Unsupervised analysis of track degradation in tramway transition zones using InSAR satellite data

Student Name: Hoedjes, T.M. (Tijmen)
Student number: 4959183

Master thesis draft report submitted in partial fulfillment of the requirements for the Degree of MSc. Civil Engineering on 28-04-2025

Specialisation Track: Traffic and Transport Engineering

Chair dr. Núñez Vicencio, A.A. (Alfredo)
Main supervisor dr. Anupam, K. (Kumar)
Company supervisor Ir. Peteroff, N. (Noa)
Company supervisor Drs. Huijsmans, M. (Michel)

Delft University of Technology
Delft, The Netherlands

Abstract

Tramway infrastructure plays a key role in transporting large numbers of people in various cities worldwide. However, ensuring the quality and operation of the tram system is not easy because the tramway infrastructure is sensitive to degradation and faces different behaviours at different locations. For instance, transition zones are areas where local stiffness changes can cause a high track geometry degradation due to traffic loads and track settlement. To be well-informed about the health conditions of the infrastructure, structural health monitoring and maintenance are crucial for the inframanager. The literature has proposed that standard deviations of track gauge can indicate high geometry deterioration. However, geometry measurements are typically not conducted frequently, opening a gap in the lack of information on critical locations. In the field of conventional railways, using InSAR satellite data has been reported to be able to provide frequent data and to detect high settlement rates leading to high rates of track degradation. In this thesis, inspired by the experiences in the railways, an evaluation of InSAR satellite data will be conducted to perform a feasibility analysis for the case of transition zones in tramways.

This thesis focuses on predicting high settlement rates at transition zones using InSAR satellite data. Because InSAR data shows more frequent deformation data, it can potentially detect earlier high settlement rates or allow a more frequent data update of critical locations without requiring dedicated measurement trains. While geometry is the core in this thesis, InSAR could also be used for other applications such as vegetation control. In the case study, validation data with structural analysis and alternative means of condition measurement were unavailable. Thus, the development of an unsupervised approach that allows a first analysis to facilitate the exploration of the characteristics of the data and its relation with infrastructure has been considered.

Three transition zones of the tramway track in Amsterdam have been selected as case studies. The InSAR data points that are closer to the transition zone are extracted to use prediction methods. The coordinate position of these data points faces uncertainties. Therefore, different regions have been considered, and a method to obtain an area using the Pearson Correlation Coefficient from InSAR and GVB geometry data has been proposed. Then, two methods to predict high settlement rates have been proposed. The first method is the stochastic rates method, which uses half-year InSAR data settlement rates to indicate if a rate is higher than the historical rates. The other method uses data from different years to show a pattern and looks for dips to indicate these as high settlement rates. Finally, the method proposes using different data sources to enhance understanding of health conditions in transition zones. Speed plots were used to showcase the possibility of data fusion and correlation analysis, as driver behaviours at transition zones could be an interesting parameter to analyse.

In the results, the measured standard deviation of the track gauge of data points on the bridge was usually lower than the standard deviation of the track gauge of data points not on the bridge. Almost the same results were returned using the same principle with twist data. The uncertainty of both approaches has an order of 10^{-2}, so they have similar performance. The stochastic rates method gives more precise and appropriate results than the prediction method based on BLUE. The two approaches have also been tested with a standard transition length of 10 meters at the Piet Wiedijkstraat and 20 meters for the other locations, instead of using the Pearson Correlation Coefficient method. The settlement rates on the bridge were mainly lower than those off the bridge. Also, the standard deviations of the track gauge on the bridge were lower than the standard deviations off the bridge. Subsequently, the stochastic rates method showed higher peaks than when using variable ranges. However, the prediction method based on BLUE still did not give a specific value like the stochastic rates approach. Finally, different data sources can be used to enhance the understanding of the behaviour of the transition zones. Speed plots have been used to reflect driver behaviour at transition zones. Still, other datasets like maintenance, bridge design, wear, precipitation, temperature, and vegetation could also be used in future research.

Acknowledgements

I want to thank Mr. dr. A.A. (Alfredo) Núñez Vicencio for allowing me to do the master thesis in his section and think about topics that fit my interests. I also want to thank Dr. Kumar Anupam for being part of the thesis committee. I want to acknowledge Noa Peteroff and Michel Huijsmans from GVB for their trust in me in creating a proposal and doing further research within their company.

I started the preparations for this thesis at the end of April 2024. In the middle of July 2024, I started the thesis, including writing in the office of GVB. In the process of this thesis, I learned much about doing research, reading articles, Python skills, and much more. With the end product, I can finish my time as a student and start looking for a job in the transport work field.

I want to thank my friends I made during the bachelor Civiele Techniek and the master Civil Engineering for helping and supporting me through both studies. I also want to thank my friends from my hometown, Castricum, with whom I shared many memorable moments throughout both of my studies. During the COVID-19 pandemic of 2020, I looked forward to the daily online calls with my groupmates and friends to discuss assignments we had to do from home. Later, in my master's degree in civil engineering, I looked forward to seeing my friends from the transport specialisation every day. Even though I still lived in Castricum with my parents at that time. Later on, when I moved to Delft, I enjoyed being with my friends from my study and sports. During the thesis, I am grateful for having the opportunity to ask my friends for advice about the thesis problems I was stuck with.

During my study period, I also developed my talent in middle-distance running. First, I was able to train with the professional running team 'Team Distance Runners' from 2019 to 2021. Later, from 2021 to 2023, I signed an athlete contract, which made me part of the team. During the two top sports training camps in Portugal in 2022 and 2023, I learned much about myself as an athlete. I want to thank my coaches and teammates from that time for supporting me and allowing me to learn a lot about the life of a top sportsman. From 2023 to now, I have been an active member of the Delftse Studentenatletiekvereniging Dodeka. I want to acknowledge everyone at Dodeka for providing an environment where I can develop on a personal level every day.

Finally, I want to acknowledge my family for supporting me during my study period and accepting my choices during both studies. The support of my family, particularly my parents and two little sisters, made it possible for me to do everything I wanted during my study period.

List of abbreviations

AI	Artificial Intelligence
ANN	Artifical Neural Networks
BLUE	Best Linear Unbiased Estimator
DIC	Digital Image Correlation
FEA	Finite Element Analysis
GPR	Ground Penetrating Radar
GVB	Gemeentevervoerbedrijf
InSAR	Interferometric Synthetic Aperture Radar
IQR	Interquartile Range
KGE	Kleinste Geografische Eenheid
KPIs	Key Performance Indicators
LSC	Least-Sqaures Collocation
NDT	Non-Destructive Testing
PCC	Pearson Correlation Coefficient
PSI	Permanent Scatterer Interferometry
RMSE	Root Mean Square Error
SAR	Synthetic Aperture Radar
VCE	Variance Component Estimation

Contents

Chapter 1

Introduction

1.1 General Background

Trams are a key component of urban transportation systems, and in some cities, they have become much more popular in recent years to achieve lively cities that are car-free [Boquet, 2017]. To ensure the quality of the train operation and extend the infrastructure's service life, structural health monitoring and maintenance are crucial for the inframanager [Koohmishi et al., 2024].

Still, the tram infrastructure is a distributed system that experiences degradation differently in different locations. One of these locations is transition zones, which are, by definition, areas where a local change in structure occurs. For instance, between tram tracks and a bridge, between a tunnel and a track, and at level crossings where tram and road infrastructure intersect [Li and Davis, 2005] [Feizhi et al., 2023]. This thesis focuses on the transition zones between the tram and bridge networks.

On a railway track, transition zones have much higher maintenance costs than the rest of the railway track. Significant changes in stiffness appear between a stiff structure and an embankment, such as road crossings and slab tracks. As a result of these abrupt changes in vertical stiffness, the track geometry will be degraded faster than in open tracks. Track geometry degradation occurs due to traffic loads and track settlement. However, severe track geometry degradation is frequently found at transition zones because of abrupt stiffness changes. Figure 1.1 shows a typical transition zone of a tramway track in Amsterdam. The degradation of the track geometry leads to a significant irregularity (dips) in the track geometry. Track irregularities harm passenger comfort and can even lead to track damage and derailment [Wang, Chang, and Markine, 2018]. The elevation changes of the rail depend on how the track modulus, k, varies and on the train's velocity [Lei and Mao, 2004] [Kerr and Bathurst, 2001]. In research on railway transition zones, the transition length is usually between 25 and 30 meters [Dingqing and David, 2005] [Sañudo et al., 2016].

As mentioned above, due to the high rates of track degradation in transition zones, transition zones in The Netherlands have more train maintenance costs than in other track locations [Coelho, 2011]. In concrete bridges, to increase damping characteristics and reduce impact between ballast and concrete, rubber pads can be used under concrete ties [Li and Davis, 2005]. Tamping is a maintenance procedure that focuses on restoring the track to its original geometrical position by re-compacting the ballast by letting it go underneath the sleepers [Lichtberger, 2010]. Tamping is performed mainly every four or five years [Czichos, 2013].

Tram tracks are constructed differently from classic rail tracks. One reason is that, in some locations, trams and road vehicles are expected to share large segments. Therefore, the tramway track is often integrated into continuous concrete foundation slabs and is not designed in sleepers on a ballast bed, as on railway tracks [Ahac and Lakusic, 2015]. The concrete slab transfers the load and provides track stability. Elastomeric components give the track resilience [Moridpour and Hesami, 2015]. A ballasted track exhibits higher differential settlements in railways than a slab track. [Charoenwong et al., 2023]. In tramways, ballast tracks occur but less than open tracks. Therefore, transition zones with severe track geometry degradation will also occur in a tram transportation network [Jover, Gaspar, and Fischer, 2020] [Kraśkiewicz, Urbaniak, and Piotrowski, 2025]. Shan et al. (2016) researched the predictions of deformations from tram track construction in a structure-embankment transition zone. They used Finite Element Analysis (FEA), which

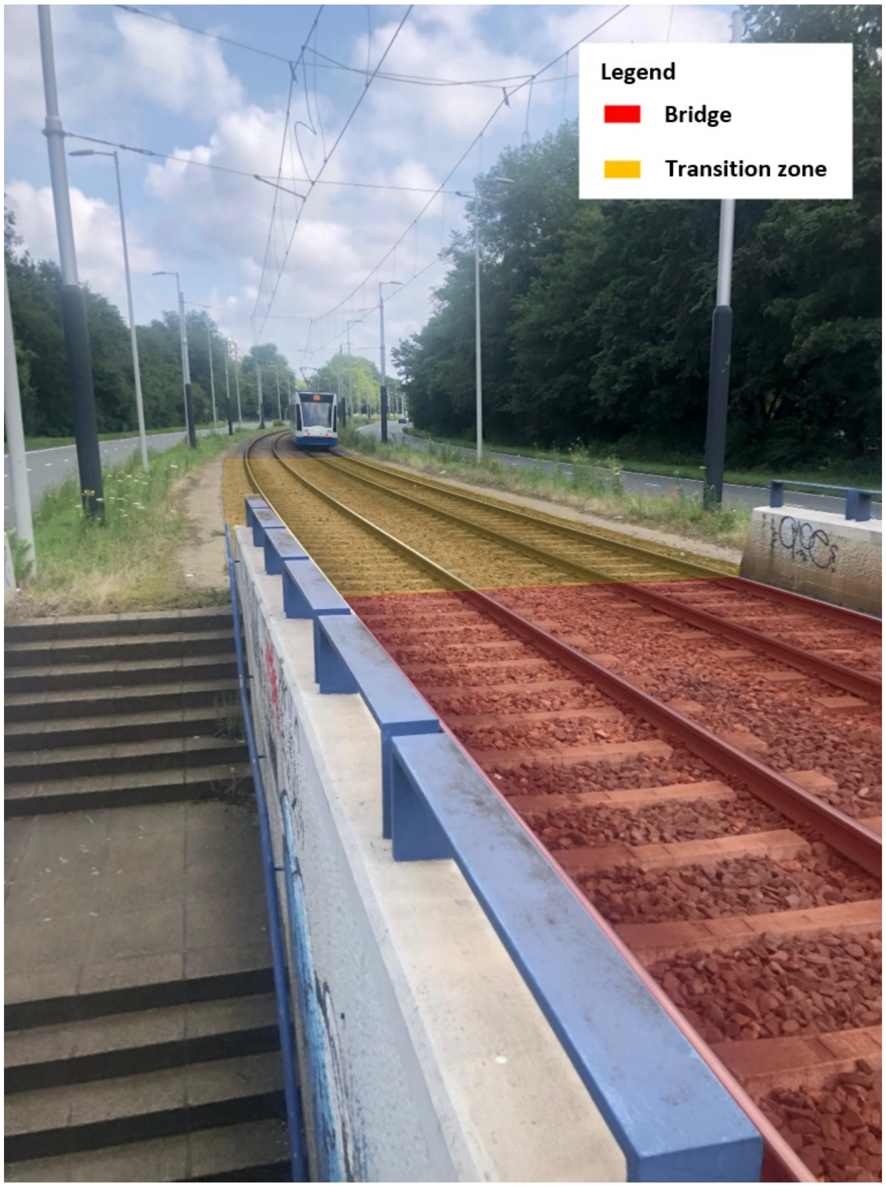


Figure 1.1: A tramway transition zone at the Piet Wiedijkstraat in Amsterdam.

resulted in an improvement of the transition zone [Shan et al., 2016]. However, research on tramways, particularly in transition zones, is limited compared to conventional railways. This is despite the societal impact that research in tramway infrastructures could have. These infrastructures are embedded in cities, and their condition can greatly impact the quality of life of people living in their surroundings.

In railway track health monitoring, non-destructive testing techniques (NDT) have been proven to be a valuable maintenance strategy for a transition zone [Koohmishi et al., 2024]. Wang et al. (2018) proposed using a NDT technique, named InSAR (Interferometric Synthetic Aperture Radar), to detect high rates of settlement to monitor track degradation and provide information when the measured settlement becomes too critical [Wang, Chang, and Markine, 2018]. InSAR is a feature that measures changes in land surface topography. In short, InSAR has frequent satellite deformation data that can be helpful in asset management [Koohmishi et al., 2024]. Since this is an NDT, InSAR is valuable information for maintenance techniques that should be investigated to determine if it will also be effective on a tramway track.

1.2 Research gap

As mentioned above, InSAR is already applied to detect the degradation of rail tracks in transition zones [Wang, Chang, and Markine, 2018] [Koohmishi et al., 2024]. It has been reported that the geometry parameters of the tram rails can give a good indication to predict the degradation of the tram track. Geometry parameters do not show direct high rates of track degradation, but can show high geometry track deterioration [Falamarzi et al., 2018] [Abadi, Sameni, and Yaghini, 2023]. However, to measure the geometry, the public transport operator should take measurements with a special train, which cannot be done every week or month due to high costs and gear availability. At the same time, InSAR satellite data can be given every 8 or 12 days for a new measurement [Tosti et al., 2020]. Therefore, InSAR deformation data can be a helpful tool for faster detection of high settlement rates that can lead to geometry deterioration. However, to the best of our knowledge, no research has been openly reported on the use of InSAR data for tramway tracks. In addition, research on tramway track transition zones is limited. Therefore, this thesis will focus on analysing transition zones of tramways in comparison with transition zones of railways and creating a prediction model for the settlement of the tramway track in some selected transition zones based on InSAR data.

1.3 Case study in Amsterdam

The capital of The Netherlands, Amsterdam, has a tramway system consisting of 15 tram lines operated by Gemeentevervoerbedrijf (called GVB) [Yap and Oded Cats, 2022]. Since this thesis will be done in cooperation with GVB, it will focus on describing some transition zones in Amsterdam. Applying the created model to the chosen locations, a prediction of when maintenance should be scheduled will be made to ensure the quality of the tram service.

GVB has measurements every year, which gives them, the ability to determine the condition of the tramway track. The geometry has been measured yearly with a vehicle from Autech shown in Figure 1.2. The car has road and rail wheels and pulls the MW3000, shown in the front of Figure 1.2, at the back, which measures the rail geometry with a laser. The MW3000 has a check rail to compare the measured profile with the original position of the rail [Peteroff, 2021]. The information provided from the geometry measurements is not directly related to the degradation of the track since geometry indicates the condition of the track, and the condition can also be affected by the embankment condition, which is not measured with the geometry. Open-source InSAR data from Bodemdalingskaart.nl gives 12 days a displacement value as experienced in the open-source data sets from Bodemdalingskaart.nl. The frequency of the data points is higher than that of GVBs. InSAR can give extra and more frequent information about the condition of the transition zone. However, it is also not a measurement that can directly measure the track degradation because other failure mechanisms can also affect the deformation. A combination of using GVB geometry data and InSAR can show a better understanding of the tramway track condition. They may even predict when a high chance of experiencing high rates of track degradation occurs.

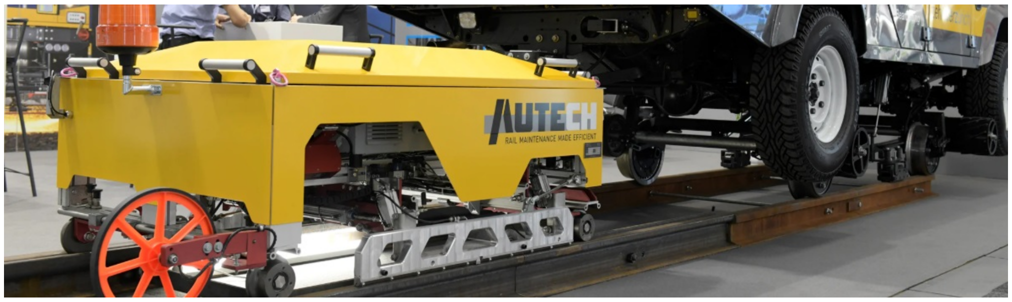


Figure 1.2: Measuring Trolley from Autech [Autech, 2020].

1.4 Research question

In this report, the InSAR method described above will be analysed and implemented in a tramway transition zone in Amsterdam. The following research question will be answered: "Is InSAR able to provide information about the condition of the transition zone?". Three sub-questions will be proposed to answer the research question, and each of them can lead to the answers to the main research question.

Three sub-questions were asked to answer the research question. Before answering the third sub-question, the first two sub-questions need to be answered separately to be able to answer sub-question 3. Sub-questions 1 and 2 do not have to be answered in the same order as set in the enumeration. The sub-questions are as follows:

1. How can geometry deterioration in tramway transition zones be indicated with current geometry parameter datasets?

2. What are the key characteristics and limitations of InSAR data for assessing the conditions of tramway transition zones?

3. Which prediction method can be used for the indication of high rates of settlement in tramway transition zones based on the current dataset of geometry parameters from GVB and InSAR deformation data?

Further research should determine what parameters from the GVB data are needed to know which parameters can indicate track degradation. It can be helpful in making a prediction model with InSAR data. Unfortunately, no GVB maintenance data are available to validate whether the prediction model's outcomes align with the applied maintenance. Therefore, the proposed prediction methods should be tested with uncertainty parameters to compare them. After answering the sub-questions, the final answer to the research question could be made.

The answer to the main question can fill the research gap on using InSAR deformation as a monitoring technique for tramways. Research using InSAR data on tramways is limited. Therefore, this thesis can contribute to improving tramway track maintenance strategies. Since a prediction method based on InSAR will also be created, this research can contribute to the knowledge about using InSAR prediction models for infrastructures.

Chapter 2 describes the literature review and discusses the most crucial information to answer the research questions. Chapter 3 discusses the methodology and scope of this thesis. Chapter 4 applies the methodology to three different locations in three case studies. Chapter 5 presents the conclusion of this thesis and the recommendations for GVB.

Chapter 2

Literature Review

This thesis proposes a methodology to use InSAR satellite data and track geometry for the tram transition zone condition prediction. The literature review is organised into two parts to understand the fundamentals of these measurement techniques and the dynamics of degradation in tramways. First, the fundamentals of InSAR data are described, with applications from different fields. Secondly, the track degradation prediction methods are discussed mainly from the field of railways, given the limited available research on tramways. The literature will inspire the proposed method for predictions about the settlement of a tram track in a transition zone.

As mentioned in Section 1.1, the literature on tramway transition zones is limited. Therefore, literature from railway research will be used to obtain supplementary information about transition zones.

2.1 InSAR data

InSAR is a non-destructive, remote sensing testing tool to monitor ground deformation. Non-destructive testing means determining the properties and structural integrity of the material without affecting the function of the material [S. Kumar and Mahto, 2013]. InSAR data can be used to monitor the deformation around the tram track structures [Koohmishi et al., 2024]. InSAR can measure with the precision of millimeters in the embankment [Chang, Dollevoet, and Hanssen, 2017]. Ground Penetrating Radar (GPR) is another NDT used in railways, and it consists of an electromagnetic-based device. GPR can be used to inspect the quality of ballast, subgrade layers, and embankments of railway tracks [Koohmishi et al., 2024].

Since its first application in 1974 to generate topographic maps, InSAR has been used in a multidisciplinary manner in geophysical monitoring, including monitoring the structural health of structures [Graham, 1974] [Massonnet and Feigl, 1998]. InSAR has been used, for instance, to detect subsidence on a railway track and to detect settlement under an old church in Spain [Tosti et al., 2020] [Tomás et al., 2012].

2.1.1 Insights in InSAR data

The idea behind InSAR is based on Synthetic Aperture Radar (called SAR), a radar remote sensing system. SAR includes a radar installed on a platform that moves forward, and this radar transmits electromagnetic pulses at a frequency ranging from a few hundred to a few thousand Hertz and receives the backscattered signal. Approximately one hundred and more than a thousand Hertz pulse repetition frequencies are sent to the Earth's surface [Moreira et al., 2013]. In Figure 2.1, the slant range represents the direction perpendicular to the y direction, and the swath's width represents the radar scene's ground range. Because SAR uses transmitted pulses to the wideband and signal processing in the phase history of signals reflected by the surface to obtain high resolution in azimuth, it can create high-resolution images of radar-mapped areas [Tomiyasu, 1978].

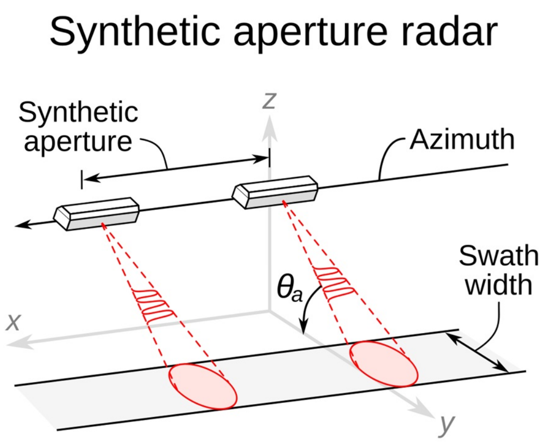


Figure 2.1: Defining the SAR system [RCraig, 2020].

SAR only provides a 2D image from a 3D object with the range and azimuth as image coordinates. The range is responsible for the radius of the circle of the surface where the satellite is focused, and the azimuth is responsible for the width and length of this circle. Because these parameters do not give information on the height of the imaged scene, InSAR requires a second antenna with a known lateral displacement from the first antenna. The height difference between the measurements of the two antennas is detected by exploiting the phase difference between the two images of the antennas, also known as the phase interferogram [Franceschetti and Lanari, 1999]. The interferogram process works with complex multiplication of the two points, which gives a value that represents the relative interferometric phase ($\phi^w \in [-\pi, \pi)$ [rad]). It is impossible to measure the absolute interferometric phase from the relative interferometric phase ϕ directly. A relative phase, modulo radians, $W\{\phi\} = \mod\{\phi+\pi, 2\pi\}-\pi$ can be measured. $W\{.\}$ is here the wrapping operator. Equation 2.1 is considered when unwrapping the operator. ϕ_{defo} is the component that represents the kinematic phase due to object or surface displacement. The other components (ϕ_{topo}, ϕ_{atmo}, and ϕ_{noise}) represent the topographic phase caused by the surface geometry, the phase contribution from the atmospheric signal delay, and other random additive noise [Wang, Chang, and Markine, 2018].

$$\phi^w = W\{\phi\} = W\{\phi_{\text{topo}} + \phi_{\text{defo}} + \phi_{\text{atmo}} + \phi_{\text{noise}}\} \tag{2.1}$$

2.1.2 Accuracy of InSAR data

As mentioned in Section 2.1, InSAR was used for the first time in 1974 by developing a radar image with displacements using the reflectivity of the terrain. Two orthogonal coordinates could be shown on this radar map, the R coordinate (distance from the flight path to the object) and the Y coordinate (distance along the flight path from some reference point to the object). The angle parameter θ is needed to know the position of the objects in the three-dimensional radar images. The angle parameter θ is also known as the depression angle. $\delta\theta$ has been proposed here as the angular accuracy by Graham and is shown in Equation 2.2. The λ stands for the wavelength, the h stands for the aperture size, the S stands for the received signal power, and N means the receiver noise power.

$$\delta\theta = \frac{\lambda}{2h\sqrt{S/N}} \tag{2.2}$$

As written in Subsection 2.1.1, two antennas catch signals that result in observing height differences in structures. The accuracy of the measurements can be written as the angular accuracy multiplied by R. The derivation of this accuracy can be done by first defining the range resolution (W_R) and the resolution in the along-track direction (W_A). As shown in Figure 2.2, $R\Delta\theta$, which is not the depression angle but the accuracy of measurements multiplied with the radar slant range (R), consists of the sum of the heights created by W_R and W_A and the angles α_R and α_A. The equation for the accuracy of the measurements can be written as the equation shown in Equation 2.3.

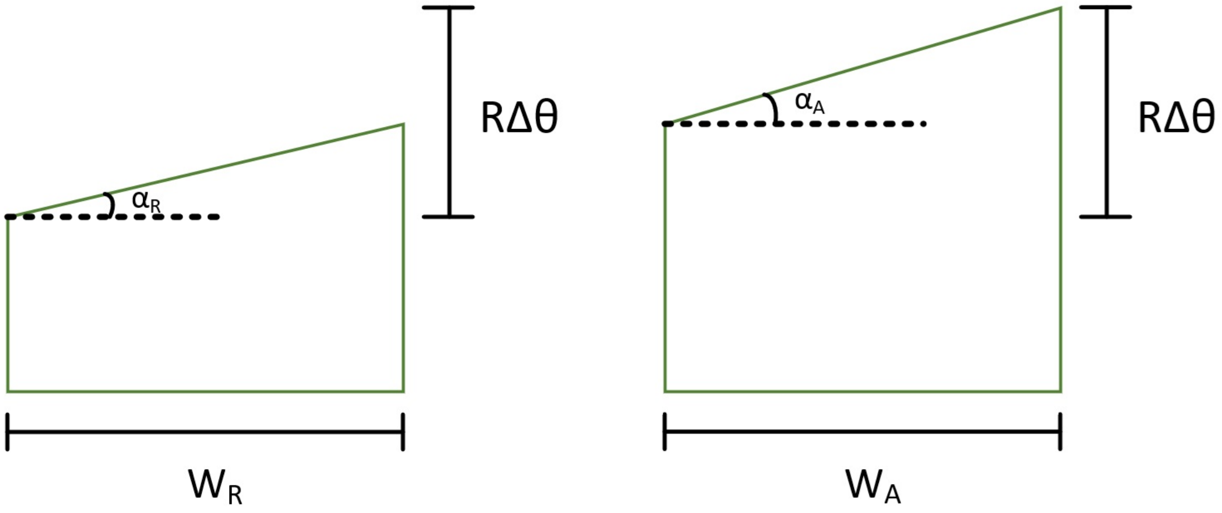


Figure 2.2: Defining W_R and W_A, inspired by: [Graham, 1974].

$$\Delta\theta = \frac{1}{R}\left(W_A\,\tan(\alpha_A) + W_R\,\tan(\alpha_R)\right) \tag{2.3}$$

The coordinates of the points appear when R and θ are transformed from cylindrical coordinates to Cartesian coordinates x, y, and z. The Cartesian coordinates are shown in Equation 2.4. As shown in Figure 2.1, y stands for the direction of the flightpath, x stands for the direction perpendicular to the flight path and parallel to the ground of the Earth, and z stands for the direction perpendicular to x and y.

$$\begin{aligned} y &= Y \\ x &= R\cos(\theta) \\ z &= R\sin(\theta) \end{aligned} \tag{2.4}$$

From equations 2.3 and 2.4, the accuracy of the location of the various objects can be determined. Suppose that accuracy is the derivative of the x- and z-directions in the x- and z-direction to the power of R and θ. In that case, the accuracy of the Cartesian coordinates can be combined by differentiating and are shown in Equation 2.5. The range error due to range resolution, measurement, and timing is taken as dR, the angular precision as $d\theta$, and the along-track error due to equivalent sources as dY.

$$\begin{aligned} dy &= dY \\ dx &= \frac{\delta x}{\delta R}\,dR + \frac{\delta x}{\delta\theta}\,d\theta = \cos(\theta)\,dR + R\,\sin(\theta)\,d\theta \\ dz &= \frac{\delta z}{\delta R}\,dR + \frac{\delta z}{\delta\theta}\,d\theta = \sin(\theta)\,dR + R\,\cos(\theta)\,d\theta \end{aligned} \tag{2.5}$$

When assuming that the differentials represent small and independent errors, the variances will be as in Equation 2.6.

$$\begin{aligned} \sigma_y^2 &= \sigma_Y^2 \\ \sigma_x^2 &= (\sigma_R\,cos(\theta))^2 + (R\sigma_\theta\,sin(\theta))^2 \\ \sigma_z^2 &= (\sigma_R\,sin(\theta))^2 + (R\sigma_\theta\,cos(\theta))^2 \end{aligned} \tag{2.6}$$

Hu et al. (2019) proposed that the position of the InSAR data points is in the order of meters, and they researched using InSAR data in combination with laser data called LiDAR to overcome the limitation of monitoring physical objects due to this high order. Airborne LiDAR gives 3D point clouds with high spatial density. With this approach, the position error ellipsoids are snapped from the scatter point to the LiDAR point cloud. The position error ellipsoid has been determined with the Signal-to-clutter (now called SCR), which is the ratio between the peak intensity and the background [Dheenathayalan et al., 2016]. Equation 2.7 then gives the variance of the subposition (range $\sigma^2_{r,P}$, and azimuth $\sigma^2_{t,P}$). Δ denotes the oversampling factor. LiDAR and a k-nearest neighbour search process provide a better view of the precision of the deformation signal [F. Hu et al., 2019].

$$\sigma^2_{r,P} = \sigma^2_{t,P} = \frac{3}{2\pi^2 \cdot \hat{SCR}_p} + \frac{1}{12\Delta^2} \tag{2.7}$$

According to Wang et al. (2018), the precision of the satellite measurements is at a millimeter level, which is relatively low, and the spatial resolution is at a meter level. According to Hu et al. (2014), the spatial decorrelation of the InSAR measurement points because of the 1-meter precision in space has been researched using the Permanent Scatterer Interferometry (now called PSI). PSI can potentially improve the accuracy and resolution of displacement measurements [J. Hu et al., 2014].

2.1.3 InSAR data on railway tracks

As mentioned in Chapter 1, Wang et al. (2018) proposed using InSAR to detect high rates of track degradation in a railway transition zone. In the research, InSAR data near the bridge were compared with Digital Image Correlation (DIC) data, which represents the height of the rail. InSAR indicated that far from the bridge, the settlement was lower than near the bridge, as was also concluded with the DIC data. Therefore, the results of the measuring coach correlate with InSAR. Because, as mentioned in Chapter 1, differential settlements lead to a poor supporting condition of the rails and sleepers and will lead to irregularities in the track, it can be concluded that the InSAR data help analyse the condition of the track. Additionally, it is cost-efficient because InSAR performs long-term health monitoring for transition zones at a high frequency [Wang, Chang, and Markine, 2018].

Zhang et al. (2019) investigated InSAR to measure deformation rates on a railway track in the Lhasa-Naqu section. The investigation led to the conclusion that before the opening of this railway section, the deformation was minimal. However, after the start of the operation, the roadbed became relatively unstable, which was shown in different uplift zones along the rail [Zhang et al., 2019].

2.2 Track degradation

As mentioned in Chapter 1, high rates of track degradation can occur due to differences in settlement rates due to different stiffnesses on a tramway track. High rates of track degradation can lead to geometry deterioration and even derailment. Each of the components of the track has an aspect in the track degradation process. In addition, some models are available to calculate numeric values to classify track degradation.

2.2.1 Track components

In railways, a ballast track consists of two main components, as shown in Figure 2.3, the superstructure, where the rails, the fastening system, the rail pads and the sleepers are lying, and the substructure, which consists of the ballast, sub ballast, and the subgrade [Hawari, 2007].
The subgrade takes all loads from the vehicle. Therefore, it should have enough bearing capacity to handle these loads and prevent excessive plastic strains [Castro et al., 2022]. The subgrade generally exists in fine-grained soils such as silt and clay, which have a lower strength and permeability than coarse-grained soils such as gravel and sand [Li and Selig, 2015].
The ballast in the substructure resists the vertical, lateral, and longitudinal forces applied to the sleepers to ensure the position of the rails [Gallagher et al., 1999]. According to Guo et al. (2022), ballast consists mainly of sand, rock, coal ash, or crushed rocks, and when this becomes wet, rapid track degradation occurs [Guo et al., 2022]. Moreover, the low strength of the subgrade will result in excessive plastic deformation and settlement [Li and Selig, 2015]. Then, the caused settlement results in the amplification of the dynamic forces acting on the track, accelerating the deterioration of the vertical and reducing the comfort of the passengers [Wang and Markine, 2018].

