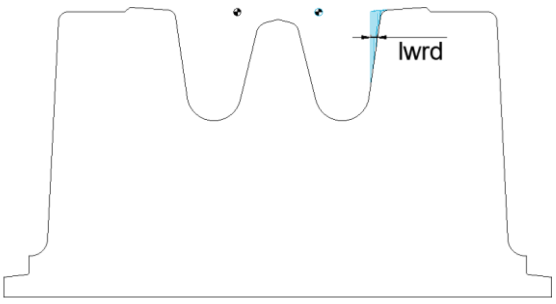


Figure 30 - Lateral wing deformation

## 3.3 Lateral nose wear

When wheels are grinding past the side of the nose (i.e. because of track alignment issues), the nose starts wearing laterally. It is important to track this process, because it is unacceptable that wheels impact the nose on a point where it's too thin to bear the wheel. This can be done in a way similar to lateral wing wear; using a template that matches the designed geometry. The difference lies in the shape of this design, for the side of the nose and the side of the wing rail. While the side of the wing rail is a simple 1:7 slope, the side of the nose is more complex. It changes from a 1:3,220 nose approach (Figure 32 green), to a 1:4 nose side (red), to a regular 1:20 slope of the full rail (blue).

This calls for a bi-linear template, that is achieved by letting the algorithm move multiple templates (with different slopes) through each other. This is exactly the same as how two milling steps in the factory would shape the crossing. An overview of these shapes is shown in Figure 32.

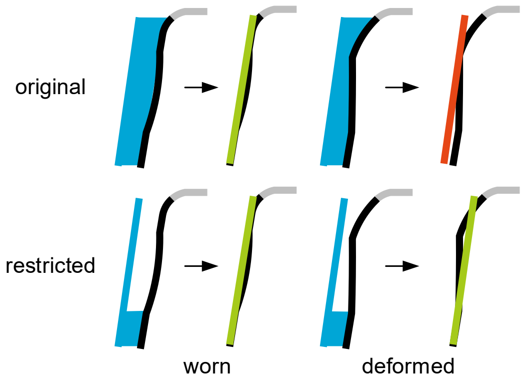


Figure 31 - Algorithm improvement

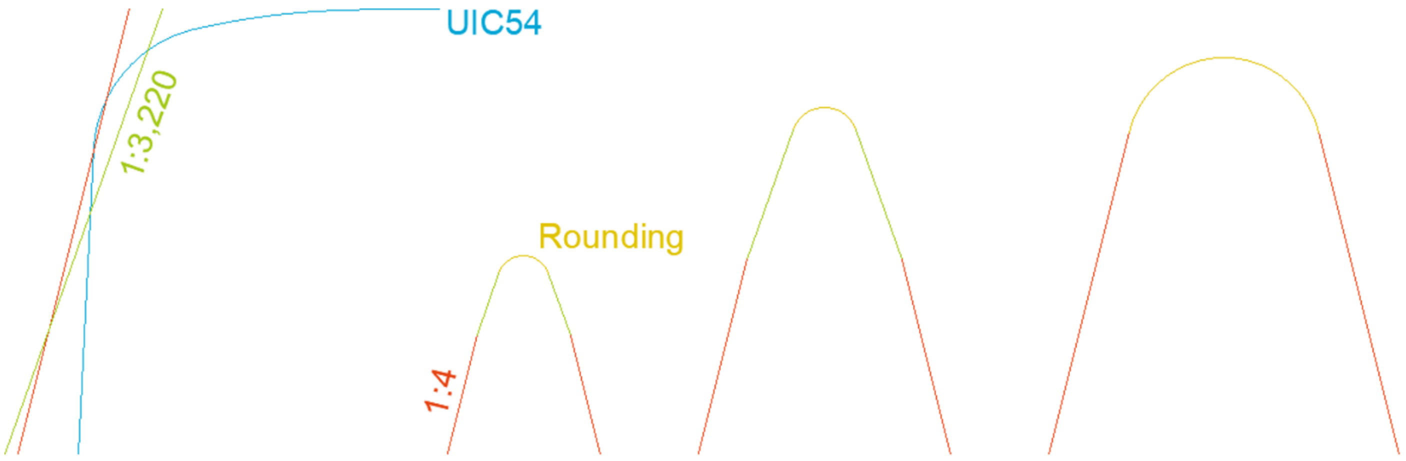


Figure 32 – Cross sections of the Dutch nose design and all shapes that play a part in the nose geometry

Testing the templates on real measurements showed that the transition between 1:3,220 and 1:4 can simply be based on the longitudinal position. The transition between 1:4 and 1:20 proved to be less precisely fabricated; there is some fluctuation in the length and location of the transition (even for the same type of crossing). This might have something to do with either the determination of the nose tip location or the production process. For this reason, the transition between 1:4 and 1:20 is done in two steps.

## 3.4 Lateral nose deformation

Lateral nose deformation can be expected when softer rail materials (like in the cast manganese crossings) are subjected to high wheel loads. The material plastically deforms, which contributes to nose height loss, but which also causes a lateral deformation. The nose gets wider, because of this mechanism. This is, like on the wing, the opposite of lateral wear. This is again addressed by adapting the lateral nose wear software a bit, to recognize and assess negative width loss.

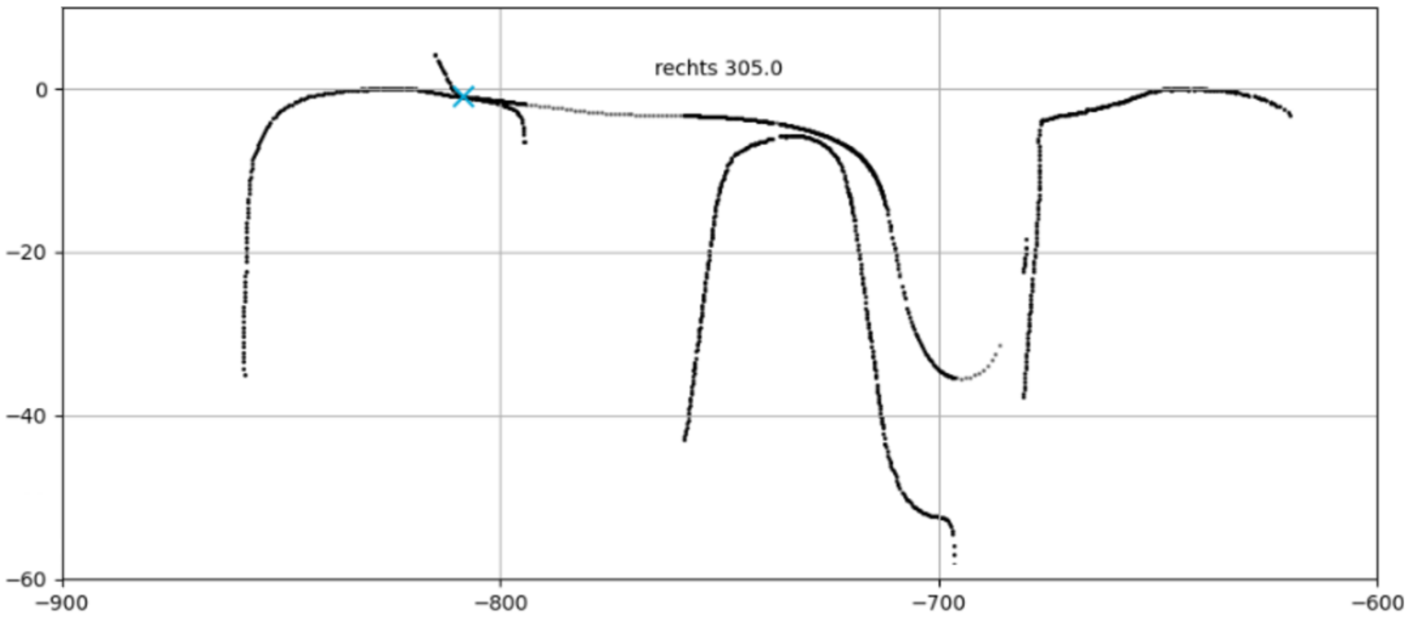


Figure 33 - Single frame from a wheel dip animation. The blue x shows the contact estimate

## 3.5 Wheel dip

Most features use algorithms similar to the earlier described method. One feature however, required a different approach. This concerns Wheel Dip. The goal of this metric is to give an estimation of how deep wheels would sink in to the crossing's flangeways. The metric can be regarded as a digital version of [4].

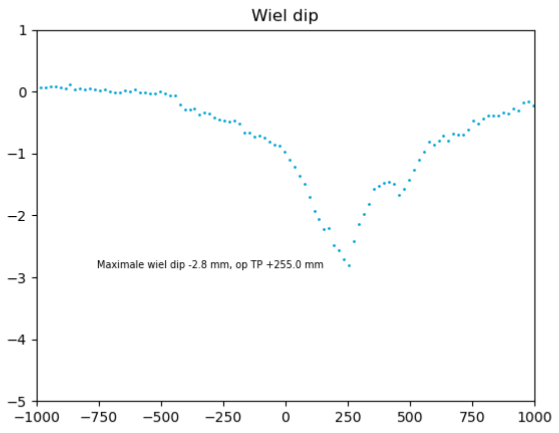


Figure 34 - Wheel dip trajectory plot

To derive this feature, a different kind of template is projected above the cross sections. From the norms [34] a continuous description of a new S1002 wheel profile has been derived. This template is then discretised above the crossing. The last step is to lower the wheel, till it hits either the wing or the nose. Doing this for all cross sections results in an animation (Figure 33) where the wheel dips in to the crossing. The blue X is the indication of where the wheel would contact the rail. The word 'indication' is important to underscore; true wheel-rail contact is much more complex. The model could (in order of priority) be improved by: using multiple wheel profiles, trying various lateral positions, coupling the wheel with the opposite wheel and (ultimately) adding physics.

While the animation is useful for insights in the state of the geometry and causes of wear / deformation, the real metric should be a single number per measurement. This is simply the maximum in Figure 34. It serves as an indication for how deep the wheel can sink in the crossing, which could tell something about wheel impacts.

## 3.6 Wing height loss

The 'extra lateral volume' has to come from somewhere; wing rails lose height because of the plastic deformation. In addition to that, extra height is lost because of wear. Keeping track of this loss is important, because the height loss on the wing promotes a wing-nose transition at the thinner parts of the nose [35] (something that should be prevented). Knowing what losses can be expected should prevent surprises and promote preventive actions.

Calculating wing height loss requires an approach tailored to the original design of the crossing. Most cast wing rails have a horizontal top (Figure 41) while constructed crossings have a UIC54 top (Figure 28) or specific milling profile (Figure 35). Measuring the loss thus requires a reliable classification of the crossing that the algorithm is dealing with. This classification was not based on ProRails SAP asset database, because the data contains too much false classifications. For example, SAP might suggest that a crossing is cast while it is a constructed high wing rail crossing. Instead of the SAP data, the measurement itself is automatically classified by an algorithm with two steps: first classify whether the crossing is cast or constructed, second determine the manufacturer (the foundry in Outreau, JEZ foundry, Voestalpine Turnout Technology Netherlands BV or Vossloh Cogifer Kloos BV). This algorithm is further explained in section B1.5 of appendix B.

As soon as the crossing design is selected, the principle works the same as the previously mentioned features: take the relevant part of the cross section, discretise a template of the right shape above the measurement and lower the template on to the measurement. The only exception is the UIC54 wing top: for this case, the template is lowered on to the crossing on several lateral positions and saved at the best fit. The reason is that a wrong lateral position of such a curved template causes large errors. The fitting process is further explained in section B1.6 of appendix B.

Good projections can lead to very precise results. Figure 35 shows an example for a crossing from manufacturer Vossloh Cogifer Kloos BV. The template (green) is a bilinear combination of a 1:40 slope and a 1:15 slope, which is the same as the milling tool [36] that is used during manufacturing. In the figure, it can easily be seen how the tip of the wheel tread cut out 2.72 mm of material on the right wing rail. What's even better though, is that small height losses like on the left wing rail can be spotted as well (both by the computer and the human eye). The area where wheels had influence on the geometry is clearly distinguishable, despite that the loss was 0.58 mm.

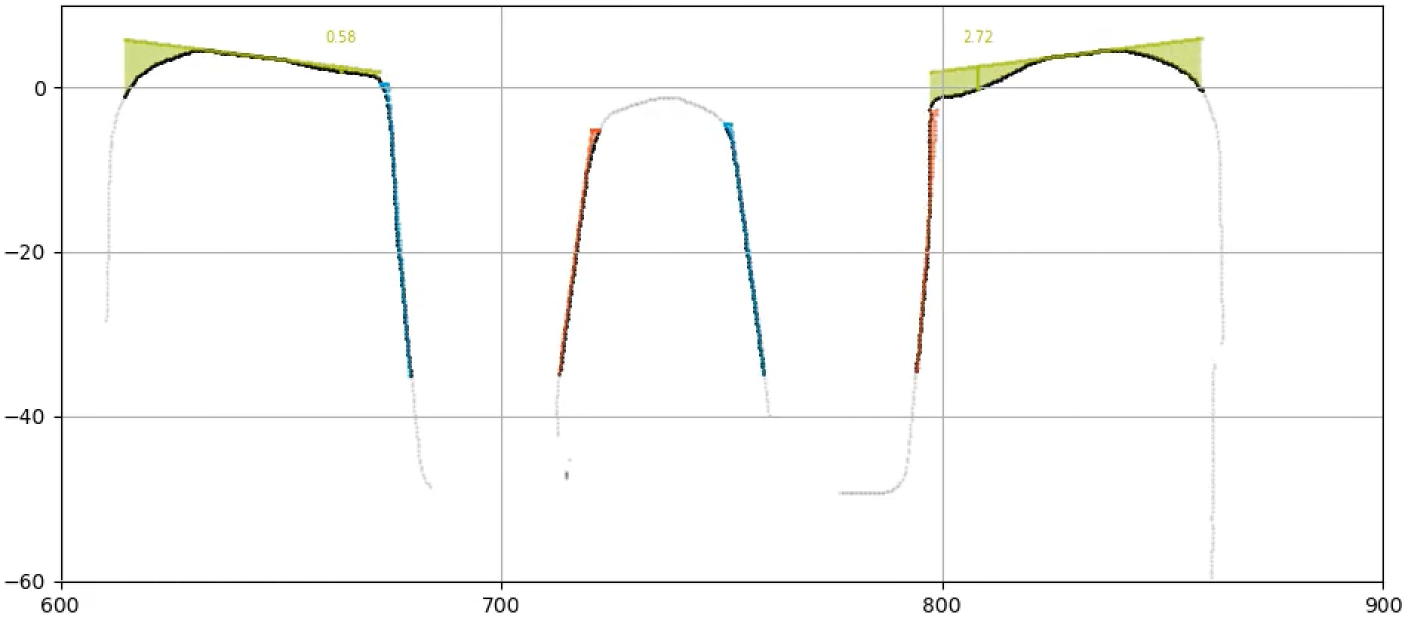


Figure 35 – Wing height loss (in mm) measured in green, on a constructed high wing rail crossing

## 3.7 Nose height loss

The last parameter, nose height loss, has a similar story as wing height loss. Again, the height loss is caused by a combination of wear and deformation. And again, the nose height loss is with respect to the original design of the crossing.

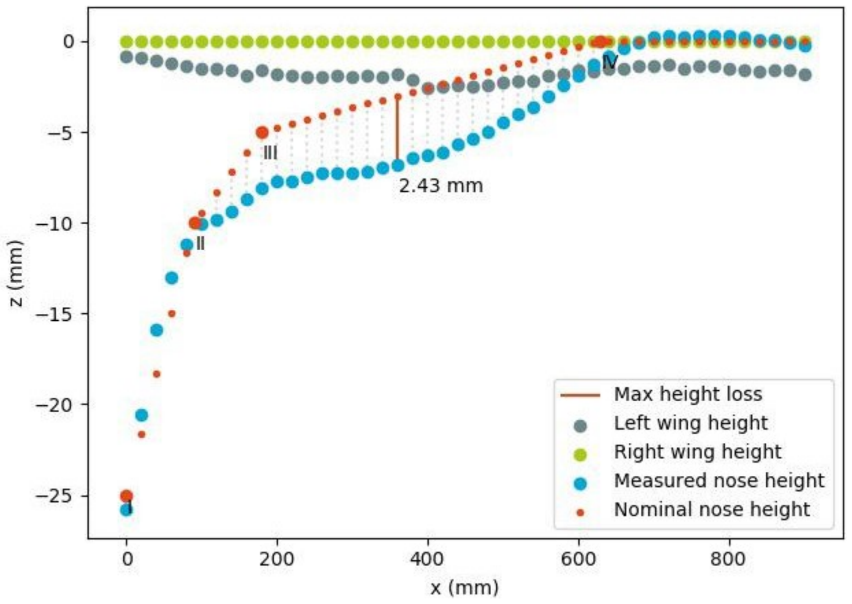


Figure 36 – Example of a nose height loss plot

With the known location of the TCP, the nominal nose height had to be derived from the design drawings. The original design is gathered in a specially made lookup table. The table contains parameters of all design drawings of common crossings in the Netherlands. All crossing drawings were downloaded from the Rail Infra Catalogue (the system on which ProRail stores design drawings) and these drawings were studied and put in to the table. The table contains information about the length of the crossing, whether it's curved, what rail profile is used, what milling tools are applied, etc. These parameters are used to form a continuous design height profile of the crossing nose, which is then queried for a single height per cross section, based on the longitudinal location of that cross section.

