ALIGNMENT OF TUNNEL POINT CLOUD DATA USING CENTRAL LINE EXTRACTION AND REGION-BASED ROTATION AS PART OF AN AUTOMATED TUNNEL LINING INSPECTION PIPELINE

A PREPRINT

Vera Dimovska
Viapontica AI
Bayes Centre, 47 Potterrow
Edinburgh, EH8 9BT
United Kingdom
vera.dimovska@viapontica.ai

Conall Dewar
School of Mathematics
James Clerk Maxwell Building
The University of Edinburgh
Edinburgh, EH9 3FD, UK
C.M.Dewar-1@sms.ed.ac.uk

Biagio Antonelli
Viapontica AI
Bayes Centre, 47 Potterrow
Edinburgh, EH8 9BT
United Kingdom
biagio.antonelli@viapontica.ai

Vesko Cholakov
Viapontica AI
Bayes Centre, 47 Potterrow
Edinburgh, EH8 9BT
United Kingdom
vesko.cholakov@viapontica.ai

Benjamin D. Goddard
School of Mathematics
and Maxwell Institute for
Mathematical Sciences
James Clerk Maxwell Building
The University of Edinburgh
Edinburgh, EH9 3FD, UK
b.goddard@ed.ac.uk

Stuart King
School of Mathematics
and Maxwell Institute for
Mathematical Sciences
James Clerk Maxwell Building
The University of Edinburgh
Edinburgh, EH9 3FD, UK
S.King@ed.ac.uk

October 2, 2023

ABSTRACT

We present a new algorithm which transforms a point cloud scan (LiDAR 3D data) of a tunnel, with any profile shape and curvature, to a form where all tunnel cross-sections are aligned. The algorithm finds the tunnel's central line by solving a series of optimisation problems and subsequently rotates (aligns) tunnel points based on regions defined from the central line. The point cloud scans of structures provide the ability to understand important information such as structural deformation and weaknesses. However, point clouds often contain small errors that must be reduced in order to obtain relevant information. In point cloud scans of tunnels, these small errors may cause the cross-sections of the tunnels to be misaligned. The proposed algorithm transforms the tunnel data by reducing this observation error between cross sections, which allows us to more easily study any tunnel lining deformation and degradation. Few previous algorithms have been developed to reduce the misalignment of cross-sections and previous work has focused on tunnels with a circular profile. The proposed algorithm therefore usefully extends the range of tunnels which can be considered.

Keywords Point Cloud · Alignment · LiDAR.

1 Introduction

In recent years, the development of the Terrestrial Laser Scanning (TLS) or LiDAR technology has enabled high speed collection of millions of 3D points which collectively make up a point cloud [3]. Initially this technology was used widely in areas of topography, historical preservation and civil engineering research. More recently it has been used as input to 3D data analysis models for the detection of deformations and defects in structures such as bridges, buildings and, to a lesser extent, tunnels [1][4].

Here, we outline the organisation of the paper. In Section 1 we introduce, and motivate the problem and we further briefly discuss previous research on tunnel LiDAR data and the similarities with our work. In Section 2 we define two datasets that we have used to develop the algorithm. In Section 3 we explain the proposed alignment algorithm by dividing it into sub-steps. In Section 4 we present our experiments on (1) algorithm speed and (2) comparative analysis with a baseline aligning algorithm on unseen data. In Section 5 we showcase the algorithm on real tunnel data of a larger tunnel. In Section 6 we discuss limitations and future work. Finally, Section 7 holds the conclusion of the work and Section 8 contains the acknowledgements.

1.1 Problem Context & Motivation

Before any defect detection algorithm can be run on tunnel point cloud data (PCD) a number of sequential preprocessing steps must take place such as data cleaning (removal of unnecessary artifacts) and alignment. The preprocessing steps have the goal of taking any 3D tunnel scan and outputting a straight tube-like tunnel running parallel to, say, the x-axis. This lets the subsequently applied defect detection models generalise well by assuming that all tunnels start at 0 on the x-axis and run along it. Without applying an alignment algorithm standard algorithms for the prediction of the true tunnel cross section shape are difficult to implement as only local information can be used. In contrast, if the tunnel data is aligned, we are able to overlay multiple overlapping cross sections and use them together to find the true tunnel cross section shape. The difference can be seen in Figure 1, with the original tunnel data presented in Figure 2.