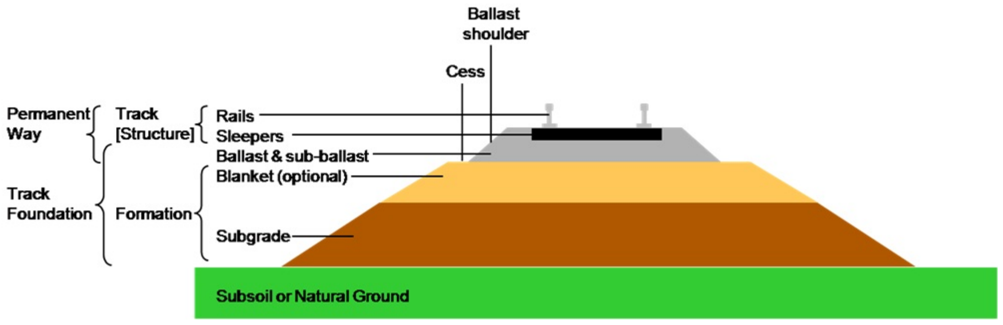


Figure 2.3: Cross section railway track [Bermicourt, 2009].

As mentioned in Chapter 1, in the crossings of the roads on tram tracks, tram tracks are constructed differently than classic rail tracks. The tramway track is often integrated into continuous concrete foundation slabs and lacks a ballast bed. However, the research using InSAR data mentioned in Chapter 1 performed research on railway tracks, which consist of a ballast bed.

2.2.2 Track degradation models

Jovanovic et al. modelled the deterioration of the railway track by using multivariate statistical analysis of the twist, gauge, alignment, cant, and level to optimise the maintenance of the railway track. The research concluded that modelling the behaviour of the deterioration of the track geometry remains challenging because it has not yet been proven scientifically that individual track components influence each other. When having more reliable data related to the railway track, statistical analysis is one of the best ways to model the deterioration of the track geometry [Jovanovic, Evren, and Guler, 2011].

Jover et al. investigated tramway track deterioration based on geometric characteristics, the length of superstructures, the speed of trams, unit costs, and lifespan. With calculations, the deterioration of a tramway in Budapest could be monitored. However, more observation about tramway track deterioration is needed to improve the knowledge about the maintenance procedure for tramway track deterioration [Jover, Gaspar, and Fischer, 2020].

Generally, track degradation models can be characterized into three categories: mechanistic models, statistical models, and Artificial Intelligence (AI) [Falamarzi et al., 2018]. Grossoni et al. (2021) proposed three mechanistic equations that give ballast settlement as a dependent variable for track degradation. The Guérin equation, shown in Equation 2.8, gives the rate of accumulation of permanent settlement of the ballast as a function of the maximum elastic deflection of the sleeper, $d_{b,max}$ [Guerin, 1996]. The Sato equation, shown in Equation 2.9, represents settlement as a function of load cycles N and the ballast pressure P, or ballast force F. The coefficients a and c depend on the power of b. The coefficients d, e, and the pressure threshold P_{th} depend on the thickness of the ballast layer. Fröhling's equation, shown in Equation 2.10, also adjusts the dynamic impact of the vehicle wheel, P_{dyn}, on the road after loading cycles N. Therefore, it also takes into account the deviatoric stress at the sleeper-ballast interface. k_{2mi} means here the measured average track stiffness in (MN/m). K_1, K_2, and K_3 are constants, P_{ref} is the reference wheel load, and w is an exponent [Grossoni et al., 2021].

$$\frac{dS_N}{dN} = 1.44 \cdot 10^{-6} \cdot {d_{b,max}}^{2.51} \tag{2.8}$$

$$S_N = \begin{cases} a \cdot F^b \cdot c \cdot N & \text{if } P \leq P_{th} \\ d \cdot (P - P_{th})^2 \cdot e \cdot N & \text{if } P > P_{th} \end{cases} \tag{2.9}$$

$$S_N = \left(\left[K_1 + K_2 \cdot \left(\frac{k_{2mi}}{K_3}\right)\right] \cdot \frac{P_{dyn}}{P_{ref}}\right)^w \cdot \ln N \tag{2.10}$$

Falamarzi et al. (2018) proposed two Artificial Intelligence (AI) models. The regression model has been set with relevant parameters track gauge deviation in the previous year, track surface, and rail type. The output variable is the track gauge for the current year, as that output gives insight into the rate of track degradation. The Artificial Neural Networks model (ANN) consists of several independent, interconnected neurons that can communicate with each other with weighted connections. Equation 2.11 provides the mathematical mechanism of a neuron in the ANN model, whereby A is the activation function, ω_j is the weight of the j^{th} input, I_j is the input, and B_i is the bias of the i^{th} neuron. After validation, Falamerzi et al. concluded that a regression model as an ANN model for the prediction of rail degradation is acceptable [Falamarzi et al., 2018].

$$O_i = A \cdot \left(\sum_{j=1}^{n} \omega_j \cdot I_j + B_i \right) \tag{2.11}$$

Statistical models are based on input and output variables [Falamarzi et al., 2018]. As input variables, geometry parameters can be used because analyzing geometry can indicate high rates of track degradation [Wang, Chang, and Markine, 2018]. According to the European standards for the quality of the track geometry, S-EN 13848-1: 2004 + A1, 2008, the parameters of the railway track geometry can be divided into five parameters. The five classes represent the longitudinal level, alignment, gauge, cant, and twist. The longitudinal level represents the geometry of the track compared with the reference line from the longitudinal vertical plane. The alignment means the geometry of the track compared with the reference line from the longitudinal horizontal plane. The gauge is the distance between two single rails. As mentioned in Chapter 1, according to Falamerzi et al. (2018), high standard deviations of the track gauge give a significant indication of high rates of track degradation [Falamarzi et al., 2018]. The cant represents the height difference between two single rails, which occurs when the rail curves are in vertical alignment [Soleimanmeigouni, Ahmadi, et al., 2020]. Finally, twist means the difference between the top of the constructed line of the rail and the current top of the rail measured at a known interval of meters. Twist can cause an unbalanced load to the left and right rails and can even lead to derailment [Abadi, Sameni, and Yaghini, 2023].

The literature in this chapter about railway transition zones provides valuable insights for understanding similar phenomena in tramways. Still, to the best of our knowledge, reports on the analysis of tramway transition zones are limited. One related study by Periard (1998) investigated curve squealing, a noise of high amplitude that appears in a curve due to wheel-rail interaction on a tramway track. In conclusion, it has been written that the rear wheelset produces a noise of much higher volume than the front wheelset. It has been stated that the contact geometry influences the creeping or sliding contact of the wheels. In addition, the squealing noise is influenced by the track dynamics at the entry and exit of the curve [Periard, 1998].

In the case of prediction, according to Ahac et al. (2015), track gauge parameters can be used to model predictive strategies for track degradation. High standard deviations of the track gauge mean more geometry deterioration. In addition, the research concluded that rail wear increases during exploitation. Subsequently, the rail degradation speed becomes lower. Earlier studies have also shown that after a period of abrasion of the rails, the speed of rail degradation is reduced [Ahac and Lakusic, 2015].

Peteroff (2021) researched the rail wear in curves at the tramway in Amsterdam in his master's thesis. Rail wear is a mechanism that costs a significant part of the maintenance budget. The research studied tram velocity and rail hardness as parameters influencing the rail wear process. In the results, it has been concluded that in rail curves, three parameters have significantly affected wear: the vehicle loading by passengers, increased velocity, and increased flange steepness of the wheels. In further detail in the thesis, it has been proposed that track degradation models could be combined with track monitoring. Track monitoring techniques that can be used are acceleration, laser, and photo-measurements. Combining these three techniques benefits from their advantages [Peteroff, 2021].

Chapter 3

Methodology

The research question in this thesis is to set up a model based on InSAR deformation data to measure the settlement of a tramway track in a transition zone in Amsterdam. Due to the research undertaken to create an innovative model, this project can be classified as a modelling project [Zwart and Vries, 2016].

Some methods are used to answer the sub-questions and the main research question. This chapter will explain the methodology of answering each sub-question and the main research question.

3.1 Selecting locations

First, a few tramway transition zones in Amsterdam will be selected to be analysed using the geometry data of GVB and the InSAR satellite data. The selected locations can be used to create a prediction model based on InSAR. To select the locations, a one-day tram trip to Amsterdam will be scheduled to look for transition zones. Furthermore, the knowledge obtained from the literature study in Chapter 2 will be used to analyse the condition of each transition zone with the geometry data GVB has available. As proposed by Falamerzi et al. (2018), the standard deviation of the track gauge is a valuable indication of geometry deterioration. However, the data used to indicate geometry deterioration should be provided with another geometry dataset of GVB to strengthen the results of the track gauge dataset used. Therefore, twist data from GVB will also be used to validate the gauge data and see if the track gauge is high when more twist differences occur near the investigated bridge. The twist dataset means a different measurement length of the twist between two rail parts. GVB has measurements available with different measurement lengths of 2 m, 6 m, and 8 m. It will be assumed that high standard deviations of twist also mean high geometry deterioration. The choice has been made to compare the standard deviations of twist data with the track gauge data because twists can also be harmful, which means that using the means can represent problematic results for twists. After all, low means can be shown due to harmful twist data rather than a higher mean.

3.2 Preprocessing InSAR data

InSAR data in Amsterdam is available from Bodemdalingskaart.nl. The used InSAR data consists of data from 2017 to 2022 from the satellites Sentinel-1a and Sentinel-1b, according to Bodemdalingskaart.nl. However, Bodemdalingskaart.nl is an open-source InSAR data platform, and its data are estimated from the received data from the stated satellites. Therefore, some statistical methods are needed to filter out the outliers. According to Laurikkala et al. (2000), the outliers of each InSAR data point can be found using box plots. The first (first 25% of the sorted data) and the third quantile (first 75% of the sorted data) should be calculated by creating box plots. Subsequently, the Interquartile Range (now called IQR) can be calculated as the third quantile minus the first quantile. When the deformation is greater or less than the first or third quantile minus or plus 1.5 times the IQR, an outlier in a data point is detected [Laurikkala, Juhola, and Kentala, 2000]. However, after filtering out all outliers at each data point, additional processing steps are required due to highly noisy data. To overcome this problem, it will be assumed that the settlement rate is linear so that a linear model can be fitted to the data. Previous research has proven a linear model as a method to predict settlement with high precision [Nadeem et al., 2021]. To test the quality of this fitted model, the Root Mean Square Error (called RMSE) will be calculated at each data point. The RMSE is a standard metric that is used in model evaluation [Hodson, 2022]. In Equation 3.1, the formula for the RMSE is shown. n means the amount of deformation data from a data point, y_i means the observed deformation value, and $\hat{y}_i$ means the predicted deformation value. After calculating the RMSE of each data point, a confidence interval with a generic percentage (80%, 90%, and 95%) will be calculated. When the RMSE of a data point does not lie in the confidence interval but instead lies above the upper quantile of the confidence interval, the data point will be filtered out because the error is relatively too high. The rest of the data points will still be used for analysis. Therefore, every approach will use three different InSAR datasets based on the three proposed confidence interval values: 80%, 90%, and 95%.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{3.1}$$

As mentioned above, a linear model will calculate the settlement rates. According to Teunissen (2007), Least-Squares Collocation (called LSC) is a useful method for calculating trends in the spatial and Earth sciences. A part of LSC is the Best Linear Unbiased Estimator (called BLUE) shown in Equation 3.2 [Teunissen, 2007]. A stands for the input matrix, and y for the measurements to be fitted. Q_{yy} means the covariance matrix [Hastie, Tibshirani, and Friedman, 2009].

$$\hat{x} = \left(A^T Q_{yy}^{-1} A\right)^{-1} A^T Q_{yy}^{-1} \cdot y \tag{3.2}$$

According to Teunissen et al. (2007), Least-squares Variance Component Estimation (called VCE) can give the variance of the satellite data [Teunissen and Amiri-Simkooei, 2007]. VCE equates the mean square error with the expected value, which can be the error variance [Searle, 1995]. According to Hastie et al. (2009), the error variance is the sum of the errors, $(y_i - \hat{y}_i)^2$, divided by the number of deformation data, N, minus the number of features, p, and one as shown in Equation 3.3 [Hastie, Tibshirani, and Friedman, 2009].

$$\hat{\sigma}^2 = \frac{1}{N-p-1}\sum_{i=1}^{N}\left(\, y_i - \hat{y}_i\right)^2 \tag{3.3}$$

After preprocessing the data, the InSAR data will be plotted in the three locations to show the settlement rates, and these rates will be classified into data points positioned on and off the bridge. In the research by Wang et al. (2018) on the use of InSAR on railways, three groups were supposed; each of them is defined in Table 3.1. However, in this research, the data will not be plotted in three groups because the data points positioned near the bridge cannot be assumed on the bridges on which the trams in Amsterdam travel due to a short span length. The mean rates for both groups will be calculated to compare the two groups. An assumption is that the mean rate of the data points on the bridge is lower than that of the data points not on the bridge because the embankment generally has a higher settlement rate than the bridge [Li and Davis, 2005].

Table 3.1: Defined groups railway in the research of Wang et al. (2018).

Far from bridge	240m from bridge
Close to the bridge	1.5m from bridge
Bridge length	100m

3.3 Prediction methods

The earlier preprocessed data will be used for two prediction approaches to predict high settlement rates. After creating and using the approaches, both models will be compared using a statistical method, which will be explained.

As mentioned in Section 3.1, GVB track gauge data and twist will be used. Because GVB has only data available from 2019 to 2024, both approaches will use InSAR data and geometry from 2019 to 2022, which

are not the same dataset used in the described step in Subsection 3.2, causing the data to have a smaller period. Due to the lack of track gauge and InSAR data, the track gauge data from 2019 to 2022 and the InSAR data from 2019 to 2022 will be used in both approaches. Then, both methods will be tested on the exact dates the track gauge data is applied, which may create contradictory approaches. Ideally, for instance, InSAR data and GVB data can be coupled from 2019 to 2022, and then InSAR data for 2023 and 2024 can be used to predict high rates of track degradation. However, this proposed approach is impossible due to the lack of available InSAR data for 2023 and 2024.
As described in Section 1.3, GVB measures the track geometry of the tramway track every year. Both approaches will use InSAR as a maintenance technique to show the advantages of using settlement rates calculated from half-year InSAR data. Half-year InSAR data will be used because, as stated in Subsection 2.1.2, since InSAR has low accuracy in position (the order of meters), InSAR can have a high variance in displacements, which can give a less accurate result in the short term. When using a span of one, two, or three months, the calculated rates are still too sensitive and take too long to obtain a positive value. Therefore, it has been decided to use only the InSAR calculated settlement rates for half a year.
In both approaches, the three preprocessed InSAR datasets (a set of data points with a confidence interval of 80%, 90%, or 95% to filter outliers) will first be linked to the GVB track gauge data. The BLUE method will calculate the settlement rates yearly to compare the InSAR data with the GVB yearly data in every transition zone. Dividing the rates and geometry data into groups containing the bridge and those not on the bridge will give a better insight into the abrupt stiffness changes. Taking the average of all the available InSAR data points will provide values for each year, which will be compared with the GVB geometry data. The yearly values of the rates and the track gauge of the data points positioned on the bridge and not on the bridge will be compared with the Pearson Correlation Coefficient (now called PCC). The PCC measures the strength and direction of the linear relationship between two random variables and has been used many times for pattern recognition. The formula of the PCC is shown in 3.4, where ρ stands for the linear correlation between datasets x and y, $E[xy]$ stands for the cross-correlation between x and y, and σ_x and σ_y are the square roots of the variances of x and y [Benesty, Chen, and Huang, 2008]. After calculating the correlation of the settlement rates on the bridge with the GVB geometry data on the bridges, the data points needed for the prediction method will be found by looking for the transition zone length when the InSAR data points not on the bridge have the highest correlation, so the highest ρ, with GVB track gauge data point not on the bridge. For the transition zone lengths, a range from zero to 100 meters with steps of five meters will be used. As an assumption, a negative link will be assumed because when high standard deviations of the track gauge appear, more significant geometry deterioration will occur, which should mean high rates of track degradation. Therefore, the regime for selecting the optimum transition zone length will be choosing the most extended transition zone length with the lowest ρ value. The found data points will be used in both approaches.

$$\rho(x, y) = \frac{E[xy]}{\sigma_x \sigma_y} \tag{3.4}$$

In linking GVB geometry data with InSAR data, the data points will be optimised with a variable transition zone length. However, as mentioned in Subsection 1.1, the transition zone length has been chiefly assumed to be 30 meters from both sides of the bridges. Therefore, the approaches do not work with a standard length for transition zones. To overcome this problem, the methods will be tested with a few standard lengths of the transition zone. In railways, transition zones have a standard length of about 30 m because of the maximum speed of 220 km/h for trains [Sañudo et al., 2016]. In transport services in urban cities, requirements such as maximum speed restrictions of approximately 70-80 km/h occur [Strainescu et al., 2008]. According to the research of Sañudo et al. (2016), a speed of 80 km/h corresponds with a transition zone length of about 11 meters. Therefore, the approaches will be tested with a standard transition zone length of 10 meters at all proposed locations. If an approach for a location will not work due to fewer data, a transition zone length of 15 or 20 meters will be used in both methods. The link between InSAR data and GVB track gauge data will be analysed for transition zones with lengths of 5, 10, 15, and 20 meters.

3.3.1 First approach: Stochastic rates

The first approach is based on stochastic rates. Before starting the first approach, there should be a reference point at each location to look for the static settlement, which explains the settlement of the tramway track due to the soft soil of Amsterdam. According to the guideline for parking structures from the University

of Michigan, parking spaces have a higher live load capacity, which will give in the analysis a settlement rate only based on the general deformation of Amsterdam and a lower settlement rate because of the used material [Michigan Engineering Department, 2020]. Therefore, a few InSAR data points on parking spaces will be searched. After finding a few data points and fitting the rate with the BLUE, the reference point with the lowest RMSE will be chosen as the reference point for that location.
After determining the reference point, the data points from each of the three data sets (a set of data points with a confidence interval of 80%, 90%, or 95% to filter outliers) gotten from linking the geometry data with InSAR data will be randomly divided into two groups: a baseline set and an updated set. The amount of the baseline set will be generic and comprise 40%, 50%, or 60% of the total data points. The baseline data points will be used to calculate degradation rates within the dataset for each half-year period. Subsequently, the rates obtained from each half-year, subtracted from the rate of the reference point, will be used to fit a normal distribution, representing the stochasticity of the track settlement rates. The remaining data points will then be used to predict if high settlement rates are leading to high rates of track degradation. For this prediction, a quantile, now called the critical area, in the standard distribution will be defined, and the rates from the updated set will be evaluated if their rates exceed this critical area. Three crucial areas, 0.0125, 0.025, and 0.05, will be used. When the rate of a data point from the updated set falls below the quantile-derived threshold from the baseline set, the data point will be classified as experiencing high settlement rates. When updating the critical areas, the value of the baseline model amount will be 50 %. When updating the baseline model amount, the critical area will be 0.025. In this model, the normal distribution for data points located on the bridge and not on the bridge will be created separately. It results in a model that can predict if a data point experiences high settlement rates every half year after the first half year. For instance, when it is expected that the half-year rates from 01-01-2020 to 01-07-2020 are experiencing high settlement rates, the half-year rates from 01-01-2019 to 01-07-2019 and 01-07-2019 to 01-01-2020 will be used as the baseline set. The half-year rates from 01-01-2020 to 01-07-2020 will then be tested on this baseline set, resulting in a percentage of data points experiencing high settlement rates. Hereby, every rate will be subtracted from the half-year rate of the reference point to ensure that the model will not predict high settlement rates due to a high rate of static settlement. It will result in a model that shows that a percentage of tramway tracks experience a high rate of track degradation. When more data points lie at the lower confidence interval, there is a higher chance of relatively high track degradation rates.

3.3.2 Second approach: Prediction based on BLUE

The second approach is based on the half-year settlement rates calculated with BLUE, mentioned in Subsection 3.2. With the variance propagation law, shown in Equation 3.5 [Deakin, 2005], the deviations of the calculated rate can be calculated when it is assumed that the rate will also be the same a half-year later. A means the input matrix, $\Sigma_{\hat{x}\hat{x}}$ means the covariance of the $\hat{x}$. Because all data points can give several deviations, only the maximum deviation will be calculated from every data point. Afterwards, the maximum deviation will be selected and plotted from all the maximum deviations of the data points to give an overview.

$$\Sigma_{yy} = A\Sigma_{\hat{x}\hat{x}}A^T \tag{3.5}$$

With the InSAR deformation data and Equation 3.5, a prediction for the degradation can be created with the help of Python. The prediction can be made by calculating the settlement rate of the last half-year InSAR data of the data points. The mean settlement will be calculated and plotted in a graph with the years 2019 to 2022 for the data points on and off the bridge. The used data points not on the bridge will also be determined by calculating the highest ρ between the data points not on the bridge and GVB geometry data to select the InSAR data points which show the best likely behaviour with GVB geometry data. The data points found will be used to calculate the linear settlement and deviation with the variance propagation law. The mean trends and deviations are the predictions for the settlement to give insight into high rates of track degradation. The calculated settlement shows how the geometry will react according to this prediction approach. Because a ρ with a value -1 or 1 gives an almost linear link between the InSAR data and GVB data. Therefore, an increasing trend in InSAR data will also show a decreasing or increasing GVB data trend, dependent on a positive or negative ρ. Instead of percentages given by the stochastic rates method, the prediction method based on BLUE gives a prediction for the settlement and a deviation. The results risk providing no clue for high settlement rates without relative context. For instance, when the half-year rates from 01-01-2020 to 01-07-2020 will be predicted, the half-year rates from 01-07-2019 to 01-01-2020 will be used.

3.4 Discussion

The main research question, as defined in Chapter 1, is: Is InSAR able to provide information about the condition of the transition zone? The literature shows that various methods could be tested to answer this question. Some of the methods require physical parameters, which are not available. Then, methods that only require data are considered. Among the many options, InSAR data was reported to be noisy, so the approach to select has to deal with this feature and be simple enough for easy implementation in practice. Therefore, as described earlier, half-year InSAR rates will be used, which will be assumed to be less noisy. Then, to address the question, the two proposed approaches will be compared: the stochastic rates approach with the prediction method based on the BLUE approach.
The stochastic rates approach is based on extreme value analysis, which analyses the probability of occurrence of an event based on the occurrence of all the earlier events [Charras-Garrido and Lezaud, 2013]. Because it has been assumed that the settlement rates are linear, linear rates will be used for events as explained, and the current rate will be tested with those events. The BLUE approach is also based on the assumption that the settlement rates are linear. Plotting BLUE overtime may show dips, which can indicate high rates of track degradation.
The decision has been made not to use machine learning methods. Because the assumption has been made that the problem is linear, linear models will be used. When a problem is defined as a linear problem, simplified models can be better than complex machine learning models [Roccetti et al., 2019]. Moreover, machine learning techniques can cause overfitting of the noisy InSAR data [Dietterich, 1995].
Both built models will be applied to uncertainty tests to determine which model works the best under the most significant uncertainties. In approach 1, the standard deviation of the data points rates used to calculate the normal distribution of the stochastic rates will be used to test how accurate approach 1 is. The prediction method based on BLUE will use the maximum predicted deviation of the settlement rate of all the data points. The deviation can also be used as an uncertainty test to compare approach 2 with approach 1.
In addition, the condition of the transition zone can be indicated with more datasets, like temperature distribution [Zeng et al., 2019]. Another dataset will be used to give better insight into the transition zone condition, which is a dataset of the mean yearly speed in the tramway track transition zones. The mean speed was measured in 2021 and 2022 by GVB. Unfortunately, the speeds of 2019 and 2020 were unavailable in the GVB data. According to Goodarzi et al. (2022), speed restrictions can be used as corrective maintenance to rectify rail defects. In his thesis, Peteroff (2021) aimed for a method to create a velocity distribution based on the detected speeds of geometrical points. In the process, every geometrical point has been linked to track points, leading to an average speed of each trackpoint every year. The thesis has led to average speeds of each trackpoint from 2021 to 2024, which can be used in the research. Knowing that a lower speed can mean a need for maintenance, a lower relative speed will be assumed to indicate high rates of track degradation. However, the speed data are still not validation data because they do not validate that low speeds appear because of track irregularities caused by track degradation due to high settlement rates. The lower speeds can also occur because of the rail wear [Soleimanmeigouni, A. Ahmadi, and U. Kumar, 2018], which falls out of the scope of this research.
In addition, an overview of the conditions of each transition zone will be created. At each location, the GVB twist, track gauge, mean speed data, and the stochastic rates and BLUE outcomes will be shown with a colour to show the condition according to the data. For GVB track gauge and twist data, if the sigma shows a higher value than usual in a year according to the rest of the data, the data of that year will be detected as an orange colour due to the deviation. If a sigma from the datapoints of a year not on the bridge differs from the sigma from that year on the bridge, the data of that year will also be detected as an orange colour. Otherwise, the data of that year will be detected as green. If the speed plot in 2021 shows a lower mean speed than 2022, the speed for 2021 will be detected as a red colour. Otherwise, it will be detected as a green colour. If the stochastic rates or BLUE model detect for one year one time half-yearly high rates, the colour of that year will be yellow. When both halves of the year are detected, the colour will be red. Otherwise, the colour will be green.
Both models have been put into a framework for better insight, as shown in Figure 3.1.

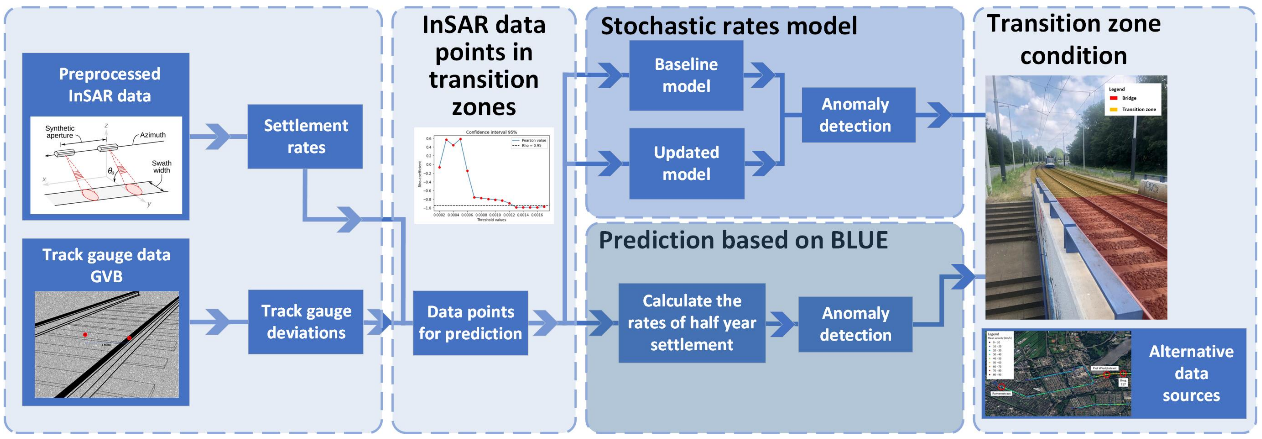


Figure 3.1: Framework for both approaches

Chapter 4

Results

4.1 Track degradation in transition zones in Amsterdam

According to research on railway track structure, high rates of track degradation will appear more on ballast railway tracks than on slab railway tracks. However, on soft soils, high rates of track degradation can happen at high speeds also on slab tracks [Madshus and Kaynia, 2000]. Because in The Netherlands, soil textures from fine sands to clays do occur [Bakker, 1979], using slab tracks can lead to high rates of track degradation, which makes every transition zone unique in its behaviour. Assuming high rates of track degradation will also occur in the tramway network in transition zones, a few locations in Amsterdam, shown in Figure 4.1, will be proposed to investigate high rates of track degradation.

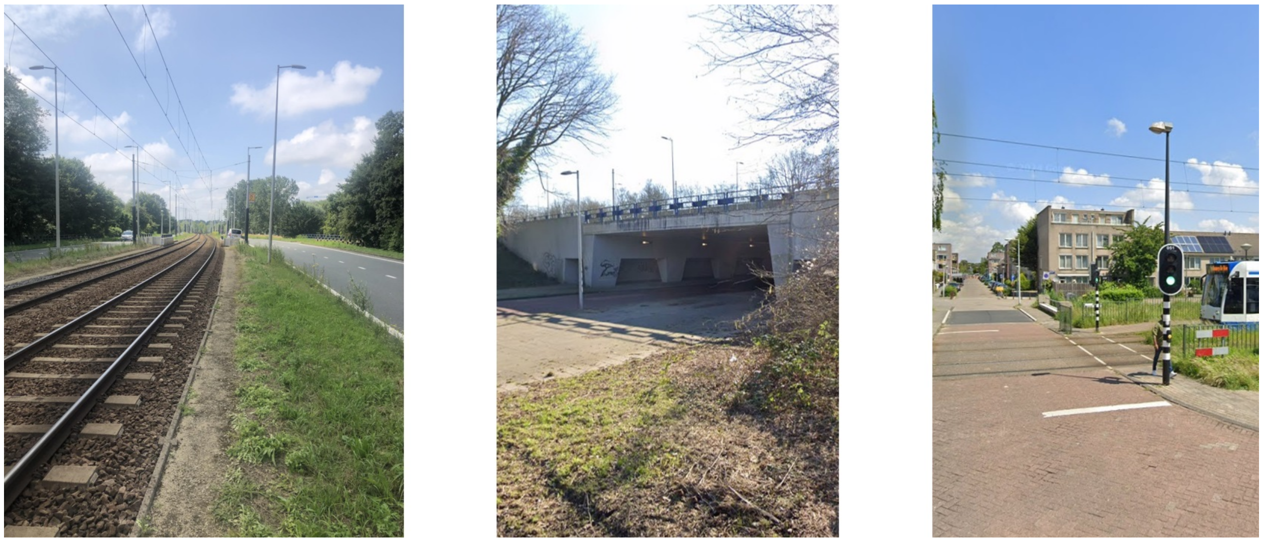


Figure 4.1: The three proposed locations probably experience high rates of track degradation, Piet Wiedijkstraat, Brug 717, and Romerostraat (third picture: Google Maps, 2024).

The Cornelis Lelylaan has a ballast track, and the tram experiences a few bridges. One is at the Piet Wiedijkstraat, as shown on the left in Figure 4.1. The Piet Wiedijkstraat contains a bridge of about 9 m. According to Arnoult (1986), when a bridge has a span of more than 6 m, it is called a bridge; otherwise, it is a culvert, which is not the case here [Arnoult, 1986]. The data from Bodemdalingskaart.nl are shown in Figure 4.2. The shaded red and blue areas represent the area of the InSAR data points used. The data points on the edge will not be used because they are not part of the ballast track. All the data points on the bridge that are part of the tramway will be used in the blue area. Therefore, the data points on both roads will not be used. Figure 4.2 shows an uneven distribution of InSAR data points that can affect research results. The whole transition zone cannot be analysed because data points do not represent several areas of the tramline. Therefore, the behaviour of some parts of the transition zone will not be considered in the research, which makes the analysis and the model less realistic with the actual behaviour of the transition zone.

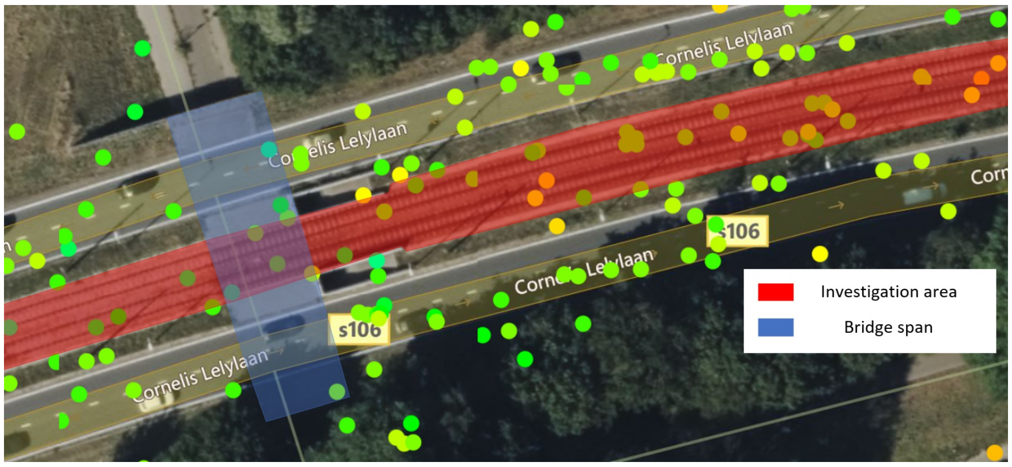


Figure 4.2: InSAR data from Bodemdalingskaart.nl at Piet Wiedijkstraat.

Using the InSAR data of Bodemdalingskaart.nl of about 100 meters from both sides of the bridge, filtering out the outliers with the boxplot method explained in Subsection 3.2. Applying the data errors as their covariance, a scatter plot can be created in Figure 4.3 with a coordinate system WGS84. The outfiltering of data points that fall outside the confidence intervals based on their RMSE has not been applied yet. In this plot, the bridge is shown as a red rectangle, which means that every data point in this rectangle is a data point on the bridge. According to Wang et al. (2018) and Hu et al. (2019) the position accuracy of InSAR data is 1 meter. Therefore, the area of the rectangle has been expanded 1 meter to the left, the right, upwards, and downwards [Dheenathayalan et al., 2016]. The data points not in the rectangle represent those not on the bridge. The blue line represents the tram lines on the Cornelis Lelylaan.