The results of this system are shown in Figure 36. The plot provides insight into the wear of the nose and whether the wing rail has lost height in a similar manner. The greatest deviation is plotted and stored as a metric for the crossing degradation. While a single number like that doesn't cover the all information of Figure 36, it is essential to break such a figure down to a single number. In that way, bigger populations of measurements can be compared.

## 3.8 Discussion

In Figure 39 to Figure 37 several detections are shown. They demonstrate how various digital templates (a 1:7 line, a UIC54-shaped template, a horizontal line, a bi-linear template) enable the measurements of various features. The templates are placed based on longitudinal position and the original design drawings. The right design drawings are chosen based on a classification algorithm that determines the type of the crossing.

The features show geometry degradation per measurement. The next step is to follow specific interesting crossings, through multiple measurements. This is done in the next chapter.

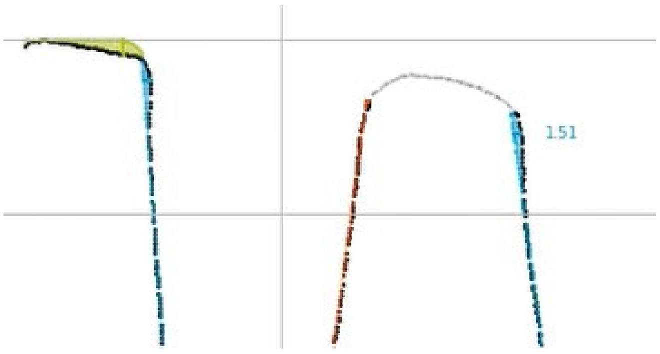


Figure 39 - Detection of lateral nose deformation (right)

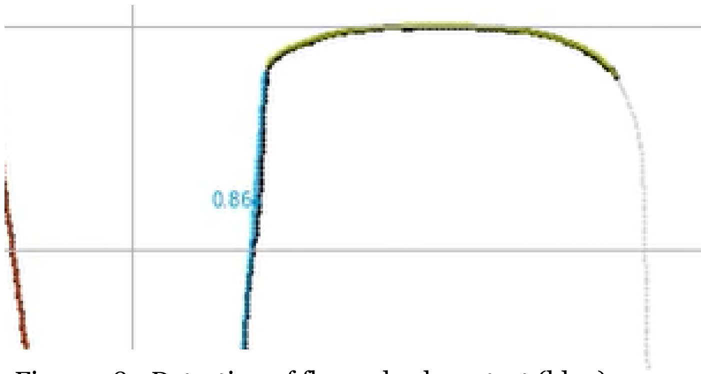


Figure 38 - Detection of flange-back contact (blue)

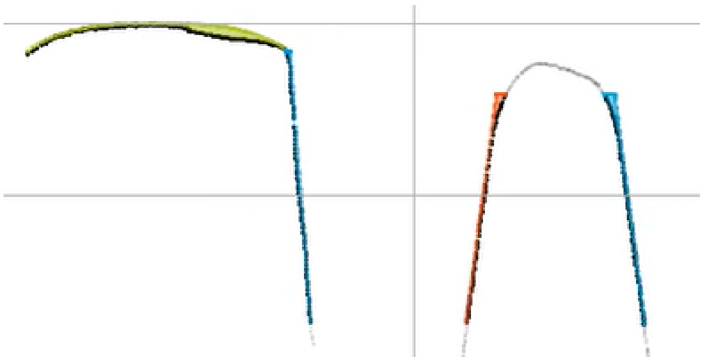


Figure 40 - Detection of height loss on a UIC54 top (green)

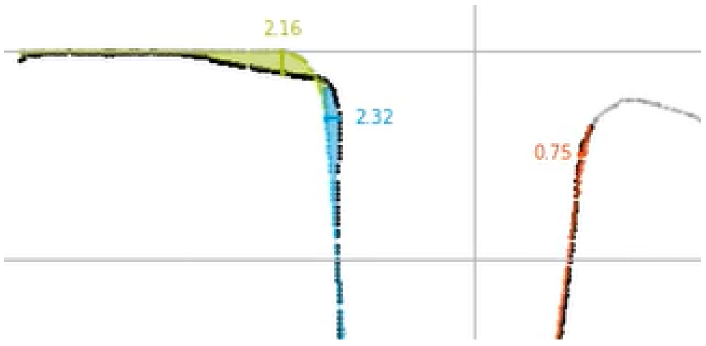


Figure 37 - Detection of lipping on a cast wing (blue)

# 4 Results

The previous chapter has described the method to generate features from the measurements. This chapter discusses the results of a first analysis of those features: de vertical degradation of wing rails. The case study serves as a proof of concept; to show that the software from chapters 2 and 3 is working and that it provides usable results.

## 4.1 Introduction

In chapter 3, 6 new features were derived. Moreover, ProRail already owned quite some existing features (other datasets like traffic load). These features can be used in two ways: find new hypotheses by analysing all data or investigate existing hypotheses by investigating an interesting part of the data. For this case study, the latter was done by using experience to define a promising scope: the correlation between vertical wing rail degradation and traffic loads from relevant directions.

The datasets that were used for this analysis were: the ProRail object database (SAP), the profile measurements and traffic loads. First, overviews of crossings with known installation date in 2015-2017 were created for 1:9 cast manganese crossings. Second, the crossings were matched to the yearly traffic load per direction (for the year after installation). Lastly, the dates of available measurements were added to the overview.

The overview of crossings, their traffic load (per direction) and their measurement dates was used to identify interesting crossings. Criteria were: a sufficient amount of measurements, wing rail loaded mainly in one direction (either facing or trailing) and relevant amount of tonnage between the first and last available measurement. These interesting crossings were inputted in to the software, by using three parallel queues; one queue per crossing (with multiple measurement dates). The number three comes from three processor cores for calculations, while keeping the fourth core in the pc available for other work.

## 4.2 Results

All interesting crossings were followed from their installation data up to their most recent scan. All degradation animations were manually checked for the maximum wing height loss, to rule out any problems with scan or assessment (like shadows and reflection).

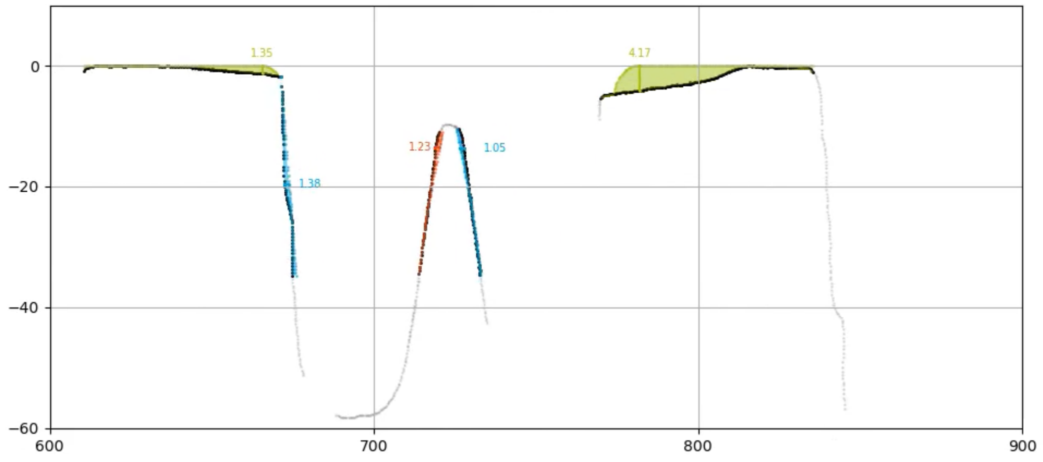


Figure 41 – The worst wing height loss from the case study (all numbers are in mm)

Figure 41 shows the example of the most degraded wing that was encountered. Note that other crossings in the country might be even worse; keep in mind that the dataset contains scans up to the 16th of May 2019, which gives about 4 years of degradation.

After doing this for 60 relevant measurements, Figure 42 and Figure 44 were plotted. From these figures and the underlying data, several things can be observed:

1. The first mm from the wing seems to be lost almost instantly
2. Trailing traffic is more hurtful to the wing height than facing traffic
3. Higher operating speeds lead to more height loss, only when operating in trailing direction

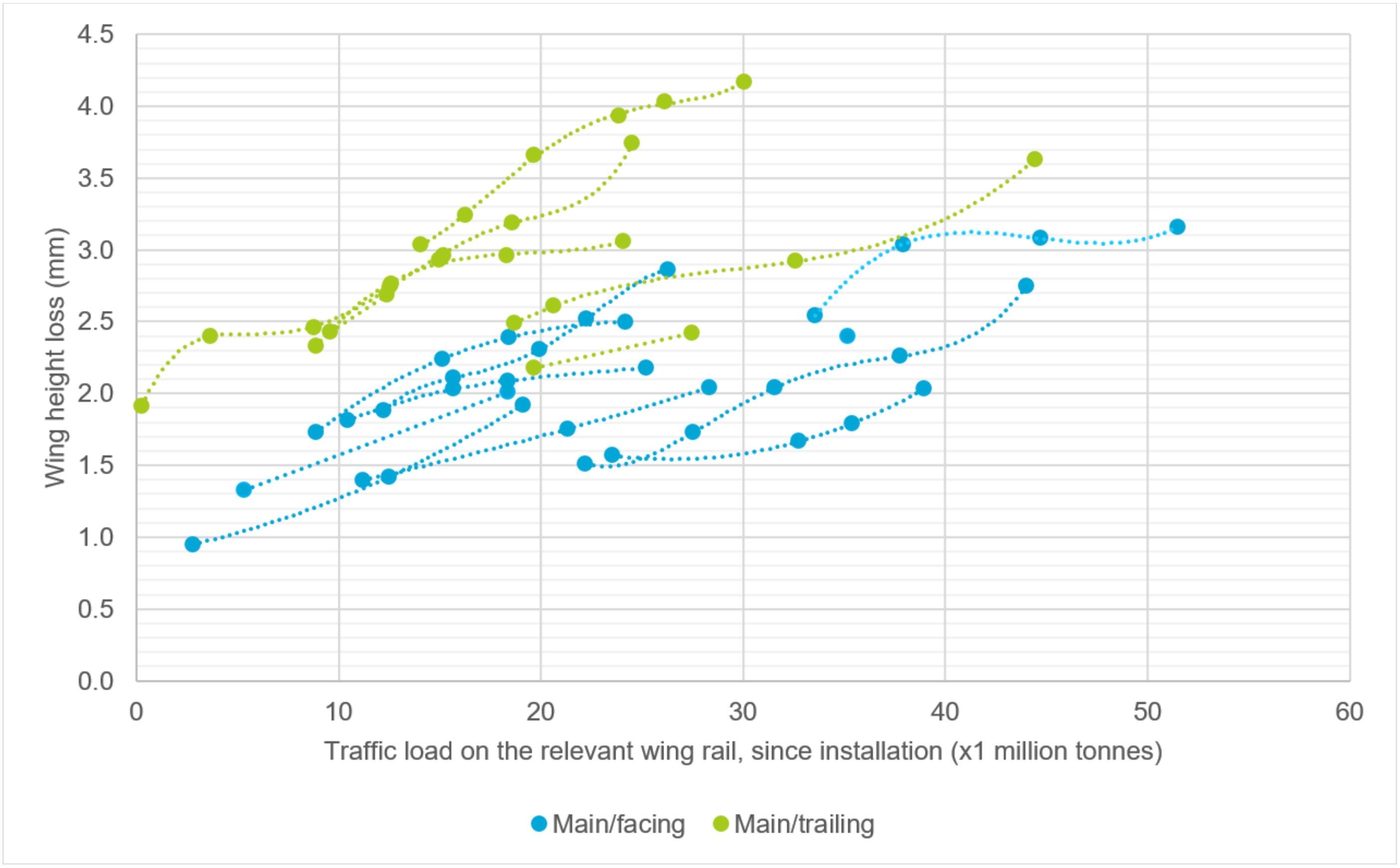


Figure 42 - Wing height loss of 60 measurements on 16 cast manganese 1:9 common crossings

## 4.3 Discussion

### Fast loss of the first mm of height

Measurements not long after installation indicate that (with both traffic directions) the first mm of height is lost almost instantly. This is caused by a combination of wear and deformation. Simulations from [12] indicate that the wheel-rail contact patch is constrained to a very narrow width in this area. The simulations further indicate that the consequences are high contact pressure (leading to deformation) and a high wear number (an indication for wear). Figure 43 gives an impression of how this is caused by the design of the wing rail: the 1:15 area of the wheel is running on a rounded edge of 8 mm radius. A second cause could be that the manganese steel first needs to harden from the first traffic.

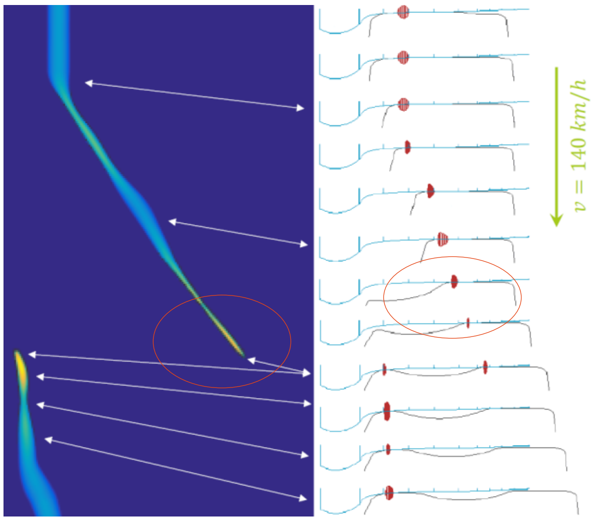


Figure 43 - Wheel-rail contact on the crossing [12]

## Trailing traffic is more hurtful than facing traffic

In the back-to-front situation, wheel-rail contact passes from the crossing nose on to the wing rail. This leads to an impact on the wing rail, characterized by a lot of wear and deformation. Like described in the previous sub-section, the front-to-back situation is also causing height loss, but simply less than the impacts from back-to-front.

## Higher operating speeds lead to more height loss, only when operating in trailing direction

The back-to-front situation has a higher height loss for higher operating speeds. The two back-to-front crossings with the lowest degradation are operated at 40 km/h. This might have something to do with the nature of the degradation; the impacts of wheels on the wing rail is dynamically amplified if the speed increases, so a faster degradation at higher speed seems logical.

The facing situation has similar height loss for all operating speeds. For example: the two front-to-back crossings with the lowest wing height loss are operated at 140 km/h. Amsterdam 255A (40 km/h) needed 23,2 MGT less load to reach a loss of 2 mm than Dedemsvaart 231B (140 km/h). This might be explained in the different nature of the wheel-rail contact; it is a more classical form of wheel-rail interaction (without impact on the wing rail). The wear and deformation therefore might be less related to operating speed.

Note that these hypotheses, on the effect of operating speeds, are drawn from a relatively small dataset. More cases are needed to consolidate them.

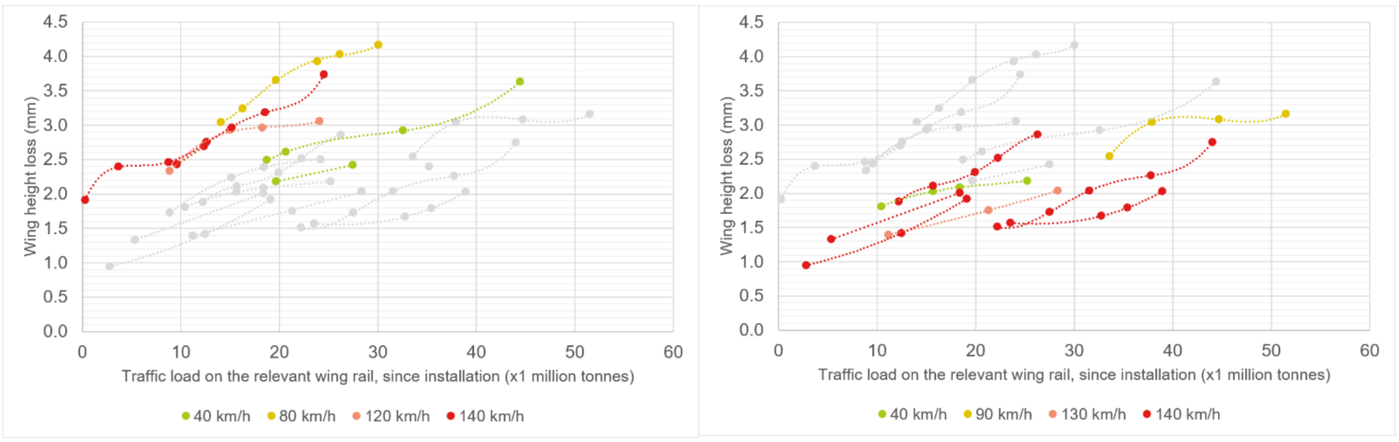


Figure 44 – Same data as Figure 42, with trailing (left) and facing (right) separated, with operating speeds
Measurements from the opposite operating direction are shown in grey, for reference

# 5 Conclusions and recommendations

## 5.1 Conclusions

Most of the time in this project was spent on the automated aligning (chapter 2). The result is that crossing measurements can now be aligned automatically, to enable the generation of features.