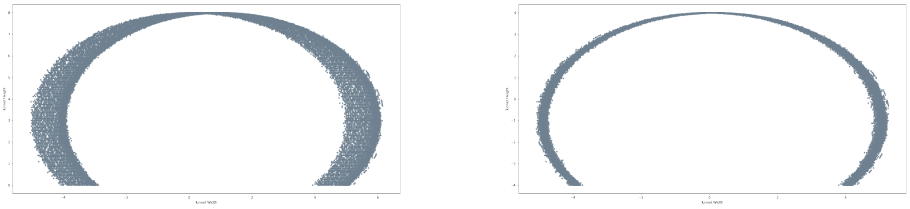

Figure 1: A visualisation of the first 10m of tunnel cross sections overlayed before alignment (left) and after alignment (right).

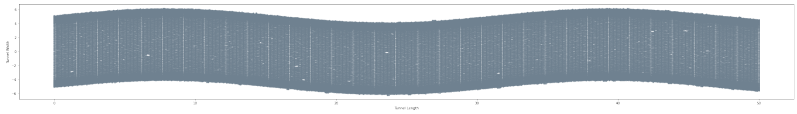

Figure 2: The tunnel before alignment (top view).

We aim to create a preprocessing data pipeline, and in turn, an alignment algorithm:

- For preprocessing "any" tunnel data, that will be agnostic to "any" data collection method, as long as we are given a tube-like 3D shape. The resulting algorithm is data-agnostic, allowing it to be applied to a wide range of data sources.
- Which will standardise the tunnel shape so further shape detection and analysis algorithms can be run on the tunnel data.

Given any shape tunnel with a "start" anywhere in the 3D space the user needs to set only a starting point and the algorithm will trace the tunnel trajectory and straighten the tunnel (align the cross sections). Unlike other approaches, our goal was to generalise to any tunnel shape[1], no matter how curved, and to be aware of the 3D data and the fact that tunnels don't necessary start at the coordinate center. For example, we performed experiments finding the tunnel trajectory by fitting two regression curves in the x-y and x-z planes. The results were case-specific and only usable if the 3D data was split into two 2D sets.

The algorithm was implemented with the goal of addressing two problems: (1) the natural tunnel curviness (e.g., the Gotthard Base Tunnel in Switzerland) and (2) local cross section misalignment.

[1] `https://civilnoteppt.com/shapes-of-tunnels/`

1.2 Previous work

There appear to be few previous papers on similar problems of aligning point cloud data representing a scanned tunnel. Most work has been performed around the extraction of the tunnel axis, but little has been done on the alignment of the cross section. However, we outline some related contributions below.

In [4], a method was derived where a cylindrical fitting technique was used to fit a tunnel with a circular profile. Then, the fitted cylinder was used to produce a 3D model that allows for the detection of deformation in the tunnel. The main restriction of this approach is that it assumes tunnels have a circular profile. In such cases, the technique is able to perform a cylinder fit to transform the dataset to a new axis aligned with the main down-tunnel direction. Post transformation, the profile of the cross section is found and a noise reduction process is carried out, after which, a 3D model of the tunnel is produced.

In [2], the process described begins by developing a noise reduction process for tunnels with a non-circular shape. This is achieved by finding the n nearest neighbours of a random tunnel point P, then fitting a plane at these points with a normal defined by the vector from the tunnel's center to P. Finally, the average distance between the plane and the neighbours is calculated. If the average distance is greater than some threshold, then P is classified as a noise point. Once the noise reduction process is complete, the tunnel is transformed to a new axis where the x-axis represents travelling across the tunnel, the y-axis represents travelling down the tunnel and the z-axis represents travelling vertically within the tunnel with a simple rotation and translation manually calculated for each tunnel.