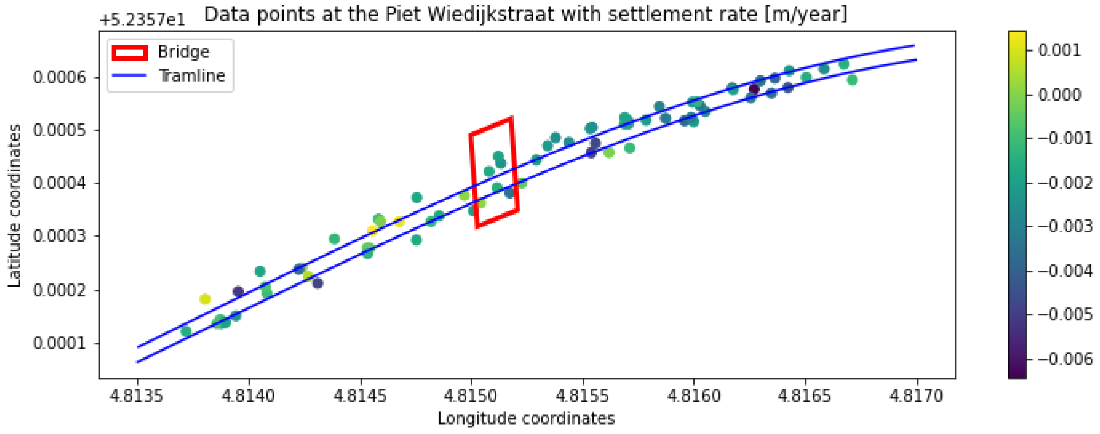


Figure 4.3: Datapoints at the Piet Wiedijkstraat.

Looking east of the Piet Wiedijkstraat bridge, another bridge is positioned at the Louis Bouwmeesterstraat, called Brug 717. The bridge is shown in the middle in Figure 4.1. Brug 717 is a bridge of 16.5 meters. It spans the Louis Bouwmeesterstraat. In Figure 4.4, the InSAR points placed in the area of Brug 717 are shown. Also, in this case, the bridge has been expanded by 1 meter due to InSAR's assumed 1 meter precision accuracy [Dheenathayalan et al., 2016].

The Romerostraat crosses the tramline on an integrated slab track of about 11.9 m. Before and after this integrated slab track, the tram line rides over a green embankment line. The bridge which will be analysed is shown in Figure 4.1. The Romerostraat consists of a track integrated with grass and a slab track to cross the Romerostraat and the tram line. Figure 4.5 shows the InSAR points placed at the Romerostraat. Also, in this case, the bridge has been expanded by 1 meter due to InSAR's assumed 1 meter precision accuracy

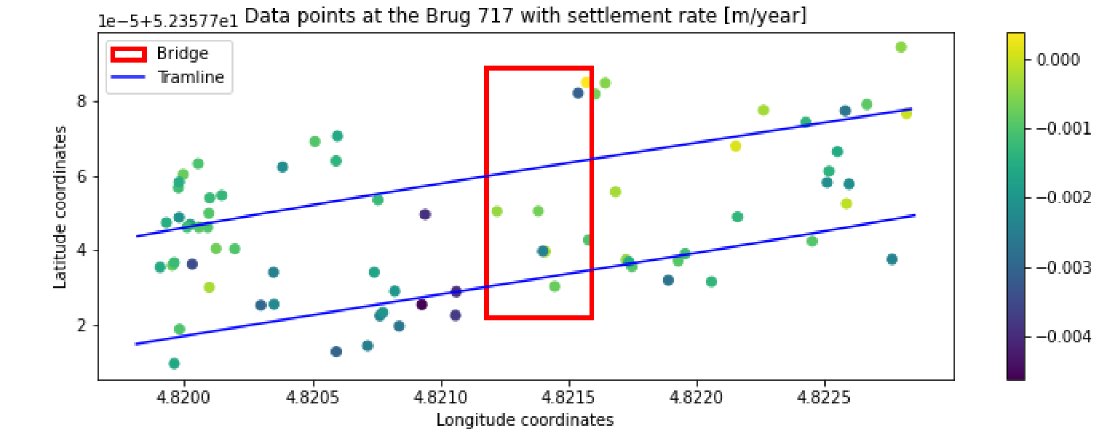


Figure 4.4: Datapoints at Brug 717.

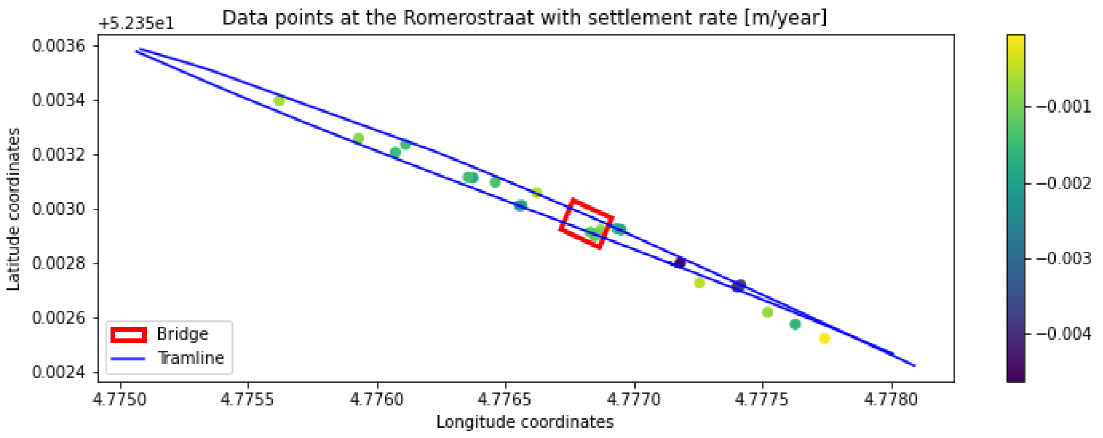


Figure 4.5: Datapoints at the Romanostraaat.

[Dheenathayalan et al., 2016].

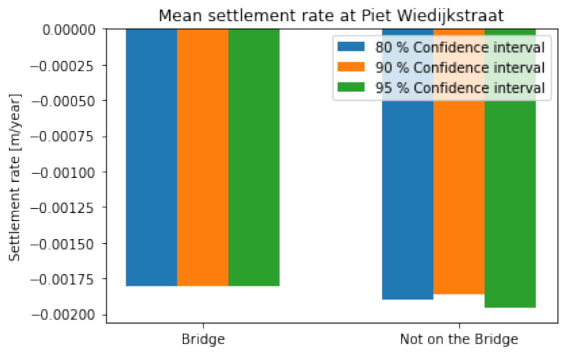
To give a visualised overview of the settlement rates, Figure 4.6 shows the mean settlement rate from 2017 to 2022 for the data points on the bridge and not in the bridge. As seen in Figure 4.6, at all the locations, the mean settlement rate, according to InSAR data, is lower on the bridge than off the bridge. The assumption that different rates appear on the bridge and not in the bridge, as shown in Subsection 3.3.1, is valid according to Figure 4.6. The Python code to calculate the in Figure 4.6 mean settlement rates is shown in Appendix A.

In addition, the deformation over time of all the data points has been averaged to find a difference in the pattern of the behaviour between the data points on the bridge and those not on the bridge. The results of average deformation data and the Python code to calculate these data are shown in Appendix B.

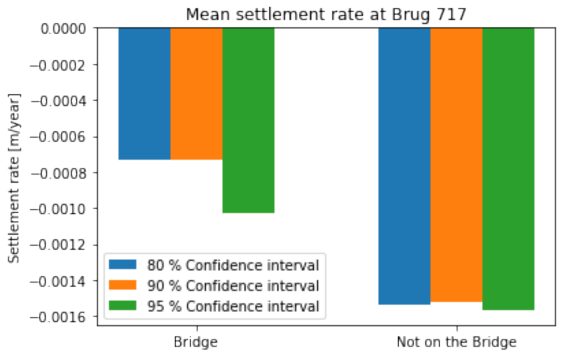
Piet Wiedijkstraat and Brug 717 are located on the same tramway track. The Romerostraat is a branch of the tramline that runs on the Piet Wiedijkstraat and at location Brug 717. In Figure 4.7, a plot with the locations of the three proposed locations with the mean velocities of 2021 on the tramline locations is shown.

4.2 Link between GVB geometry data and InSAR data

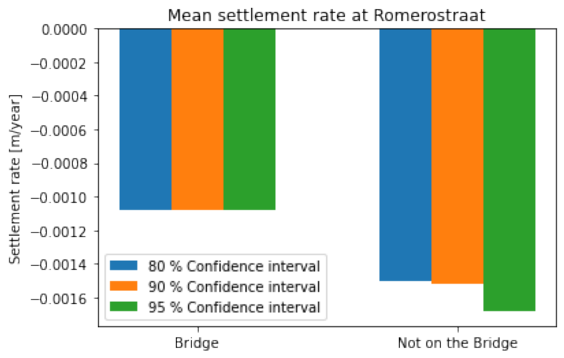
Applying the methodology described in Section 3.3 for linking InSAR data with the GVB track gauge data to Piet Wiedijkstraat gives the most critical data points that best fit the GVB data. The PCC has been



(a) Mean settlement rates at the Piet Wiedijkstraat with using confidence intervals of 80 %, 90 %, and 95 %.



(b) Mean settlement rates at Brug 717 with using confidence intervals of 80 %, 90 %, and 95 %.



(c) Mean settlement rates at the Romerostraat with using confidence intervals of 80 %, 90 %, and 95 %.

Figure 4.6: Optimizing data points and link between track gauge and settlement rates for the Piet Wiedijkstraat.

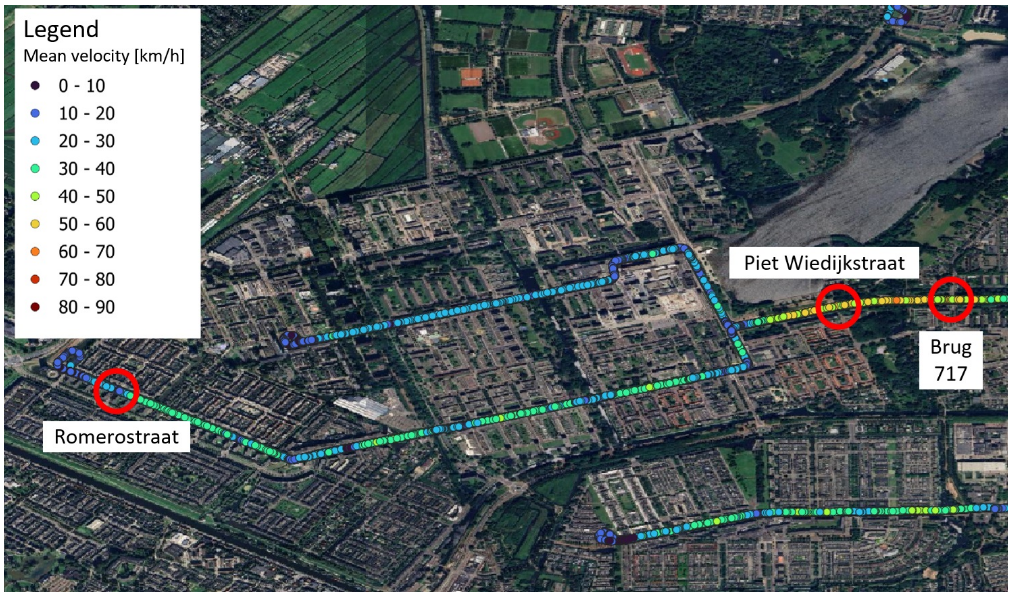
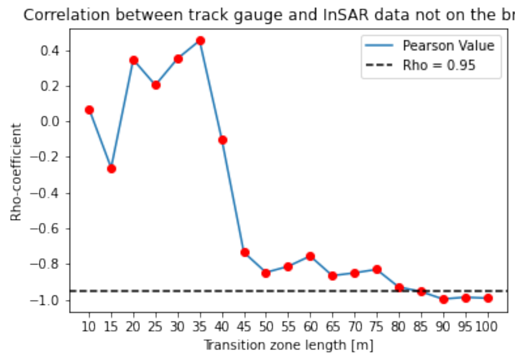
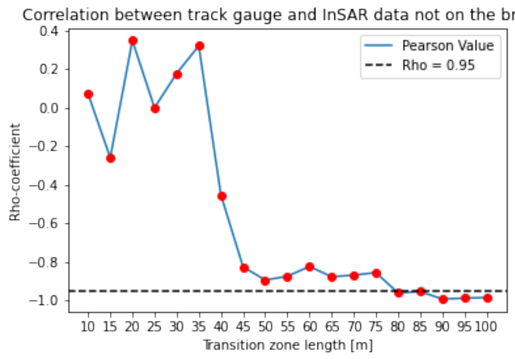


Figure 4.7: Overview of the three proposed locations with the mean velocities of the trams in 2021.

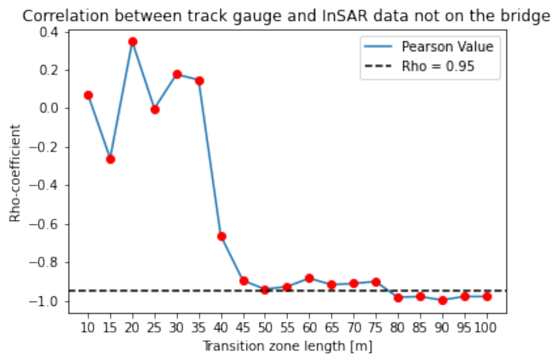
used on the data, not on the bridge, because the data on the bridge will not change with other transition zone lengths. Since it has been mentioned in Subsection 3.2 that data points will be out-filtered with the RMSE of the BLUE of each data point. At the Piet Wiedijkstraat, the Python code has been applied three times with out-filtering the data points with confidence intervals of 80 %, 90 %, and 95 %. Figure 4.8 shows the outcomes of out-filtering all the data points. The Python script, written to separate the gauge deviation values into datasets with data points on and off the bridge, is shown in Appendix C.



(a) Rho-coefficients of settlement rates of data points not on the bridge per transition zone length, when an out-filtering data points confidence interval of 80% has been applied.



(b) Rho-coefficients of settlement rates of data points not on the bridge per transition zone length, when an out-filtering data points confidence interval of 90% has been applied.



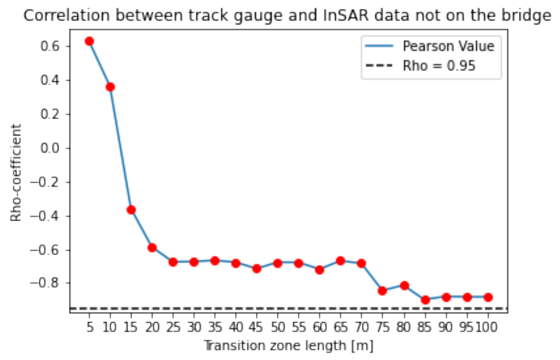
(c) Rho-coefficients of settlement rates of data points not on the bridge per transition zone length, when an out-filtering data points confidence interval of 95% has been applied.

Figure 4.8: Optimizing data points and link between track gauge and settlement rates for the Piet Wiedijkstraat.

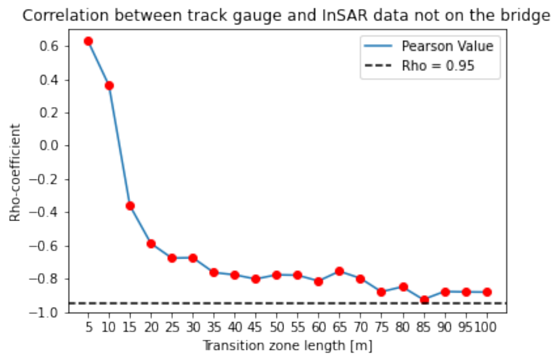
A threshold value of $\rho = -0.95$ has also been plotted in every plot as shown in Figure 4.8 because, in the first calculations, many ranges lay under this value, so a regime for selecting an optimum transition zone length is needed. As mentioned in Subsection 3.3, the algorithm for choosing the optimised transition zone length picks the lowest ρ. The transition zone length of 90 meters has been selected in all the different confidence intervals because this length has in all the cases the lowest ρ. In Table 4.1, all the correlation coefficients when different confidence intervals have been used are shown between the rates of the InSAR data and the sigmas of GVB data. Table 4.1 shows that the previously described assumed link between gauge deviation and InSAR data is almost entirely present in the data, not from the bridge.

Table 4.1: Calculated PCC and amount of datapoints at Piet Wiedijkstraat

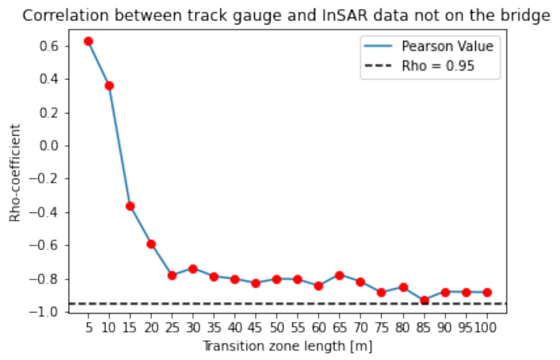
	$PCC_{\sigma-\hat{r},\, bridge}$	$PCC_{\sigma-\hat{r},\, not\, on\, the\, bridge}$	Data points
Confidence interval 80 %	-0.72	-0.99	57
Confidence interval 90 %	-0.72	-0.99	61
Confidence interval 95 %	-0.72	-0.99	63



(a) Rho-coefficients of settlement rates of data points not on the bridge per transition zone length, when an out-filtering data points confidence interval of 80 % has been applied.



(b) Rho-coefficients of settlement rates of data points not on the bridge per transition zone length, when an out-filtering data points confidence interval of 90 % has been applied.



(c) Rho-coefficients of settlement rates of data points not on the bridge per transition zone length, when an out-filtering data points confidence interval of 95 % has been applied.

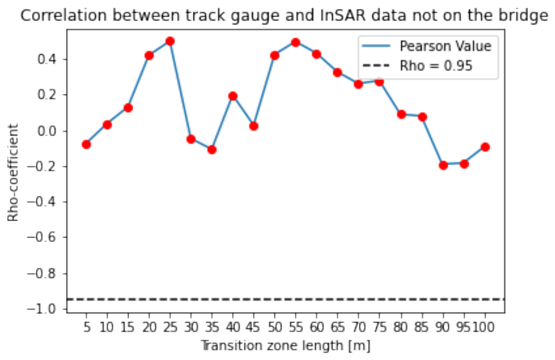
Figure 4.9: Optimizing data points and link between track gauge and settlement rates for the location Brug 717.

The optimisation graphs based on the transition zone lengths at Brug 717 are shown in Figure 4.9. Looking at Figure 4.9, it can be concluded that a transition zone length of 85 meters has the most points with the lowest ρ. The calculated parameters and the number of data points are shown in Table 4.2. Remarkably, the rates of the bridge and the sigmas on the bridge correspond with a positive linear link between them.

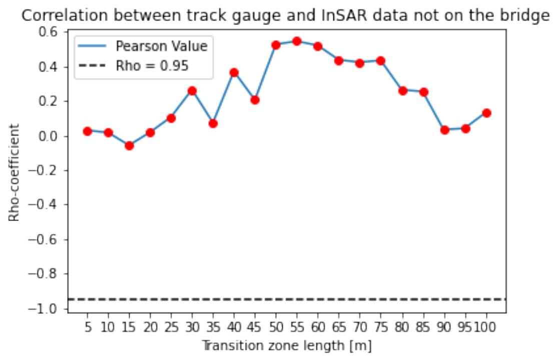
Table 4.2: Calculated parameters Brug 717

	$PCC_{\sigma-\hat{r},\, bridge}$	$PCC_{\sigma-\hat{r},\, not\, on\, the\, bridge}$	Data points
Confidence interval 80 %	0.97	-0.90	59
Confidence interval 90 %	0.97	-0.92	61
Confidence interval 95 %	0.96	-0.93	63

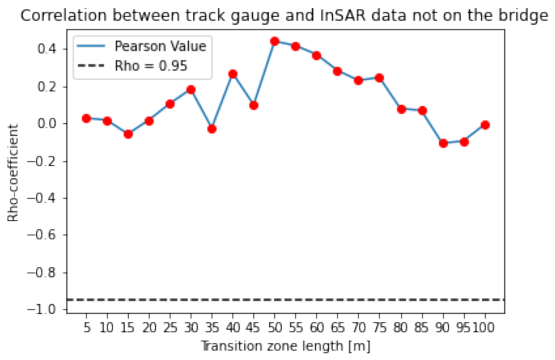
The optimisation graphs based on the range of the data points at the Romerostraat are shown in Figure 4.10. The most remarkable phenomenon discovered in the three graphs is that none of the ranges shows a clear negative linear relationship between GVB data and InSAR data. Since it has been assumed that high values of standard deviations of the track gauge data mean high settlement rates and negative InSAR data, it can be concluded that the proposed theory does not apply to this case. When using a confidence interval of 80 % and 95 %, a transition zone length of 90 meters gives the lowest ρ. When using a 90 % confidence interval, an opposite transition zone length of 15 meters gives the lowest ρ. The calculated parameters are



(a) Rho-coefficients of settlement rates of data points not on the bridge per transition zone length, when an out-filtering data points confidence interval of 80% has been applied.



(b) Rho-coefficients of settlement rates of data points not on the bridge per transition zone length, when an out-filtering data points confidence interval of 90% has been applied.



(c) Rho-coefficients of settlement rates of data points not on the bridge per transition zone length, when an out-filtering data points confidence interval of 95% has been applied.

Figure 4.10: Optimizing data points and link between track gauge and settlement rates for the Romerostraat.

shown in Table 4.3. The confidence interval 90 % gives many fewer data points than the other confidence intervals due to the described lower ρ in that transition zone length.

Table 4.3: Calculated parameters Romerostraat

	$PCC_{\sigma-\hat{r},\,bridge}$	$PCC_{\sigma-\hat{r},\,not\,on\,the\,bridge}$	Data points
Confidence interval 80 %	0.86	-0.18	20
Confidence interval 90 %	0.86	-0.06	7
Confidence interval 95 %	0.86	-0.10	22

4.3 Track gauge and settlement

Figure 4.11 shows the results of the track gauge and BLUE settlement rates in a transition zone length of 90 meters, outfiltered data points with a confidence interval of 95%. In the plot, it is shown that, generally, when the settlement rate is high, the sigma of the track gauge is high. The pattern described should be expected because when settlement is high, high rates of track degradation and track deterioration will appear, resulting in more significant track gauge deviations [Ahac and Lakusic, 2015].

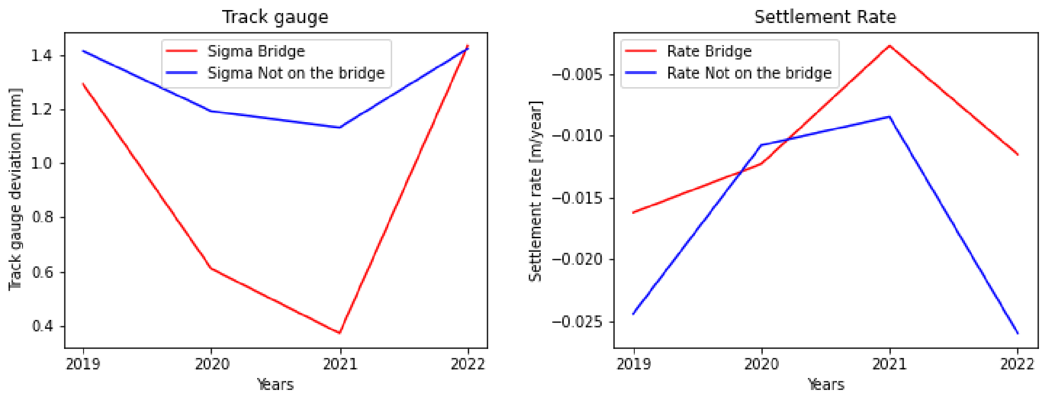


Figure 4.11: Link between GVB track gauge data and InSAR data at Piet Wiedijkstraat on the bridge and not on the bridge in a transition zone with a length of 90 meters with out-filtering data points with a confidence interval of 95%.

The results of the calculated optimum transition zone length of 85 meters with a confidence interval of 95% at Brug 717 are shown in Figure 4.12. However, it is remarkable that the rates of the bridge and the sigmas on the bridge correspond with a positive linear link between each other. At a confidence interval of 80% and 90%, a range of 75 meters has been used, which gives more data points when using a higher confidence interval instead of fewer data points at the location Piet Wiedijkstraat. Another remarkable phenomenon discovered in Figure 4.12 is that the settlement rate of 2020 of the data points on the bridge, which are higher (higher negative values) than the settlement of the data points not on the bridge. The settlement rates in Figure 4.12 are affected by the fact that the data points on the bridge do not show the same behaviour as the data points on the bridge at the Piet Wiedijkstraat.

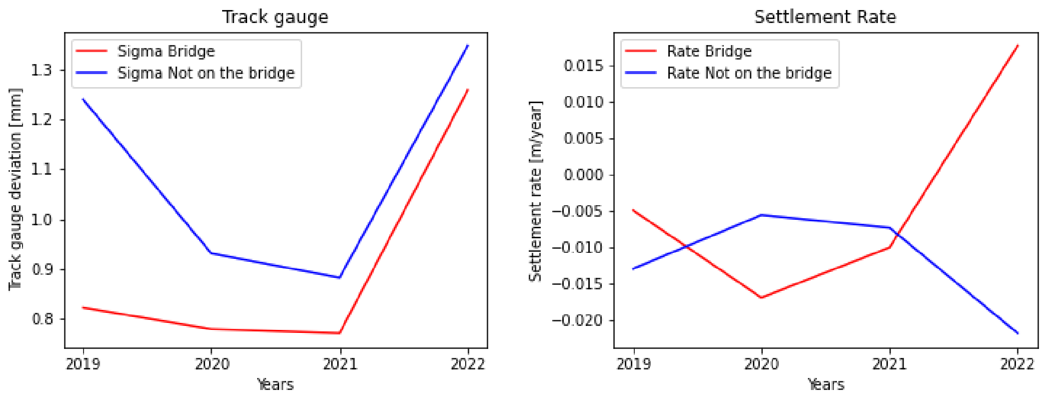


Figure 4.12: Link between GVB track gauge data and InSAR data at Brug 717 on the bridge and not on the bridge in a transition zone with a length of 85 meters with out-filtering data points with a confidence interval of 95 %.

The results of the calculated optimum transition zone length of 90 meters with a confidence interval of 80 % at the Romerostraat are shown in Figure 4.13. As seen in Figure 4.12 where the settlement rate of the data points on the bridge of the year 2020 is much higher than the rate of the data points not on the bridge, this phenomenon has also been discovered in Figure 4.13 for the years 2020 and 2022. Therefore, the settlement rates in Figure 4.13 are also affected by the fact that the data points on the bridge do not show the same behaviour as the data points on the bridge at the Piet Wiedijkstraat.

As mentioned in Subsection 3.3, standard transition zone lengths of 5, 10, 15, and 20 meters will be used to compare the link between track gauge and InSAR data with the variable transition zone approach. At

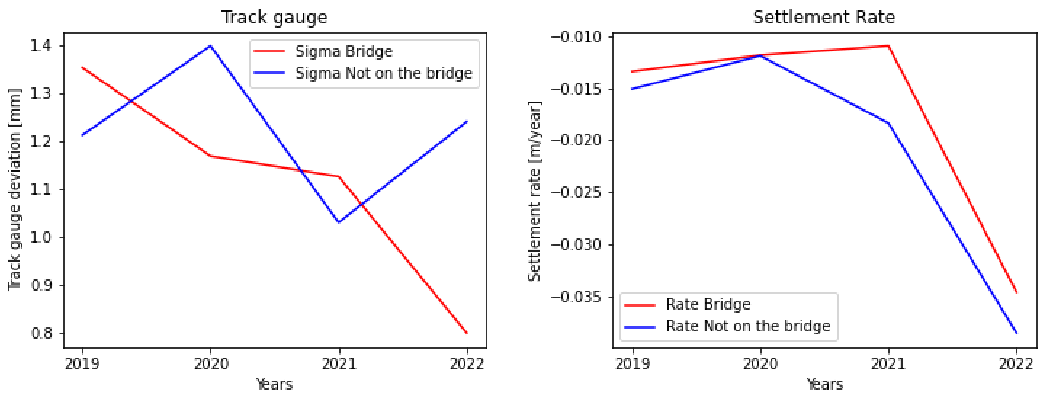


Figure 4.13: Link between GVB track gauge data and InSAR data at the Romerostraat on the bridge and not on the bridge in a transition zone with a length of 90 meters with out-filtering data points with a confidence interval of 80 %.

the Piet Wiedijkstraat, InSAR data points are unavailable when using a transition zone length of 5 meters. Therefore, standard transition zone lengths of 10, 15, and 20 meters have been used, leading to the correlation coefficients shown in Table 4.4. The strong negative linear relationship between the yearly settlement rates and the yearly track deviation shown in Table 4.1 is not shown in Table 4.4. The fewer available InSAR data points compared with a transition zone length of 90 meters can cause less well PCC. Also, Figure 4.14 shows a settlement rate pattern which is not recognisable with the pattern from Figure 4.11. On the other hand, the pattern of the track gauge in Figure 4.14 is similar to the pattern in Figure 4.11. Therefore, using a standard transition zone length differs from optimising the transition zone length with the correlation between track gauge data and InSAR data at the Piet Wiedijkstraat.

Table 4.4: Calculated parameters Piet Wiedijkstraat with a transition zone length of 10, 15 and 20 meters.

	$l_{transition\ zone}$ [m]	$PCC_{\sigma-\hat{r},bridge}$	$PCC_{\sigma-\hat{r},\ not\ on\ the\ bridge}$	Data points
CI 80 %	10	-0.72	0.07	9
	15	-0.72	-0.26	10
	20	-0.72	0.35	12
CI 90 %	10	-0.72	0.07	9
	15	-0.72	-0.26	10
	20	-0.72	0.35	12
CI 95 %	10	-0.72	0.07	9
	15	-0.72	-0.26	10
	20	-0.72	0.35	12

Figure 4.15 shows the link between GVB track gauge data and InSAR data at Brug 717 for the transition zones when lengths of 5, 10, 15, and 20 meters are used with a confidence interval of 95 %. The settlement rates when using a transition zone length of 5 meters are too high to show in the settlement rate plot. Therefore, only the settlement rates of the transition zones with lengths of 10, 15, and 20 meters are shown. The track gauge shows a different pattern than the track gauge in Figure 4.12. However, the settlement rates pattern looks almost the same as in Figure 4.12. Only the mean rate in 2020 of the data points not on the are lower than the mean rate of the data points on the bridge. The PCC of all the confidence intervals and the generic transition zone lengths are shown in Table 4.5. The table shows that a longer transition zone length gives a better link between GVB track gauge data and InSAR data. However, the optimised PCC value in Table 4.2 has not been reached.

Figure 4.16 shows the pattern of GVB track gauge data and InSAR data for transition zone lengths of 5, 10, 15, and 20 meters with a confidence interval of 80 %. The settlement rates with a transition zone length of five meters give too high rates, as in Figure 4.15. Therefore, the settlement rates for a transition zone length of five meters are also not shown in Figure 4.16. Most of the settlement rates give the same

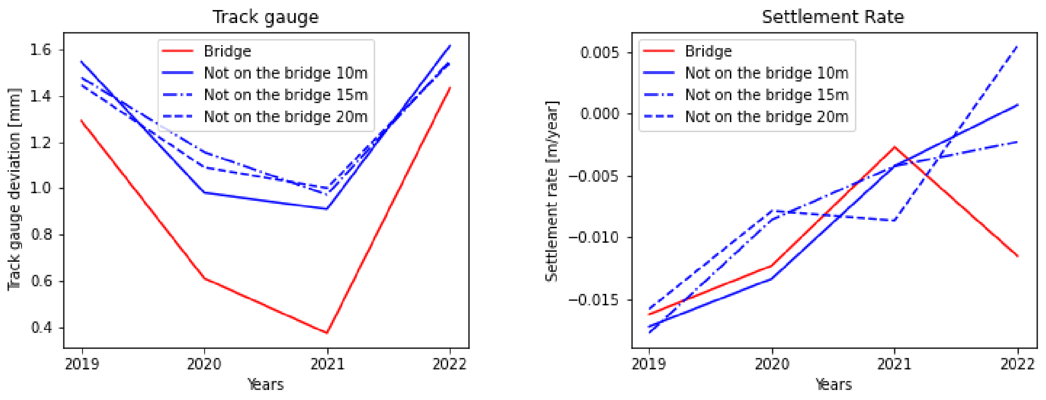


Figure 4.14: Link between GVB track gauge data and InSAR data at the Piet Wiedijkstraat on the bridge and not on the bridge with a transition zone length of 10, 15, and 20 meters with out-filtering data points with a confidence interval of 95 %.

Table 4.5: Calculated parameters Brug 717 with a transition zone length of 5, 10, 15 and 20 meters.

	$l_{transition\ zone}$ [m]	$PCC_{\sigma-\hat{r},bridge}$	$PCC_{\sigma-\hat{r},\ not\ on\ the\ bridge}$	Data points
CI 80 %	5	0.97	0.63	8
	10	0.97	0.36	9
	15	0.97	-0.36	15
	20	0.97	-0.59	16
CI 90 %	5	0.97	0.63	8
	10	0.97	0.36	9
	15	0.97	-0.36	15
	20	0.97	-0.59	16
CI 95 %	5	0.96	0.63	9
	10	0.96	0.36	10
	15	0.96	-0.36	16
	20	0.96	-0.59	17

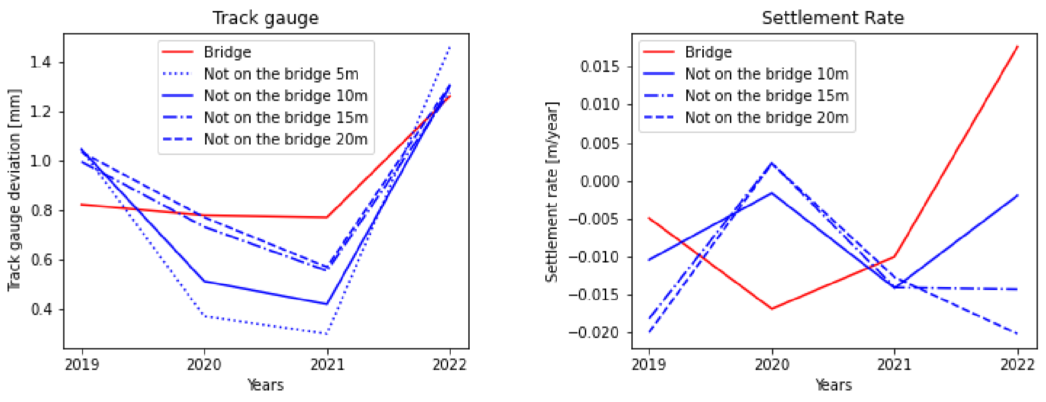


Figure 4.15: Link between GVB track gauge data and InSAR data at Brug 717 on the bridge and not on the bridge with a transition zone length of 5, 10, 15, and 20 meters with out-filtering data points with a confidence interval of 95 %.

pattern as Figure 4.13. The pattern of GVB track gauge looks almost the same as in Figure 4.13. The transition zone with a length of 5 meters gives a different pattern, showing a dip in 2020 instead of a peak in 2020.