The main goal of the project was to see whether prioritizing and predicting is possible with the existing dataset. The software can be used to detect dangerous geometry. Think for example of the lipping phenomenon, where lateral deformations make the crossing vulnerable to breakage. This covers the prioritizing part. Moreover, predicting is possible by combining a useful set of assessments in to an overview like Figure 42. With such an overview, measurements can be placed between a larger population to detect and predict dangerous situations, overperformance and underperformance. These insights can be used to suggest improvements, like in the next section.

## 5.2 Recommendations

### Work towards basing the measurement priorities on traffic loads / operating speeds

Degradation trends like Figure 42 are based on crossings with ‘relevant tonnages’ and ‘relevant operating speeds’. This means that there is an interesting amount of change between two measurements. This often relates to main line tracks. Sidings and yards often yield less tonnage and thus their geometry changes much slower. As soon as these degradation speeds are clear enough, one could consider basing the measurement interval on the load. This could even be done with the same amount of measurements: heavily loaded crossings could be surveyed more often and rarely used crossings could be surveyed less often.

An alternative option would be to determine whether false/missing measurements should be redone, or that it is ok to miss one measurement from that specific crossing. Logistics can decide to catch up a false measurement on a heavily loaded crossing, while skipping a missed measurement on a rarely operated spur.

### Do a similar analysis on the switch panel and closure panel

This project and the future work (as described in section 5.3) focus exclusively on fixed common crossings. The turnout geometry measurements also contain switches, closure rails, fixed obtuse crossings and movable crossings (both common and obtuse). The wear and deformation of these parts could also be studied, along with the degradation of their alignment parameters (like free-wheel clearance in switches).

### Automatically detect part replacements

The turnout measurement trains are able to measure cross sections every (longitudinal) 2 to 3 cm. With this precision, the detection of welds and insulated rail joints should be possible. When such locations shift between two measurements parts of the turnout must have been replaced (welds cannot be cut out and cast at the same location).

## 5.3 Future work

### Analysis of more parameters and 1:9 crossing types

Figure 42 showed a first example of degradation relation: wing height loss (as described in section 3.7) on cast manganese 1:9 crossings. The software calculates 6 other parameters however (sections 3.1 to 3.6). Moreover, measurements from other (constructed) crossing designs are available. The first priority is to analyse all 7 parameters for both constructed and cast 1:9 crossings, like in the case study from the previous chapter.

Moreover, the wheel dip calculations could be extended to provide extra parameters related to that dip. [3] suggests three additional features:  both wheel dip angles (like described in Figure 6) and the distance between impact and TCP. Also, the difference between designed and worn dip depth from [4] is a feature that could be studied in the future.

Another option to gain extra features is to put the current parameters in a different perspective. Lateral parameters can be translated in to other measurements like gauge, check gauge and flangeway width. These are simply parameters that are a less complicated (to calculate) version of the parameters that the system already provides.

## Analysis of other crossing angles

The current project focused on 1:9 common crossings. This already covers 32% of all crossings (and 40% of all measurements). After analysing the 1:9 results, other crossing angles will be added to the software. The software has been prepared for this. In most cases it is a matter of changing numbers (not formulas) but in some cases some extra functionality has to be added (i.e. for curved crossings and other rail profiles). This will be done based on the same careful way as for the 1:9 crossings and will be scheduled based on priorities that are set by ProRails turnouts department.

## Sharing results with stakeholders

The insights from this project benefit multiple stakeholders. Think of contractors, manufacturers, inspectors, regional managers, cost engineers and policy makers. ProRail already has the Branche Breed Monitoring Systeem (BBMS) system for sharing data with those stakeholders. As soon as parameters are mature enough, they can be implemented in BBMS. Up to that time, the developments are done in ProRails Big Data Analysis Platform (BDAP): Azure. The BDAP environment is the location where ProRails software projects are being developed, for research purposes or as a preparation on production in BBMS. This project has already migrated to Azure and contacts have been laid to ensure a smooth implementation.

## Creating the right visualizations for the various stakeholders

Like described in the previous sub-section, the project has multiple potential end-users. Interviews with those stakeholders have shown that they will only take full advantage of the results if they trust the usability. Visualization is a key element in implementing the results of this project in their processes.

## Finding the relation between geometry & RCF

One of the first images in the report (Figure 4) showed pictures of rolling contact fatigue (RCF). Geometry measurements don't show RCF and thus don't replace RCF inspections. However the final goal is to find and predict geometry states that are sensitive to RCF. To do this, the geometrical features from this project will be compared with reports of RCF damage (mostly created through ultrasonic measurements).

The aim is to get a reliable dataset of geometry stages (captured in features) and RCF reports on the same asset. In Figure 45, such a setup is shown for one geometrical feature in comparison with the tonnage. More than one geometry feature is available though. Various techniques are available to make an analysis like Figure 45 in 3D or higher (i.e. Principal Component Analysis).

Combinations of RCF and geometrical features should yield insights about when the RCF damages happen. This would make RCF predictable and thus easier preventable. By automating measurement alignment and feature generation, this project has delivered a large step towards such predictions. A large step in the transition from reactive to preventive maintenance.

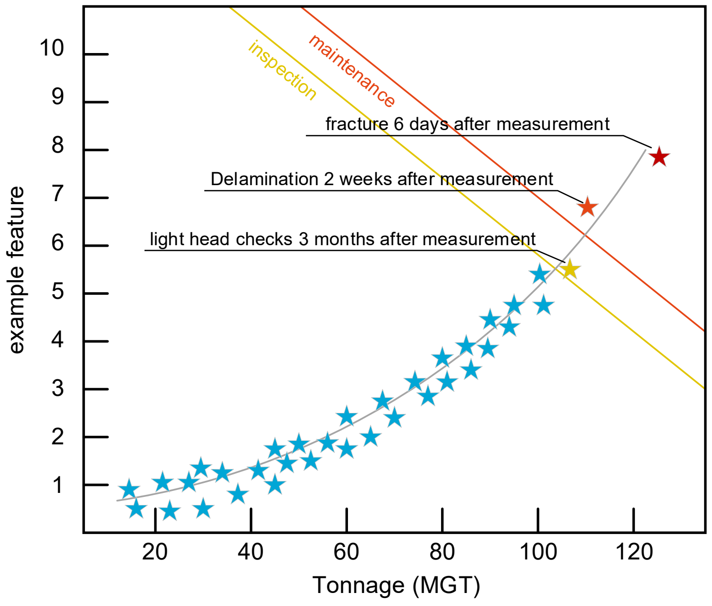


Figure 45 - How to relate RCF to geometry

# Bibliography

[1] I. Shevtsov, „Photo's from damaged crossings (as presented on Vakdag Baan 2020)," 2020.

[2] M. Steenbergen, „Doctoral thesis: Wheel-rail interaction at short-wave irregularities | TU Delft repository," 5 6 2008. [Online]. Available: https://repository.tudelft.nl/islandora/object/uuid%3A98a5de70-3464-4487-b2c0-e4bfd51fac40?collection=research. [Geopend 7 4 2021].

[3] D. Nicklisch en V. Sauer, „Messung und Bewertung der Überlaufgeometrie starrer Herzstücke," *EI-Eisenbahningenieur,* nr. September, pp. 18-23, 2013.

[4] U. Gerber en W. Fengler, „Belastung von Weichen mit starrer Herzstückspitze," *ZEVrail Glasers Annalen,* vol. 131, nr. 5, pp. 202-214, 2007.

[5] U. Gerber, A. Zoll en W. Fengler, „Fahrzeugbasierte Beurteilung des Herzstückverschleißes," *Der Eisenbahningenieur,* pp. 26-30, 2013.

[6] X. Liu, V. Markine, H. Wang en I. Shevtsov, „Experimental Tools for Railway Crossing Condition Monitoring," *Measurement,* vol. 129, pp. 424-435, 2018.

[7] Z. Wei, „Evaluating degradation at railway crossings using axle box acceleration measurements," *Sensors,* vol. 17, nr. 2236, 22 5 2017.

[8] X. Shu, N. Wilson, C. Sasaoka en J. Elkins, „Development of a real-time wheel/rail contact model in NUCARS® and application to diamond crossing and turnout design simulations," *Vehicle System Dynamics,* vol. 44, nr. 1, pp. 251-260, 2006.

[9] E. Kassa, C. Andersson en J. Nielsen, „Simulation of dynamic interaction between train and railway turnout," *Vehicle System Dynamics,* vol. 44, nr. 3, pp. 247-258, 2007.

[10] International Union of Railways (UIC), „6.1.2.2 Crossing panel – stiffness and geometry," in *Innotrack – Concluding Technical Report*, Solna, Intellecta Infolog, 2010, pp. 164-165.

[11] X. Liu en V. Markine, „MBS Vehicle-Crossing Model for Crossing Structural Health Monitoring," *Sensors,* vol. 20, nr. 10, p. 2880, 2020.

[12] J. Wegdam, „Assessing crossing geometry (MSc. thesis)," 12 12 2018. [Online]. Available: http://resolver.tudelft.nl/uuid:4505c798-0446-47b0-862f-19fd8632c8f7.

[13] C. Wang, V. Markine en I. Shevtsov, „Analysis of train/turnout vertical interaction using a fast numerical model and validation of that model," *Journal of Rail and Rapid Transit,* vol. 228, nr. 7, pp. 730-743, 2014.

[14] DB Systemtechnik, „flyer_ESAH_D_2018_pad_flyer_8_seiten," 8 2018. [Online]. Available: https://www.db-

systemtechnik.de/resource/blob/3233610/763f3cb992a3cf3ab0fbdf6dfe21a18f/Aktuell_weichenherzst ueckdiagnose_esah_deutsch-data.pdf. [Geopend 6 4 2021].

[15] K. Kassa en J. Nielsen, „Dynamic interaction between train and railway turnout: full-scale field test and validation of simulation models," *Vehicle System Dynamics,* vol. 46, nr. 1, pp. 521-534, 2008.

[16] U. Gerber, A. Zoll en W. Fengler, „Fahrzeugbasierte Beurteilung des Herzstückverschleißes," *EI-Eisenbahningenieur,* nr. September, pp. 26-30, 2013.

[17] VI-grade GmbH, VI-Rail 16.0 Documentation, Marburg: VI-grade GmbH, 2014.

[18] E. Kassa en G. Johansson, „Simulation of train–turnout interaction and plastic deformation of rail profiles," *Vehicle System Dynamics,* vol. 44, nr. 1, pp. 349-359, 2007.

[19] International Union of Railways (UIC), „Crossing panel – stiffness and geometry," *Innotrack Concluding Technical Report,* pp. 164-165, 2010.

[20] R. Skrypnyk, U. Ossberger, B. Pålsson, M. Ekh en J. Nielsen, „Long-term rail profile damage in a railway crossing: Field measurements and numerical simulations," *Wear,* Vols. %1 van %2472-473, p. Article 203331, 2021.

[21] ProRail, „OHD00022-1-V008 Onderhoudsdocument: Instandhoudingsspecificaties wissels en kruisingen - Geometrie," 1 2 2017. [Online]. Available: https://prorailbv.sharepoint.com/:b:/r/sites/RIC/Extern/OHD00022-1-V008/OHD00022-1-V008.pdf. [Geopend 9 4 2021].

[22] ProRail, „IHS00002-1 Instandhoudingsspecificatie Wissels en kruisingen Deel 1: Onderhoudswaarden, Interventie waarden & Onmiddellijke actiewaarden," 1 1 2019. [Online]. Available: https://prorailbv.sharepoint.com/sites/RIC/Extern/IHS00002-1-V001/IHS00002-1-V001.pdf. [Geopend 14 4 2021].

[23] ProRail, „OHD00033-1-V007 Instandhoudingspecificaties: Spoorinfra – Baan en Overwegen," 1 1 2015. [Online]. Available: https://prorailbv.sharepoint.com/sites/RIC/Extern/OHD00033-1-V007/OHD00033-1-V007.pdf. [Geopend 14 4 2021].

[24] ProRail, „OHD00129-1-V004 Onderhoudsdocument: Instandhoudingsspecificaties wissels en kruisingen - Geometrie," 1 2 2017. [Online]. Available: https://prorailbv.sharepoint.com/sites/RIC/Extern/OHD00129-2-V001/OHD00129-2-V001.pdf. [Geopend 14 4 2021].

[25] ProRail, „RLN00399-1-V003 Detectie en beoordeling van Spoorstaafdefecten - Deel 1: Beheersmethodiek," 1 10 2016. [Online]. Available: https://prorailbv.sharepoint.com/sites/RIC/Extern/RLN00399-1-V003/RLN00399-1-V003.pdf. [Geopend 14 4 2021].

[26] European Railway Agency, „Technical Specifications for Interoperability | ERA," 16 6 2019. [Online]. Available: https://www.era.europa.eu/activities/technical-specifications-interoperability_en#meeting2. [Geopend 14 4 2021].

[27] Mermec, „Turnouts Inspection and Measurement System_EN0816.1,” 2008. [Online]. Available: Brochure, requested through https://www.mermecgroup.com/press-room/943/resource-library.php/&res=1088. [Geopend 6 4 2021].

[28] M. Rusu, „Doctoral thesis: Automation of Railway Switch and Crossing Inspection,” 12 2015. [Online]. Available: https://core.ac.uk/download/pdf/83926136.pdf. [Geopend 7 4 2021].

[29] ProRail, „RLN00195-V002 Richtlijn: Eisen aan meettreinen en meetdata voor liggingsgeometrie en wisselinspectie.,” 1 6 2013. [Online]. Available: https://prorailbv.sharepoint.com/sites/RIC/Extern/RLN00195-V002/RLN00195-V002.pdf. [Geopend 14 4 2021].

[30] ProRail, „ProRail Informatieportaal,” ProRail, [Online]. Available: https://informatieportaal.spoordata.nl/. [Geopend 05 02 2021].

[31] ProRail, „Wissel profielviewer - BBMS-P,” ERDMANN-Softwaregesellschaft mbH, 1998. [Online]. Available: https://bbms.prorail.nl/web/. [Geopend 06 02 2021].

[32] ProRail, „ProRail meet wissels volautomatisch met lasers - YouTube,” 14 03 2018. [Online]. Available: https://www.youtube.com/watch?v=HIREMMNXzPo. [Geopend 05 02 2021].

[33] Vossloh Cogifer Kloos BV, „Installatievoorschrift voor geconstrueerde verlijmde Kloos puntstukken en kruisstukken 54E1 1:4,5; 1:9; 1:12; 1:15,” Nieuw-Lekkerland, 2010.

[34] UIC, „UIC code 510-2 Trailing stock: wheels and wheelsets. Conditions concerning the use of wheels of various diameters,” International Union of Railways (UIC), Paris, 2004.

[35] C. Wan, „Optimisation of vehicle-track interaction at railway crossings | TU Delft Repositories,” 7 9 2016. [Online]. Available: https://repository.tudelft.nl/islandora/object/uuid%3A8dac8f02-fe9e-4baa-9503-e9b3d79dd1aa?collection=research. [Geopend 14 05 2018].

[36] R. Kuyper, „Drawing 1091030-105: composition common crossing 1:9 54E1 1 to 9 L + R R=195,” Vossloh Cogifer Kloos BV, Nieuw-Lekkerland, 2019.

[37] C. Esveld, N. Frank, S. Jovanovic, A. Kok, A. de Man, V. Markine, R. Oswald, P. Scheepmaker, R. Wenty, G. van der Werf, J. van 't Zand, J. Zoeteman en J. Zwarthoed, Modern Railway Track, Zaltbommel: MRT-Productions, 2001.

[38] I. Shevtsov, „Rolling Contact Fatigue problems at railway turnouts – experience of ProRail,” 13 11 2013. [Online]. Available: http://www.sim-flanders.be/sites/default/files/events/Meeting_Materials_Nov2013/mm_13112013_ivan_shevtsov_prorail.pdf. [Geopend 27 10 2018].