Table 4.6: Calculated parameters Romerostraat with a transition zone length of 5, 10, 15 and 20 meters.

	$l_{transition\,zone}$ [m]	$PCC_{\sigma-\hat{r},bridge}$	$PCC_{\sigma-\hat{r},\,not\,on\,the\,bridge}$	Data points
CI 80 %	5	0.86	-0.07	5
	10	0.86	0.04	5
	15	0.86	0.13	6
	20	0.86	0.42	8
CI 90 %	5	0.86	0.03	6
	10	0.86	0.17	6
	15	0.86	-0.06	7
	20	0.86	0.02	9
CI 95 %	5	0.86	0.03	6
	10	0.86	0.02	6
	15	0.86	-0.06	7
	20	0.86	0.02	9

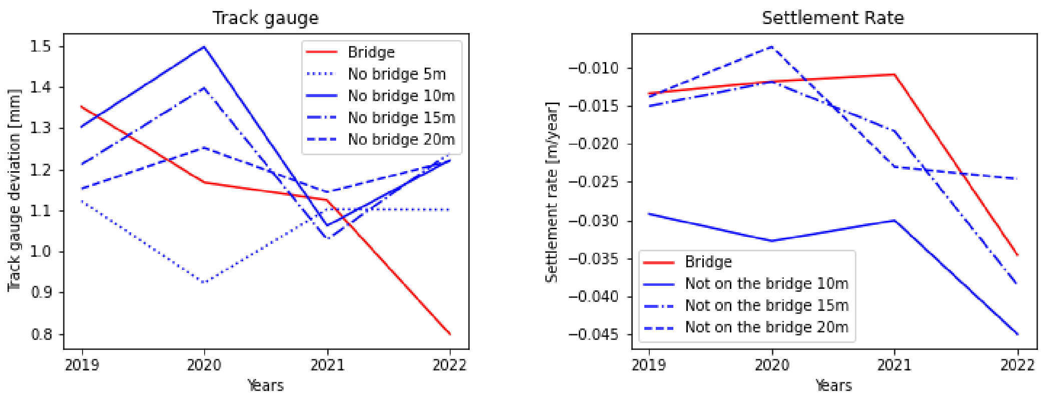


Figure 4.16: Link between GVB track gauge data and InSAR data at Romerostraat on the bridge and not on the bridge with a transition zone length of 5, 10, 15, and 20 meters with out-filtering data points with a confidence interval of 95 %.

4.4 Twist data

Figure 4.17 shows three plots of GVB twist data at Piet Wiedijkstraat with a transition zone length of 90 meters. The three plots shown in Figure 4.11 represent each other's distances between twist measurements, 2 m, 4 m, and 8 m. Unfortunately, data from 2021 on the bridge is not available. In Figure 4.17, it is shown that, as also seen in Figure 4.11, in 2021, a dip is shown, which indicates low rates of track degradation. For this case, the track gauge has been strengthened with twist data as an appropriate parameter to predict high geometry deterioration. The Python script applied for the twist data is shown in Appendix D.

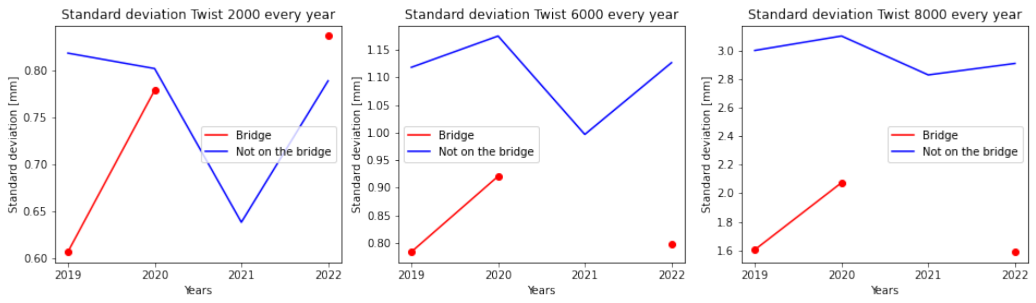


Figure 4.17: Twist data from 2019 and 2022 at the Piet Wiedijkstraat

In Figure 4.18, the standard deviations for three different distances of twist measurements on location Brug 717 are shown to compare it with the track degradation. The results are calculated for GVB data points with a transition zone length of 85 meters. It is shown that, as also seen in Figure 4.12, in 2021, a dip indicates low track degradation rates. However, after the dip in 2021, the twist does not go up in two of the three plots shown in Figure 4.12. Therefore, the track gauge is not strictly well validated with the twist for this location.

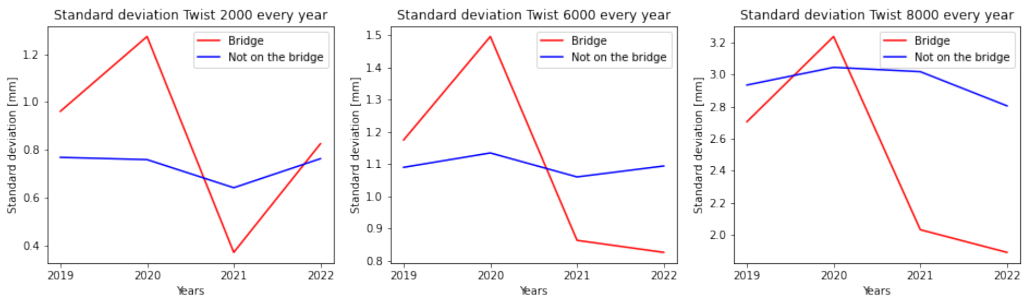


Figure 4.18: Twist data from 2019 and 2022 at location Brug 717

In Figure 4.19, the standard deviations for three different distances of twist measurements on the Romerostraat are shown to compare them with the track degradation. The results are calculated for GVB data points with a transition zone length of 90 meters. Unfortunately, data from 2019 is missing. It is shown that, as also seen in Figure 4.13, linear progress is shown with the deviations not on the bridge. However, in 2021 in Twist 2000, a dip is shown, indicating low rates of track degradation, which is not shown in Figure 4.13. Therefore, the track gauge is not strictly validated with the twist for this location.

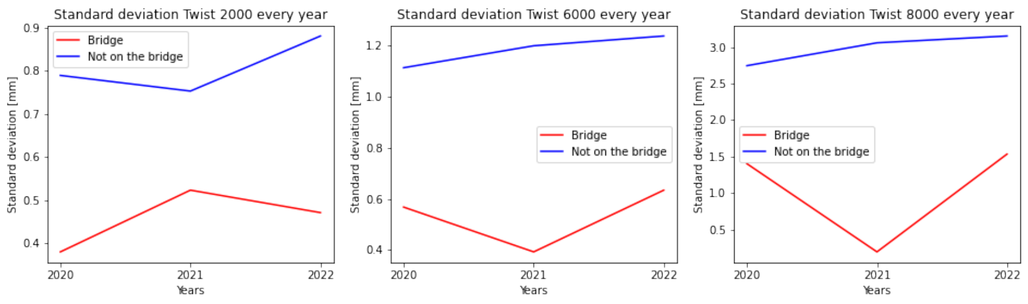


Figure 4.19: Twist data from 2020 to 2022 at Romerostraat

The results of the GVB twist data for the used transition zone length from picking the lowest ρ in Section 4.2 have been compared with using standard transition zone lengths of 5, 10, 15, and 20 meters. Figure 4.20 shows the standard deviations of the twist data on the bridge and in the transition zones with lengths of 5, 10, 15, and 20 meters at the Piet Wiedijkstraat. Figure 4.20 shows almost the same pattern as Figure 4.17. There are little differences in twist data between the transition zones in Figure 4.20. However, the twist data in Figure 4.20 show a fraction higher standard deviations.

Figure 4.21 shows the standard deviations of the twist data on the bridge and in the transition zones with lengths of 5, 10, 15, and 20 meters at Brug 717. Figure 4.21 shows twist standard deviation values which are a fraction lower than in Figure 4.18. Between the transition zone lengths, the patterns look almost the same.

Figure 4.22 shows the standard deviations of the twist data on the bridge and in the transition zones with lengths of 5, 10, 15, and 20 meters at the Romerostraat. Figure 4.22 shows twist standard deviation values which are a fraction lower than in Figure 4.22. Between the transition zone lengths, the patterns look almost the same.

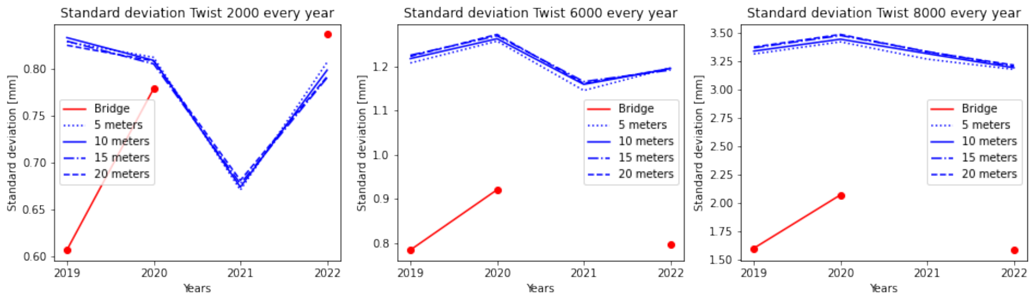


Figure 4.20: Twist data from 2019 to 2022 with different transition zone lengths at the Piet Wiedijkstraat.

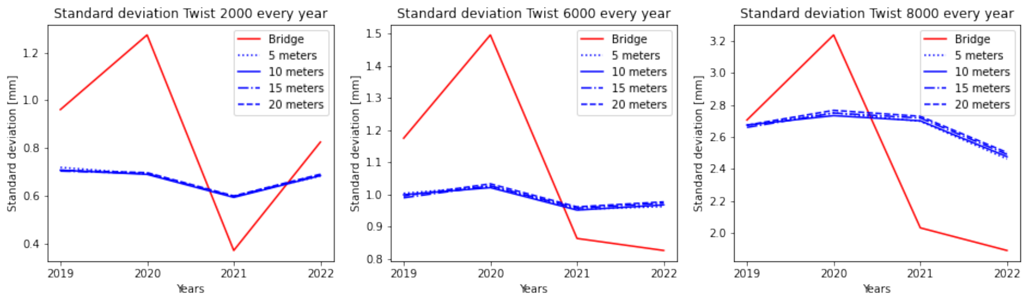


Figure 4.21: Twist data from 2019 to 2022 with different transition zone lengths at Brug 717.

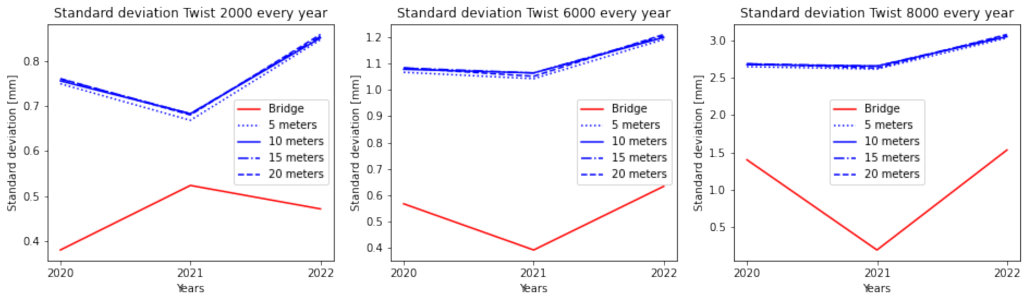


Figure 4.22: Twist data from 2020 to 2022 with different transition zone lengths at the Romerostraat.

4.5 Stochastic rates

Before using the stochastic rates approach, a reference point near the investigation area was looked for to compare the deformation with the static settlement of Amsterdam. As shown in Figure 4.23, four points have been determined as reference points near the Piet Wiedijkstraat, which is positioned on a parking place.
Table 4.7 shows the results of the stochastic rates approach for every confidence interval at the Piet Wiedijkstraat. In this table, two high peaks of the stochastic rates model have been demonstrated with the standard deviation of the baseline set of data points on the bridge and the data points not on the bridge. For the results, the standard deviation of the set of data points on the bridge is more important because of their expanded length compared with the set of data points on the bridge. Therefore, when looking for optimum model parameters, a low standard deviation value of the baseline set not on the bridge has a higher priority than a low standard deviation of the baseline set on the bridge. Appendix E shows the whole Python code applied for this method.
In general, Piet Wiedijkstraat's locations, 2019-01-01 and 2021-07-01, are where high settlement rates have been experienced. For 2019-01-01, there is a relatively high peak (50 %) and a relatively low standard deviation when a confidence interval of 95 %, a critical area of 0.025, and a baseline model amount of 50 % have been applied. However, for the date 2021-07-01, there is a relatively high peak (50 %) and a relatively low standard deviation when a confidence interval of 90 %, a critical area of 0.025, and a baseline model amount

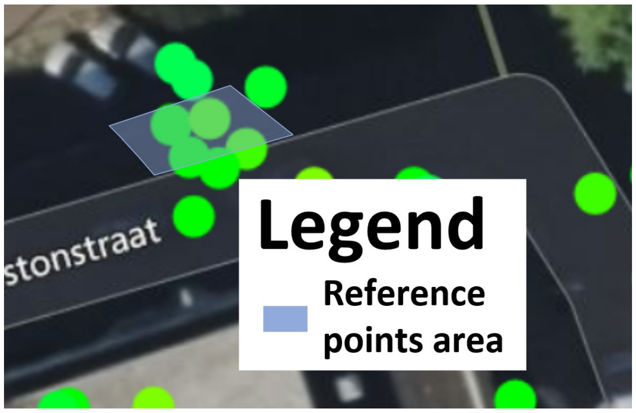


Figure 4.23: Reference point Piet Wiedijkstraat.

Table 4.7: Results of the stochastic model applied at the Piet Wiedijkstraat with different critical areas of 0.0125, 0.025, and 0.05 and different baseline model amounts of 40 %, 50 %, and 60 %. The general values (values which do not change when the other parameters have changed) are for the baseline model amount 50 %, and for the critical area 0.025.

		Peak 1	% Peak 1	σ Peak 1	Peak 2	% Peak 2	σ Peak 2
CI 80%	$\alpha = 0.0125$	2019-01-01	16.67 %	0.003712 and 0.005986	2020-07-01	16.67 %	0.004553 and 0.007312
	$\alpha = 0.025$	2019-01-01	33.33 %	0.003024 and 0.008621	2019-07-01	33.33 %	0.004956 and 0.007229
	$\alpha = 0.05$	2019-01-01	16.67 %	0.003387 and 0.007336	2020-07-01	16.67 %	0.006139 and 0.006533
	$n = 0.4$	2019-01-01	25.0 %	0.003703 and 0.007357	2021-07-01	62.5 %	0.005262 and 0.006547
	$n = 0.5$	2019-01-01	16.67 %	0.003387 and 0.006500	2019-07-01	33.33 %	0.005159 and 0.005459
	$n = 0.6$	2020-07-01	33.33 %	0.007256 and 0.007272	2022-07-01	16.67 %	0.008574 and 0.008223
CI 90%	$\alpha = 0.0125$	2019-01-01	33.33 %	0.001912 and 0.005486	2021-07-01	33.33 %	0.003536 and 0.007456
	$\alpha = 0.025$	2019-07-01	16.67 %	0.003668 and 0.007024	2021-07-01	33.33 %	0.003536 and 0.008006
	$\alpha = 0.05$	2019-01-01	33.33 %	0.001664 and 0.007661	2022-07-01	33.33 %	0.006105 and 0.008051
	$n = 0.4$	2019-07-01	16.67 %	0.005927 and 0.005845	2020-07-01	33.33 %	0.004560 and 0.006630
	$n = 0.5$	2019-01-01	33.33 %	0.001664 and 0.006832	2021-07-01	50.0 %	0.004918 and 0.008332
	$n = 0.6$	2019-01-01	25.0 %	0.001632 and 0.007059	2019-07-01	25.0 %	0.004176 and 0.006978
CI 95%	$\alpha = 0.0125$	2020-07-01	33.33 %	0.004560 and 0.007077	2021-07-01	50.0 %	0.005262 and 0.008186
	$\alpha = 0.025$	2019-01-01	16.67 %	0.003703 and 0.007077	2022-07-01	33.33 %	0.006105 and 0.008042
	$\alpha = 0.05$	2019-01-01	33.33 %	0.001664 and 0.006367	2021-07-01	50.0 %	0.005262 and 0.007725
	$n = 0.4$	2019-07-01	16.67 %	0.004127 and 0.005092	2021-07-01	50.0 %	0.005262 and 0.008550
	$n = 0.5$	2019-01-01	50.0 %	0.001912 and 0.005679	2019-07-01	16.67 %	0.003668 and 0.006668
	$n = 0.6$	2019-07-01	25.0 %	0.003084 and 0.007700	2022-07-01	25.0 %	0.008374 and 0.007759

of 50 % have been applied. Therefore, it can be concluded that the stochastic rates model cannot give an optimum model in this case, but it can be concluded that a baseline model amount of 50 % and a critical area of 0.025 is a well-standard model.

As shown in Figure 4.24, four points have been determined as reference points near Brug 717. At the Piet Wiedijkstraat, the reference points are positioned in a parking place because of their higher live load capacity. The reference point of this location will be, as at the Piet Wiedijkstraat, selected by the lowest RMSE when assuming a linear trend calculated by BLUE.
Table 4.8 shows the results for every confidence interval at the Brug 717. At this location, it can be concluded that 2019-01-01 and 2019-07-01 are dates where high settlement rates have been experienced. For 2019-01-01, there is a relatively high peak (50%) and a relatively low standard deviation when a confidence interval of 90%, a critical area of 0.025, and a baseline model amount of 50% have been applied. For 2019-07-01, these model parameters also give a relatively high peak (25%) and a relatively low standard deviation. The same outcome has also been shown at a confidence interval of 80% with the same model parameters. Therefore, also in the case of location Brug 717, a baseline model amount of 50% and a critical area of 0.025 is a well-standard model.

As shown in Figure 4.25, four points have been determined as reference points near the location Romeros-

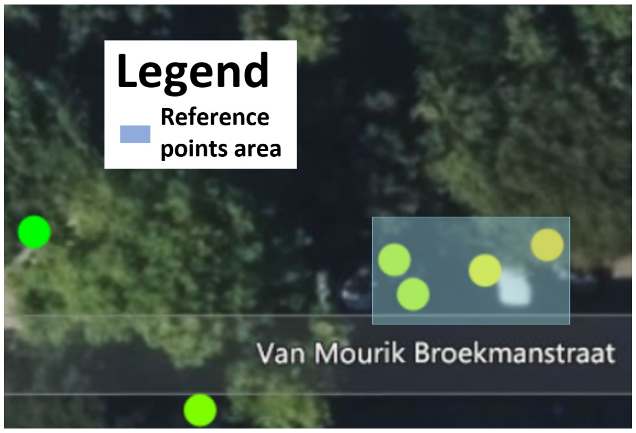


Figure 4.24: Reference point Brug 717.

Table 4.8: Results of the stochastic model applied at the location Brug 717 with different critical areas of 0.0125, 0.025, and 0.05 and different baseline model amounts of 40%, 50%, and 60%. The general values (values which do not change when the other parameters have changed) are for the baseline model amount 50% and the critical area 0.025.

		Peak 1	% Peak 1	σ Peak 1	Peak 2	% Peak 2	σ Peak 2
CI 80%	$\alpha = 0.0125$	2019-01-01	12.5 %	0.008875 and 0.005553	2019-07-01	25.0 %	0.006196 and 0.005537
	$\alpha = 0.025$	2019-07-01	25.0 %	0.005515 and 0.005261			
	$\alpha = 0.05$	2019-01-01	12.5 %	0.008138 and 0.003246	2019-07-01	37.5 %	0.006373 and 0.005281
	$n = 0.4$	2019-01-01	50.0 %	0.000294 and 0.006550	2019-07-01	20.0 %	0.006505 and 0.006211
	$n = 0.5$	2019-01-01	12.5 %	0.006926 and 0.004388	2019-07-01	37.5 %	0.005536 and 0.005257
	$n = 0.6$	2019-01-01	16.67 %	0.005104 and 0.004619	2019-07-01	16.67 %	0.006434 and 0.006742
CI 90%	$\alpha = 0.0125$	2019-01-01	12.5 %	0.005554 and 0.004198	2019-07-01	12.5 %	0.008909 and 0.004920
	$\alpha = 0.025$	2019-01-01	50.0 %	0.001930 and 0.004078	2019-07-01	25.0 %	0.005515 and 0.006164
	$\alpha = 0.05$	2019-07-01	25.0 %	0.008538 and 0.006579			
	$n = 0.4$	2019-01-01	40.0 %	0.001546 and 0.005020	2019-07-01	20.0 %	0.005646 and 0.006189
	$n = 0.5$	2019-01-01	25.0 %	0.005532 and 0.005595	2019-07-01	25.0 %	0.009023 and 0.006664
	$n = 0.6$	2019-01-01	33.33 %	0.004837 and 0.004754	2020-07-01	16.67 %	0.008787 and 0.007427
CI 95%	$\alpha = 0.0125$	2019-07-01	12.5 %	0.006434 and 0.005662	2020-01-01	37.5 %	0.006882 and 0.006461
	$\alpha = 0.025$	2019-01-01	12.5 %	0.005101 and 0.004459	2020-01-01	37.5 %	0.008783 and 0.006240
	$\alpha = 0.05$	2019-01-01	25.0 %	0.007845 and 0.003975	2019-07-01	12.5 %	0.008149 and 0.005366
	$n = 0.4$	2019-07-01	20.0 %	0.008721 and 0.005951	2020-01-01	10.0 %	0.007219 and 0.005271
	$n = 0.5$	2019-07-01	12.5 %	0.006688 and 0.005642			
	$n = 0.6$	2019-01-01	12.5 %	0.005101 and 0.005255	2019-07-01	12.5 %	0.007450 and 0.006752

traat. At the Piet Wiedijkstraat and the Brug 717, the reference points are positioned in a parking place. However, as shown in Figure 4.25, not all the data points in the parking place will be used in the selection because these data points are located in the vegetation, and InSAR has a low accuracy in vegetation [Ferretti, Prati, and Rocca, 2001].
Table 4.9 shows the results for every confidence interval. Looking at these results, it is revealed that on 2022-07-01, the highest peak of 50 % exists. When looking at the standard deviations, applying a confidence interval of 95 % and a critical area of 0.025 results in the lowest uncertainty.

Table 4.10 shows the results for a transition zone length of 10 meters at the Piet Wiedijkstraat. Because there were no differences in outfiltering data points with different confidence intervals, the stochastic rates approach has been conducted on one confidence interval. Table 4.10 indicates high settlement rates on 2019-07-01 and 2020-07-01. The peaks are higher and differ from those in 4.7, which are mainly on 2019-01-01 and 2021-07-01. The highest peaks appear when using a baseline model amount of 50 % and a critical area of 0.025, which are the same model parameters when using the variable transition zone length method.

At Brug 717, the stochastic rates approach does not work with a standard transition zone length of 10 meters due to fewer available InSAR data points. Therefore, as described in Section 3.3, a transition zone

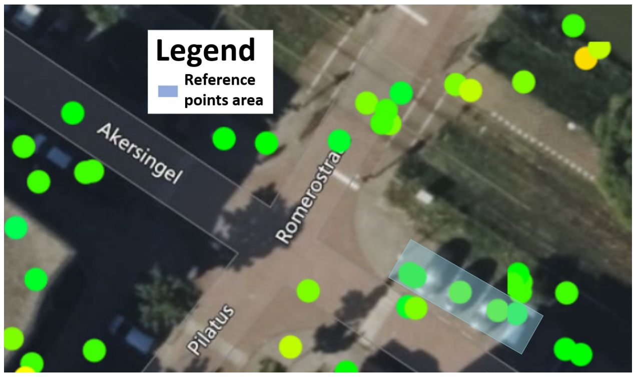


Figure 4.25: Reference point Romerostraat.

Table 4.9: Results of the stochastic model applied at the Romerostraat with different critical areas of 0.0125, 0.025, and 0.05 and different baseline model amounts of 40%, 50%, and 60%. The general values (values that do not change when the other parameters have changed) are for the baseline model amount 50% and the critical area 0.025.

		Peak 1	% Peak 1	σ Peak 1	Peak 2	% Peak 2	σ Peak 2
CI 80 %	$\alpha = 0.0125$	2020-07-01	25.0 %	0.011449 and 0.009896	2021-07-01	25.0 %	0.011259 and 0.010423
	$\alpha = 0.025$	2022-07-01	25.0 %	0.011449 and 0.009896			
	$\alpha = 0.05$	2020-07-01	25.0 %	0.011449 and 0.010426	2022-07-01	25.0 %	0.010670 and 0.010966
	$n = 0.4$	2020-07-01	33.33 %	0.008429 and 0.009837	2021-07-01	33.33 %	0.008636 and 0.011968
	$n = 0.5$	2021-07-01	25.0 %	0.009200 and 0.010460	2022-07-01	25.0 %	0.010667 and 0.010261
	$n = 0.6$	2020-07-01	25.0 %	0.007706 and 0.009922	2021-07-01	25.0 %	0.011259 and 0.010512
CI 90 %	$\alpha = 0.0125$	2021-07-01	25.0 %	0.009200 and 0.010902	2022-07-01	25.0 %	0.011378 and 0.012184
	$\alpha = 0.025$	2021-07-01	25.0 %	0.011378 and 0.012184	2022-01-01	25.0 %	0.011378 and 0.012184
	$\alpha = 0.05$	2021-07-01	25.0 %	0.009200 and 0.010800	2022-07-01	50.0 %	0.009200 and 0.010800
	$n = 0.4$	2021-07-01	16.67 %	0.009180 and 0.009920	2022-07-01	33.33 %	0.009847 and 0.011084
	$n = 0.5$	2020-07-01	25.0 %	0.008537 and 0.010257	2022-07-01	25.0 %	0.010670 and 0.010675
	$n = 0.6$	2021-07-01	25.0 %	0.008746 and 0.010068	2022-07-01	50.0 %	0.009084 and 0.011145
CI 95 %	$\alpha = 0.0125$	2019-01-01	25.0 %	0.000315 and 0.008312	2022-07-01	25.0 %	0.010670 and 0.010955
	$\alpha = 0.025$	2020-07-01	25.0 %	0.008537 and 0.010395	2022-07-01	50.0 %	0.008537 and 0.010395
	$\alpha = 0.05$	2020-07-01	25.0 %	0.008537 and 0.010402	2021-07-01	25.0 %	0.008746 and 0.010958
	$n = 0.4$	2021-07-01	33.33 %	0.008746 and 0.010958	2022-07-01	33.33 %	0.008203 and 0.011319
	$n = 0.5$	2021-07-01	25.0 %	0.008203 and 0.011319	2022-07-01	25.0 %	0.010667 and 0.011391
	$n = 0.6$	2021-07-01	25.0 %	0.011353 and 0.010508	2022-07-01	25.0 %	0.010670 and 0.012161

Table 4.10: Results of the stochastic model applied at the Piet Wiedijkstraat with a transition zone length of 10 meters with different critical areas of 0.0125, 0.025, and 0.05 and different baseline model amounts of 40%, 50%, and 60%. The general values (values that do not change when the other parameters have changed) are for the baseline model amount 50% and the critical area 0.025.

	Peak 1	% Peak 1	σ Peak 1	Peak 2	% Peak 2	σ Peak 2
$\alpha = 0.0125$	2019-07-01	40.0 %	0.006189 and 0.006663	2020-07-01	60.0 %	0.009305 and 0.010096
$\alpha = 0.025$	2019-07-01	80.0 %	0.006230 and 0.002178	2020-07-01	60.0 %	0.008201 and 0.010096
$\alpha = 0.05$	2019-07-01	40.0 %	0.007729 and 0.004988	2020-07-01	80.0 %	0.006916 and 0.005632
$n = 0.4$	2019-07-01	50.0 %	0.005794 and 0.002178	2020-07-01	33.33 %	0.010141 and 0.009619
$n = 0.5$	2019-07-01	40.0 %	0.006189 and 0.006663	2020-07-01	60.0 %	0.008201 and 0.010096
$n = 0.6$	2019-07-01	40.0 %	0.006088 and 0.006663	2020-07-01	60.0 %	0.007611 and 0.009619

length of 15 meters will be used. However, the stochastic rates approach still does not work with this length. Therefore, a transition zone length of 20 meters has been used. Table 4.11 shows the results of the stochastic rates approach with a transition zone length of 20 meters at Brug 717. Because there were only differences in outfiltering data points with confidence intervals of 80 % and 90 % and a confidence interval of 95 %, the stochastic rates approach has been conducted on two confidence intervals. The results show clear peaks at 2019-01-01 and 2020-01-01, which differ from Table 4.8. Based on the highest peaks and lowest uncertainty, the confidence intervals 80 % and 90 % with model parameters $\alpha = 0.025$ and $n = 0.5$ give the best results.

Table 4.11: Results of the stochastic model applied at Brug 717 with a transition zone length of 20 meters with different critical areas of 0.0125, 0.025, and 0.05 and different baseline model amounts of 40%, 50%, and 60%. The general values (values that do not change when the other parameters have changed) are for the baseline model amount 50% and the critical area 0.025.

		Peak 1	% Peak 1	σ Peak 1	Peak 2	% Peak 2	σ Peak 2
CI 80-90 %	$\alpha = 0.0125$	2019-01-01	100.0 %	0.004737 and 0.002105	2020-01-01	55.56 %	0.013582 and 0.014896
	$\alpha = 0.025$	2019-01-01	100.0 %	0.001244 and 0.001853	2020-01-01	66.67 %	0.009022 and 0.011599
	$\alpha = 0.05$	2019-01-01	100.0 %	0.005572 and 0.003382	2020-01-01	66.67 %	0.009022 and 0.014068
	$n = 0.4$	2019-01-01	100.0 %	0.006726 and 0.003224	2020-01-01	54.55 %	0.015906 and 0.009278
	$n = 0.5$	2019-01-01	88.89 %	0.009499 and 0.002195	2020-01-01	55.56 %	0.015652 and 0.011674
	$n = 0.6$	2019-01-01	85.71 %	0.007785 and 0.002203	2020-01-01	57.14 %	0.014277 and 0.009922
CI 95 %	$\alpha = 0.0125$	2019-01-01	100.0 %	0.004849 and 0.003363	2020-01-01	66.67 %	0.011641 and 0.014967
	$\alpha = 0.025$	2019-01-01	88.89 %	0.008803 and 0.001674	2020-01-01	55.56 %	0.014121 and 0.010259
	$\alpha = 0.05$	2019-01-01	77.78 %	0.008481 and 0.003438	2020-01-01	66.67 %	0.011049 and 0.015928
	$n = 0.4$	2019-01-01	90.91 %	0.007165 and 0.001819	2020-01-01	72.73 %	0.008790 and 0.013451
	$n = 0.5$	2019-01-01	88.89 %	0.009965 and 0.002578	2020-01-01	55.56 %	0.012524 and 0.010697
	$n = 0.6$	2019-01-01	100.0 %	0.006940 and 0.001977	2020-01-01	62.5 %	0.011858 and 0.012300

The stochastic rates approach does not work for a transition zone length of 10 and 15 meters at the Romerostraat as well. Therefore, a transition zone length of 20 meters has also been used. Table 4.12 shows the results of the stochastic rates approach with a transition zone length of 20 meters at the Romerostraat. Because there were only differences in outfiltering data points with confidence intervals of 80 % and 90 % and a confidence interval of 95 %, the stochastic rates approach has been conducted on two confidence intervals. Table 4.12 indicates high settlement rates on 2021-07-01 and 2022-07-01. The peaks on 2022-07-01 are lower than on 2021-07-01, which should indicate lower track degradation in 2022 than in 2021. The highest peaks appear when using model parameters of $n = 0.6$ and $\alpha = 0.025$ for confidence intervals of 90 % and 95 %.

4.6 Prediction method based on BLUE

The results of the prediction method based on BLUE with applying confidence intervals of 80%, 90%, and 95% at the Piet Wiedijkstraat are shown in Figure 4.26. It is shown that, in 2021, the rates of the bridge and those not on the bridge are higher than the rest of the rates. 2021 represents data from 01-01-2020 to 01-07-2020, and gives a prediction for 01-01-2021. However, when using this model in practice, not all the graphs can be shown, which means that the results can look slightly unclear due to the lack of a given relative

Table 4.12: Results of the stochastic model applied at the Romerostraat with a transition zone length of 20 meters with different critical areas of 0.0125, 0.025, and 0.05 and different baseline model amounts of 40%, 50%, and 60%. The general values (values that do not change when the other parameters have changed) are for the baseline model amount 50% and the critical area 0.025.