[39] X. Liu, *Pictures of damaged crossings.*

[40] Nextsense, „Wheelset measurement & rail inspection | Railway - NEXTSENSE,” Rittler & Co, [Online]. Available: https://www.nextsense-worldwide.com/en/industries/railway.html. [Geopend 1 4 2021].

[41] Greenwood Engineering A/S, „MiniProf BT Rail - Greenwood Engineering,” [Online]. Available: https://greenwood.dk/railway/instruments/miniprof-bt-rail/. [Geopend 1 4 2021].

[42] Mermec, „Turnout & Crossing Measurement System,” [Online]. Available: https://www.mermecgroup.com/inspect/track-measurement/1090/turnout-e-crossing-.php. [Geopend 14 4 2021].

# A Norms

## A1 Norms for alignment and geometry

Documents OHD 22 [21], OHD 33 [23], OHD 129 [24] and IHS 2 [22] contain the norms for maintenance on switch and crossing geometry. Through a set of figures, these norms are described. Nominal values are green, OW and BW values (maintenance limits) are yellow and IW and VW values (safety limits) are red. Note that document OHD 129 [24] is a special version of OHD 22, which is applicable only on the Hanzelijn railway and at the Kijfhoek hump yard. Because of the operating speeds on the Hanzelijn, the norm has been made suitable for speeds up to 200 km/h (while OHD 22 is valid only up to 160 km/h).

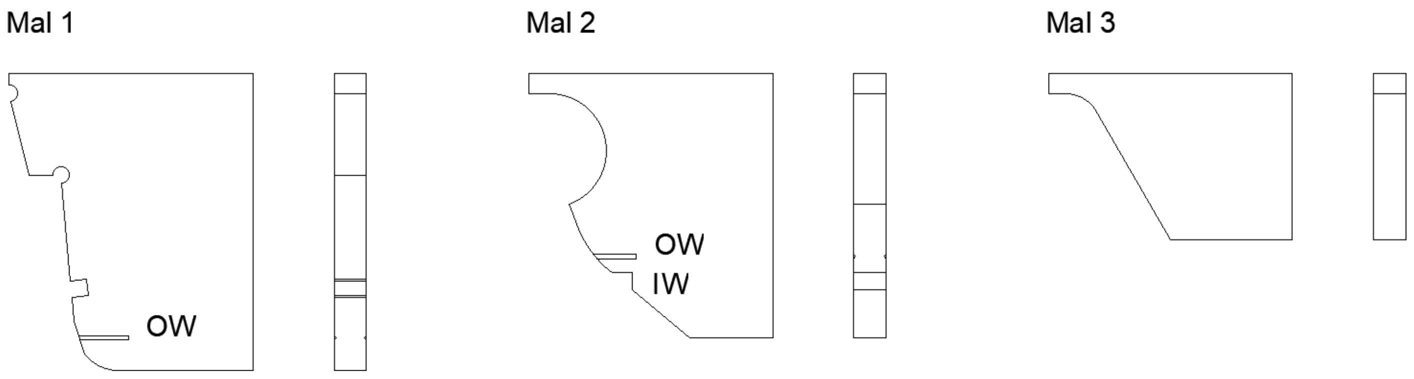


Figure 46 - Moulds to assess the gauge corner slope [section 2.1] (less relevant around crossings)

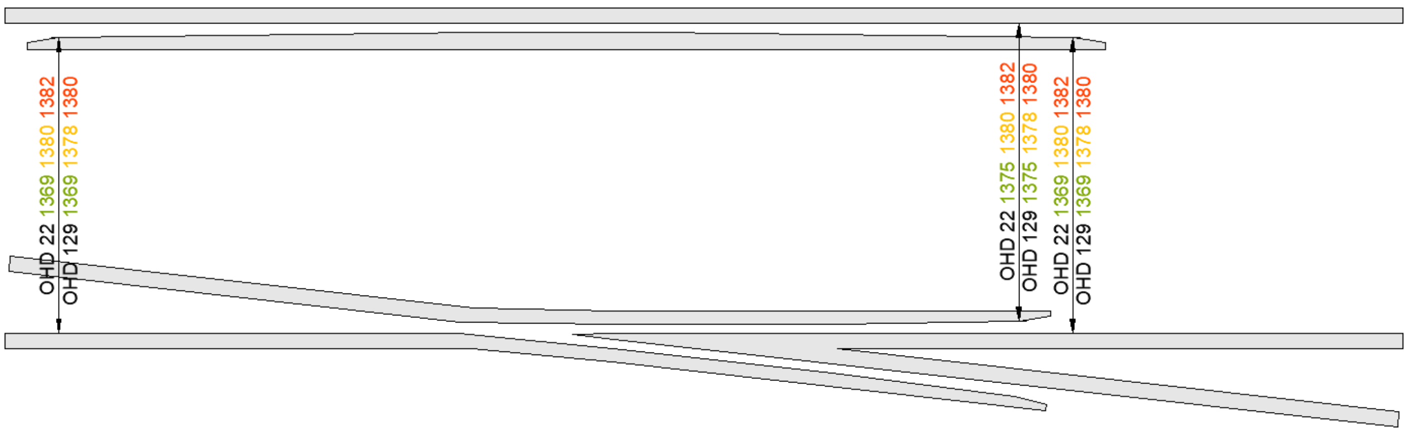


Figure 47 - Norms for lead gauges [section 2.2]

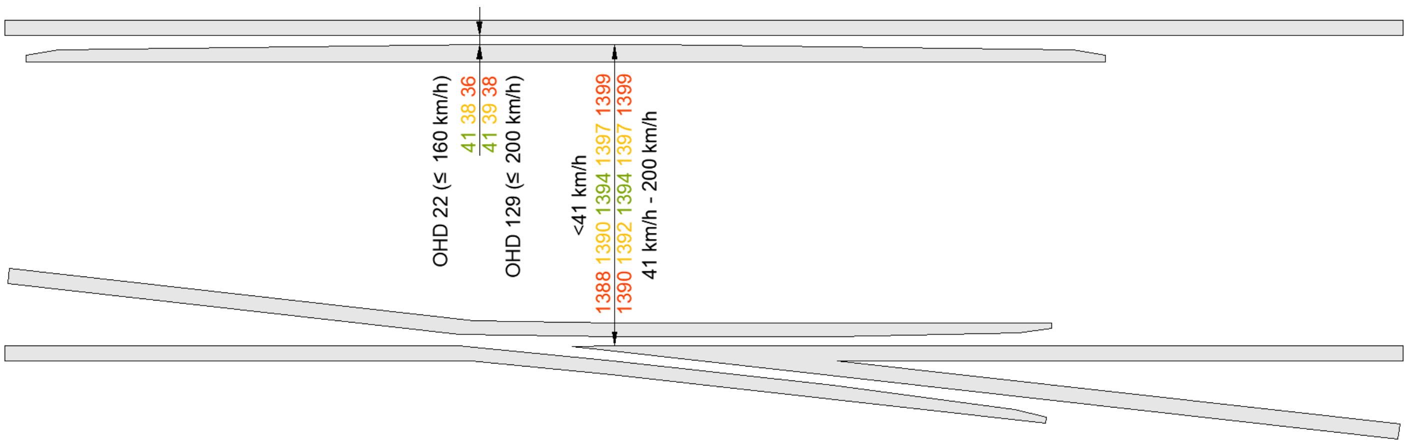


Figure 48 - Norms for the flangeway clearance (left) and check rail gauge (right) [section 2.3]

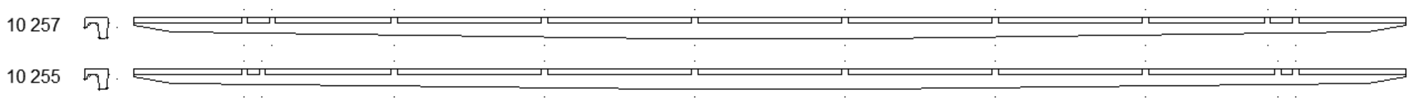


Figure 49 - Two common configurations of check rail supports for 1:9 crossings. The norms state that check rails should not be bent (laterally) too much, because this could make the lead angles too steep. Neighbouring supports should have a lateral difference of less than 2 mm and all supports should not differ more than 4 mm. [section 2.3]. Lateral difference in check rail supports is not allowed in OHD 129.

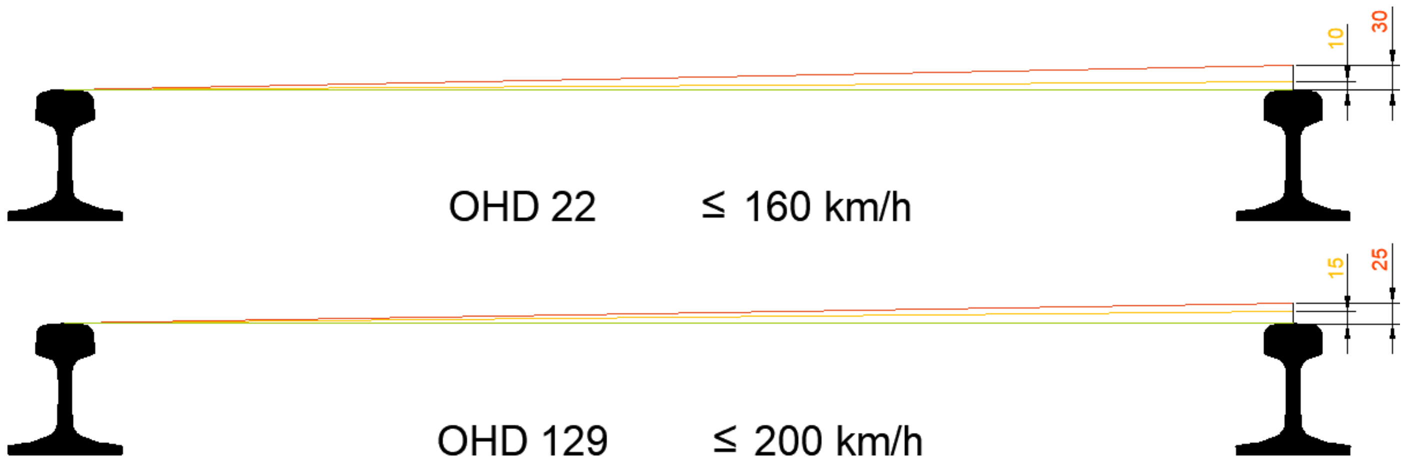


Figure 50 - Norms for cant deviations [section 2.4] for both OHD 22 and OHD 129. Drawing shows the situation where the designed cant was zero. Note that cant deviations clockwise and counter clockwise are treated the same way.

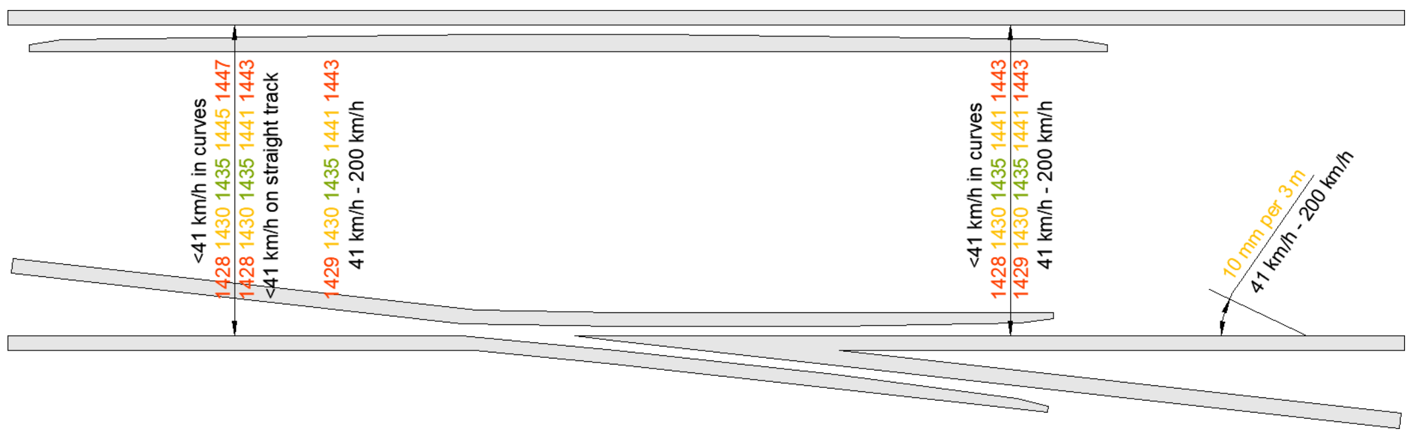


Figure 51 - Norms for the gauge on the front and back of the crossing and maximum gauge gradient [section 2.5]

Note that section 2.6 discusses the switch panel flangeway and thus is irrelevant for the crossing panel.

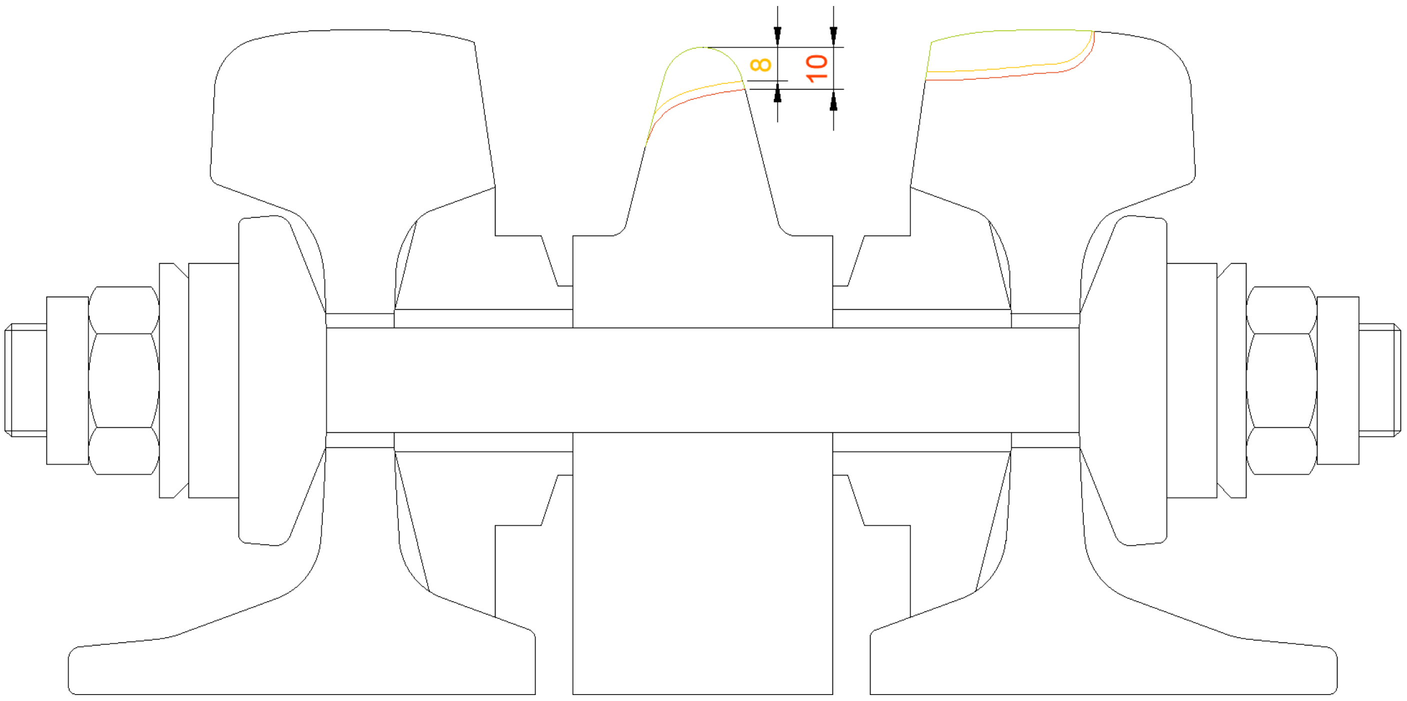


Figure 52 - Norms for vertical constructed crossing wear, to prevent flanges hitting the distance blocks [section 2.7]