		Peak 1	% Peak 1	σ Peak 1	Peak 2	% Peak 2	σ Peak 2
CI 80 %	$\alpha = 0.0125$	2021-07-01	50.0 %	0.011259 and 0.007964	2022-07-01	25.0 %	0.010088 and 0.009071
	$\alpha = 0.025$	2020-07-01	25.0 %	0.008537 and 0.006212	2021-07-01	50.0 %	0.011259 and 0.007770
	$\alpha = 0.05$	2019-01-01	25.0 %	0.000315 and 0.005620	2021-07-01	50.0 %	0.011338 and 0.008470
	$n = 0.4$	2021-07-01	66.67 %	0.009180 and 0.005622	2022-07-01	33.33 %	0.008203 and 0.009846
	$n = 0.5$	2021-07-01	50.0 %	0.011338 and 0.005347	2022-07-01	25.0 %	0.010670 and 0.010644
	$n = 0.6$	2020-07-01	25.0 %	0.007706 and 0.008335	2021-07-01	75.0 %	0.011353 and 0.010283
CI 90-95 %	$\alpha = 0.0125$	2021-07-01	60.0 %	0.008746 and 0.007770	2022-07-01	20.0 %	0.010667 and 0.011325
	$\alpha = 0.025$	2021-07-01	60.0 %	0.011259 and 0.007964	2022-01-01	20.0 %	0.010670 and 0.010869
	$\alpha = 0.05$	2020-07-01	20.0 %	0.007706 and 0.010712	2021-07-01	80.0 %	0.011353 and 0.008674
	$n = 0.4$	2021-07-01	66.67 %	0.008771 and 0.009773	2022-07-01	33.33 %	0.008203 and 0.011355
	$n = 0.5$	2021-07-01	60.0 %	0.009364 and 0.007770	2022-07-01	40.0 %	0.009084 and 0.011325
	$n = 0.6$	2021-07-01	75.0 %	0.009200 and 0.009032	2022-07-01	50.0 %	0.009084 and 0.009971

context. Another remarkable phenomenon is that all three plots look the same, so filtering outliers does not work on this location with this method. Appendix F shows the whole Python code applied for this method.

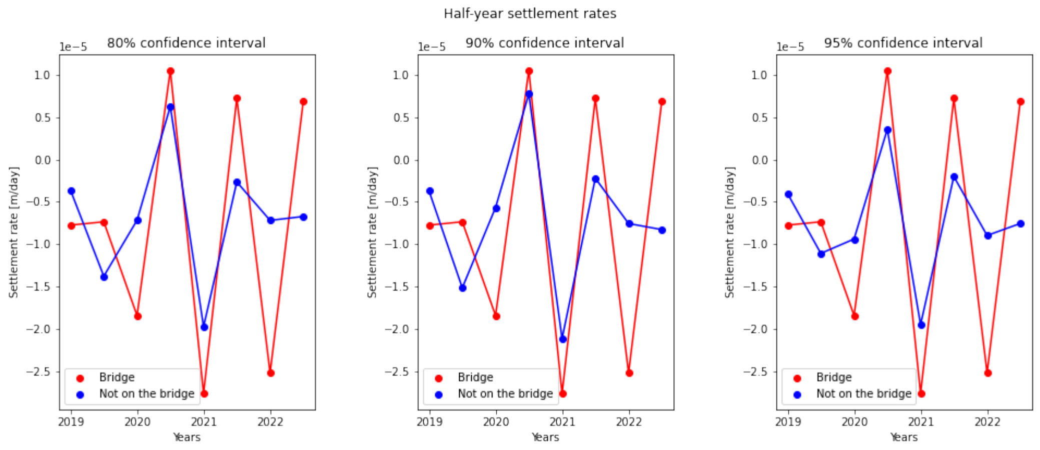


Figure 4.26: Results prediction method based on BLUE applied at the Piet Wiedijkstraat with confidence interval of 80%, 90%, and 95%

The maximum uncertainties of the prediction method based on BLUE, with applying confidence intervals of 80 %, 90 %, and 95 % are shown in Figure 4.27. In general, it can be concluded that for this case, the maximum uncertainty of the model has an order of 10^{-2}. It is also remarkable that the prediction of 01-07-2022 has a relatively high maximum uncertainty. The prediction for this date consists of the displacements from 01-07-2021 to 01-01-2022. According to this remarkable phenomenon, all the data points are different from the fitted line by the BLUE method.

The same process described above has also been applied to location Brug 717. The results of the prediction method based on BLUE with applying confidence intervals of 80%, 90%, and 95% are shown in Figure 4.28. It is also shown at this location that, in 2021, the bridge rates and those not on the bridge are higher than the rest. 2021 represents also data from 01-01-2020 to 01-07-2020, and gives a prediction for 01-01-2021.

The maximum uncertainties of the prediction method based on BLUE with applying confidence intervals of 80%, 90%, and 95% are shown in Figure 4.29. In general, it can be concluded that for this case, the maximum uncertainty of the model has an order of 10^{-2}. When applying a confidence interval of 95%, the maximum uncertainties lie closer to each other than in confidence intervals of 80% and 90%. It is also remarkable that the prediction of 01-07-2022 has a relatively high maximum uncertainty in the confidence intervals of 80% and 90%.

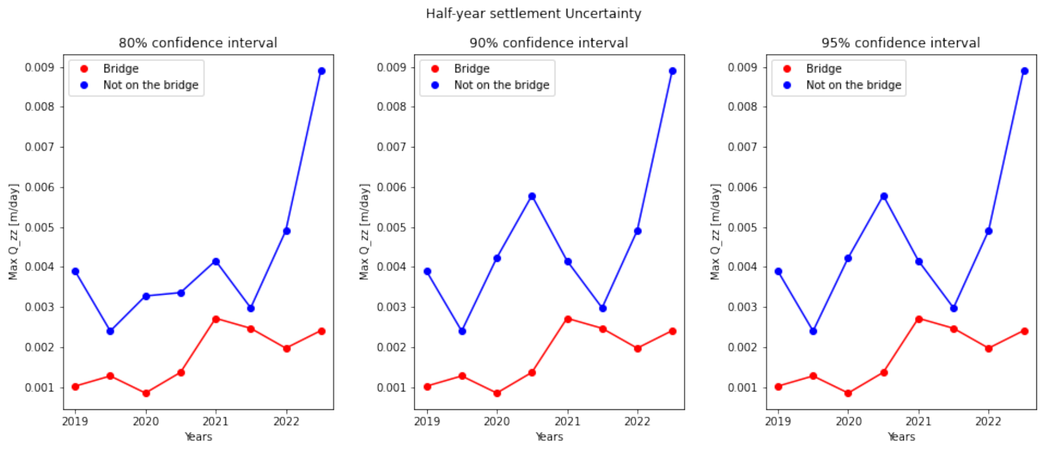


Figure 4.27: Uncertainties prediction method based on BLUE applied at the Piet Wiedijkstraat with confidence intervals of 80 %, 90 %, and 95 %

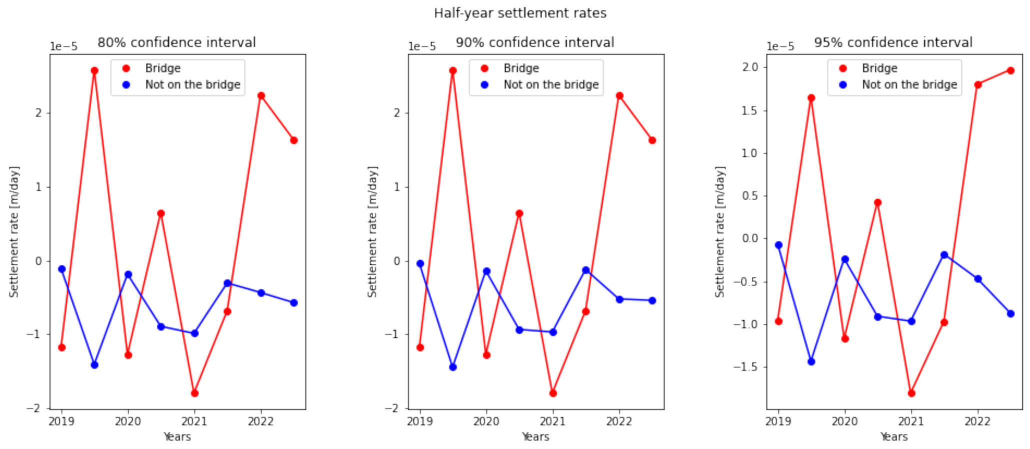


Figure 4.28: Results prediction method based on BLUE at location Brug 717 with confidence interval of 80%, 90%, and 95%

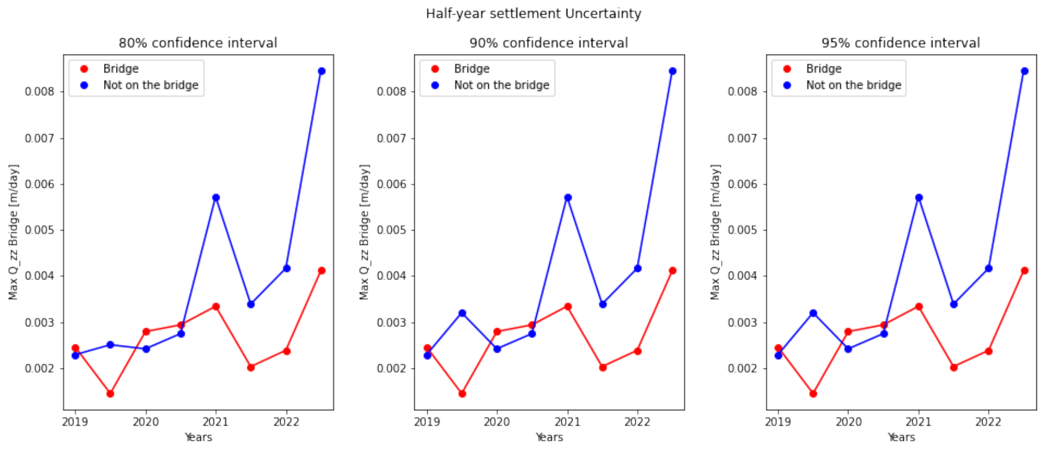


Figure 4.29: Uncertainties prediction method based on BLUE at location Brug 717 with confidence intervals of 80%, 90%, and 95%

The results of the prediction method based on BLUE for the Romerostraat with applying confidence intervals of 80 %, 90 %, and 95 % are shown in Figure 4.30. It is also at this location that it was demonstrated that, in 2021, the rates of the bridge and not on the bridge are higher than the rest of the rates. Another remarkable phenomenon is the varying peaks of the data points on the bridge and not on the bridge on 01-01-2022 and 01-07-2022.

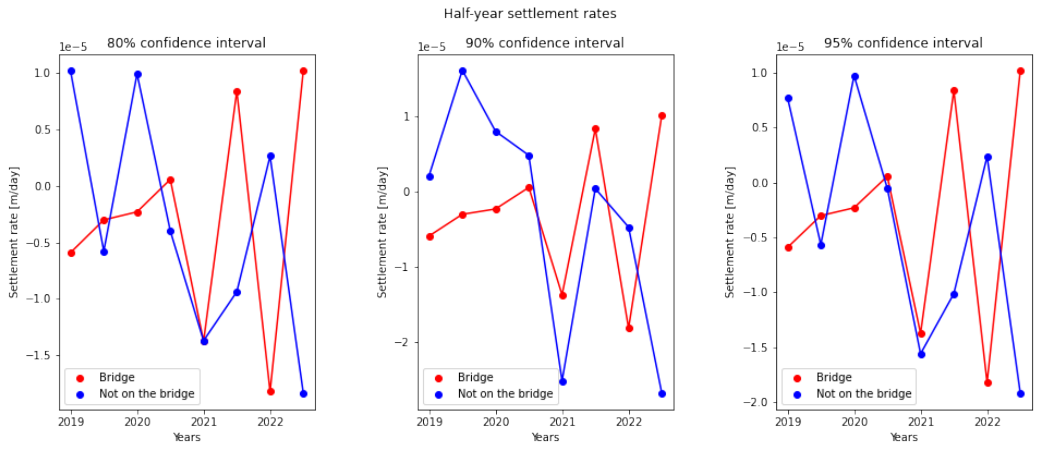


Figure 4.30: Results prediction method based on BLUE at the Romerostraat with confidence interval of 80%, 90%, and 95%

The maximum uncertainties of the prediction method based on BLUE with applying confidence intervals of 80 %, 90 %, and 95 % are shown in Figure 4.31. In general, it can be concluded that for this case, the maximum uncertainty of the model has an order of 10^{-2}. Also, in this case, it is remarkable that the predictions of 01-01-2022 and 01-07-2022 have a relatively high maximum uncertainty. However, the relatively high peak is lower than the two other cases. After analysing the three cases and experiencing this phenomenon, it may be concluded that the InSAR data showed high uncertainties at that time.

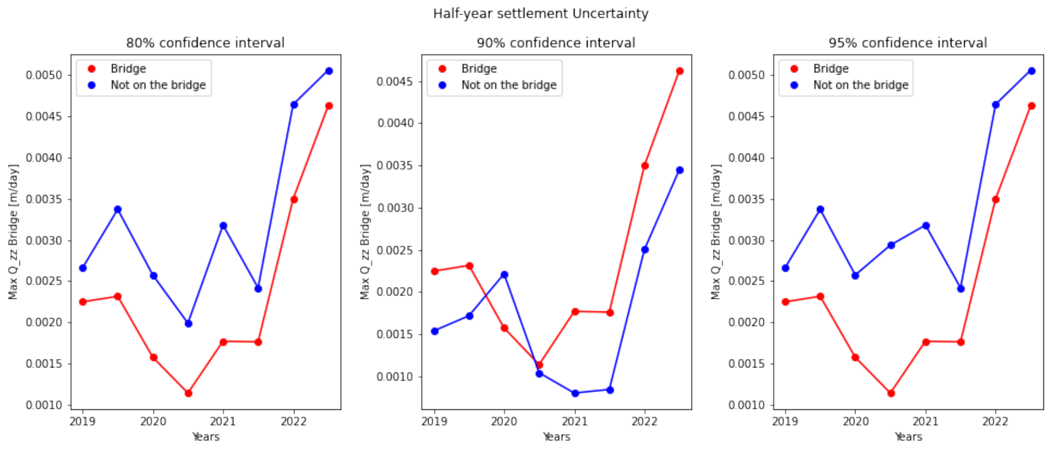


Figure 4.31: Uncertainties prediction method based on BLUE at the Romerostraat with confidence intervals of 80 %, 90 %, and 95 %

The prediction method based on BLUE has also been compared with a standard transition zone length of 10 meters. However, because Brug 717 and the Romerostraat have fewer InSAR data points, a transition zone length of 20 meters has been used at those locations.
Figure 4.32 shows the results for a transition zone length of 10 meters at the Piet Wiedijkstraat. Because there were, as described in Subsection 4.5, no differences in outfiltering data points with different confidence

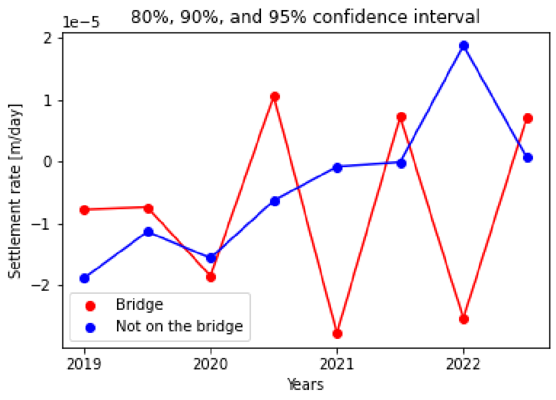


Figure 4.32: Results prediction method based on BLUE at the Piet Wiedijkstraat with a transition zone length of 10 meters.

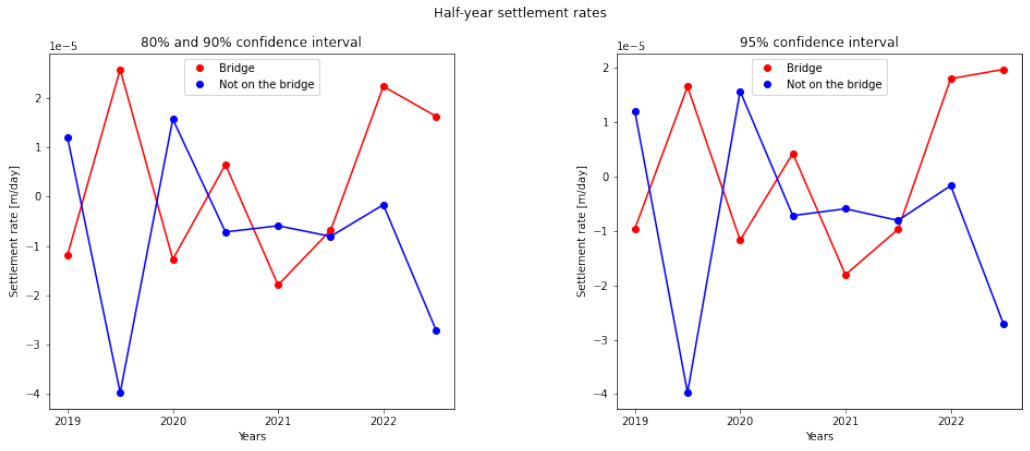


Figure 4.33: Results prediction method based on BLUE at Brug717 with a transition zone length of 20 meters.

intervals, so the stochastic rates approach has been conducted on one confidence interval. Figure 4.32 indicates high settlement rates in 2019 and 2020 for data points on the bridge and not on the bridge at the Piet Wiedijkstraat. The high settlement rates shown in Figure 4.26 in 2021 are not shown in Figure 4.32. In general, the uncertainty of the BLUE plots from the Piet Wietdijkstraat with a transition zone length of 10 meters and the other locations have the same order as the plots with the variable transition zone length method, which is 10^{-2} m/day.

Figure 4.33 indicates high settlement rates on 01-07-2019 and 01-07-2022 at Brug 717. Figure 4.28 showed that in 2021, high settlement rates on and off the bridge were experienced. Therefore, the outcomes of Figure 4.33 differ from the results of Figure 4.28. The dips on 01-07-2019 and 01-07-2022 are much lower than those in Figure 4.28. Therefore, the prediction method based on BLUE gives clearer results when using a lower transition zone length than the variable transition zone length method for this case.

Figure 4.34 indicates high settlement rates on 01-01-2021 and 01-07-2022 at the Romerostraat. It is the same pattern experienced in Figure 4.30. The remarkable phenomenon of varying dips of the data points on the bridge and not on the bridge on 01-01-2022 and 01-07-2022 is also experienced in Figure 4.34. The dips on 01-01-2021 and 01-07-2021 are much lower than those in Figure 4.30, as shown at Brug 717. Therefore, for the Romerostraat, the prediction method based on BLUE gives clearer results when using a transition zone length of 20 meters than when using variable transition zone lengths.

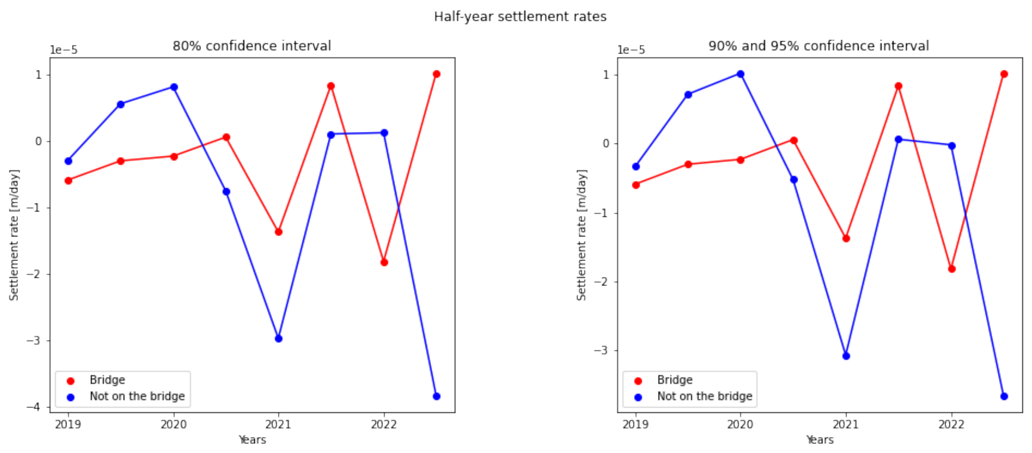


Figure 4.34: Results prediction method based on BLUE at the Romerostraat with a transition zone length of 20 meters.

4.7 Speed plots

At the Piet Wiedijkstraat, the mean velocities in 2021 are shown in Figure 4.35. Figure 4.36 shows the velocities in 2022. The bridge has been shaded in red. The figures show that in 2021, the mean velocity of the trams was lower than in 2022. The shrinking velocity in 2021 could be caused by a dip before the bridge. The stochastic rates method and the prediction method based on BLUE detected high rates of track degradation in July 2021. Therefore, the results of the methods have been strengthened by this speed analysis.

At location Brug 717, the velocities in 2021 are shown in Figure 4.37. Figure 4.38 shows the velocities in 2022. At the Piet Wiedijkstraat, the mean velocity was lower in 2021 than in 2022. It can also be caused by a dip before the bridge. The prediction method based on BLUE detected a high settlement rate in July 2021; the stochastic rates method did not.

At the Romerostraat, the velocities in 2021 are shown in Figure 4.39. Figure 4.40 shows the velocities in 2022. The standard transition zone is shown in both figures. In 2022, the velocities are also higher than in 2021, but not with a significant difference from the other two locations. When looking at the transition zone, in 2022, lower speeds appeared than in 2021. The stochastic rates method detected high rates of track degradation in 2022, while the prediction method based on BLUE detected high rates of track degradation in 2021. Both methods detected high rates of track degradation in 2021.

4.8 Maintenance data

As written in Section 1.4, GVB maintenance data is unavailable, unfortunately. Therefore, the outcomes of the prediction models and the analysed GVB geometry data cannot be validated. For Piet Wiedijkstraat and Brug 717 locations, tamping maintenance data is unavailable. Moreover, maintenance data could also not be found for the Romerostraat. Maintenance tamping data was found for other tramlines in Amsterdam. Furthermore, the data consists of the applied tamping maintenance sets and handwritten start and end points. Longitude and latitude coordinates did not exist in the maintenance data set, which makes it challenging to use the data in Python.

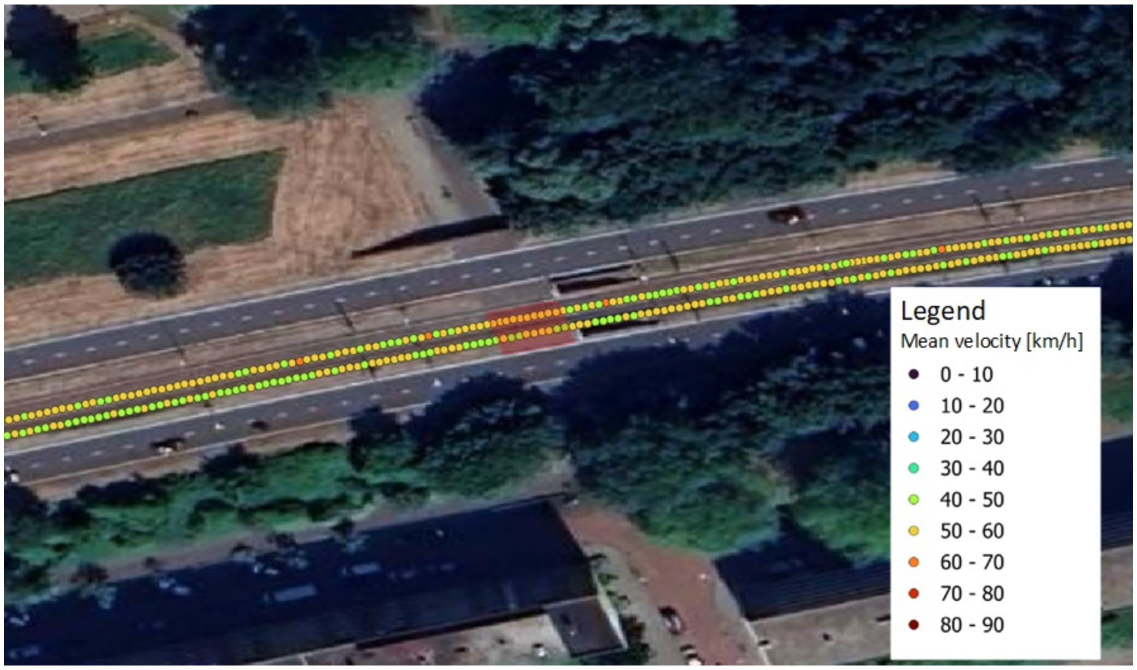


Figure 4.35: Mean velocity tramway transition zone Piet Wiedijkstraat in 2021. The bridge has been shaded in red.

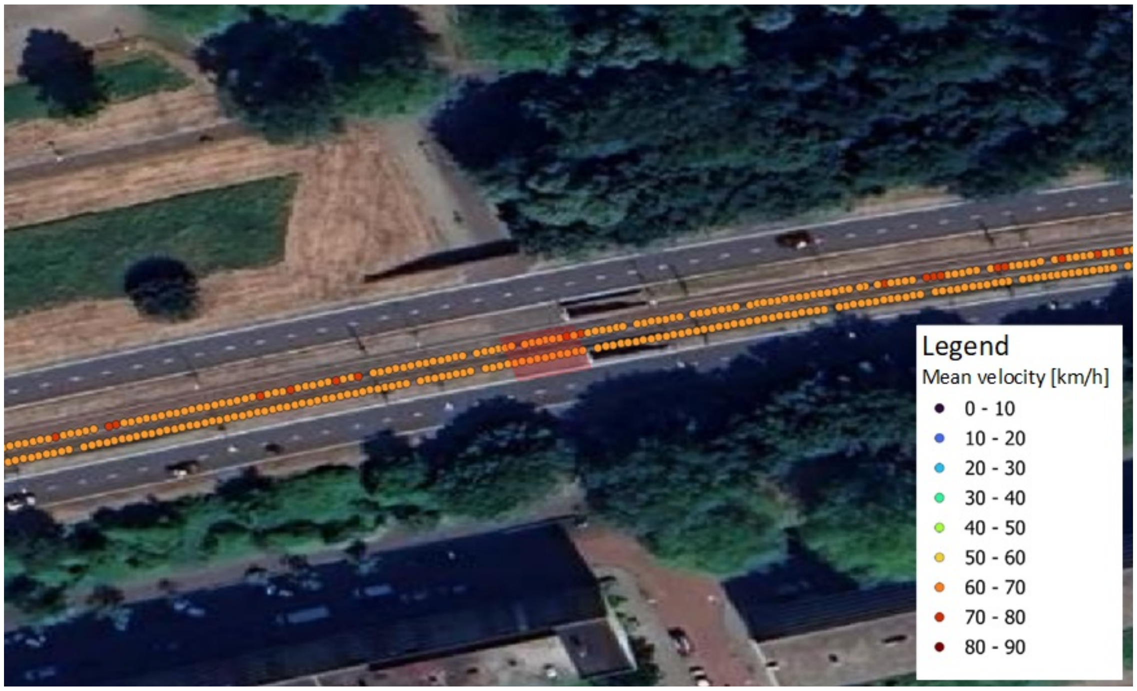


Figure 4.36: Mean velocity tramway transition zone Piet Wiedijkstraat in 2022. The bridge has been shaded with red.

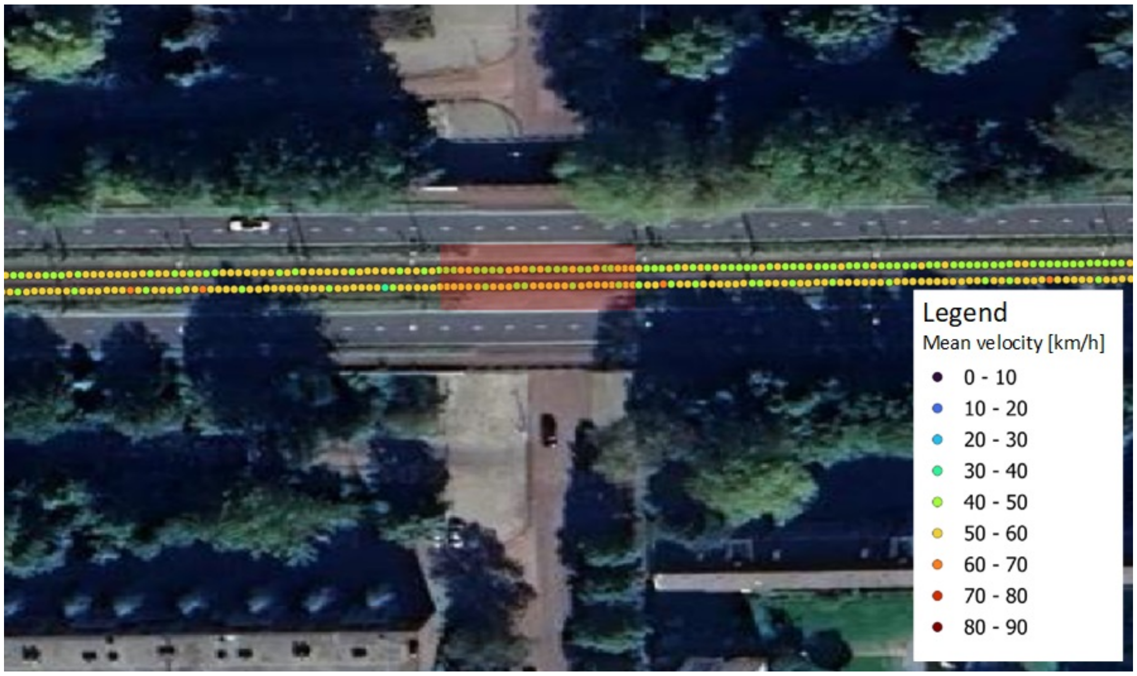


Figure 4.37: Mean velocity tramway transition zone Brug 717 in 2021. The bridge has been shaded with red.

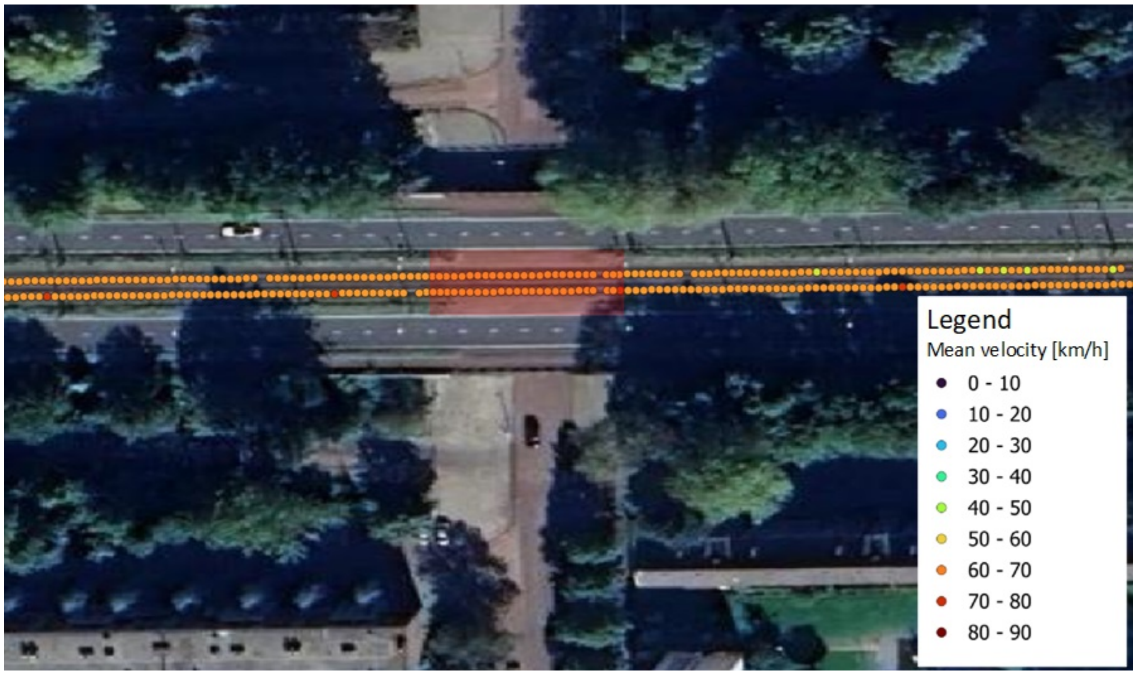


Figure 4.38: Mean velocity tramway transition zone Brug 717 in 2022. The bridge has been shaded with red.

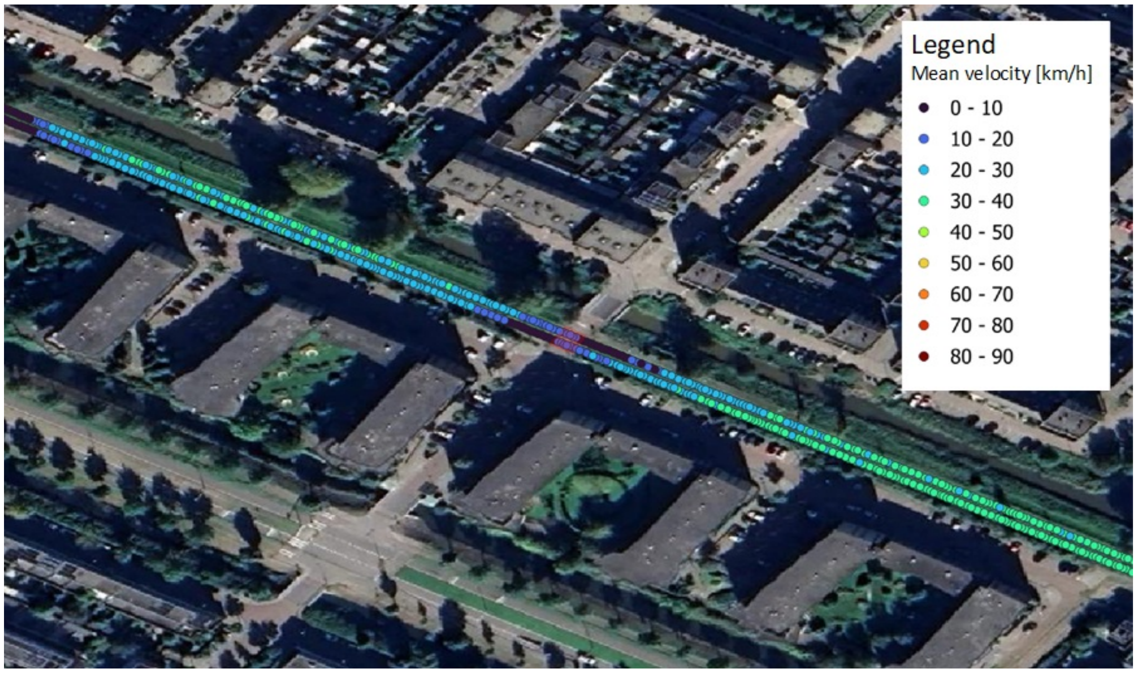


Figure 4.39: Mean velocity tramway transition zone Romerostraat in 2021. The bridge has been shaded in red.

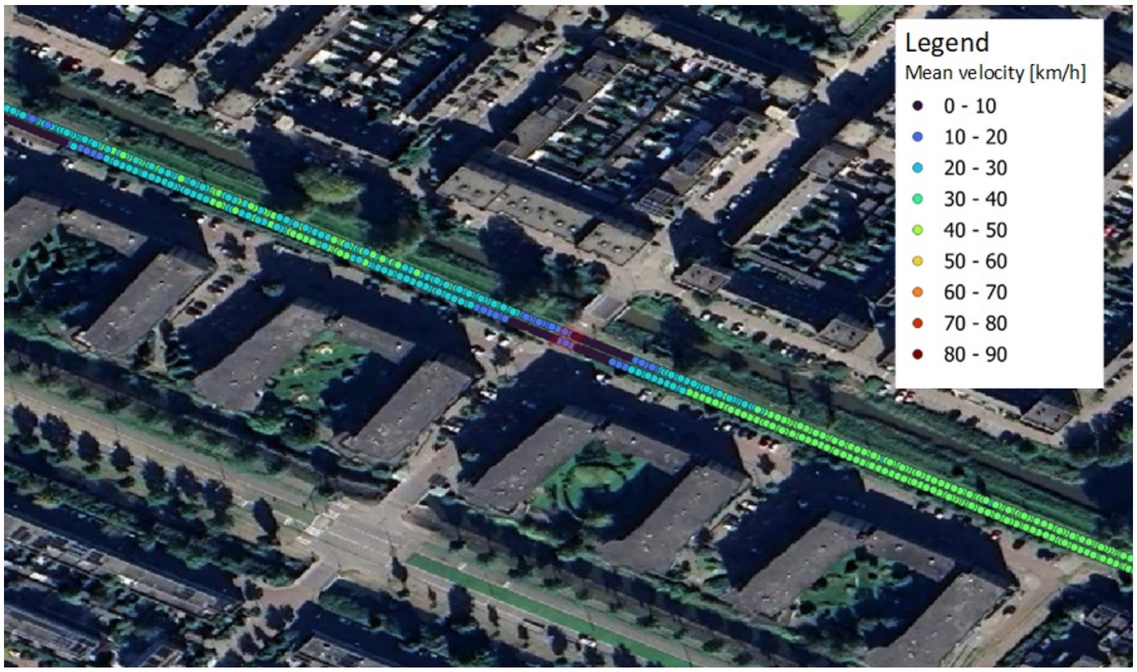


Figure 4.40: Mean velocity tramway transition zone Romerostraat in 2022. The bridge has been shaded in red.

Chapter 5

Conclusion

5.1 Selecting locations

The selected locations in Amsterdam, the Piet Wiedijkstraat, Brug 717, and the Romerostraat, have been tested with the available GVB geometry data. The GVB geometry datasets, track gauge, and twist data provided helpful information about the transition zone condition. As an assumption, it has been assumed that, according to Ahac et al. (2015), the standard deviations of track gauge and twist data on the bridge are higher than the standard deviations of those data not on the bridge. The stated assumption is partly valid for using track gauge data for Piet Wiedijkstraat and Brug 717 locations. For the twist data, the assumption is valid for all the locations. It makes the results of both datasets in line with the literature. The measurements of the datasets are still not directly related to the track degradation. Because, as stated in Section 1.3, track degradation can also be caused by a poor embankment condition, which is not measured with the geometry data. Moreover, as stated in Subsection 2.2.2, Jovanic et al. (2011) concluded that modelling the behaviour of the deterioration of the track geometry remains challenging because it has not yet been proven that individual track components influence each other.

5.2 Preprocessing InSAR data

The InSAR data at the three locations are unevenly distributed. Table 5.1 shows the total available InSAR data points when using a maximum transition zone length of 100 meters without filtering the data points with the described method in Subsection 3.2. The used InSAR data also has a position accuracy of one meter. The defined characteristics of InSAR showed noisy data at every data point. Paying for InSAR data combined with LiDAR data, as Hu et al. (2015) did, can give more data points with better position accuracy. Moreover, the precision of InSAR data would be in millimeters when paying for it.

Table 5.1: The available InSAR data points in a range of 100 meters from both sides of the bridges.

Location	Data points
Piet Wiedijkstraat	60
Brug 717	40
Romerostraat	20

5.3 Prediction method

The stochastic rates approach and the prediction approach based on BLUE are proposed to predict high settlement and have been performed. To overcome the noisy data, linear rates of half-year InSAR data have been decided to be used in both prediction methods. Both methods are linear methods to make simplified models. As mentioned in Section 3.4, the decision has been made not to use machine learning methods due

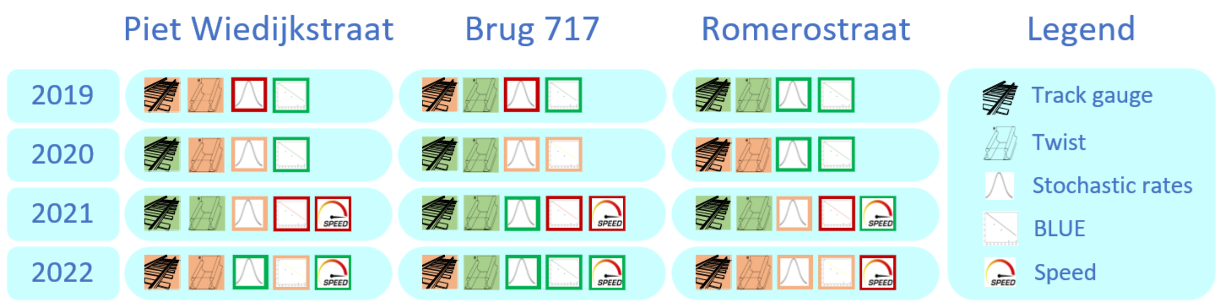


Figure 5.1: An overview of the condition of each transition zone for every year.

to the assumed simplified problem and the risk of overfitting.
Both proposed prediction methods have shown the same uncertainty in Chapter 4, which has an order of 10^{-2}. However, the stochastic rates approach gave a better insight into when high settlement rates appear because it detected when data points exceeded extreme values of the baseline set. The two methods have also been tested with a standard transition zone length of 10 meters at all the locations, instead of using the transition zone length with the highest correlation between GVB track gauge data and InSAR data. Only at the Romerostraat were the same outcomes when using a short transition zone length.
Unfortunately, the outcomes of the prediction methods cannot be strongly underpinned due to the unavailability of historical maintenance data. With maintenance data, the results of the prediction methods could be validated and give better insight into calibrating the ideal stochastic rates model.
With the outcomes of both prediction methods shown in Chapter 4, the overview has been made with the methodology shown in Section 3.4. Figure 5.1 gives a poor condition for the transition zone at the Romerostraat, which is strengthened by the speed plots. The transition zone at the Piet Wiedijkstraat also had a poor condition in 2019, according to GVB data sets and the stochastic rates method.

5.4 Research question: Is InSAR able to provide information about the condition of the transition zone?

The first sub-question, defined in Chapter 1, has been formulated as: "How can geometry deterioration in tramway transition zones be indicated with current geometry parameter datasets?" It can be concluded that the presently available track gauge dataset can provide a valuable indication of geometry deterioration. As stated in 5.1, geometry does not directly measure track degradation since the embankment can also cause track degradation. Moreover, since GVB did not have historical maintenance data available, the proposed dataset for detecting high rates of track degradation did not have the opportunity to be validated.

The second sub-question, defined in Chapter 1, has been formulated as: "What are the key characteristics and limitations of InSAR data for assessing the conditions of tramway transition zones?" When using the InSAR to link it to GVB track gauge data, it has been shown that the link has appropriate Pearson Correlation Coefficients at not every location. At Brug 717, the data point not on the bridge showed a less negative linear relationship than at the Piet Wiedijkstraat. Subsequently, at the Romerostraat, the link between InSAR data and GVB track gauge data showed the least negative linear relationship. An explanation for this is the fewer InSAR data points available in a range of 100 meters at the second and third locations, as shown in Table 5.1. When using generic transition zone lengths of 5, 10, 15, and 20 meters for all three locations, a link between more data points and a higher negative link between track gauge data and InSAR data has been observed. The PCC outcomes of all the locations and the transition zone lengths plotted against the number of data points have been shown in Figure 5.2.

The third sub-question, defined in Chapter 1, has been formulated as: "Which prediction method can be used for the indication of high rates of settlement in tramway transition zones based on the current dataset of geometry parameters from GVB and InSAR deformation data?" The uncertainty of the stochastic rates method and the prediction method based on BLUE has an order of 10^{-2}, so both approaches have the same

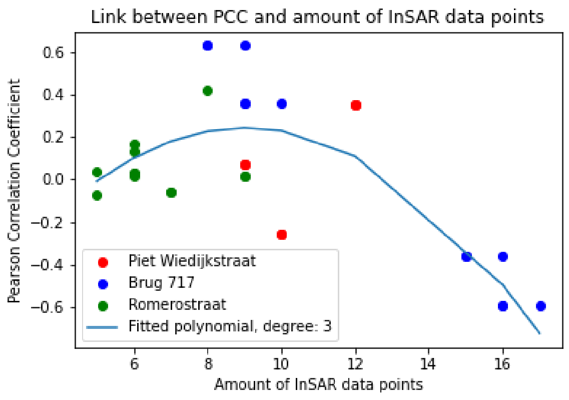


Figure 5.2: Relationship between GVB track gauge data and InSAR data at the three locations with a fitted polynomial.

uncertainty and could indicate high rates of track degradation.

After answering all the sub-questions, the main question can be answered. The main question, defined in Chapter 1, has been formulated as: 'Is InSAR able to provide information about the condition of the transition zone?" The stochastic rates method is appropriate for the uncertainties and the outcomes. It peaks when the behaviour is disproportionate, and it does not need the prediction of the years before to provide a significant result, as the prediction method is based on BLUE needs. However, no real validation is used to validate the results of both methods. The additional mean yearly speed dataset has been used to give more insight into the outcomes of both prediction methods. Still, it cannot be used as validation data because they do not focus only on track irregularities that can be caused by track degradation. Therefore, validation with historical maintenance data is needed to provide a better, helpful vision of the stochastic rates method to predict maintenance in a tramway track transition zone. Also, other datasets like bridge design, wear, precipitation, temperature, and vegetation can be helpful when indicating the conditions of transition zones.

5.5 Recommendations for GVB

From the conclusions, a few recommendations for GVB can be made. First, a historical maintenance database can calibrate and further validate prediction methods. Future approaches for transition zones, such as physical or other data-driven methods, can be proven or made better with this database. GVB can create a database with all the transition zones and add the historical maintenance data in these areas. For the speed and maintenance data, GVB uses its own coordinate system, KGE (Kleinste Geografische Eenheid, Lowest Geographical Unity), and tramline name, which is only available in QGIS. In the future, GVB can add longitude and latitude coordinates to improve Python models.

GVB can start monitoring the transition zones with the values of parameters observed from the yearly measurements. Mainly, the differences between the values of parameters of data points on the bridge and those not on the bridge can help indicate the condition of the transition zones. The length of a transition zone can be variable and based on finding the highest PCC, but a length of 10 or 15 meters also indicates a proper transition zone condition. With the database of historical maintenance data proposed in the paragraph above, prediction methods can be created to predict maintenance earlier. A dashboard that monitors the transition zones based on a traffic light, as shown in Figure 5.1, can also be considered by GVB. In this dashboard, other datasets can also be used to indicate conditions of transition zones like bridge design, wear, precipitation, temperature, and vegetation.

Before GVB decides to purchase InSAR to indicate conditions in the transition zone and use it as an indication for a potential dashboard, further research should be conducted to validate the effectiveness of InSAR data. Again, the effectiveness of InSAR should be tested and validated with the historical maintenance already proposed in the earlier paragraphs. As suggested by Hu et al. (2019), GVB can decide to overcome the limitation of monitoring physical objects due to the low accuracy by using InSAR data in combination with LiDAR laser data. Superiorly, the effectiveness of InSAR data can also be tested with a physical approach. Unfortunately, it was impossible to use a physical approach in this research due to the non-existence of wheel load values, which are needed to use a physical approach.

The research focused on applying InSAR data on transition zones to prevent track degradation on ballast and grass tracks. However, GVB mentioned during the study that when InSAR has been proven valuable, it can also be applied to old bridges in the city center of Amsterdam. Due to many old arch bridges, land deformation on both sides can lead to steeper arches, which can cause less passenger comfort and safety problems. Also, in this case, GVB should monitor the deformation with InSAR in combination with LiDAR laser data to overcome the low position accuracy.

5.6 Future research

InSAR data are unevenly distributed across the tramway track found in Section 4.1. These unevenly distributed data could affect the case study results and prediction approaches. A future model that connects the InSAR data points with the nearest tramline can overcome this undesirable situation. In addition, a weighted model can be used for analysis. Therefore, points far from the tramline can be given a lower weight than points closer to the tramline to create a more realistic settlement model of the tramway track. Also, interpolation to give tramline parts without data points can provide insight into their behaviour. It can be used to better indicate the settlement of the transition zone. In addition, in this research, only two approaches have been used. In further research, other approaches like machine learning can be used to compare the results of this research. As stated in Section 5.5, physical approaches can also help validate the results of the InSAR data prediction approaches. Using another NDT, GPR, can give additional information for the InSAR prediction approaches.

As mentioned in Subsection 3.3, due to the lack of geometry and InSAR data, the geometry data from 2019 to 2022 and the InSAR data from 2019 to 2022 have been used in both approaches. Then, both methods were tested on the exact dates the geometry data were applied, which may have created contradictory approaches. In future research, GVB data from 2019 to 2022 can be used to look at each transition zone, and InSAR data points are needed to predict high settlement rates for 2023 and 2024.

The findings on the behaviour of the InSAR data in transition zones contribute to the research on the maintenance strategy in tramway transition zones. The findings can be used in other transition zones in Amsterdam. However, historical maintenance data is necessary to improve the stochastic rates approach by modifying the model parameters for further application and improvement. The model can also research transition zones in other cities when the tramway operation company has track gauge and historical maintenance data.

Bibliography

Abadi, E., M. Sameni, and M. Yaghini (2023). “Analysis of the relationship between geometric parameters of railway track and twist failure by using data mining techniques”. In: *Engineering Failure Analysis* 143, p. 106862. DOI: 10.1016/j.engfailanal.2022.106862.

Ahac, M. and S. Lakusic (2015). “Tram track maintenance-planning by gauge degradation modelling”. In: *Transport* 30, pp. 430–436. DOI: 10.3846/16484142.2015.1116464.

Arnoult, J. (1986). *Culvert Inspection Manual.* Byrd, Tallamy, MacDonald and Lewis.

Bakker, H. de (1979). *Major soils and soil regions in The Netherlands.* Springer Nature. ISBN: 978-9-061-93590-2.

Benesty, J., J. Chen, and Y. Huang (2008). “On the importance of the Pearson correlation coefficient in noise reduction”. In: *IEEE transactions on audio, speech, and language processing* 16.4, pp. 757–765. DOI: 10.1109/TASL.2008.919072.

Boquet, Y. (2017). “The renaissance of tramways and urban redevelopment in France”. In: *Miscellanea Geographica* 21, pp. 5–18. DOI: 10.1515/mgrsd-2017-0005.

Castro et al. (2022). “Evaluating environmental effects on the structural behavior of the railroad track subgrade considering different sub-ballast design solutions”. In: *Transportation Geotechnics* 34, p. 100761. DOI: 10.1016/j.trgeo.2022.100761.

Chang, L., R. Dollevoet, and R. Hanssen (2017). “Nationwide railway monitoring using satellite SAR interferometry”. In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing.* DOI: 10.1109/JSTARS.2016.2584783.

Charoenwong, C. et al. (2023). “Railway slab vs ballasted track: A comparison of track geometry degradation”. In: *Construction and Building Materials* 378, p. 131121. DOI: 10.1016/j.conbuildmat.2023.131121.

Charras-Garrido, M. and P. Lezaud (2013). “Extreme Value Analysis: an Introduction”. In: *Journal de la société française de statistique* 154.2, pp. 66–97. DOI: 1316.62063.

Coelho, B. (2011). “Dynamics of railway transition zones in soft soils”. PhD thesis. Delft University of Technology.

Czichos, H. (2013). *Handbook of technical diagnostics.* Springer, pp. 11–22. ISBN: 978-3-642-25849-7. DOI: 10.1007/978-3-642-25850-3_2.

Deakin, R. (2005). *Notes on least squares.* RMIT Geospatial Science School of Mathematical and Geospatial Science.

Dheenathayalan, P. et al. (2016). “High-precision positioning of radar scatterers”. In: *Journal of Geodesy* 90, p. 20. DOI: 10.1007/s00190-015-0883-4.

Dietterich, T. (1995). “Overfitting and Undercomputing in Machine Learning”. In: *ACM Computing Surveys.*

Dingqing, L. and D. David (2005). “Transition of railroad bridge approaches”. In: *Journal of Geotechnical and Geoenvironmental Engineering* 131.11, pp. 1392–1398. DOI: 10.1061/(ASCE)1090-0241(2005)131:11(1392).

Falamarzi, A. et al. (2018). “Rail degradation prediction models for tram system: Melbourne case study”. In: *Journal of Advanced Transportation.* DOI: 10.1155/2018/6340504.

Feizhi, X. et al. (2023). “Critical transverse differential settlement between modern tram pile-plank-supported subgrade and surrounding pavement subgrade”. In: *Transportation Geotechnics* 38, p. 100896. DOI: 10.1016/j.trgeo.2022.100896.

Ferretti, A., C. Prati, and F. Rocca (2001). “Permanent scatterers in SAR interferometry”. In: *Geoscience and Remote Sensing, IEEE Transactions on* 39, pp. 8–20. DOI: 10.1109/36.898661.

Franceschetti, G. and R. Lanari (1999). *Synthetic Aperture Radar Processing.* Electronic Engineering Systems. Taylor & Francis. ISBN: 9780849378997.

Gallagher, G. et al. (1999). “The application of time domain ground penetrating radar to evaluate railway track ballast”. In: *NDT & E International* 32.8, pp. 463–468. DOI: 10.1016/S0963-8695(99)00025-0.

Graham, L.C. (1974). “Synthetic interferometer radar for topographic mapping”. In: *Proceedings of the IEEE* 62.6, pp. 763–768. DOI: 10.1109/PROC.1974.9516.

Grossoni, I. et al. (2021). “Modelling railway ballasted track settlement in vehicle-track interaction analysis”. In: *Transportation Geotechnics* 26, p. 100433. DOI: 10.1016/j.trgeo.2020.100433.

Guerin, N. (1996). “Approche expérimentale et numérique du comportement du ballast des voies ferrées”. PhD thesis. École nationale des ponts et chaussées.

Guo, Y. et al. (2022). “Railway ballast material selection and evaluation: A review”. In: *Construction and Building Materials* 344, p. 128218. DOI: 10.1016/j.conbuildmat.2022.128218.

Hastie, T., R. Tibshirani, and J. Friedman (2009). *The elements of statistical learning: Data mining, inference, and prediction*. Springer. ISBN: 978-0-387-84857-0.

Hawari, H. (2007). “Minimising track degradation through managing vehicle/track interaction”. PhD thesis. Queensland University of Technology.

Hodson, T. (2022). “Root-mean-square error (RMSE) or mean absolute error (MAE): when to use them or not”. In: *Geoscientific Model Development* 15.14, pp. 5481–5487. DOI: 10.5194/gmd-15-5481-2022.

Hu, F. et al. (2019). “Monitoring deformation along railway systems combining multi-temporal InSAR and LiDAR Data”. In: *Remote Sensing* 11.19. DOI: 10.3390/rs11192298.

Hu, J. et al. (2014). “Resolving three-dimensional surface displacements from InSAR measurements: A review”. In: *Earth-Science Reviews* 133, pp. 1–17. DOI: 10.1016/j.earscirev.2014.02.005.

Jovanovic, S., G. Evren, and H. Guler (2011). “Modelling railway track geometry deterioration”. In: *Proceedings of the ICE - Transport* 164, pp. 65–75. DOI: 10.1680/tran.2011.164.2.65.

Jover, V., L. Gaspar, and S. Fischer (2020). “Investigation of geometrical deterioration of tramway tracks”. In: *Science and Transport Progress* 8.2(86), pp. 46–59. DOI: 10.15802/stp2020/204152.

Kerr, A. and L. Bathurst (2001). *A method for upgrading the performance at track transitions for high-speed service*. U.S. Department of Transportation.

Koohmishi, M. et al. (2024). “Advancing railway track health monitoring: Integrating GPR, InSAR and machine learning for enhanced asset management”. In: *Automation in Construction* 162, p. 105378. DOI: 10.1016/j.autcon.2024.105378.

Kraśkiewicz, C., M. Urbaniak, and A. Piotrowski (2025). “Experimental study on the elastic support in a discrete rail fastening system used in ballastless tram track structures”. In: *Materials* 18.1. DOI: 10.3390/ma18010141.

Kumar, S. and D. Mahto (2013). “Recent trends in industrial and other engineering applications of non destructive testing: A review”. In: 4.

Laurikkala, J., M. Juhola, and E. Kentala (2000). “Informal identification of outliers in medical data”. In: *Fifth international workshop on intelligent data analysis in medicine and pharmacology*, p. 5.

Lei, X. and L. Mao (2004). “Dynamic response analyses of vehicle and track coupled system on track transition of conventional high speed railway”. In: *Journal of Sound and Vibration* 271.3, pp. 1133–1146. DOI: 10.1016/S0022-460X(03)00570-4.

Li, D. and D. Davis (2005). “Transition of Railroad Bridge Approaches”. In: *Journal of Geotechnical and Geoenvironmental Engineering* 131.11, pp. 1392–1398. DOI: 10.1061/(ASCE)1090-0241(2005)131:11(1392).

Li, D. and T. Selig (2015). “Evaluation of railway subgrade problems”. In: *Transportation Research Record.*

Lichtberger, B. (2010). *Handbuch gleis: Unterbau - oberbau - instandhaltung - wirtschaftlichkeit*. Eurailpress in DVV Media Group. ISBN: 978-3-777-10400-3.

Madshus, C. and A Kaynia (2000). “High-speed railway lines on soft ground: dynamic behaviour at critical train speed”. In: *Journal of Sound and Vibration* 231.3. Cited by: 386, pp. 689–701. DOI: 10.1006/jsvi.1999.2647.

Massonnet, Didier and Kurt L. Feigl (1998). “Radar interferometry and its application to changes in the earth’s surface”. In: *Reviews of Geophysics* 36.4, pp. 441–500. DOI: 10.1029/97RG03139.

Michigan Engineering Department, University of (2020). *Design guideline 5.6 parking structures*. Online guide providing structural codes and standards.

Moreira, A. et al. (2013). “A tutorial on synthetic aperture radar”. In: *IEEE Geoscience and Remote Sensing Magazine (GRSM)* 1, pp. 6–43. DOI: 10.1109/MGRS.2013.2248301.

Moridpour, S. and R. Hesami (2015). "Degradation and performance specification of Melbourne tram tracks". In: *2015 International Conference on Transportation Information and Safety (ICTIS)*, pp. 270–276. DOI: 10.1109/ICTIS.2015.7232120.

Nadeem, M. et al. (2021). "Investigation of the settlement prediction in soft soil by Richards Model: based on a linear least squares-iteration method". In: *Archives of Civil Engineering* 67.2, pp. 491–506. DOI: 10.24425/ace.2021.137181.

Periard, F. (1998). "Wheel-rail noise generation: Curve squealing by trams". PhD thesis. Delft university of technology.

Peteroff, N. (2021). "Rail wear in curves at the tramway of Amsterdam". MA thesis. Delft university of technology.

Roccetti, M. et al. (2019). "Is bigger always better? A controversial journey to the center of machine learning design, with uses and misuses of big data for predicting water meter failures". In: *Journal of Big Data*.

Sañudo, R. et al. (2016). "Track transitions in railways: A review". In: *Construction and Building Materials* 112, pp. 140–157. DOI: 10.1016/j.conbuildmat.2016.02.084.

Searle, S. (1995). "An overview of variance component estimation". In: *Metrika* 42.3-4, pp. 215–230. DOI: 10.1007/BF01894301.

Shan, Y. et al. (2016). "Some predictions of deformations from tram track construction in a structure-embankment transition zone". In: *Procedia Engineering* 143, pp. 1160–1168. DOI: 10.1016/j.proeng.2016.06.129.

Soleimanmeigouni, I., A Ahmadi, et al. (2020). "Investigation of the effect of the inspection intervals on the track geometry condition". In: *Structure and Infrastructure Engineering* 16.8, pp. 1138–1146. DOI: 10.1080/15732479.2019.1687528.

Soleimanmeigouni, I., A. Ahmadi, and U. Kumar (2018). "Track geometry degradation and maintenance modelling: A review". In: *Proceedings of the Institution of Mechanical Engineers, Part F: Journal of Rail and Rapid Transit*.

Strainescu, I. et al. (2008). "Speed control of subways and trams". In: *Urban Transport XIV*.

Teunissen, P. (2007). "Least-squares prediction in linear models with integer unknowns". Undefined/Unknown. In: *Journal of Geodesy* 81.9, pp. 565–579. DOI: 10.1007/s00190-007-0138-0.

Teunissen, P. and A. Amiri-Simkooei (2007). "Least-squares variance component estimation". In: *Journal of Geodesy, 82 (2)* 82. DOI: 10.1007/s00190-007-0157-x.

Tomás, R. et al. (2012). "Subsidence damage assessment of a gothic church using differential interferometry and field data". In: *Structural Health Monitoring* 11.6, pp. 751–762. DOI: 10.1177/1475921712451953.

Tomiyasu, K. (1978). "Tutorial review of synthetic-aperture radar (SAR) with applications to imaging of the ocean surface". In: *Proceedings of the IEEE* 66.5, pp. 563–583. DOI: 10.1109/PROC.1978.10961.

Tosti, F. et al. (2020). "Transport infrastructure monitoring by data fusion of GPR and SAR imagery information". In: *Transportation Research Procedia* 45. Transport Infrastructure and systems in a changing world. Towards a more sustainable, reliable and smarter mobility.TIS Roma 2019 Conference Proceedings, pp. 771–778. DOI: 10.1016/j.trpro.2020.02.097.

Wang, H., L. Chang, and V. Markine (2018). "Structural Health Monitoring of Railway Transition Zones Using Satellite Radar Data". In: *Sensors*. DOI: 10.3390/s18020413.

Wang, H. and V. Markine (2018). "Modelling of the long-term behaviour of transition zones: Prediction of track settlement". In: *Engineering Structures* 156, pp. 294–304. DOI: 10.1016/j.engstruct.2017.11.038.

Yap, M. and O. Oded Cats (2022). "Analysis and prediction of ridership impacts during planned public transport disruptions". In: *Journal of Public Transportation* 24, p. 100036. DOI: 10.1016/j.jpubtr.2022.100036.

Zeng, Z. et al. (2019). "An analytical calculation method for displacement and force on continuous welded rails in temperature-transition zone". In: *Construction and Building Materials* 207, pp. 228–237. DOI: 10.1016/j.conbuildmat.2019.02.120.

Zhang, Q. et al. (2019). "InSAR technique applied to the monitoring of the Qinghai–Tibet Railway". In: *Natural Hazards and Earth System Sciences* 19.10, pp. 2229–2240. DOI: 10.5194/nhess-19-2229-2019.

Zwart, S. and M. Vries (2016). "Philosophy of technology after the empirical turn". In: Springer. Chap. Methodological Classification of Innovative Engineering Projects.

Appendix A

Python code to calculate the average rates of InSAR data points

Appendix A contains the Python code to calculate the average rates of all the data points applied in the case study of each location. The out-filtered data points with the confidential interval method are considered in this code.

```
import numpy as np
import pandas as pd
%matplotlib inline
import matplotlib.pyplot as plt
from matplotlib.patches import Rectangle
from pandas import read_csv
import pyproj
from datetime import datetime
import math
from scipy.stats import norm

def outliers(data):
    # Get data
    data = read_csv(f'{data}', index_col=[0])
    data = data.iloc[19:, 0][:]
    data = {'Data' : data}
    data = pd.DataFrame(data=data)

    # Change data
    x_data = np.zeros(len(data))
    for i in range(len(x_data)):
        x_data[i] = data.iloc[i]

    # Compute statistics
    q1 = np.percentile(x_data, 25)
    q3 = np.percentile(x_data, 75)
    iqr = q3 - q1

    #Algorithm to determine outliers
    index = []
    outlier = []
    for i in range(len(data)):
        if x_data[i] < q1 - 1.5 * iqr or x_data[i] > q3 + 1.5 * iqr:
            outlier.append(i)
        else:
            index.append(i)

        return index

def dataframe(data, threshold_bridge):
    # Define coordinates
    index = outliers(data)
    data = read_csv(f'{data}', index_col=[0])
    plot_data = np.zeros(11)
    plot_data[0] = data.loc['pnt_lat'][0]
    plot_data[1] = data.loc['pnt_lon'][0]
    plot_data[2] = data.loc['pnt_height'][0]

    # Define bridge
    angle = 180 * (math.atan(((threshold_bridge[3] - threshold_bridge[1]) /
    (threshold_bridge[2] - threshold_bridge[0])))/np.pi)
    rate = math.tan((angle * np.pi) / 180)
    height = np.sqrt((threshold_bridge[4] - threshold_bridge[2])**2 +
    (threshold_bridge[5] - threshold_bridge[3])**2)
    if rate == 0:
        x1_point = threshold_bridge[0]
        y1_point = threshold_bridge[1] + height
        x2_point = threshold_bridge[2]
        y2_point = threshold_bridge[3] + height
    else:
        x1_point = threshold_bridge[0] + height / ((1 / rate) * -1)
        y1_point = threshold_bridge[1] + np.sqrt(((height)**2)
        - ((threshold_bridge[0] - x1_point)**2))
        x2_point = threshold_bridge[2] + height / ((1 / rate) * -1)
        y2_point = threshold_bridge[3] + np.sqrt(((height)**2)
        - ((threshold_bridge[2] - x2_point)**2))

    # Cluster points
    longitude = {'Longitude' : data.loc['pnt_lon']}
    latitude = {'Latitude' : data.loc['pnt_lat']}
    coordinates = pd.DataFrame(data[longitude['Longitude'].iloc[0], l
    atitude['Latitude'].iloc[0]])

    lon_lat = np.zeros(2)
    for i in range(len(lon_lat)):
        lon_lat[i] = coordinates.iloc[i][0]

    count = 0
    if (threshold_bridge[2] - threshold_bridge[0]) * (lon_lat[1] -
    threshold_bridge[1]) - (lon_lat[0] - threshold_bridge[0]) *
    (threshold_bridge[3] - threshold_bridge[1]) > 0 and (x1_point - x2_point) *
    (lon_lat[1] - y2_point) - (lon_lat[0] - x2_point) *
    (y1_point - y2_point) > 0 and (x2_point - threshold_bridge[2]) *
    (lon_lat[1] - y2_point) - (lon_lat[0] - x2_point) *
    (y2_point - threshold_bridge[3]) > 0 and (threshold_bridge[0] - x1_point) *
    (lon_lat[1] - threshold_bridge[1]) - (lon_lat[0] - threshold_bridge[0]) *
    (threshold_bridge[3] - y2_point) > 0:
        count += 2
    else:
        count = 0
    plot_data[4] = count
```

```
# Define displacements
data = data.iloc[19:, 0][:]
data = {'Data' : data}
data = pd.DataFrame(data=data)
data = data.iloc[index]

# Calculate difference in years between first point and last point
difference = datetime.strptime(str(data.iloc[-1].name)[2:10], "%Y%m%d") -
datetime.strptime(str(data.iloc[0].name)[2:10], "%Y%m%d")
difference_in_years = (difference.days + difference.seconds/86400)/365.2425

# Calculate number of data points
data_points = np.arange(0, len(data))

# Set up A matrix
A = np.column_stack((np.ones((len(data), 1)), data_points))

# Set up Q_yy
Q_yy = np.diag(np.ones(len(data)))

# Set up y-vector
y_vector = np.zeros(len(data))
for i in range(len(y_vector)):
    y_vector[i] = data.iloc[i]

# Calculate x-hat
x_hat = np.linalg.inv(np.transpose(A) @ np.linalg.inv(Q_yy)
@ A) @ np.transpose(A) @ np.linalg.inv(Q_yy) @ y_vector

# Calculate y-hat
y_hat = A @ x_hat

# Calculate Q_yy_new
Q_yy_new = np.diag((y_vector - y_hat)**2)

# Calculate x-hat
x_hat = np.linalg.inv(np.transpose(A) @ np.linalg.inv(Q_yy_new) @ A)
@ np.transpose(A) @ np.linalg.inv(Q_yy_new) @ y_vector

# Calculate fitted line
y_hat = A @ x_hat

# Calculate Sigma_x_hat
Sigma_x_hat = np.linalg.inv(np.transpose(A) @ np.linalg.inv(Q_yy_new) @ A)
plot_data[8] = ((np.sqrt(len(data))) / (np.sqrt(difference_in_years))) *
np.sqrt(np.diag(Sigma_x_hat))[1]

# Calculate rate
rate = len(data) * (x_hat[1] / difference_in_years)
plot_data[3] = rate

# Calculate RMSE
nsum = np.sum((y_vector - (A @ x_hat)) ** 2)
E = np.sqrt((1/len(data))*nsum)
plot_data[5] = E
```

```
    # Calculate Mean and Standard Deviation
    plot_data[6] = np.mean(y_vector)
    plot_data[7] = np.std(y_vector)
    return plot_data

def ecdf(files, threshold_bridge, quantile, location):
    # Get RMSEs
    data = []
    for i in range(len(files)):
        data.append(dataframe(files[i], threshold_bridge)[5])

    df = pd.DataFrame(data={'Data' : data}, index=files)
    df = df.sort_values(by='Data', ascending=True)

    # Sort data
    data = sorted(data)

    # Calculate probabilities
    p_data = np.zeros(len(data))
    for i in range(len(p_data)):
        p_data[i] = (i+1) / (len(p_data) + 1)

    # Fit normal distribution
    x_values = np.linspace(np.min(data), np.max(data), 100)
    y_values = norm.cdf(x_values, loc=np.mean(data), scale=np.std(data))

    # Filter out outliers
    alist = []
    for i in range(len(p_data)):
        if p_data[i] > quantile:
            alist.append(i)

    erased_files = []
    for i in range(len(alist)):
        erased_files.append(df.iloc[alist[i]].name)

    return erased_files

def erase_files(files, threshold_bridge, quantile, location):
    erased_files = ecdf(files, threshold_bridge, quantile, location)

    # Generate which should be removed from files
    alist = []
    for i in range(len(files)):
        for j in range(len(erased_files)):
            if files[i] == erased_files[j]:
                alist.append(i)

    # Filter files
    filtered_files = [item for idx, item in enumerate(files) if idx not in alist]
    return filtered_files

def simulation(files, threshold_bridge, quantile, location):
    files = erase_files(files, threshold_bridge, quantile, location)
    # Get rates and errors from dataframe function
    not_on_the_bridge = []
    bridge = []
    for i in range(len(files)):
        if dataframe(files[i], threshold_bridge)[4] == 2:
            bridge.append(dataframe(files[i], threshold_bridge)[3])
        else:
            not_on_the_bridge.append(dataframe(files[i], threshold_bridge)[3])

    return np.mean(bridge), np.mean(not_on_the_bridge)

def plot_simulation(files, threshold_bridge, quantile, location):
    data = np.zeros((len(quantile), 2))
    for i in range(len(quantile)):
        for j in range(len(simulation(files, threshold_bridge, quantile[i],
                                      location))):
            data[i, j] = simulation(files, threshold_bridge, quantile[i],
                                    location)[j]

    # Create data for plot
    labels = ['Bridge', 'Not on the Bridge']
    confidence_80 = [data[0, 0], data[0, 1]]
    confidence_90 = [data[1, 0], data[1, 1]]
    confidence_95 = [data[2, 0], data[2, 1]]

    x = np.arange(len(labels))  # the label locations
    width = 0.2  # the width of the bars

    fig, ax = plt.subplots()
    rects1 = ax.bar(x - width, confidence_80, width,
                    label='80 % Confidence interval')
    rects2 = ax.bar(x, confidence_90, width,
                    label='90 % Confidence interval')
    rects3 = ax.bar(x + width, confidence_95, width,
                    label='95 % Confidence interval')

    # Add some text for labels, title and custom x-axis tick labels, etc.
    ax.set_ylabel('Settlement rate [m/year]')
    ax.set_title(f'Mean settlement rate at {location}')
    ax.set_xticks(x)
    ax.set_xticklabels(labels)
    ax.legend()
    plt.savefig(f'Meansettlementrate{location}.png')
    return
```

Appendix B

Average deformation and the Python code

Appendix B contains the average deformation plots of all the available data points at all three locations on and off the bridge. It also contains the Python code to calculate this average deformation.