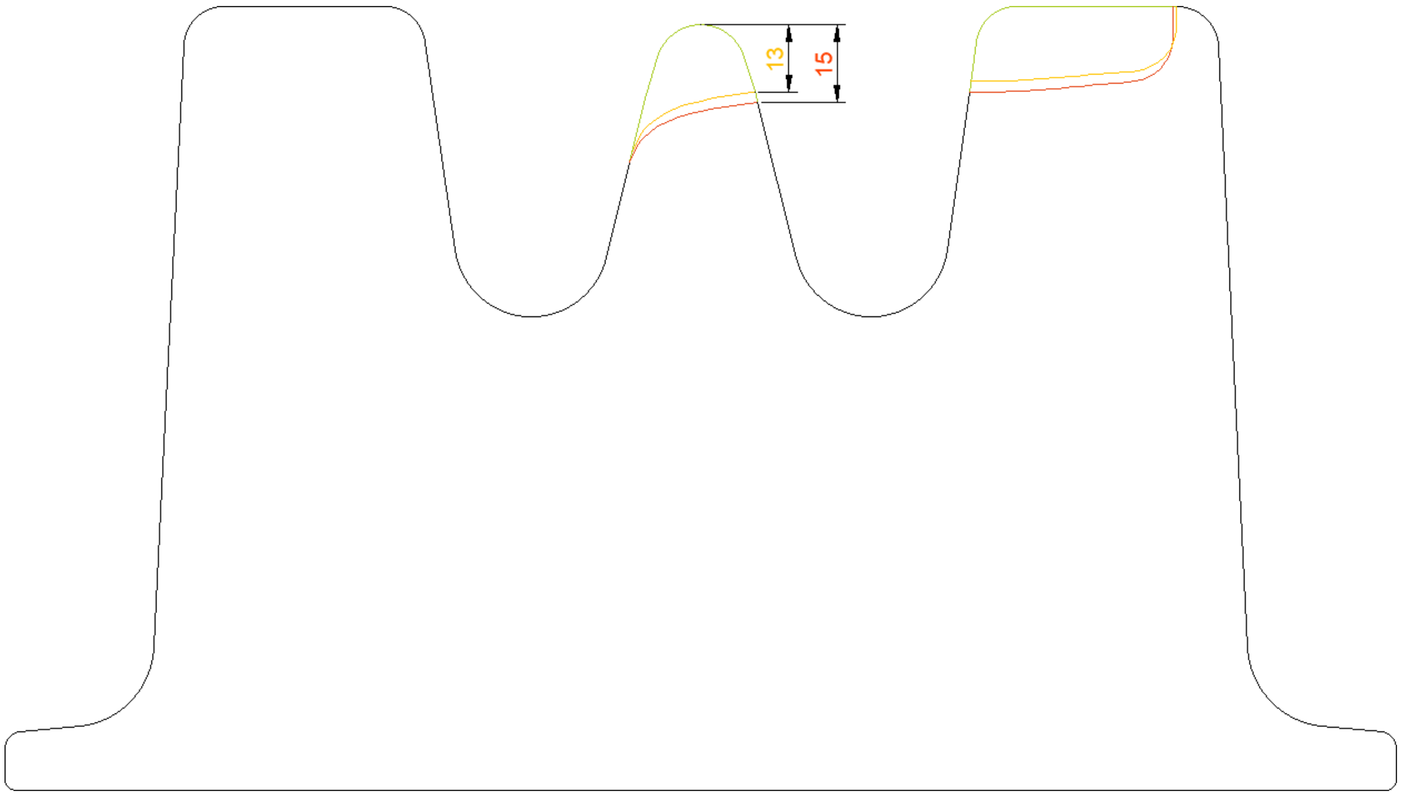


Figure 53 - Norms for vertical cast crossing wear, to prevent flanges touching the bottom of the flangeway [section 2.7]

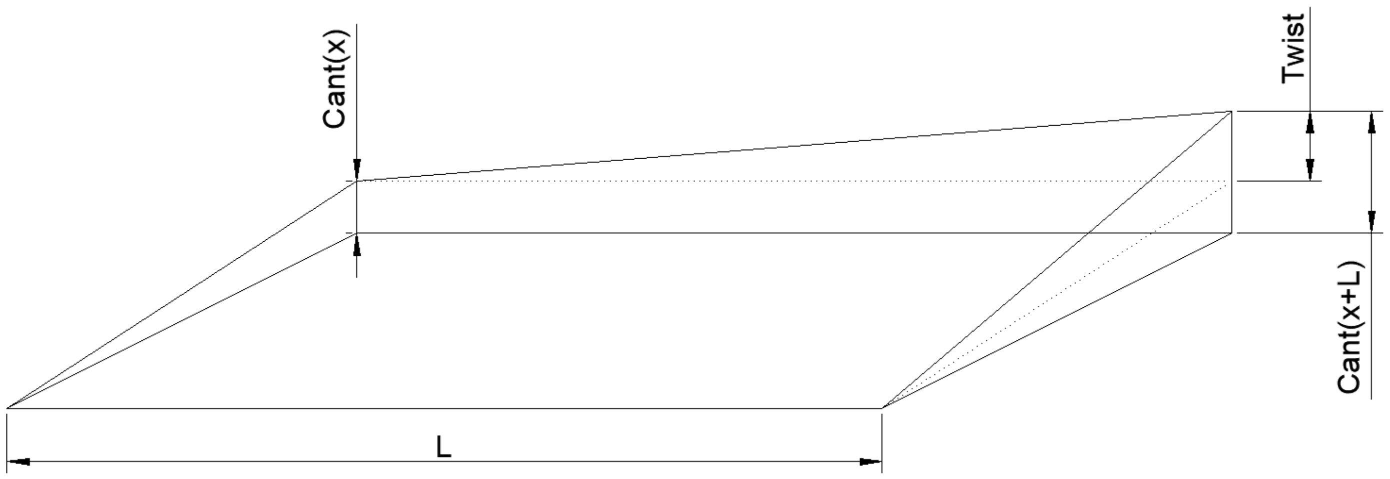


Figure 54 - The definition of Twist [27]

| Speed (km/h) | Base length (m) | BW norm (mm) | VW norm (mm) |
|---|---|---|---|
| ≤160 | 3 | 13 | 18 |
| ≤160 | 12 | 30 | 40 |
| ≤200 | 3 | $12^H$ | $16^H$ |
| ≤200 | 12 | $30^H$ | $40^H$ |

Table 1 - Norms for twist [section 2.8]

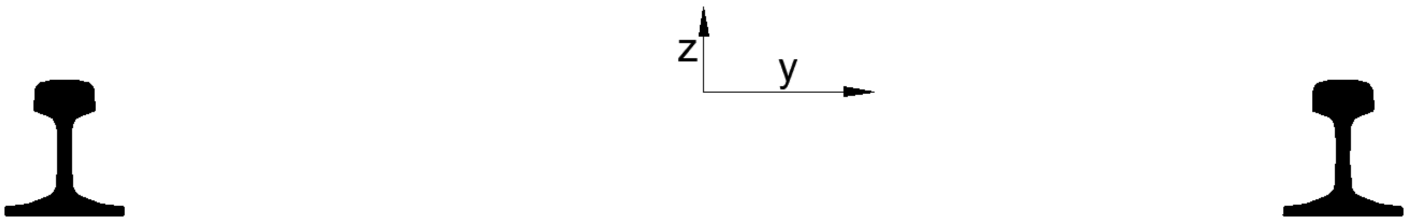


Figure 55 - Definitions for lateral (y) and vertical (z) track centre line deviations

| Speed (km/h) | Maintenance value BW (mm) | | Safety value VW (mm) | |
|---|---|---|---|---|
| | 1 m chord | 9 m chord | 1 m chord | 9 m chord |
| ≤ 40 | 3,7 | 22 | 5,0 | 30 |
| ≤ 80 | 3,0 | 13 | 4,0 | 18 |
| ≤ 160 | 1,8 | 7 | 2,4 | 10 |
| ≤ 200 | $1{,}5^H$ | $7^H$ | $1{,}6^H$ | $9^H$ |

Table 2 - Norms for maximum lateral deviations [section 2.9]

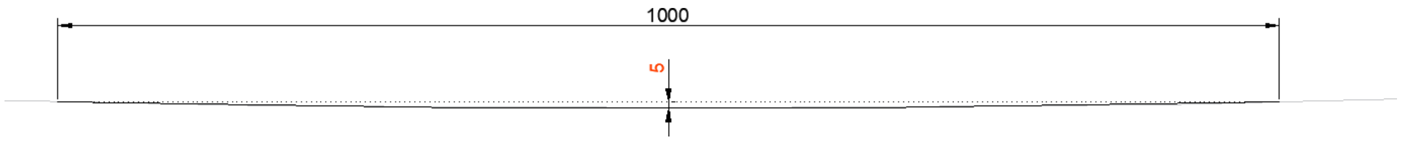


Figure 56 - Example of a lateral chord measurement: 1 m chord safety value for operating speeds up to 40 km/h. The lateral position of the track centre line is measured over a chord of 1 m, with a maximum deviation of 5 mm.

| Speed (km/h) | Maintenance value BW (mm) | | Safety value VW (mm) | |
|---|---|---|---|---|
| | 1 m chord | 10 m chord | 1 m chord | 10 m chord |
| ≤ 40 | 1,8 | 12 | 5,0[H] | 31[H] |
| ≤ 80 | 1,1 | 8 | 3,0[H] | 28[H] |
| ≤ 120 | 0,8 | 8 | 2,0[H] | 24[H] |
| ≤ 160 | 0,6 | 8 | 1,3[H] | 21[H] |
| ≤ 200 | 0,5[H] | 7[H] | 1,2[H] | 18[H] |

Table 3 - Norms for vertical deviations in the crossing panel [section 2.10]

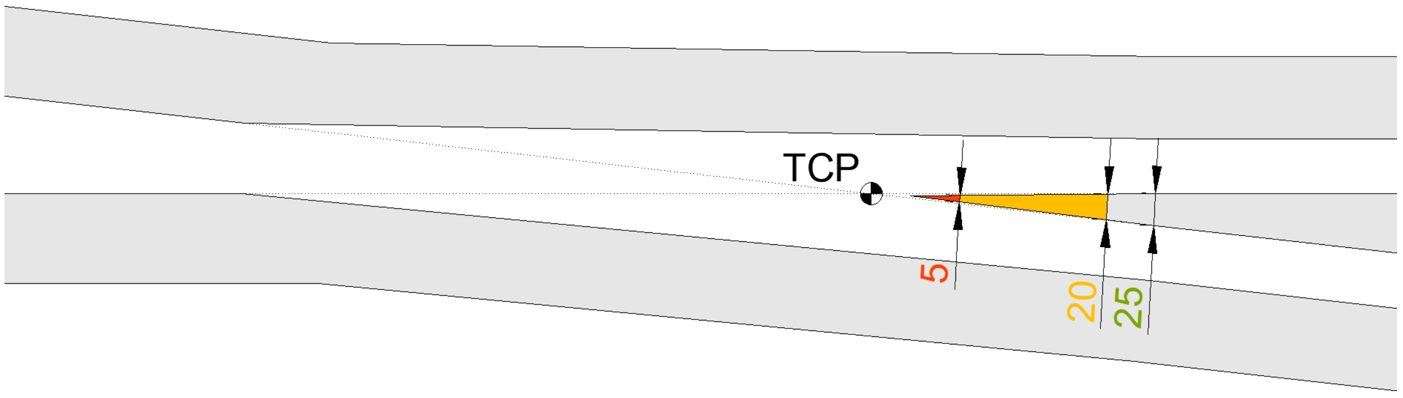


Figure 57 - Norms for the transition zone; the point where first contact is made on the nose [22]

OHD 33 (an old version of the norms) states that at the start of the transition point, the nose height is 4 to 6 mm below rail top. This is inconsistent with the designed height of 3.15 for cast crossings and in general the design of raised wing rail crossings. In newer versions, this requirement is absent.

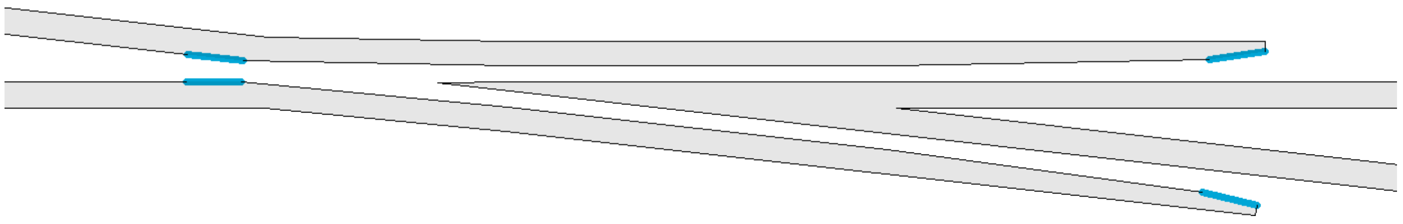


Figure 58 - Flange-back contact on the wing entrances and knuckle entrance (mentioned in [22] but not enforced)

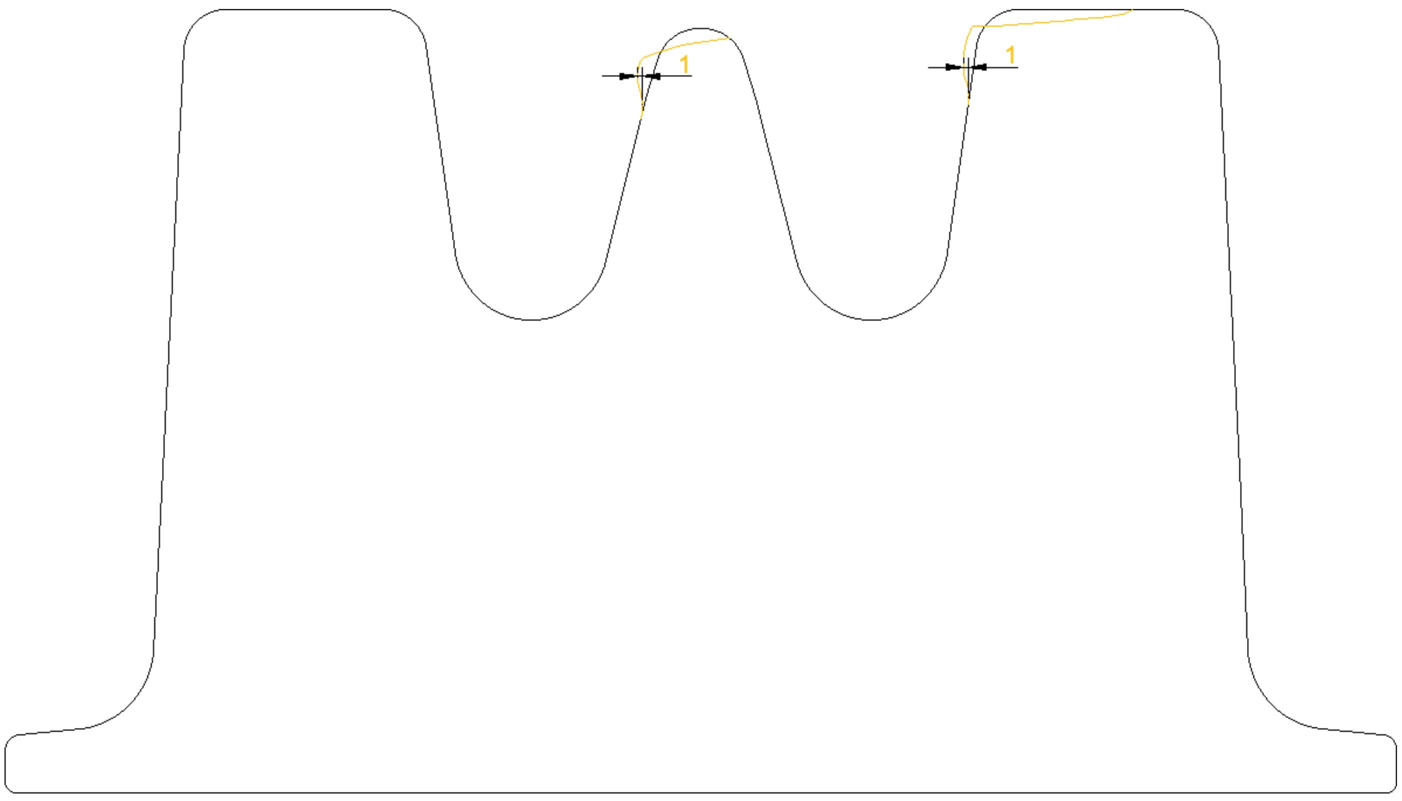


Figure 59- Norms for lipping [22]