To show the displacements of each data point visibly, a plot with the average displacements of the data points on the bridge and not on the bridge has been created, shown in Figure B.1. In Figure B.1, more than 50 data points not on the bridge are shown, and about 5 data points on the bridge are shown. The plot contains all the data points, including the not yet out filtered ones with their RMSEs. The plot shows that the data points on the bridge generally have slower settlement rates than those not on the bridge.

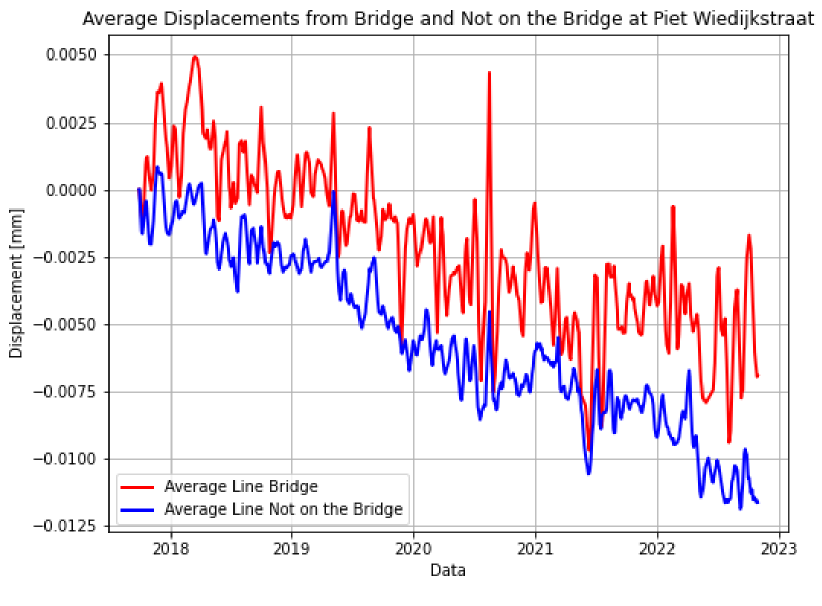


Figure B.1: Average displacements at Piet Wiedijkstraat from data points on the bridge and not on the bridge.

Figure B.2 shows that the displacements of the data points not on the bridge are more significant than those on the bridge. However, the average displacements of the data points on the bridge are sometimes more significant than those of the data points not on the bridge. A reason for this phenomenon is that the number of available data points is less than at the Piet Wiedijkstraat, which is 40 instead of 60. The results will show how this will affect the proposed prediction models.
Figure B.3 shows that the displacements of the data points not on the bridge are more significant than those

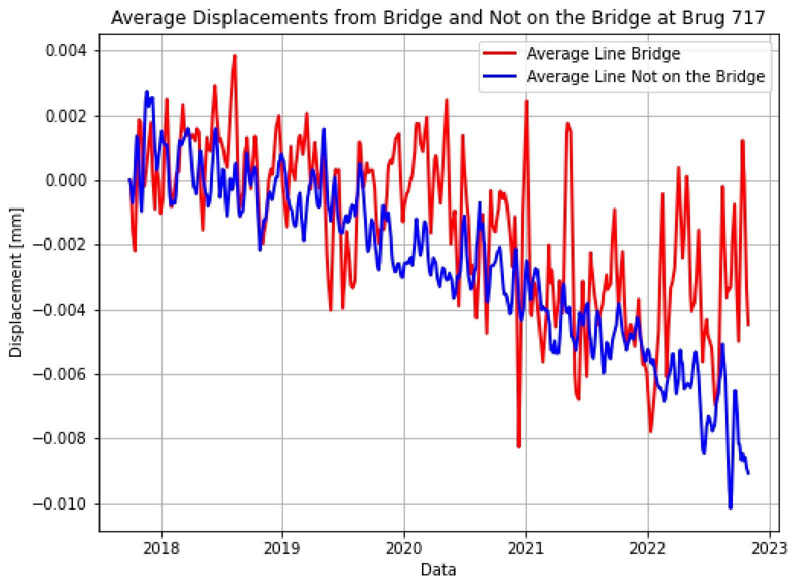


Figure B.2: Average displacements at Brug 717 from data points on the bridge and not on the bridge.

on the bridge.

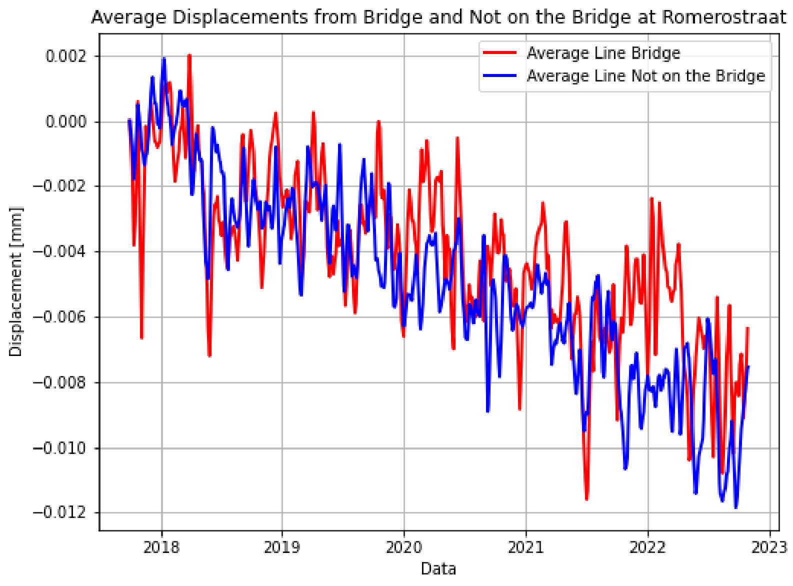


Figure B.3: Average displacements at the location Romerostraat from data points on the bridge and not on the bridge.

```
# Importing packages
import numpy as np
%matplotlib inline
import matplotlib.pyplot as plt
import pandas as pd
```

```
from pandas import read_csv
from scipy import spatial
from datetime import datetime
import time
import math
from matplotlib.lines import Line2D

def InSAR_points(data, threshold_bridge):
    # Get data
    data = read_csv(f'{data}', index_col=[0])

    # Define coordinates
    lon_lat = np.zeros(2)
    lon_lat[0] = data.loc['pnt_lon'][0]
    lon_lat[1] = data.loc['pnt_lat'][0]

    # Define bridge
    angle = 180 * (math.atan(((threshold_bridge[3] - threshold_bridge[1]) /
    (threshold_bridge[2] - threshold_bridge[0]))))/np.pi)
    rate = math.tan((angle * np.pi) / 180)
    height = np.sqrt((threshold_bridge[4] - threshold_bridge[2])**2 +
    (threshold_bridge[5] - threshold_bridge[3])**2)
    if rate == 0:
        x1_point = threshold_bridge[0]
        y1_point = threshold_bridge[1] + height
        x2_point = threshold_bridge[2]
        y2_point = threshold_bridge[3] + height
    else:
        x1_point = threshold_bridge[0] + height / ((1 / rate) * -1)
        y1_point = threshold_bridge[1] + np.sqrt(((height)**2)
        - ((threshold_bridge[0] - x1_point)**2))
        x2_point = threshold_bridge[2] + height / ((1 / rate) * -1)
        y2_point = threshold_bridge[3] + np.sqrt(((height)**2)
        - ((threshold_bridge[2] - x2_point)**2))

    # Calculate if point is lying on bridge or not on the bridge
    count = 0
    if (threshold_bridge[2] - threshold_bridge[0]) * (lon_lat[1] -
    threshold_bridge[1]) - (lon_lat[0] - threshold_bridge[0]) *
    (threshold_bridge[3] - threshold_bridge[1]) > 0 and (x1_point - x2_point) *
    (lon_lat[1] - y2_point) - (lon_lat[0] - x2_point) *
    (y1_point - y2_point) > 0 and (x2_point -
    threshold_bridge[2]) * (lon_lat[1] - y2_point) -
    (lon_lat[0] - x2_point) * (y2_point - threshold_bridge[3])
    > 0 and (threshold_bridge[0] - x1_point) * (lon_lat[1] -
    threshold_bridge[1]) - (lon_lat[0] - threshold_bridge[0]) *
    (threshold_bridge[3] - y2_point) > 0:
        count += 1
    else:
        count = 0

    # Define displacements
    data = data.iloc[19:, 0][:]
    data_values = np.zeros(len(data))
    for i in range(len(data_values)):
        data_values[i] = data.iloc[i]
    # Give the time values
    time_values = np.zeros(len(data))
    for i in range(len(time_values)):
        time_values[i] = data.index[i][2:10]

    time_steps = np.zeros((3, len(time_values)))
    for i in range(len(time_values)):
        time_steps[0, i] = time.strptime(str(int(time_values[i])), "%Y%m%d")[0]
        time_steps[1, i] = time.strptime(str(int(time_values[i])), "%Y%m%d")[1]
        time_steps[2, i] = time.strptime(str(int(time_values[i])), "%Y%m%d")[2]

    data = {'Year' : time_steps[0, :], 'Month' : time_steps[1, :],
    'Day' : time_steps[2, :]}
    df_time = pd.DataFrame(data=data)
    for i in range(len(df_time)):
        for j in range(len(df_time.keys())):
            df_time.iloc[i, j] = str(int(df_time.iloc[i, j]))

    df_time['date'] = pd.to_datetime(df_time[['Year', 'Month', 'Day']])

    # Give the displacements
    if count == 1:
        df_displacements = {'Bridge' : data_values}
    else:
        df_displacements = {'Not on the bridge' : data_values}

    df_displacements = pd.DataFrame(data=df_displacements, index=df_time['date'])
    return df_displacements, count

    def create_dfs(files, threshold_bridge):
    # Append all the dataframes
    dfs_bridge = []
    dfs_not_on_the_bridge = []
    for i in range(len(files)):
        if InSAR_points(files[i], threshold_bridge)[1] == 1:
            dfs_bridge.append(InSAR_points(files[i], threshold_bridge)[0])
        else:
            dfs_not_on_the_bridge.append(InSAR_points(files[i],
            threshold_bridge)[0])
    return dfs_bridge, dfs_not_on_the_bridge

def plot_displacement_points_bridge(files, threshold_bridge, title):
    dfs_bridge = create_dfs(files, threshold_bridge)[0]

    ax = None

    for i, df in enumerate(dfs_bridge):
        # For each dataframe, plot on the same axis
        ax = df.plot(ax=ax, legend=False, figsize=(6, 4))

    # Make the plot smooth
    ax.set_ylabel('Displacement [m]')
    ax.set_title(f'Plot of displacements at the bridge at {title}')
    ax.set_ylim(-0.04 , 0.03)
    return

def plot_displacement_points_not_on_the_bridge(files, threshold_bridge, title):
    dfs_not_on_the_bridge = create_dfs(files, threshold_bridge)[1]

    ax = None

    for i, df in enumerate(dfs_not_on_the_bridge):
        # For each dataframe, plot on the same axis
        ax = df.plot(ax=ax, legend=False, figsize=(6, 4))

    # Make the plot smooth
    ax.set_ylabel('Displacement [m]')
    ax.set_title(f'Plot of displacements not on the bridge at {title}')
    ax.set_ylim(-0.04 , 0.03)

    return

def create_average_line(df):
    # Create x-data
    all_x = np.unique(np.concatenate([d.index for d in df]))
    data = pd.DataFrame(data={'Data' : all_x})

    # Interpolate y-values for each DataFrame in the list
    interpolated_y = []
    for d in df:
        interpolated_y.append(np.interp(data, d.index, d[f'{df[0].keys()[0]}']))

    # Calculate the mean y-values
    average_y = np.mean(interpolated_y, axis=0)
    return all_x, average_y

def plot_averages(files, threshold_bridge, location):
    # Get the data
    dfs_bridge, dfs_not_on_the_bridge = create_dfs(files, threshold_bridge)
    all_x_bridge, average_y_bridge = create_average_line(dfs_bridge)
    all_x_not_on_the_bridge, average_y_not_on_the_bridge =
    create_average_line(dfs_not_on_the_bridge)

    # Plot the average line
    plt.figure(figsize=(8, 6))
    plt.plot(all_x_bridge, average_y_bridge, label="Average Line Bridge",
    color="red", linewidth=2)
    plt.plot(all_x_not_on_the_bridge, average_y_not_on_the_bridge,
    label="Average Line Not on the Bridge", color="blue", linewidth=2)

    # Customize the plot
    plt.title(f"Average Line from Bridge and Not on the Bridge at {location}")
    plt.xlabel("Data")
    plt.ylabel("Displacement [m]")
    plt.legend()
    plt.grid(True)

    # Show the plot
    plt.show()
    return
```

Appendix C

Python code to calculate link InSAR data and GVB data

Appendix C contains the Python code to calculate and plot the link between InSAR data and GVB Track gauge data.

```
import numpy as np
import pandas as pd
from pandas import read_csv
from datetime import datetime
import math

def outliers(data):
    # Get data
    data = read_csv(f'{data}', index_col=[0])
    data = data.iloc[19:, 0][:]
    data = {'Data' : data}
    data = pd.DataFrame(data=data)

    # Change data
    x_data = np.zeros(len(data))
    for i in range(len(x_data)):
        x_data[i] = data.iloc[i]

    # Compute statistics
    q1 = np.percentile(x_data, 25)
    q3 = np.percentile(x_data, 75)
    iqr = q3 - q1

    #Algorithm to determine outliers
    index = []
    outlier = []
    for i in range(len(data)):
        if x_data[i] < q1 - 1.5 * iqr or x_data[i] > q3 + 1.5 * iqr:
            outlier.append(i)
        else:
            index.append(i)

    return index

def bridge(threshold_bridge):
    # Define bridge
    angle = 180 * (math.atan(((threshold_bridge[3] - threshold_bridge[1]) /
    (threshold_bridge[2] - threshold_bridge[0]))))/np.pi)
    rate = math.tan((angle * np.pi) / 180)
    height = np.sqrt((threshold_bridge[4] - threshold_bridge[2])**2 +
    (threshold_bridge[5] - threshold_bridge[3])**2)
    if rate == 0:
        x1_point = threshold_bridge[0]
        y1_point = threshold_bridge[1] + height
        x2_point = threshold_bridge[2]
        y2_point = threshold_bridge[3] + height
    else:
        x1_point = threshold_bridge[0] + height / ((1 / rate) * -1)
        y1_point = threshold_bridge[1] + np.sqrt(((height)**2) -
        ((threshold_bridge[0] - x1_point)**2))
        x2_point = threshold_bridge[2] + height / ((1 / rate) * -1)
        y2_point = threshold_bridge[3] + np.sqrt(((height)**2) -
        ((threshold_bridge[2] - x2_point)**2))

    # Middle point of the threshold circle
    x_middle_1 = x1_point + ((threshold_bridge[0] - x1_point) / 2)
    y_middle_1 = y1_point + ((threshold_bridge[1] - y1_point) / 2)
    x_middle_2 = x2_point + ((threshold_bridge[2] - x2_point) / 2)
    y_middle_2 = y2_point + ((threshold_bridge[3] - y2_point) / 2)
    x_middle = np.min([x_middle_1, x_middle_2]) + (np.abs(x_middle_2 -
    x_middle_1) / 2)
    y_middle = np.min([y_middle_1, y_middle_2]) + (np.abs(y_middle_2 -
    y_middle_1) / 2)
    return x1_point, y1_point, x2_point, y2_point, x_middle, y_middle

def calculate_E(data, threshold_bridge):
    # Define coordinates
    index = outliers(data)
    data = read_csv(f'{data}', index_col=[0])

    # Get coordinates of the bridge
    x1_point, y1_point, x2_point, y2_point, x_middle, y_middle =
    bridge(threshold_bridge)

    # Define displacements
    data = data.iloc[19:, 0][:]
    data = {'Data' : data}
    data = pd.DataFrame(data=data)
    data = data.iloc[index]

    # Calculate difference in years between first point and last point
    difference = datetime.strptime(str(data.iloc[-1].name)[2:10], "%Y%m%d") -
    datetime.strptime(str(data.iloc[0].name)[2:10], "%Y%m%d")
    difference_in_years = (difference.days + difference.seconds/86400)/365.2425

    # Calculate number of data points
    data_points = np.arange(0, len(data))

    # Set up A matrix
    A = np.column_stack((np.ones((len(data), 1)), data_points))

    # Set up Q_yy
    Q_yy = np.diag(np.ones(len(data)))

    # Set up y-vector
    y_vector = np.zeros(len(data))
    for i in range(len(y_vector)):
        y_vector[i] = data.iloc[i]

    # Calculate x-hat
    x_hat = np.linalg.inv(np.transpose(A) @ np.linalg.inv(Q_yy) @ A) @
    np.transpose(A) @ np.linalg.inv(Q_yy) @ y_vector

    # Calculate y-hat
    y_hat = A @ x_hat

    # Calculate Q_yy_new
    Q_yy_new = np.diag((y_vector - y_hat)**2)

    # Calculate x-hat
    x_hat = np.linalg.inv(np.transpose(A) @ np.linalg.inv(Q_yy_new) @ A) @
    np.transpose(A) @ np.linalg.inv(Q_yy_new) @ y_vector

    # Calculate RMSE
    nsum = np.sum((y_vector - (A @ x_hat)) ** 2)
    E = np.sqrt((1/len(data))*nsum)
    return E

def ecdf(files, threshold_bridge, quantile):
    # Get RMSEs
    data = []
    for i in range(len(files)):
        data.append(calculate_E(files[i], threshold_bridge))

    df = pd.DataFrame(data={'Data' : data}, index=files)
    df = df.sort_values(by='Data', ascending=True)

    # Sort data
    data = sorted(data)

    # Calculate probabilities
    p_data = np.zeros(len(data))
    for i in range(len(p_data)):
        p_data[i] = (i+1) / (len(p_data) + 1)

    # Fit normal distribution
    x_values = np.linspace(np.min(data), np.max(data), 100)
    y_values = norm.cdf(x_values, loc=np.mean(data), scale=np.std(data))

    # Filter out outliers
    alist = []
    for i in range(len(p_data)):
        if p_data[i] > quantile:
            alist.append(i)

    erased_files = []
    for i in range(len(alist)):
        erased_files.append(df.iloc[alist[i]].name)
    return erased_files

def erase_files(files, threshold_bridge, quantile):
    erased_files = ecdf(files, threshold_bridge, quantile)

    # Generate which should be removed from files
    alist = []
    for i in range(len(files)):
        for j in range(len(erased_files)):
            if files[i] == erased_files[j]:
                alist.append(i)

    # Filter files
    filtered_files = [item for idx, item in enumerate(files)
    if idx not in alist]
    return filtered_files

def BLUE_year(data, year, threshold, threshold_bridge):
    # Define coordinates
    data = read_csv(f'{data}', index_col=[0])
    plot_data = np.zeros(5)
    plot_data[0] = data.loc['pnt_lat'][0]
    plot_data[1] = data.loc['pnt_lon'][0]
    plot_data[2] = data.loc['pnt_height'][0]

    # Get coordinates of the bridge
    x1_point, y1_point, x2_point, y2_point, x_middle, y_middle =
    bridge(threshold_bridge)

    # Cluster points
    longitude = {'Longitude' : data.loc['pnt_lon']}
    latitude = {'Latitude' : data.loc['pnt_lat']}
    coordinates = pd.DataFrame(data=[longitude['Longitude'].iloc[0],
    latitude['Latitude'].iloc[0]])

    lon_lat = np.zeros(2)
    for i in range(len(lon_lat)):
        lon_lat[i] = coordinates.iloc[i][0]

    count = 0
    if (threshold_bridge[2] - threshold_bridge[0]) * (lon_lat[1] -
    threshold_bridge[1]) - (lon_lat[0] - threshold_bridge[0]) *
    (threshold_bridge[3] - threshold_bridge[1]) > 0 and (x1_point - x2_point) *
    (lon_lat[1] - y2_point) - (lon_lat[0] - x2_point) * (y1_point - y2_point)
    > 0 and (x2_point - threshold_bridge[2]) * (lon_lat[1] - y2_point) -
    (lon_lat[0] - x2_point) * (y2_point - threshold_bridge[3]) > 0 and
    (threshold_bridge[0] - x1_point) * (lon_lat[1] - threshold_bridge[1]) -
    (lon_lat[0] - threshold_bridge[0]) * (threshold_bridge[3] - y2_point) > 0:
        count += 2
    elif threshold - np.sqrt(((lon_lat[0] - x_middle)**2) + ((lon_lat[1] -
    y_middle)**2)) < 0:
        count += 1
    else:
        count = 0
    plot_data[4] = count
```

```
# Separate data with chosen year
chosen_year = []
chosen_year_index = []
for i in range(len(data)):
    if data.iloc[i].name[2:6] == f'{year}':
        chosen_year.append(data.iloc[i][0])
        chosen_year_index.append(data.iloc[i].name[2:10])

# Create new dataframe
new_df = {'Index' : chosen_year_index, 'Data' : chosen_year}
new_df = pd.DataFrame(data=new_df)

# Filter out outliers
x_data = np.zeros(len(new_df['Data']))
for i in range(len(x_data)):
    x_data[i] = new_df['Data'][i]

q1 = np.percentile(x_data, 25)
q3 = np.percentile(x_data, 75)
iqr = q3 - q1

index = []
outlier = []
for i in range(len(x_data)):
    if x_data[i] < q1 - 1.5 * iqr or x_data[i] > q3 + 1.5 * iqr:
        outlier.append(i)
    else:
        index.append(i)

new_df = new_df.iloc[index]

# Calculate difference in years between first point and last point
difference = datetime.strptime(new_df['Index'].iloc[-1], "%Y%m%d") -
datetime.strptime(new_df['Index'].iloc[0], "%Y%m%d")
difference_in_years = (difference.days + difference.seconds/86400)/365.2425

# Calculate number of data points
data_points = np.arange(0, len(new_df))

# Set up A matrix
A = np.column_stack((np.ones((len(new_df), 1)), data_points))

# Set up Q_yy
Q_yy = np.diag(np.ones(len(new_df)))

# Set up y-vector
y_vector = np.zeros(len(new_df))
for i in range(len(y_vector)):
    y_vector[i] = new_df['Data'][new_df['Data'].keys()[i]]

# Calculate x-hat
x_hat = np.linalg.inv(np.transpose(A) @ np.linalg.inv(Q_yy) @ A) @
np.transpose(A) @ np.linalg.inv(Q_yy) @ y_vector

# Calculate y-hat
```

```
    y_hat = A @ x_hat

    # Calculate Q_yy_new
    Q_yy_new = np.diag((y_vector - y_hat)**2)

    # Calculate x-hat
    x_hat = np.linalg.inv(np.transpose(A) @ np.linalg.inv(Q_yy_new) @ A) @
    np.transpose(A) @ np.linalg.inv(Q_yy_new) @ y_vector

    # Calculate rate
    rate = len(data) * (x_hat[1] / difference_in_years)
    plot_data[3] = rate

    return plot_data

def calculation_degradation_per_year(files, year, threshold, threshold_bridge):
    # Get the rates of data points on the bridge and not on the bridge
    rates_bridge = []
    rates_not_on_the_bridge = []
    for i in range(len(files)):
        if BLUE_year(files[i], year, threshold, threshold_bridge)[4] == 2:
            rates_bridge.append(BLUE_year(files[i], year, threshold,
            threshold_bridge)[3])
        elif BLUE_year(files[i], year, threshold, threshold_bridge)[4] == 0:
            rates_not_on_the_bridge.append(BLUE_year(files[i], year, threshold,
            threshold_bridge)[3])
    return rates_bridge, rates_not_on_the_bridge

def calculation_mean_degradation_per_year(files, year, threshold,
threshold_bridge):
    rates = calculation_degradation_per_year(files, year, threshold,
    threshold_bridge)
    data = {'Bridge' : np.mean(rates[0]), 'Not-on-the-bridge' : np.mean(rates[1])}
    return np.mean(rates[0]), np.mean(rates[1])

def track_gauge_updated(year, threshold, threshold_bridge):
    # Get coordinates of the bridge
    x1_point, y1_point, x2_point, y2_point, x_middle, y_middle =
    bridge(threshold_bridge)

    #Get data from Track gauge
    data_track_gauge = read_csv(f'Trackgauge{year}.csv', sep=';')['Gauge']
    data_lon = read_csv(f'Trackgauge{year}.csv', sep=';')['Start-Long']
    data_lat = read_csv(f'Trackgauge{year}.csv', sep=';')['Start-Lat']
    lon_lat_gauge = np.zeros((3, len(data_track_gauge)))

    # Convert commas to dots
    for i in range(len(data_track_gauge)):
        data_lon[i] = data_lon[i].replace(',', '.')
        data_lat[i] = data_lat[i].replace(',', '.')
        data_track_gauge[i] = data_track_gauge[i].replace(',', '.')
        lon_lat_gauge[0, i] = data_lon[i]
        lon_lat_gauge[1, i] = data_lat[i]
        lon_lat_gauge[2, i] = data_track_gauge[i]

    # Set up a DataFrame
    df = pd.DataFrame(data={'Longitude' : lon_lat_gauge[0, :], 'Latitude' :
    lon_lat_gauge[1, :], 'Gauge' : lon_lat_gauge[2, :]})

    # Separate groups in the dataset
    bridge_data = []
    not_on_the_bridge_data = []
    not_on_the_bridge_longitude = []
    for i in range(len(df)):
        if (threshold_bridge[2] - threshold_bridge[0]) * (df['Latitude'][i] -
        threshold_bridge[1]) - (df['Longitude'][i] - threshold_bridge[0]) *
        (threshold_bridge[3] - threshold_bridge[1]) > 0 and (x1_point -
        x2_point) * (df['Latitude'][i] - y2_point) - (df['Longitude'][i] -
        x2_point) * (y1_point - y2_point) > 0 and (x2_point -
        threshold_bridge[2]) * (df['Latitude'][i] - y2_point) -
        (df['Longitude'][i] - x2_point) * (y2_point - threshold_bridge[3]) > 0
        and (threshold_bridge[0] - x1_point) * (df['Latitude'][i] -
        threshold_bridge[1]) - (df['Longitude'][i] - threshold_bridge[0]) *
        (threshold_bridge[3] - y2_point) > 0:
            bridge_data.append(df['Gauge'][i])
        elif threshold - np.sqrt(((df['Longitude'][i] - x_middle)**2) +
        ((df['Latitude'][i] - y_middle)**2)) < 0:
            not_on_the_bridge_data.append(df['Gauge'][i])

    return np.std(bridge_data), np.std(not_on_the_bridge_data)

def making_dataframe(files, year, threshold, threshold_bridge):
    data = []
    for i in range(len(year)):
        data.append({'Rate-Bridge' : calculation_mean_degradation_per_year(files,
        year[i], threshold, threshold_bridge)[0], 'Rate-Not-on-the-bridge' :
        calculation_mean_degradation_per_year(files, year[i], threshold,
        threshold_bridge)[1], 'Sigma-Bridge' : track_gauge_updated(year[i],
        threshold, threshold_bridge)[0], 'Sigma-Not-on-the-bridge' :
        track_gauge_updated(year[i], threshold, threshold_bridge)[1]})
    return pd.DataFrame(data=data, index=year)

def algorithm_threshold(files, year, threshold, threshold_bridge):
    # Determine Pearson Correlation Coefficients
    corr_values = np.zeros(len(threshold))
    for i in range(len(corr_values)):
        corr_values[i] = making_dataframe(files, year, threshold[i],
        threshold_bridge).corr().iloc[1, 3]

    return corr_values
```