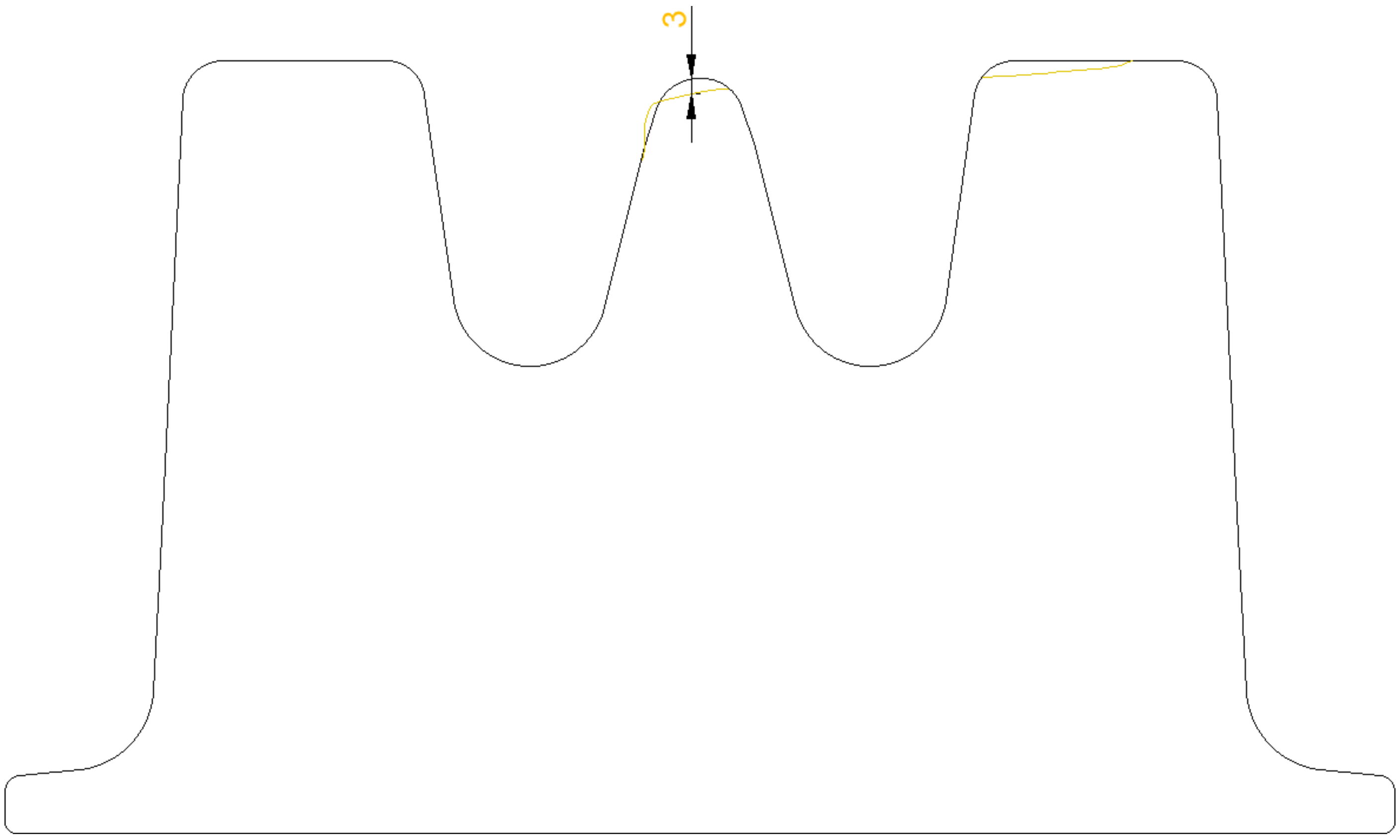


Figure 60 - Norms for vertical wear of wing and nose, around the transition zone [22]

## A2 RCF related norms

Everything concerning RCF can be found in RLN 399. The guideline consists of several parts. The first part [25] contains information about inspection regimes and classification of inspection results. Turnouts are inspected both by hand (Table 4) and by train (Figure 61).

| S&C in tracks with | Maximum inspection interval | Maximum exceedance |
|---|---|---|
| Operating speeds ≥160 km/h | 3 months | 3 weeks |
| UIC714 groups 1, 2 and 3 | 4 months | 4 weeks |
| UIC714 groups 4 and 5 | 6 months | 6 weeks |
| UIC714 group 6 | 12 months | 6 weeks |
| ≤ 40 km/h in regular timetable | 12 months | 6 weeks |
| ≤ 40 km/h not in timetable | Does not apply | Does not apply |

| Usage class | Maximum inspection interval | Maximum exceedance |
|---|---|---|
| A | 6 months | 6 weeks |
| B | 6 months | 6 weeks |
| C | 12 months | 6 weeks |
| D | Does not apply | Does not apply |

Table 4 - Ultrasonic hand inspection frequencies for turnouts. Highest frequency counts [25]

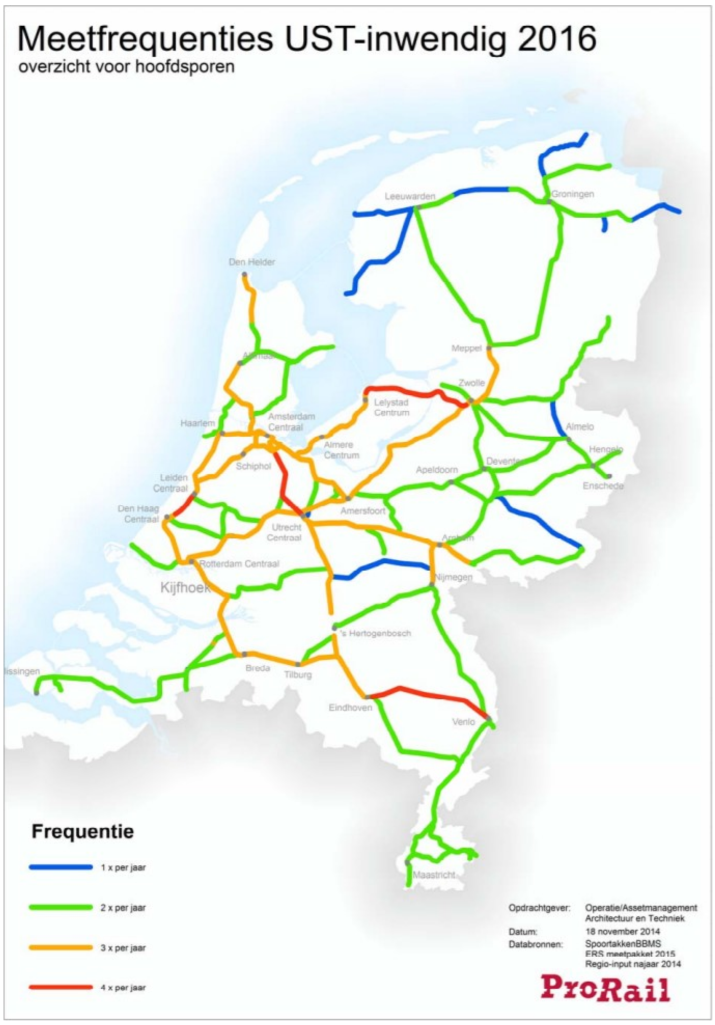


Figure 61 - Frequencies of ultrasonic train inspections [25]

# B Explanation of the software

## B1 Explanation per script

This section contains a thorough explanation of the Python scripts. The explanation starts with the executables (the place where the software is started) and goes on with deeper and deeper layers.

### B1.1 Executables

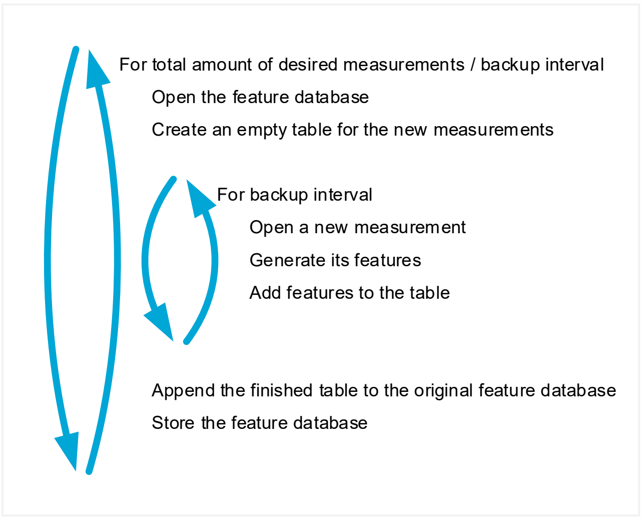
The scripts that are used to start running the code are all in the folder named `environment`. The default way to run the software is to execute `main.py`. Additionally, the scripts `side.py`, `side2.py` and `side3.py` have been created to run multiple parallel instances of the software in an easy way.

The executables have a few options. The most important ones are the `mode` and `wish_list` variables. The `mode` variable allows the user to specify what measurement to analyse: pick a random measurement, a specific asset number, an ideal measurement or the previously used measurement. The `wish_list` variable allows the user what features should be generated from the measurement (i.e. construction method and manufacturer). Features that are not selected are not calculated (and thus save calculation time).

### B1.2 Algorithms

The folder `algorithms` contains the so called pipelines; sets of tasks. For now, the folder only contains a feature generation pipeline, named `generate.py`. Therefore, all executables point to this pipeline. The structure of the feature generation script is shown in Figure 62.

The script consists of an inner loop and an outer loop, allowing the generated features to be backed up (in a .csv file) at a specified interval. This is convenient when generating features from a lot of measurements; the csv can be duplicated and the copy can be studied, while more measurements are being processed.



```
For total amount of desired measurements / backup interval
    Open the feature database
    Create an empty table for the new measurements

    For backup interval
        Open a new measurement
        Generate its features
        Add features to the table

    Append the finished table to the original feature database
    Store the feature database
```

Figure 62 – Pseudo code of generate.py

One piece of extra functionality is not mentioned in Figure 62. The `kick.py` script checks whether a measurement is in scope, before a measurement is accepted for feature generation. If the measurement is out of scope (not a 1:9 turnout, part of scissors crossover, etc.) the kick script causes the software to skip feature generation (and addition of the features to the table). This bring the software back to opening a new measurement.

In the future, other pipelines can be added to the `algorithms` folder. For example, a pipeline that aligns measurements without generating its features.

## B1.3 gather.py

In Figure 62 the step 'generate its features' is mentioned. This is done with the script `gather.py`. The structure of this script is described in Figure 63. In step a) some properties are gathered from the databases mentioned in section B4. Step b) creates an empty table, in which each row will contain a so-called feature vector; a list of properties that were derived from a measurement. Note that the loop behind that allows the creation of multiple feature vectors in one turnout. This is purely to facilitate single slip switches, double slip switches, diamond crossings and scissors crossovers in the future; these turnouts have multiple common and obtuse crossings (within one point cloud).

```
a) Get turnout properties
   • Unique turnout ID (equipment number)
   • Turnout drawing number
   • Crossing drawing number
   • Measurement filenames
   • Measurement vehicle direction (facing/trailing)
   • Crossing side (left/right with respect to vehicle)
   • Crossing tangent
   • Longitudinal locations of all TCPs in the turnout
b) Create an empty table for the feature vectors
   For all crossings in the turnout
      I) Open the point cloud file
      II) Crop the point cloud longitudinally, to the crossing panel
      III) Do a translation longitudinally, to shift the origin from switch blade to TCP
      IV) Align the measurement (and generate some features whilst doing that)
      V) Generate some more features
      VI) Plot and save some graphs
      VII) Store all generated features for this crossing in a row in the table
c) Return the full table with feature vectors
```

Figure 63 – Pseudo code of gather.py

Step I) turns the CSV file in to a Pandas DataFrame (a tabular format from the Pandas library, which is described in B3.1). Step II) immediately makes use of the functionality of Pandas; all rows with a longitudinal coordinate outside the crossing panel are removed from the DataFrame. This simplifies the point cloud and speeds up later processes. Step III) shifts the origin of the point cloud longitudinally; the initial point cloud uses the switch blade tip as $x_0$ and the shifted point cloud uses the crossings TCP as $x_0$. The shift is purely to make calculations more convenient (i.e. easier calculation of nose width, through $width = x \cdot crossing\_tangent$). In step IV) the point cloud is aligned vertically, laterally and in terms of roll (Figure 11). This step is discussed in chapter 2 conceptually and B1.4 describes the steps / math in the scripts. Note that both step IV) and step V) contribute to the feature vector. This is purely because of performance (speed). Step IV) already needs to loop through the point cloud. Using that loop for some feature generation steps saves time. Step V) is discussed in chapter 3.

```
a) Repair upside-down measurements
   • If the centroid of the rail has z > 0 flip vertically (z = -z)
b) Repair the lateral alignment
   • Check whether the stock rail is visible at y <852.5 and z<40
       True: look at 772.5 < y < 852.5
       False (outlier): look at 852.5 < y
   • At the specified lateral positions:
       Find the y coordinate, closest to z =14
       Find the y coordinate, closest to z =20
       Find the y coordinate, closest to z =25
       Find the y coordinate, closest to z =30
       Find the y coordinate, closest to z =35
   • If the slope of the stock rail back is steeper than 1:10
       True (good stock rail back visibility): use y14
       False (noisy stock rail back): interpolate from neighboring cross sections
   • Translate laterally, to make
       y = y + 787.5 - stock_rail_back
c) Repair the roll
   • Find the highest point on both the left and right wing rail
       Look in lateral windows in which the wings are expected
       Take the highest point (but remove outliers!)
   • Determine whether the highest points are local maxima
   • Remove points that aren't and interpolate those, by using
     both the other wing rail and lateral neighbors
   • Determine the required rotation, based on both wing tops
   • Shift y0 from track center line to the center of rotation
   • Apply a rotation matrix on the cross section
   • Shift y0 back to the original position
d) Compensate for raised wing rails
   • For each cross section shift vertically, based on the raised wing design
```

Figure 64 - Pseudo code of the alignment algorithm

## B1.4 repair.py

The script `repair.py` aligns measurements in such a way that the original (unworn) rail top is at $z_0$ and that the calculation of the track centre line $y_0$ takes the unguided opening of the crossing in to account. The structure of this script is shown in Figure 64. Step a) was necessary, because sometimes measurements are delivered upside down. Step b) consists of lateral translations, that are determined by the lateral coordinate of the stock rail back. Note that this coordinate is determined on 5 different heights and checked for really being a ~1:20 slope (like can be expected). Figure 16 shows an example of a measurement where this check is needed.

Step c) is the most complicated step. The challenge was not to rotate the cross section, but to validate the wing rail tops. The algorithm first checks whether the highest point on the wings is an outlier (relatively far away from other points) and then checks whether the point is a local maximum. Local maximum is validated by verifying that the point has close neighbours that are lower; being the highest point on a slope that ends in the shadow is not good enough. Figure 65 visualizes how the algorithm creates visibility areas (black) and checks whether the highest wing point is at the edge of the visibility (and shadow). Points that are invalid are interpolated with the aid of longitudinal neighbours and with the aid of the other wing rail.

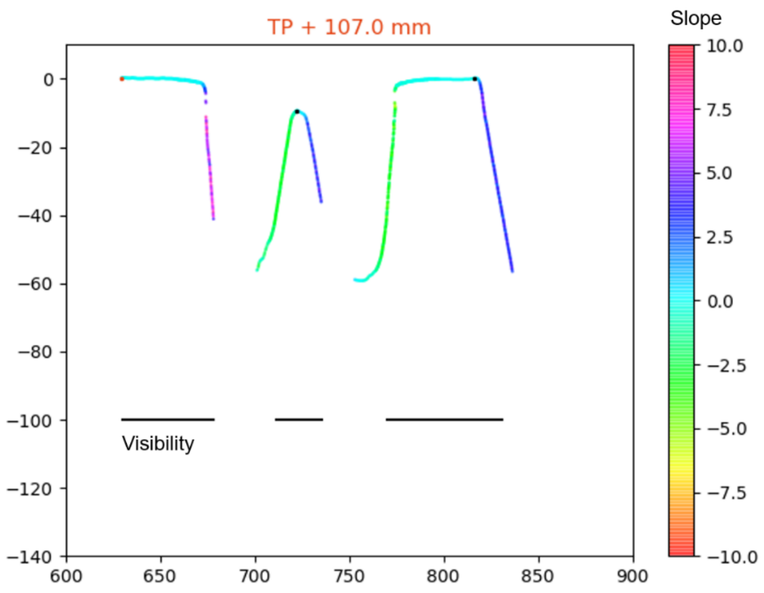


Figure 65 Detection of wrong (left) and good (right) wing tops

Per cross section, the slope between the left and right top-of-wing is determined. This slope is input for a rotation matrix. The math per vector is:

$$y_{new} = y_{old} \cos\varphi - z_{old} \sin\varphi$$

$$z_{new} = y_{old} \sin\varphi + z_{old} \cos\varphi$$

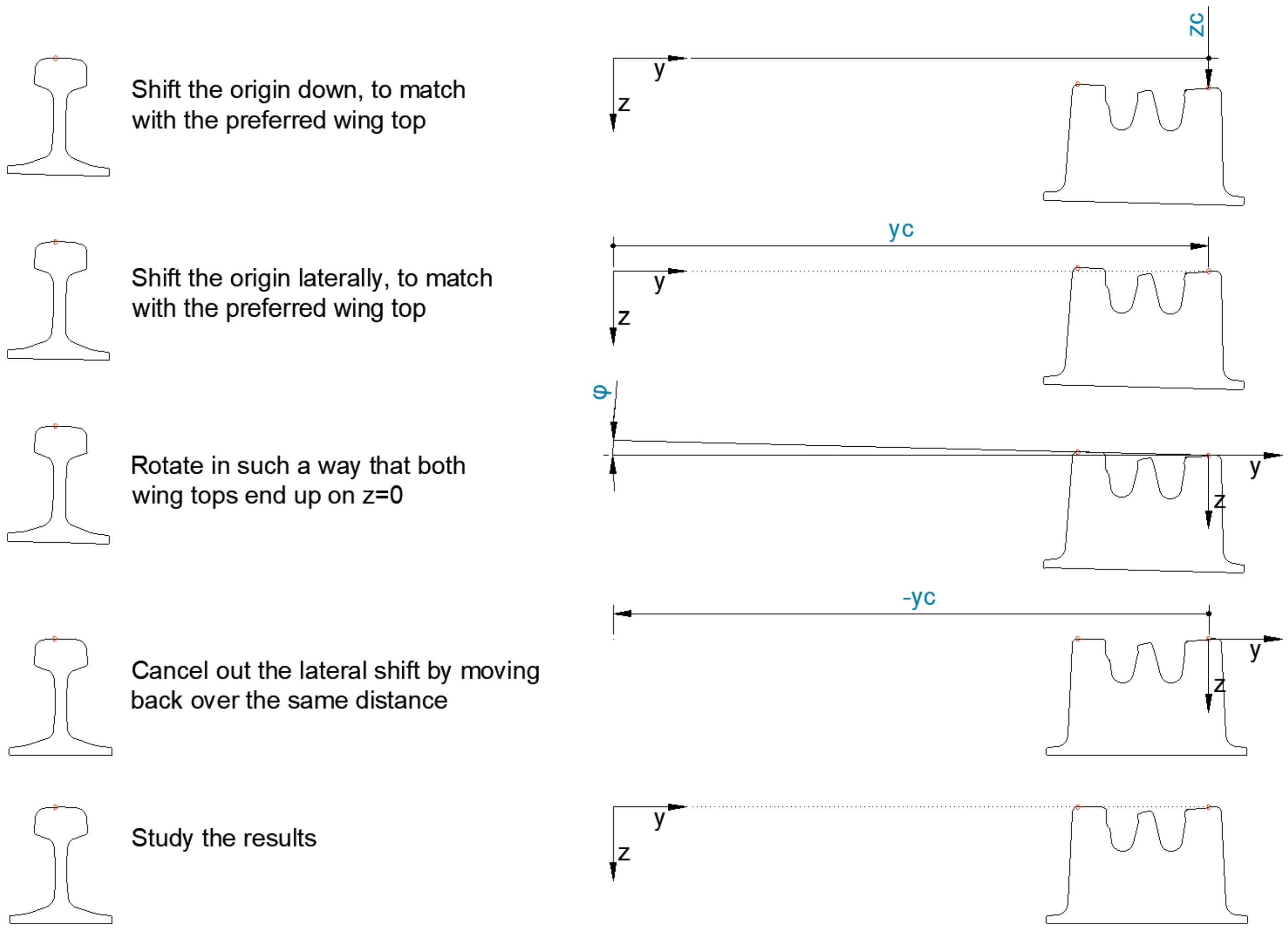
Note that the algorithm shifts the origin laterally back and forth, around the rotation step (Figure 66). This is to rotate around the crossing, rather than around the track centre line.

The last step is to compensate for raised wing rails. Two algorithms determine whether the crossing is cast or constructed (classifier based on wing rail width) and whether it has raised wing rails or not (classifier based on the presence/absence of web plates). If the crossing is a raised wing rail crossing, each crossing is moved upwards based on a lookup table. The lookup table contains the height (above rail top) of the raised wing rails for each longitudinal location. The table was based on an analysis of the crossing design in AutoCAD (Figure 27).

## B1.5 Classification of crossing type and manufacturer

Two of the classification algorithms (classifiers) used by `repair.py` require some additional explanation. The classifiers determine whether the crossing is cast or constructed (step 1) and whether constructed crossings were produced by manufacturer Vossloh Cogifer Kloos BV (Kloos) or Voestalpine Turnout Technologies Netherlands BV (TTN) (step 2).

The first step is based on a difference between cast and constructed crossings that is often visible in the measurement, not subjected to wear/deformation and clear enough to detect reliably. This difference was found in the width of the wing rail, at 1000 to 500 mm before TCP. The theory behind this is shown in Figure 67. In this area, constructed crossings should have a wing width of 70 mm whilst cast crossings are designed with a width of 84 mm. Note that this is only verified for Dutch 1:9 UIC54 common crossings; other crossing designs might differ and thus need a different classifier. Figure 68 shows how some wing width patterns from cast and constructed crossings were studied to verify the theory. After implementation of the algorithm, the classifier was further verified by manually comparing classifications with cross sections. The classifier now delivers very accurate results with no manual intervention.



Shift the origin down, to match with the preferred wing top

Shift the origin laterally, to match with the preferred wing top

Rotate in such a way that both wing tops end up on z=0

Cancel out the lateral shift by moving back over the same distance

Study the results

Figure 66 - Graphical representation of the rotation algorithm

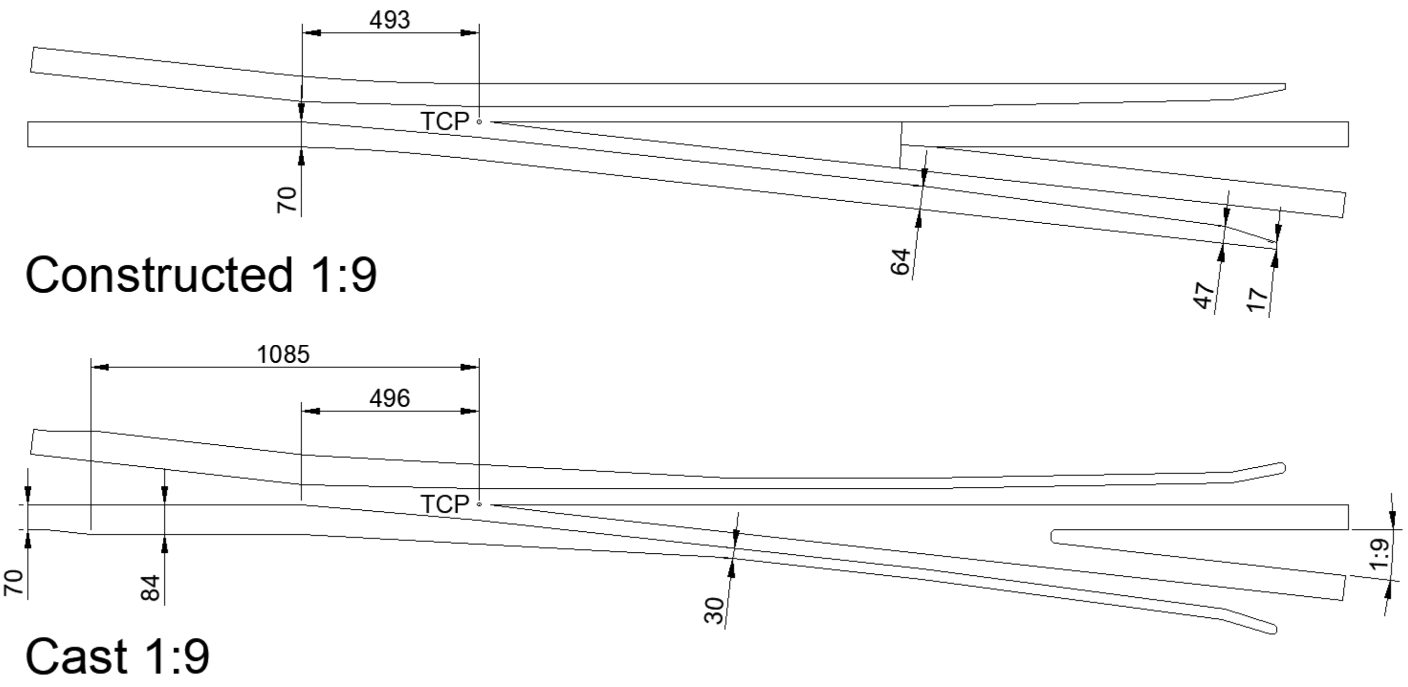


Figure 67 - Theory behind the longitudinal search area of the construction method classifier

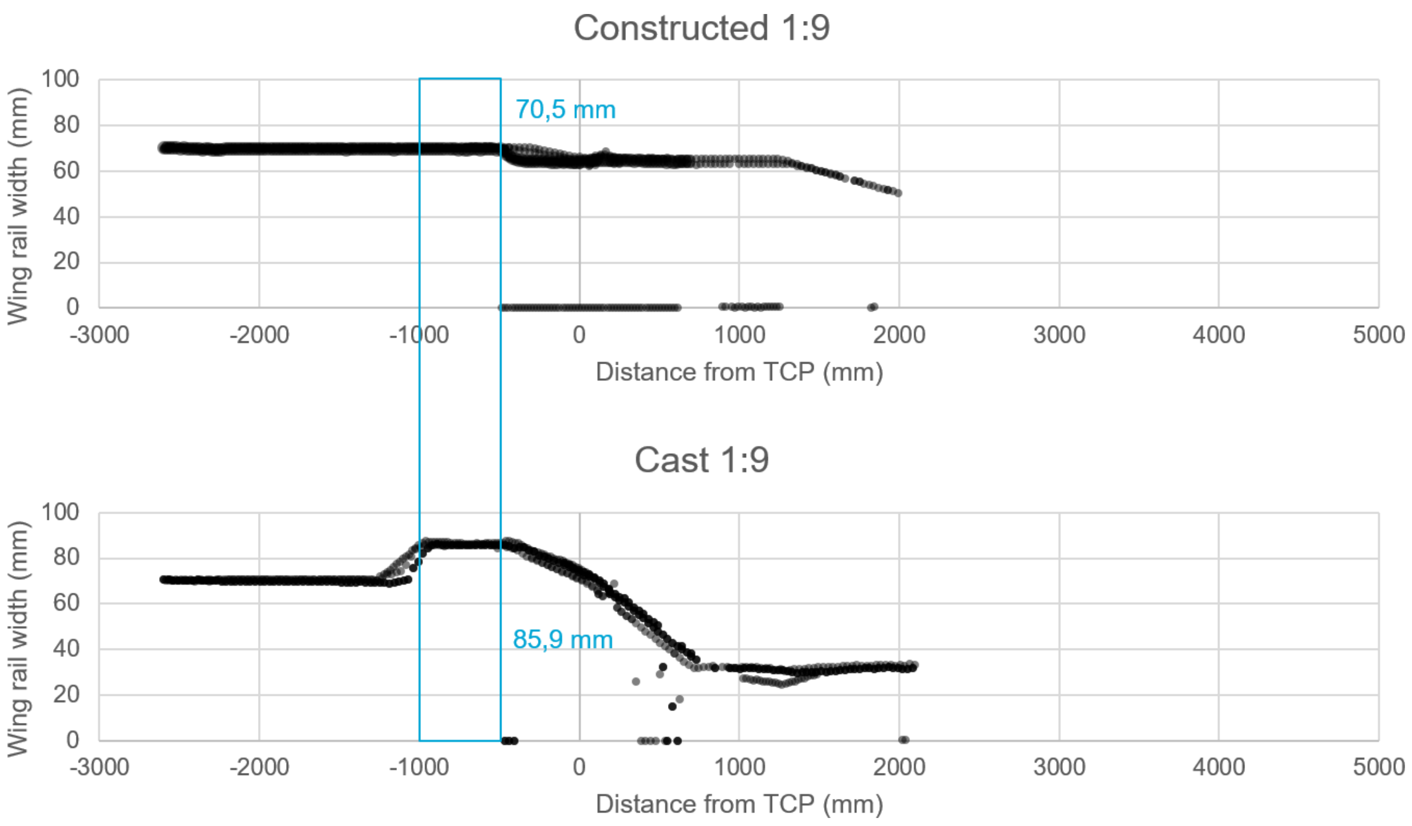


Figure 68 - Initial test dataset for the construction method classifier (average values in search area in blue)

If the crossing is classified as constructed in step 1, a second classification step is performed. Whilst cast crossings of different ages/manufacturers have very similar designs, constructed crossings don't. Crossings from Kloos have a raised wing rail, which needs a different assessment method than the regular wing rails from TTN. These designs are classified by determining whether the crossing features blocks that reinforce the outer webs (Figure 69, in blue). The classifier does this by calculating the difference in lateral position at 14 and 90 mm below the rail head. Those differences should be 2.1 mm for Kloos and 27 mm for TTN. Note that both crossings feature through bolts; horizontal bolts that keep wings, distance blocks and nose together. These bolts disturb the classifier; negative values will be found at the location of the bolts. This issue is solved by involving as many samples as possible (both wing rails, between TCP and TCP + 1010 mm).

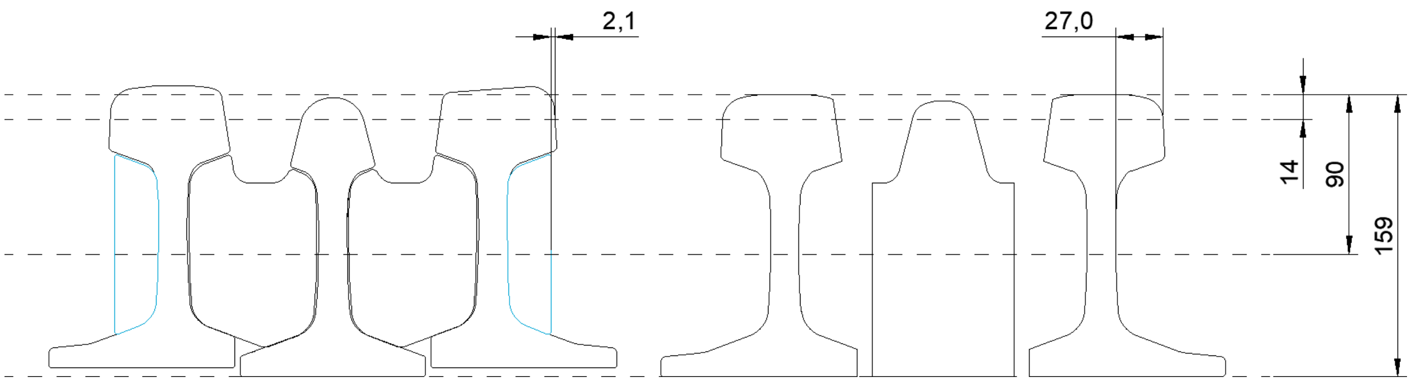


Figure 69 – Theory behind the algorithm that classifies the manufacturer of constructed crossings

## B1.6 Fitting UIC54 rail head profiles

The `repair.py` script also makes use of a UIC54 fitting algorithm. The `measure.py` script has a function called `_fit_uic54`. Note that the function name starts with a _ to indicate that it should only be used within the `measure.py` script (more information in section B2.2). The `_fit_uic54` is used to find the lateral position of a UIC54 rail head. Its most important steps are shown in Figure 70. The function needs an estimate of the lateral position of the rail (red) and measured rail profile (black). The measured rail may be milled, worn and deformed (as shown in the example). Moreover, the measurement only shows a part of that rail.

In step 1 of Figure 70 a UIC54-shaped template (yellow) is added above the rail, with a centre line (blue) 5 mm left of the estimated centre line (red). Note that the estimated centre line differs from the true centre line (green). In step 2, the template is lowered on to the rail until in touches the rail. In step 3, an area is created between the template and 0.5 mm below the template. The area is searched for point cloud points and each point cloud point in the area is awarded with a score up to 0.5. Points that exactly match the vertical location of the template get a score of 0.5. Points that are 0.3 mm off the template receive a score of 0.2. Points that are 0.4 mm off receive a 0.1 score, etc. The score of all point cloud points in the area is summed and the algorithm continues to step 4. Step 4 is the same as step 1, but the template now has an offset of 4.9 mm left of the estimated centre line. With this 4.9 mm, steps 2 and 3 are also repeated and the combined score of this offset is stored. This process repeats itself in steps of 0.1 mm, till the template centre lies 5 mm right of the estimated centre line. The offset with the highest score is used to determine the best estimate of the lateral position of the centre line of the rail.