Appendix D

Python code to plot the yearly twist data

Appendix D contains the Python code to plot the yearly twist of the data points on the bridge and not on the bridge to support the track gauge data.

```
    import numpy as np
import pandas as pd
%matplotlib inline
import matplotlib.pyplot as plt
from pandas import read_csv
import math

def bridge(threshold_bridge):
    # Define bridge
    angle = 180 * (math.atan(((threshold_bridge[3] - threshold_bridge[1]) /
    (threshold_bridge[2] - threshold_bridge[0])))/np.pi)
    rate = math.tan((angle * np.pi) / 180)
    height = np.sqrt((threshold_bridge[4] - threshold_bridge[2])**2 +
    (threshold_bridge[5] - threshold_bridge[3])**2)
    if rate == 0:
        x1_point = threshold_bridge[0]
        y1_point = threshold_bridge[1] + height
        x2_point = threshold_bridge[2]
        y2_point = threshold_bridge[3] + height
    else:
        x1_point = threshold_bridge[0] + height / ((1 / rate) * -1)
        y1_point = threshold_bridge[1] + np.sqrt(((height)**2) -
        ((threshold_bridge[0] - x1_point)**2))
        x2_point = threshold_bridge[2] + height / ((1 / rate) * -1)
        y2_point = threshold_bridge[3] + np.sqrt(((height)**2) -
        ((threshold_bridge[2] - x2_point)**2))

    # Middle point of the threshold circle
    x_middle_1 = x1_point + ((threshold_bridge[0] - x1_point) / 2)
    y_middle_1 = y1_point + ((threshold_bridge[1] - y1_point) / 2)
    x_middle_2 = x2_point + ((threshold_bridge[2] - x2_point) / 2)
    y_middle_2 = y2_point + ((threshold_bridge[3] - y2_point) / 2)
    x_middle = np.min([x_middle_1, x_middle_2]) + (np.abs(x_middle_2 -
    x_middle_1) / 2)
    y_middle = np.min([y_middle_1, y_middle_2]) + (np.abs(y_middle_2 -
    y_middle_1) / 2)
    return x1_point, y1_point, x2_point, y2_point, x_middle, y_middle

def twist(year, threshold, threshold_bridge):
    # Get coordinates of the bridge
    x1_point, y1_point, x2_point, y2_point, x_middle, y_middle =
    bridge(threshold_bridge)

    #Get data from Twist
    data = read_csv(f'Twist{year}.csv', sep=';')
    data_lon = data['Start-Long']
    data_lat = data['Start-Lat']
    lon_lat_twist = np.zeros((len(data_lon), 2))

    # Convert commas to dots
    for i in range(len(lon_lat_twist)):
        lon_lat_twist[i, 0] = data_lon.iloc[i].replace(',', '.')
        lon_lat_twist[i, 1] = data_lat.iloc[i].replace(',', '.')

    # Set up a DataFrame
    df = pd.DataFrame(data={'Longitude' : lon_lat_twist[:, 0],
    'Latitude' : lon_lat_twist[:, 1]})

    # Separate groups in the dataset
    bridge_data_2000 = []
    bridge_data_6000 = []
    bridge_data_8000 = []
    values_not_on_the_bridge_data_2000 = []
    values_not_on_the_bridge_data_6000 = []
    values_not_on_the_bridge_data_8000 = []
    for i in range(len(df)):
        if (threshold_bridge[2] - threshold_bridge[0]) * (df['Latitude'][i] -
        threshold_bridge[1]) - (df['Longitude'][i] - threshold_bridge[0]) *
        (threshold_bridge[3] - threshold_bridge[1]) > 0 and (x1_point - x2_point) *
        (df['Latitude'][i] - y2_point) - (df['Longitude'][i] - x2_point) *
        (y1_point - y2_point) > 0 and (x2_point - threshold_bridge[2]) *
        (df['Latitude'][i] - y2_point) - (df['Longitude'][i] - x2_point) *
        (y2_point - threshold_bridge[3]) > 0 and (threshold_bridge[0] -
        x1_point) * (df['Latitude'][i] - threshold_bridge[1]) -
        (df['Longitude'][i] - threshold_bridge[0]) * (threshold_bridge[3] -
        y2_point) > 0:
                bridge_data_2000.append(data['Twist-2000'][i])
                bridge_data_6000.append(data['Twist-6000'][i])
                bridge_data_8000.append(data['Twist-8000'][i])
        elif threshold - np.sqrt(((df['Longitude'][i] - x_middle)**2) +
        ((df['Latitude'][i] - y_middle)**2)) < 0:
            values_not_on_the_bridge_data_2000.append(data['Twist-2000'][i])
            values_not_on_the_bridge_data_6000.append(data['Twist-6000'][i])
            values_not_on_the_bridge_data_8000.append(data['Twist-8000'][i])
    return bridge_data_2000, bridge_data_6000, bridge_data_8000,
    values_not_on_the_bridge_data_2000, values_not_on_the_bridge_data_6000,
    values_not_on_the_bridge_data_8000

def dataframes(data):
    new_data = np.zeros(len(data))
    for i in range(len(data)):
        data[i] = str(data[i]).replace(',', '.')
        new_data[i] = data[i]
    return new_data[~np.isnan(new_data)]

def sigmas_data(year, threshold, threshold_bridge):
    bridge_data_2000, bridge_data_6000, bridge_data_8000,
    values_not_on_the_bridge_data_2000, values_not_on_the_bridge_data_6000,
    values_not_on_the_bridge_data_8000 = twist(year, threshold, threshold_bridge)
    return np.std(dataframes(bridge_data_2000)),
    np.std(dataframes(bridge_data_6000)),
    np.std(dataframes(bridge_data_8000)),
    np.std(dataframes(values_not_on_the_bridge_data_2000)),
    np.std(dataframes(values_not_on_the_bridge_data_6000)),
    np.std(dataframes(values_not_on_the_bridge_data_8000))

def making_dataframe(year, threshold, threshold_bridge):
    data = []
    for i in range(len(year)):
        data.append({'Sigma-Bridge-Twist-2000' : sigmas_data(year[i], threshold,
        threshold_bridge)[0], 'Sigma-Bridge-Twist-6000' : sigmas_data(year[i],
        threshold, threshold_bridge)[1], 'Sigma-Bridge-Twist-8000' :
        sigmas_data(year[i], threshold, threshold_bridge)[2],
            'Sigma-Not-on-the-bridge-Twist-2000' : sigmas_data(year[i], threshold,
            threshold_bridge)[3],
            'Sigma-Not-on-the-bridge-Twist-6000' : sigmas_data(year[i], threshold,
            threshold_bridge)[4],
            'Sigma-Not-on-the-bridge-Twist-8000' : sigmas_data(year[i], threshold,
            threshold_bridge)[5]})
    return pd.DataFrame(data=data, index=year)
```

Appendix E

Python code to apply stochastic rates method

Appendix E contains the Python code to apply the proposed stochastic rates method to the three proposed locations.

```
import numpy as np
import pandas as pd
%matplotlib inline
import matplotlib.pyplot as plt
from matplotlib.patches import Rectangle
from pandas import read_csv
import pyproj
from datetime import datetime
import math

def outliers(data):
    # Get data
    data = read_csv(f'{data}', index_col=[0])
    data = data.iloc[19:, 0][:]
    data = {'Data' : data}
    data = pd.DataFrame(data=data)

    # Change data
    x_data = np.zeros(len(data))
    for i in range(len(x_data)):
        x_data[i] = data.iloc[i]

    # Compute statistics
    q1 = np.percentile(x_data, 25)
    q3 = np.percentile(x_data, 75)
    iqr = q3 - q1

    #Algorithm to determine outliers
    index = []
    outlier = []
    for i in range(len(data)):
        if x_data[i] < q1 - 1.5 * iqr or x_data[i] > q3 + 1.5 * iqr:
            outlier.append(i)
        else:
            index.append(i)
    return index

def bridge(threshold_bridge):
    # Define bridge
    angle = 180 * (math.atan(((threshold_bridge[3] - threshold_bridge[1]) /
    (threshold_bridge[2] - threshold_bridge[0]))))/np.pi)
    rate = math.tan((angle * np.pi) / 180)
    height = np.sqrt((threshold_bridge[4] - threshold_bridge[2])**2 +
    (threshold_bridge[5] - threshold_bridge[3])**2)
    if rate == 0:
        x1_point = threshold_bridge[0]
        y1_point = threshold_bridge[1] + height
        x2_point = threshold_bridge[2]
        y2_point = threshold_bridge[3] + height
    else:
        x1_point = threshold_bridge[0] + height / ((1 / rate) * -1)
        y1_point = threshold_bridge[1] + np.sqrt(((height)**2)
        - ((threshold_bridge[0] - x1_point)**2))
        x2_point = threshold_bridge[2] + height / ((1 / rate) * -1)
        y2_point = threshold_bridge[3] + np.sqrt(((height)**2)
        - ((threshold_bridge[2] - x2_point)**2))

    # Middle point of the threshold circle
    x_middle_1 = x1_point + ((threshold_bridge[0] - x1_point) / 2)
    y_middle_1 = y1_point + ((threshold_bridge[1] - y1_point) / 2)
    x_middle_2 = x2_point + ((threshold_bridge[2] - x2_point) / 2)
    y_middle_2 = y2_point + ((threshold_bridge[3] - y2_point) / 2)
    x_middle = np.min([x_middle_1, x_middle_2]) + (np.abs(x_middle_2 -
    x_middle_1) / 2)
    y_middle = np.min([y_middle_1, y_middle_2]) + (np.abs(y_middle_2 -
    y_middle_1) / 2)
    return x1_point, y1_point, x2_point, y2_point, x_middle, y_middle

def RMSE_reference_point(data):
    # Get the data of the reference points
    index = outliers(data)
    data = read_csv(f'{data}', index_col=[0])
    data = data.iloc[19:, 0][:]
    data = {'Data' : data}
    data_values = pd.DataFrame(data=data)
    data_values = data_values.iloc[index]

    # Calculate difference in years between first point and last point
    difference = datetime.strptime(str(data_values.iloc[-1].name)[2:10],
    "%Y%m%d") - datetime.strptime(str(data_values.iloc[0].name)[2:10], "%Y%m%d")
    difference_in_years = (difference.days + difference.seconds/86400)/365.2425

    # Give the time values
    time_values = np.zeros(len(data_values))
    for i in range(len(time_values)):
        time_values[i] = data_values.index[i][2:10]

    time_steps = np.zeros((3, len(time_values)))
    for i in range(len(time_values)):
        time_steps[0, i] = time.strptime(str(int(time_values[i])), "%Y%m%d")[0]
        time_steps[1, i] = time.strptime(str(int(time_values[i])), "%Y%m%d")[1]
        time_steps[2, i] = time.strptime(str(int(time_values[i])), "%Y%m%d")[2]

    data = {'Year' : time_steps[0, :], 'Month' : time_steps[1, :],
    'Day' : time_steps[2, :]}
    df_time = pd.DataFrame(data=data)
    for i in range(len(df_time)):
        for j in range(len(df_time.keys())):
            df_time.iloc[i, j] = str(int(df_time.iloc[i, j]))

    df_time['date'] = pd.to_datetime(df_time[['Year', 'Month', 'Day']])

    # Calculate number of data points
    data_points = np.arange(0, len(data_values))

    # Set up A matrix
    A = np.column_stack((np.ones((len(data_values), 1)), data_points))

    # Set up Q_yy
    Q_yy = np.diag(np.ones(len(data_values)))

    # Set up y-vector
    y_vector = np.zeros(len(data_values))
    for i in range(len(y_vector)):
        y_vector[i] = data_values['Data'][data_values['Data'].keys()[i]]

    # Calculate x-hat
    x_hat = np.linalg.inv(np.transpose(A) @ np.linalg.inv(Q_yy) @ A) @
    np.transpose(A) @ np.linalg.inv(Q_yy) @ y_vector

    # Calculate y-hat
    y_hat = A @ x_hat

    # Calculate Q_yy_new
    Q_yy_new = np.diag((y_vector - y_hat)**2)

    # Calculate x-hat
    x_hat = np.linalg.inv(np.transpose(A) @ np.linalg.inv(Q_yy_new) @ A) @
    np.transpose(A) @ np.linalg.inv(Q_yy_new) @ y_vector

    # Calculate Root Mean Square Error
    nsum = np.sum((y_vector - (A @ x_hat)) ** 2)
    E = np.sqrt((1/len(data_values))*nsum)

    # Calculate rate
    rate = len(data_values) * (x_hat[1] / difference_in_years)

    return rate, E, pd.DataFrame(data={'Data' : y_vector},
    index=df_time['date'])

def lowest_E(files):
    data = []
    E = []
    for i in range(len(files)):
        data.append(RMSE_reference_point(files[i])[0])
        E.append(RMSE_reference_point(files[i])[1])
    return RMSE_reference_point(f'{files[np.argmin(E)]}')[2]
```

```python
def rate_reference_point(files, start_date, end_date):
    # Get data and filter based on start and end dates
    df = lowest_E(files)
    filtered_df = df[(df.index >= start_date) & (df.index <= end_date)]

    # Calculate difference in years between first point and last point
    difference = datetime.strptime(str(filtered_df.iloc[-1].name)[0:10].
    replace("-", ""), "%Y%m%d") -
    datetime.strptime(str(filtered_df.iloc[0].name)[0:10].
    replace("-", ""), "%Y%m%d")
    difference_in_years = (difference.days + difference.seconds/86400)/365.2425

    # Calculate number of data points
    data_points = np.arange(0, len(filtered_df))

    # Set up A matrix
    A = np.column_stack((np.ones((len(filtered_df), 1)), data_points))

    # Set up Q_yy
    Q_yy = np.diag(np.ones(len(filtered_df)))

    # Set up y-vector
    y_vector = np.zeros(len(filtered_df))
    for i in range(len(y_vector)):
        y_vector[i] = filtered_df['Data'][filtered_df['Data'].keys()[i]]

    # Calculate x-hat
    x_hat = np.linalg.inv(np.transpose(A) @ np.linalg.inv(Q_yy) @ A) @
    np.transpose(A) @ np.linalg.inv(Q_yy) @ y_vector

    # Calculate y-hat
    y_hat = A @ x_hat

    # Calculate Q_yy_new
    Q_yy_new = np.diag((y_vector - y_hat)**2)

    # Calculate x-hat
    x_hat = np.linalg.inv(np.transpose(A) @ np.linalg.inv(Q_yy_new) @ A) @
    np.transpose(A) @ np.linalg.inv(Q_yy_new) @ y_vector

    # Calculate Root Mean Square Error
    nsum = np.sum((y_vector - (A @ x_hat)) ** 2)
    E = np.sqrt((1/len(filtered_df))*nsum)

    # Calculate rate
    rate = len(filtered_df) * (x_hat[1] / difference_in_years)
    return rate

def rate(data, reference_files, threshold_bridge, start_date, end_date):
    # Get the data of the reference points
    index = outliers(data)
    data = read_csv(f'{data}', index_col=[0])
    df_data = np.zeros(3)
    df_data[0] = data.loc['pnt_lat'][0]
    df_data[1] = data.loc['pnt_lon'][0]
```

```
df_data[2] = data.loc['pnt_height'][0]
data = data.iloc[19:, 0][:]
data = {'Data' : data}
data_values = pd.DataFrame(data=data)
data_values = data_values.iloc[index]

# Get the coordinates of the bridge
x1_point, y1_point, x2_point, y2_point, x_middle, y_middle =
bridge(threshold_bridge)

# Cluster points
coordinates = pd.DataFrame(data=[df_data[1], df_data[0]])

lon_lat = np.zeros(2)

for i in range(len(lon_lat)):
    lon_lat[i] = coordinates.iloc[i][0]

count = 0
if (threshold_bridge[2] - threshold_bridge[0]) * (lon_lat[1] -
threshold_bridge[1]) - (lon_lat[0] - threshold_bridge[0]) *
(threshold_bridge[3] - threshold_bridge[1]) > 0 and
(x1_point - x2_point) * (lon_lat[1] - y2_point) -(lon_lat[0] - x2_point) *
(y1_point - y2_point) > 0 and (x2_point - threshold_bridge[2]) *
(lon_lat[1] - y2_point) - (lon_lat[0] - x2_point) * (y2_point -
threshold_bridge[3]) > 0 and (threshold_bridge[0] - x1_point) *
(lon_lat[1] - threshold_bridge[1]) - (lon_lat[0] - threshold_bridge[0]) *
(threshold_bridge[3] - y2_point) > 0:
    count += 2
else:
    count = 0

# Give the time values
time_values = np.zeros(len(data_values))
for i in range(len(time_values)):
    time_values[i] = data_values.index[i][2:10]

time_steps = np.zeros((3, len(time_values)))
for i in range(len(time_values)):
    time_steps[0, i] = time.strptime(str(int(time_values[i])), "%Y%m%d")[0]
    time_steps[1, i] = time.strptime(str(int(time_values[i])), "%Y%m%d")[1]
    time_steps[2, i] = time.strptime(str(int(time_values[i])), "%Y%m%d")[2]

df_time = {'Year' : time_steps[0, :], 'Month' : time_steps[1, :],
'Day' : time_steps[2, :]}
df_time = pd.DataFrame(data=df_time)
for i in range(len(df_time)):
    for j in range(len(df_time.keys())):
        df_time.iloc[i, j] = str(int(df_time.iloc[i, j]))

df_time['date'] = pd.to_datetime(df_time[['Year', 'Month', 'Day']])

# Get data and filter based on start and end dates
data_array = []
for i in range(len(data_values)):
    data_array.append(data_values['Data'].iloc[i])
    df = pd.DataFrame(data={'Data' : data_array}, index=df_time['date'])
    filtered_df = df[(df.index >= start_date) & (df.index <= end_date)]

    # Calculate difference in years between first point and last point
    difference = datetime.strptime(str(filtered_df.iloc[-1].name)[0:10].
    replace("-", ""), "%Y%m%d") -
    datetime.strptime(str(filtered_df.iloc[0].name)[0:10].
    replace("-", ""), "%Y%m%d")
    difference_in_years = (difference.days + difference.seconds/86400)/365.2425

    # Calculate number of data points
    data_points = np.arange(0, len(filtered_df))

    # Set up A matrix
    A = np.column_stack((np.ones((len(filtered_df), 1)), data_points))

    # Set up Q_yy
    Q_yy = np.diag(np.ones(len(filtered_df)))

    # Set up y-vector
    y_vector = np.zeros(len(filtered_df))
    for i in range(len(y_vector)):
        y_vector[i] = filtered_df['Data'][filtered_df['Data'].keys()[i]]

    # Calculate x-hat
    x_hat = np.linalg.inv(np.transpose(A) @ np.linalg.inv(Q_yy) @ A) @
    np.transpose(A) @ np.linalg.inv(Q_yy) @ y_vector

    # Calculate y-hat
    y_hat = A @ x_hat

    # Calculate Q_yy_new
    Q_yy_new = np.diag((y_vector - y_hat)**2)

    # Calculate x-hat
    x_hat = np.linalg.inv(np.transpose(A) @ np.linalg.inv(Q_yy_new) @ A) @
    np.transpose(A) @ np.linalg.inv(Q_yy_new) @ y_vector

    # Calculate Root Mean Square Error
    nsum = np.sum((y_vector - (A @ x_hat)) ** 2)
    E = np.sqrt((1/len(filtered_df))*nsum)

    # Calculate rate
    rate = len(filtered_df) * (x_hat[1] / difference_in_years)

    # Get rate from reference point
    rate_reference = rate_reference_point(reference_files, start_date, end_date)
    return rate, count

def stochastic_rates_new_version(data, reference_files, threshold_bridge, dates,
start_date, end_date, prediction_date):
    # Create an array with all the dates which will be used as train values
    dates_data = []
    for i in range(len(dates)):
        if end_date == dates[i]:
            break
        dates_data.append(dates[i])
    dates_data.append(end_date)

    # Calculate the rates of one point
    rates = []
    for i in range(len(dates_data)-1):
        rates.append(rate(data, reference_files, threshold_bridge,
        dates_data[i], dates_data[i+1])[0])

    rate_prediction = rate(data, reference_files, threshold_bridge,
    dates_data[-1], prediction_date)[0] - rate_reference_point(reference_files,
    dates_data[-1], prediction_date)

    # Determine if point is positioning on the bridge or not on the bridge
    count = 0
    if rate(data, reference_files, threshold_bridge, start_date, end_date)[1] == 2:
        count = 1
    return rates, count, rate_prediction

def testing_stochastic_rates(files, reference_files, threshold_bridge, dates,
start_date, end_date, prediction_date, amount_train, quantile):
    # Get all the data of rates on the bridge and not on the bridge
    bridge_data = []
    not_on_the_bridge_data = []
    bridge_data_test = []
    not_on_the_bridge_data_test = []
    for i in range(len(files)):
        if stochastic_rates_new_version(files[i], reference_files,
        threshold_bridge, dates, start_date, end_date, prediction_date)[1] == 1:
            bridge_data.append(stochastic_rates_new_version(files[i],
            reference_files, threshold_bridge, dates, start_date, end_date,
            prediction_date)[0])
            bridge_data_test.append(stochastic_rates_new_version(files[i],
            reference_files, threshold_bridge, dates, start_date, end_date,
            prediction_date)[2])
        else:
            not_on_the_bridge_data.append(stochastic_rates_new_version(files[i],
            reference_files, threshold_bridge, dates, start_date, end_date,
            prediction_date)[0])
            not_on_the_bridge_data_test.append(stochastic_rates_new_version(
            files[i], reference_files, threshold_bridge, dates, start_date,
            end_date, prediction_date)[2])

    # Separate a training and test data set
    index_train_bridge = random.sample(list(np.arange(0, len(bridge_data))),
    k=int(amount_train * len(bridge_data)))
    index_train_not_on_the_bridge = random.sample(list(np.arange(0,
    len(not_on_the_bridge_data))), k=int(amount_train *
    len(not_on_the_bridge_data)))
    index_test_bridge = [num for num in list(np.arange(0, len(bridge_data_test)))
    if num not in index_train_bridge]
    index_test_not_on_the_bridge = [num for num in list(np.arange(0,
    len(not_on_the_bridge_data_test))) if num not in index_train_not_on_the_bridge]

    df_bridge_train = [bridge_data[i] for i in index_train_bridge]
    df_not_on_the_bridge_train = [not_on_the_bridge_data[i] for i in
```

```
index_train_not_on_the_bridge]
df_bridge_test = [bridge_data_test[i] for i in
index_test_bridge]
df_not_on_the_bridge_test = [not_on_the_bridge_data_test[i] for i in
index_test_bridge]

train_data_bridge = []
for i in range(len(df_bridge_train)):
    for j in range(len(df_bridge_train[0])):
        train_data_bridge.append(df_bridge_train[i][j])

train_data_not_on_the_bridge = []
for i in range(len(df_not_on_the_bridge_train)):
    for j in range(len(df_not_on_the_bridge_train[0])):
        train_data_not_on_the_bridge.append(df_not_on_the_bridge_train[i][j])

# Create normal distribution to detect high rates of track degradation on points
# lying not on the bridge
train_data_not_on_the_bridge = sorted(train_data_not_on_the_bridge)
p_data_not_on_the_bridge = np.zeros(len(train_data_not_on_the_bridge))
for i in range(len(p_data_not_on_the_bridge)):
    p_data_not_on_the_bridge[i] = (i+1) / (len(p_data_not_on_the_bridge) + 1)
x_values_not_on_the_bridge = np.linspace(np.min(train_data_not_on_the_bridge),
np.max(train_data_not_on_the_bridge), 100)
y_values_not_on_the_bridge = norm.cdf(x_values_not_on_the_bridge,
loc=np.mean(train_data_not_on_the_bridge),
scale=np.std(train_data_not_on_the_bridge))
index_not_on_the_bridge = 0
for i in range(len(y_values_not_on_the_bridge)):
    if y_values_not_on_the_bridge[i] > quantile:
        index_not_on_the_bridge = i-1
        break
critical_value_not_on_the_bridge =
x_values_not_on_the_bridge[index_not_on_the_bridge]
count_not_on_the_bridge = 0
for i in range(len(df_not_on_the_bridge_test)):
    if df_not_on_the_bridge_test[i] < critical_value_not_on_the_bridge:
        count_not_on_the_bridge += 1

# Create normal distribution to detect high rates of track degradation on points
# lying on the bridge
train_data_bridge = sorted(train_data_bridge)
p_data_bridge = np.zeros(len(train_data_bridge))
for i in range(len(p_data_bridge)):
    p_data_bridge[i] = (i+1) / (len(p_data_bridge) + 1)
x_values_bridge = np.linspace(np.min(train_data_bridge),
np.max(train_data_bridge), 100)
y_values_bridge = norm.cdf(x_values_bridge, loc=np.mean(train_data_bridge),
scale=np.std(train_data_bridge))
index_bridge = 0
for i in range(len(y_values_bridge)):
    if y_values_bridge[i] > quantile:
        index_bridge = i-1
        break
critical_value_bridge = x_values_bridge[index_bridge]
count_bridge = 0
```

```
    for i in range(len(df_bridge_test)):
        if df_bridge_test[i] < critical_value_bridge:
            count_bridge += 1
    return ((count_bridge + count_not_on_the_bridge) / (len(df_bridge_test) +
    len(df_not_on_the_bridge_test))) * 100,
    np.std(train_data_bridge), np.std(train_data_not_on_the_bridge)


def simulation(files, reference_files, threshold_bridge, dates, amount_train,
quantile):
    simulation_data = np.zeros(len(dates)-2)
    sigma_bridge = np.zeros(len(dates)-2)
    sigma_not_on_the_bridge = np.zeros(len(dates)-2)
    for i in range(len(simulation_data)):
        simulation_data[i], sigma_bridge[i],
        sigma_not_on_the_bridge[i] = testing_stochastic_rates(files,
        reference_files, threshold_bridge, dates,
        dates[0], dates[i+1], dates[i+2], amount_train, quantile)

    data = {'Percentage' : simulation_data, 'Sigma-bridge' : sigma_bridge,
    'Sigma-not-on-the-bridge' : sigma_not_on_the_bridge}
    return pd.DataFrame(data=data, index=dates[2:])
```

Appendix F

Python code to apply prediction method based on BLUE

Appendix F contains the Python code to apply the prediction method based on BLUE to all three proposed locations.

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from pandas import read_csv
from datetime import datetime
from matplotlib.patches import Rectangle
import time
import math

def outliers(data):
    # Get data
    data = read_csv(f'{data}', index_col=[0])
    data = data.iloc[19:, 0][:]
    data = {'Data' : data}
    data = pd.DataFrame(data=data)

    # Change data
    x_data = np.zeros(len(data))
    for i in range(len(x_data)):
        x_data[i] = data.iloc[i]

    # Compute statistics
    q1 = np.percentile(x_data, 25)
    q3 = np.percentile(x_data, 75)
    iqr = q3 - q1

    #Algorithm to determine outliers
    index = []
    outlier = []
    for i in range(len(data)):
        if x_data[i] < q1 - 1.5 * iqr or x_data[i] > q3 + 1.5 * iqr:
            outlier.append(i)
        else:
            index.append(i)

    return index
```

```
def bridge(threshold_bridge):
    # Define bridge
    angle = 180 * (math.atan(((threshold_bridge[3] - threshold_bridge[1]) /
    (threshold_bridge[2] - threshold_bridge[0]))) / np.pi)
    rate = math.tan((angle * np.pi) / 180)
    height = np.sqrt((threshold_bridge[4] - threshold_bridge[2])**2 +
    (threshold_bridge[5] - threshold_bridge[3])**2)
    if rate == 0:
        x1_point = threshold_bridge[0]
        y1_point = threshold_bridge[1] + height
        x2_point = threshold_bridge[2]
        y2_point = threshold_bridge[3] + height
    else:
        x1_point = threshold_bridge[0] + height / ((1 / rate) * -1)
        y1_point = threshold_bridge[1] + np.sqrt(((height)**2) -
        ((threshold_bridge[0] - x1_point)**2))
        x2_point = threshold_bridge[2] + height / ((1 / rate) * -1)
        y2_point = threshold_bridge[3] + np.sqrt(((height)**2) -
        ((threshold_bridge[2] - x2_point)**2))

    # Middle point of the threshold circle
    x_middle_1 = x1_point + ((threshold_bridge[0] - x1_point) / 2)
    y_middle_1 = y1_point + ((threshold_bridge[1] - y1_point) / 2)
    x_middle_2 = x2_point + ((threshold_bridge[2] - x2_point) / 2)
    y_middle_2 = y2_point + ((threshold_bridge[3] - y2_point) / 2)
    x_middle = np.min([x_middle_1, x_middle_2]) + (np.abs(x_middle_2 -
    x_middle_1) / 2)
    y_middle = np.min([y_middle_1, y_middle_2]) + (np.abs(y_middle_2 -
    y_middle_1) / 2)
    return x1_point, y1_point, x2_point, y2_point, x_middle, y_middle

def InSAR_points(data, start_date, end_date, threshold_bridge):
    # Get the coordinates of the bridge
    x1_point, y1_point, x2_point, y2_point, x_middle, y_middle =
    bridge(threshold_bridge)

    # Get the data of the reference points
    index = outliers(data)
    data = read_csv(f'{data}', index_col=[0])
    lon = data.loc['pnt_lon'][0]
    lat = data.loc['pnt_lat'][0]
    data = data.iloc[19:, 0][:]
    data = {'Data' : data}
    data_values = pd.DataFrame(data=data)
    data_values = data_values.iloc[index]

    # Cluster points
    longitude = {'Longitude' : lon}
    latitude = {'Latitude' : lat}
    coordinates = pd.DataFrame(data=[longitude['Longitude'],
    latitude['Latitude']])

    lon_lat = np.zeros(2)
    for i in range(len(lon_lat)):
        lon_lat[i] = coordinates.iloc[i][0]
```

```
count = 0
if (threshold_bridge[2] - threshold_bridge[0]) * (lon_lat[1] -
threshold_bridge[1]) - (lon_lat[0] - threshold_bridge[0]) *
(threshold_bridge[3] - threshold_bridge[1]) > 0 and (x1_point - x2_point) *
(lon_lat[1] - y2_point) - (lon_lat[0] - x2_point) * (y1_point - y2_point) > 0
and (x2_point - threshold_bridge[2]) * (lon_lat[1] - y2_point) -
(lon_lat[0] - x2_point) * (y2_point - threshold_bridge[3]) > 0 and
(threshold_bridge[0] - x1_point) * (lon_lat[1] - threshold_bridge[1]) -
(lon_lat[0] - threshold_bridge[0]) * (threshold_bridge[3] - y2_point) > 0:
    count += 2
else:
    count = 0

# Set up the displacements in an y-vector
y_vector = np.zeros(len(data_values))
for i in range(len(y_vector)):
    y_vector[i] = data_values.iloc[i, 0]

# Calculate difference in years between first point and last point
difference = datetime.strptime(str(data_values.iloc[-1].name)[2:10],
"%Y%m%d") - datetime.strptime(str(data_values.iloc[0].name)[2:10], "%Y%m%d")
difference_in_years = (difference.days + difference.seconds/86400)/365.2425

# Give the time values
time_values = np.zeros(len(data_values))
for i in range(len(time_values)):
    time_values[i] = data_values.index[i][2:10]

time_steps = np.zeros((3, len(time_values)))
for i in range(len(time_values)):
    time_steps[0, i] = time.strptime(str(int(time_values[i])), "%Y%m%d")[0]
    time_steps[1, i] = time.strptime(str(int(time_values[i])), "%Y%m%d")[1]
    time_steps[2, i] = time.strptime(str(int(time_values[i])), "%Y%m%d")[2]

data = {'Year' : time_steps[0, :], 'Month' : time_steps[1, :],
'Day' : time_steps[2, :]}
df_time = pd.DataFrame(data=data)
for i in range(len(df_time)):
    for j in range(len(df_time.keys())):
        df_time.iloc[i, j] = str(int(df_time.iloc[i, j]))

df_time['date'] = pd.to_datetime(df_time[['Year', 'Month', 'Day']])

# Create a DataFrame and filter points out based on start date and end date
df = pd.DataFrame(data={'Data' : y_vector}, index=df_time['date'])
filtered_df = df[(df.index >= start_date) & (df.index <= end_date)]

# Calculate number of data points
days = [(date - filtered_df.index[0]).days + 1 for date in filtered_df.index]
data_points = np.arange(0, len(data_values))

# Set up A matrix
A = np.column_stack((np.ones((len(filtered_df), 1)), days))

# Set up Q_yy
Q_yy = np.diag(np.ones(len(filtered_df)))
```

```
    # Set up y-vector
    y_vector = np.zeros(len(filtered_df))
    for i in range(len(y_vector)):
        y_vector[i] = filtered_df['Data'][filtered_df['Data'].keys()[i]]

    # Calculate x-hat
    x_hat = np.linalg.inv(np.transpose(A) @ np.linalg.inv(Q_yy) @ A) @
    np.transpose(A) @ np.linalg.inv(Q_yy) @ y_vector

    # Calculate y-hat
    y_hat = A @ x_hat

    # Calculate Q_yy_new
    Q_yy_new = np.diag((y_vector - y_hat)**2)

    # Calculate x-hat updated
    x_hat = np.linalg.inv(np.transpose(A) @ np.linalg.inv(Q_yy_new) @ A) @
    np.transpose(A) @ np.linalg.inv(Q_yy_new) @ y_vector

    return A, Q_yy_new, x_hat, y_vector, count

def prediction(data, start_date, end_date, threshold_bridge):
    # Get A-matrix and x-hat
    A, Q_yy, x_hat, y_vector, count = InSAR_points(data, start_date, end_date,
    threshold_bridge)

    # Get Q_xx
    Q_xx = np.linalg.inv(A.T @ np.linalg.inv(Q_yy) @ A)

    # Get Q_zz
    A_zz = []
    for i in range(len(A)):
        A_zz.append(A[i, 1])
    A_zz = A_zz + A[-1, 1]
    A_zz = np.column_stack((np.ones((len(A_zz), 1)), A_zz))
    Q_zz = A_zz @ Q_xx @ A_zz.T

    # Calculate minimum and maximum rates
    A_linear = np.column_stack((np.ones((len(A_zz), 1)), np.arange(1,
    len(A_zz)+1)))
    min_rate = np.linalg.inv(A_linear.T @ A_linear) @ A_linear.T @
    ((A_zz @ x_hat) + np.sqrt(np.diag(Q_zz)))
    max_rate = np.linalg.inv(A_linear.T @ A_linear) @ A_linear.T @
    ((A_zz @ x_hat) - np.sqrt(np.diag(Q_zz)))
    return np.max(np.sqrt(np.diag(Q_zz))), x_hat[1], count

def predicted_rates(files, start_date, end_date, threshold_bridge):
    # Set up dataframes with rates
    df = np.zeros((3, len(files)))
    for i in range(len(df)):
        for j in range(len(files)):
            df[i, j] = prediction(files[j], start_date, end_date,
            threshold_bridge)[i]

    # Separate rates on the bridge and not on the bridge
    data = {'Max-Q_zz' : df[0, :], 'Rate' : df[1, :], 'Count' : df[2, :]}
    df = pd.DataFrame(data=data)
    df_bridge = df[df['Count'] == 2]
    df_not_on_the_bridge = df[df['Count'] == 0]

    # Calculate mean of the rates
    rates_bridge = np.zeros(2)
    rates_bridge[0] = np.max(df_bridge['Max-Q_zz'])
    rates_bridge[1] = np.mean(df_bridge['Rate'])
    rates_not_on_the_bridge = np.zeros(2)
    rates_not_on_the_bridge[0] = np.max(df_not_on_the_bridge['Max-Q_zz'])
    rates_not_on_the_bridge[1] = np.mean(df_not_on_the_bridge['Rate'])

    return rates_bridge, rates_not_on_the_bridge

def predicted_rates_dates(files, dates, threshold_bridge):
    data = []
    for i in range(len(dates)-2):
        data.append({'Rate-Bridge' : predicted_rates(files, dates[i+1],
        dates[i+2], threshold_bridge)[0][1],
        'Max-Q_zz-Bridge' : predicted_rates(files, dates[i+1], dates[i+2],
        threshold_bridge)[0][0], 'Rate-Not-on-the-bridge' : predicted_rates(files,
        dates[i+1], dates[i+2], threshold_bridge)[1][1],
        'Max-Q_zz-Not-on-the-bridge' : predicted_rates(files,
        dates[i+1], dates[i+2], threshold_bridge)[1][0]})
    return pd.DataFrame(data=data, index=dates[2:])
```