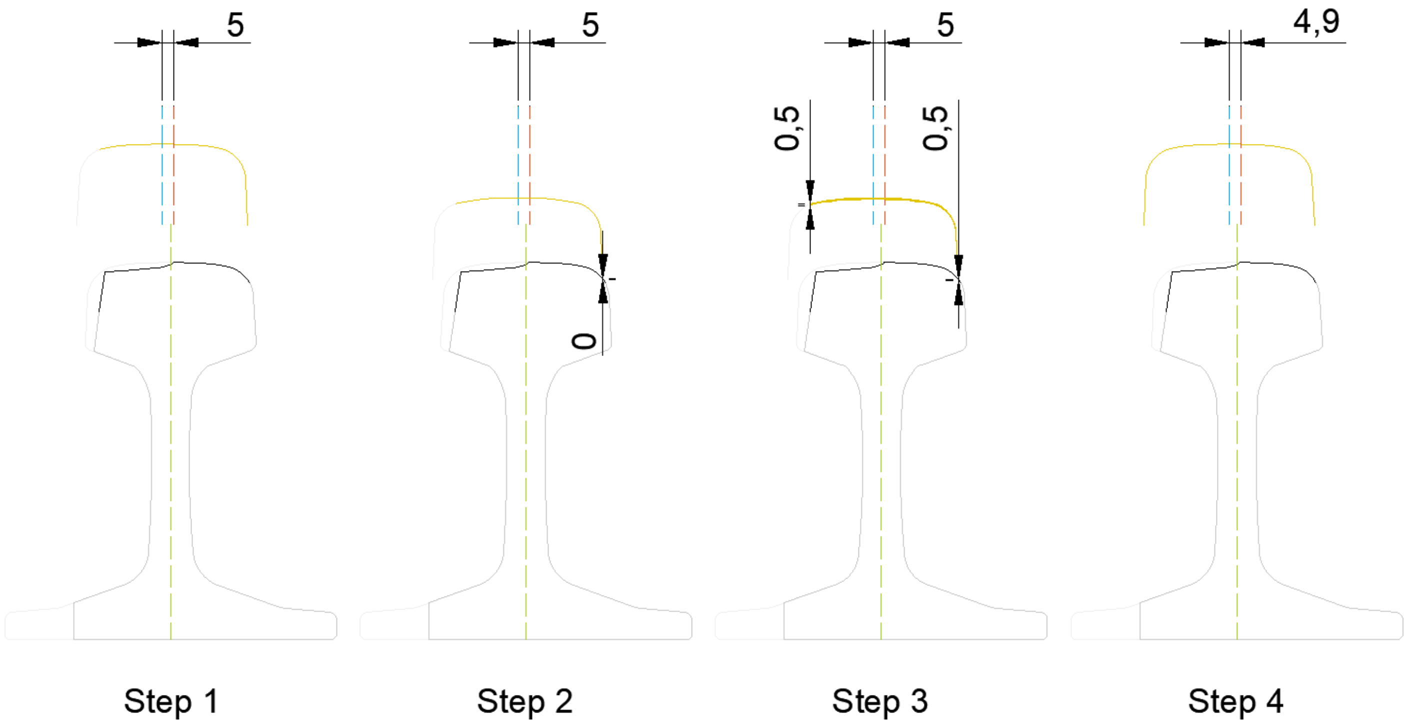


Figure 70 - Visualization of some of the steps in the measure._fit_uic54 function

## B1.7 Clustering the faces of the crossing

To calculate certain properties of a rail, it is useful to divide it in several sides (faces). These faces can then be named (front, top, back) and characterised; with parameters like orientation (slope). The division of the faces is done with a clustering algorithm. Clustering algorithms try to categorize (classify) observations (in this case: point cloud data points) based on one or multiple properties (features) of those observations. Good features have good separability; for each category feature values have little to no overlap. To cluster faces of a rail, local slope is used as the sole feature to distinguish faces. The local slope $dz/dy$ in point $i$ is calculated as the slope of the line between point $i$ and point $i-1$:

$$\frac{dz}{dy}_i = \frac{z_i - z_{i-1}}{y_i - y_{i-1}}$$

Additionally, for this example, the inverse of this slope $dy/dz$ is calculated:

$$\frac{dy}{dz}_i = \frac{y_i - y_{i-1}}{z_i - z_{i-1}}$$

Calculating the slope and inverse slope for all data points of the cross section in Figure 71 on the left, would result in a plot like Figure 71 on the right. With the criterion $\text{abs}(dz/dy) < 1$ it can be determined whether a face more vertical-like or more horizontal-like (the criterion lays the boundary at $dy = dz$, which is where the local slope is $45°$). The criterion is not enough to cluster however. The algorithm contains some extra rules on how to handle noise, how many faces should be created, how to handle invisible faces and how to name the faces.

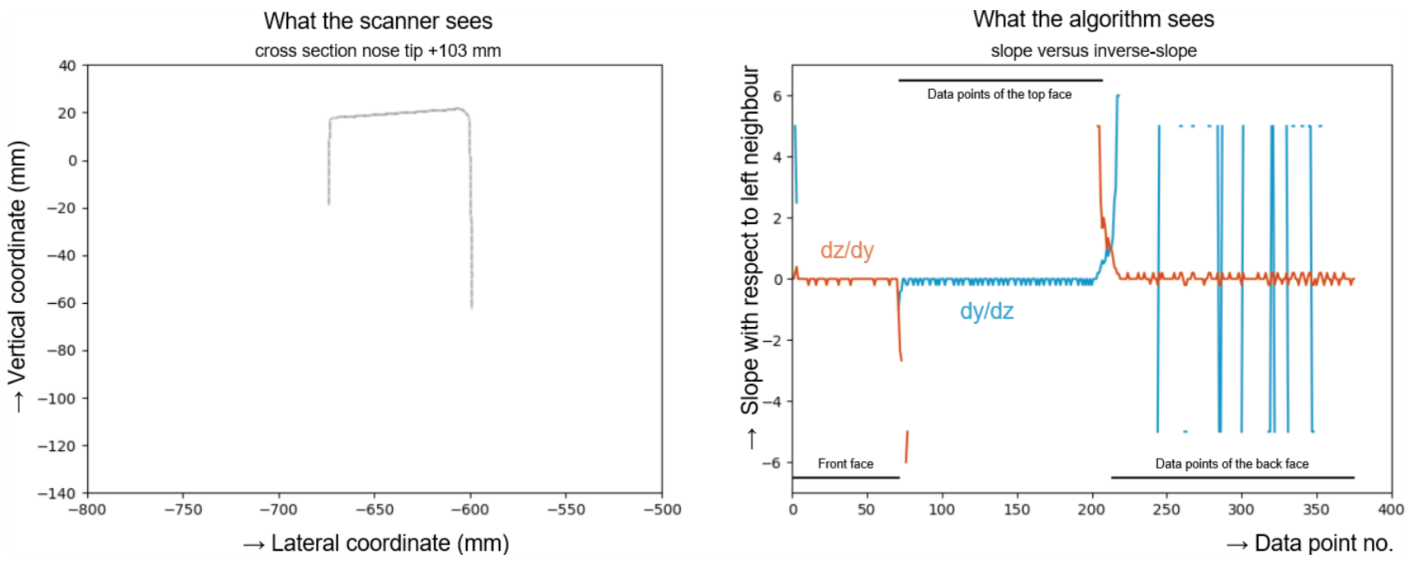


Figure 71 - Explanation of the face clustering algorithm for check rails

## B2 Environment

### B2.1 Language

At the start of this project, the author only had experience with coding in Matlab. However, choosing Matlab for this project would have led to complications. ProRail doesn't have and aspire Matlab licences and many colleagues use Python (https://www.python.org/). On top of that, the Python language has a better eco system; there are more libraries and guides available. For these reasons, Python was chosen instead of Matlab (which made the project start with some online Python courses).

### B2.2 Style

Code styling is a list of layout conventions. Examples are: where to put empty lines / spaces and in what order should a script be organised. The styling of the code has been done in accordance with the PEP8 standard. This is the most commonly used python style and can be found on https://www.python.org/dev/peps/pep-0008/.

### B2.3 VCS

The codebase initially started without a Version Control System (VCS). The Google Drive backup function did store some older versions on itself, but this was no real VCS. The benefits of a real VCS are numerous, but most importantly it allows to write a note (commit message) each time you submit some changes (commit) to the repository. On the 5th of May 2020, the codebase was uploaded to a private GitHub repository.

This served as the VCS up till the 12th of February 2021. At that date, the repository was transferred from the private GitHub repository to a Microsoft Azure. ProRail uses Azure Repos as a central location for repos. Moving the repository to Azure prepares it to use live data in ProRails Big Data Analysis Platform (BDAP), instead of manually downloaded local copies. Moreover, it allows the repo to be reviewed by colleagues. A timeline of the VCS solutions is shown in Figure 72 and the repo is nowadays available through https://dev.azure.com/ProRail/AssetDegeneratie/_git/Puntstukken.

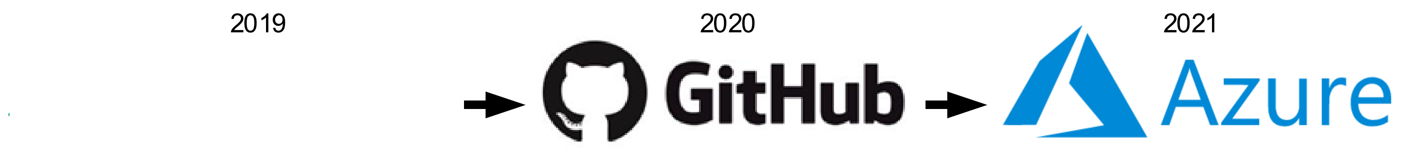


Figure 72 - Timeline of the VCS solutions

### B2.4 Contributions

The Azure repository is edited with the fork and pull method (Figure 73). This is a method to allow (in the future) multiple people to work on the same repository.

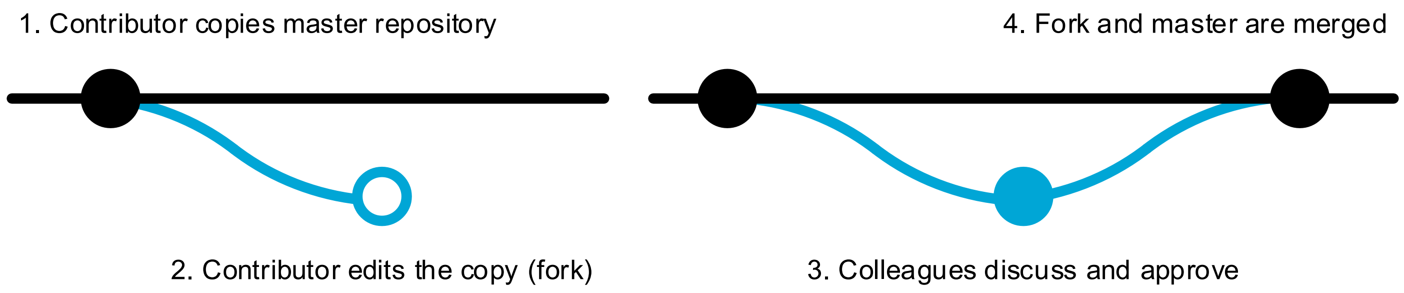

1. Contributor copies master repository
4. Fork and master are merged
2. Contributor edits the copy (fork)
3. Colleagues discuss and approve

Figure 73 - Fork and pull procedure

### B2.5 IDE

An Integrated Development Environment (IDE) is the editor in which code is developed. Some of the functions are highlighting certain parts of the code, helping with code styling, communicating with the VCS, incorporating the interpreter to run the code and showing graphs. The IDE used for this project is PyCharm, which is commonly used and most suitable for larger projects (another commonly used alternative Jupyter Notebook would be more suitable for smaller projects).

### B2.6 Interpreter

An interpreter handles versions. Mainly, it selects the right version of the programming language. On top of that, it makes sure all libraries are accessed with the right version. For this project, Anaconda was used.

## B3 List of used libraries

If a certain function is used very often (i.e. matrix multiplication), it is probably available in a public library. In such a case, it is better to use a library than to write the function yourself. Commonly used libraries are often more thought-through (more efficient/faster and more reliable). On the other hand, libraries that are custom tailored for very specific applications are often less reliable and only a few people know how to work with them.

Since the start of the project, the aim has been to keep the codebase reliable and maintainable. This is reflected in the choices for libraries. As much as possible, widely-used libraries have been chosen. Often it can be just a bit easier/faster to use some more custom libraries, but this has been avoided.

### B3.1 Pandas

The project relies heavily on the pandas library. Pandas is one of the most widely used data handling packages. Moreover, ProRails DataLab department uses it too, which helps with maintainability.

The library can be seen as a way to store tabular data. It is able to handle all kinds of classes in the cells and allows naming of columns and rows. These tables can be analysed swiftly, with various selection and editing functions. The project started with Pandas 0.25.1 and the newest version (whilst writing this thesis) is 1.2.4.

### B3.2 Matplotlib

All visualizations are done through the Matplotlib library. Matplotlib is the most commonly used plotting library and thus easy to maintain. Many people know how to work with it and manuals are numerous. The project started in version 3.1.1 and the newest version is 3.4.1.

## B3.3 Full list

| Package | Version | Latest version |
|---|---|---|
| blas | 1.0 | 1.0 |
| ca-certificates | 2020.12.8 | 2021.4.13 |
| certifi | 2020.12.5 | 2020.12.5 |
| cycler | 0.10.0 | 0.10.0 |
| ffmpeg | 4.2.2 | 4.2.2 |
| freetype | 2.10.4 | 2.10.4 |
| icu | 58.2 | 68.1 |
| intel-openmp | 2020.2 | 2020.2 |
| jpeg | 9b | 9b |
| kiwisolver | 1.3.0 | 1.3.1 |
| libpng | 1.6.37 | 1.6.37 |
| libtiff | 4.1.0 | 4.2.0 |
| lz4-c | 1.9.2 | 1.9.3 |
| matplotlib | 3.3.2 | 3.3.4 |
| matplotlib-base | 3.3.2 | 3.3.4 |
| mkl | 2020.2 | 2020.2 |
| mkl-service | 2.3.0 | 2.3.0 |
| mkl_fft | 1.2.0 | 1.3.0 |
| mkl_random | 1.1.1 | 1.1.1 |
| numpy | 1.19.2 | 1.19.2 |
| numpy-base | 1.19.2 | 1.19.2 |
| olefile | 0.46 | 0.46 |
| openssl | 1.1.1i | 1.1.1k |
| pandas | 1.1.3 | 1.2.4 |
| pillow | 8.0.1 | 8.2.0 |
| pip | 20.3.1 | 21.0.1 |
| pyparsing | 2.4.7 | 2.4.7 |
| pyqt | 5.9.2 | 5.9.2 |
| python | 3.8.0 | 3.9.2 |
| python-dateutil | 2.8.1 | 2.8.1 |
| pytz | 2020.4 | 2021.1 |
| qt | 5.9.7 | 5.9.7 |
| setuptools | 51.0.0 | 52.0.0 |
| sip | 4.19.13 | 4.19.25 |
| six | 1.15.0 | 1.15.0 |
| sqlite | 3.33.0 | 3.35.4 |
| tk | 8.6.10 | 8.6.10 |
| tornado | 6.1 | 6.1 |
| vc | 14.2 | 14.2 |
| vs2015_runtime | 14.27.29016 | 14.27.29016 |
| wheel | 0.36.1 | 0.36.2 |
| wincertstore | 0.2 | 0.2 |
| xz | 5.2.5 | 5.2.5 |
| zlib | 1.2.11 | 1.2.11 |
| zstd | 1.4.5 | 1.4.9 |

Table 5 - Full list of used libraries and their versions

## B4 List of used databases

The project makes use of several datasets. All these dependencies are mentioned in this section. Note that the databases are all local copies; an export at a certain moment in time, from the live database. ProRail is currently working on live database connections, in the BDAP environment. In February 2021 this project was moved to that location, to prepare it for live database connections in the future.

### B4.1 Geometry measurements

The main database contains switch & crossing geometry measurements and is named WLGEOM_T. The point clouds. The database can be viewed through https://bbms.prorail.nl/. The editable version of the data (a BBBMS export) was provided by ProRails Inframonitoring department, in May 2019 (one month after starting this project). It was delivered as an export of the main database, on a hard disk (353 GB of data). The export contained over 35.000 measurements, since the 3rd of August 2016 and up till the 16th of May 2019.

Each measurement consists of a .zip file and contains five .csv files: profiel, meta_header, meta, discreet and continue. The profile file mainly contains the point cloud data; X, Y and Z coordinates. The two meta files contain extra information, like measurement company and measurement vehicle. The discrete file contains discrete properties of the measurement, like (measured) length of the turnout. The continuous file contains continuous (measured) properties, like gauge.

### B4.2 Object database

All of ProRails assets are registered in its object database. ProRail uses the SAP software package and publishes the database through https://informatieportaal.spoordata.nl/. All turnouts and crossings have their own object code on this database, which allows coupling with the geometry measurements.

### B4.3 Technical drawings

All turnout and crossing objects refer to a specific technical drawing number. These are published on the Rail Infra Catalogue (RIC) through https://prorailbv.sharepoint.com/sites/ric. The crossing designs have been parametrized; parameters like crossing length and nose profile were transferred to a tabular format (.csv) to allow the scripts to look up drawing properties.

### B4.4 Traffic load

ProRails axle counters are coupled in ProRails TROTS system. This network generates a list of traffic loads per track and a list of traffic loads per turnout. The turnout traffic load database Wisselberijding contains parameters like tonnages, traffic types and amounts of axles. The parameters are available per traffic directions (facing/trailing, straight/deviating). This dataset is available (both per month and per year) through https://prorailbv.sharepoint.com/teams/T2017_0058/bieb1/Forms/AllItems.aspx. It was used for the case study of multiple measurements.