In [5], the tunnel axis is found using an iterative fitting optimization method. The original tunnel design axis is initially used to find cross sections. These cross sections are then further fitted, with circles, and the circles' centers are approximated with a B-Spline curve. The method of fitting and approximation is repeated until it converges, i.e., each circle center is withing a set threshold of the current tunnel axis. This method assumes the cross sections have a circular shape and that the original design axis of the tunnel is known.

In [6], the axis was extracted by "obtaining the stretching direction of the drawing surface". The orientation of the tunnel was found by minimizing the projected area of the point cloud onto a plane using the density variance. This algorithm disregards tunnel curviness (bending) because they work with a short tunnel part.

In order to improve upon the techniques derived in each of these papers, we propose an alternate approach which is able to take curvature of the tunnel into consideration, as well as any shaped profile. We first briefly describe the types of both real and synthetic data used to develop an alignment algorithm, then in the subsequent section describe the algorithm in several stages of centre line determination, rotation and alignment. Lastly we describe the application of the algorithm to a range of scanned tunnel data.

2 Data

To test and develop the proposed algorithms for tunnel alignment we used two datasets: (1) a real-world PCD scan of the first 300m of a UK railway tunnel and (2) synthetic PCD. A visualisation of both can be seen in Figure 3.

The tunnel data contains several interesting features. At periodic intervals along the tunnel, there are indents in the tunnel walls; these are alcoves for staff to stand as trains pass through the tunnel. Moreover, a scattering of points can be seen above the tunnel at approximately the midpoint. This is an air-shaft which is used to bring fresh air into the tunnel for those working within. Therefore, in order to be successful, the output of the our alignment algorithm should maintain such features. The final feature, the tunnel's profile, is notably non-circular.

The synthetic data we have principally used is a cylindrical tunnel generated by finding 5000 random points with radius of the cylinder equal to 5 and overall length equal to 20. After generating points at random precisely on the surface of this cylinder they are further perturbed to simulate observation noise by moving each point by a distance value generated from the Gaussian distribution $N(0, 0.1^2)$ in a direction given by selection of two angles from a uniform distribution between 0 and 2π. Finally, to add curvature to the tunnel, the y components of the data points were adjusted using the following formula

$$y_{\text{component}} = y_{\text{component}} + 0.1 \sin(x_{\text{component}}). \tag{1}$$

The reason for using synthetic data is so we can precisely quantify the accuracy of any developed techniques, as the ground truth is known.

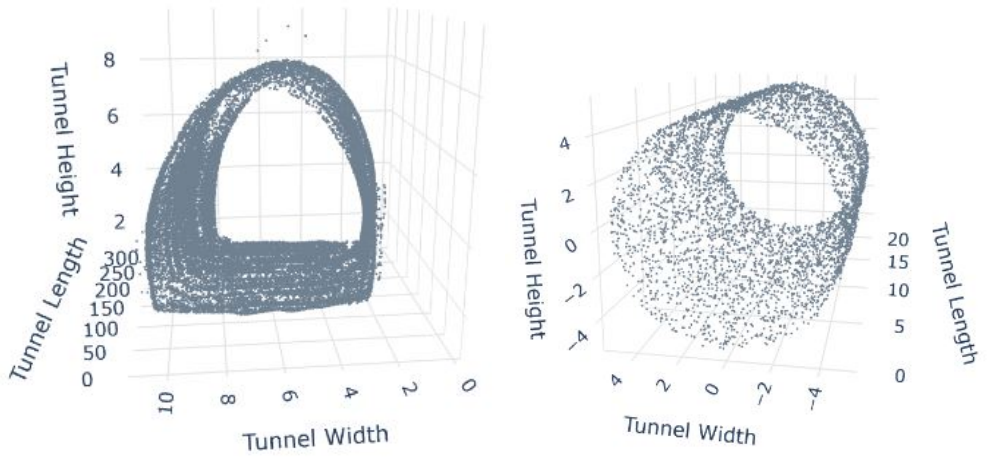

Figure 3: A visualisation of the raw PCD representing a railway tunnel (left), and the synthetic PCD (right).

3 Alignment algorithm

Each of the three subsections below represent a stage in the proposed overall process to align the tunnel cross sections. We have outlined the steps in the diagram in Figure 4. We now describe the mathematical concepts underlying each algorithm, along with their implementation.

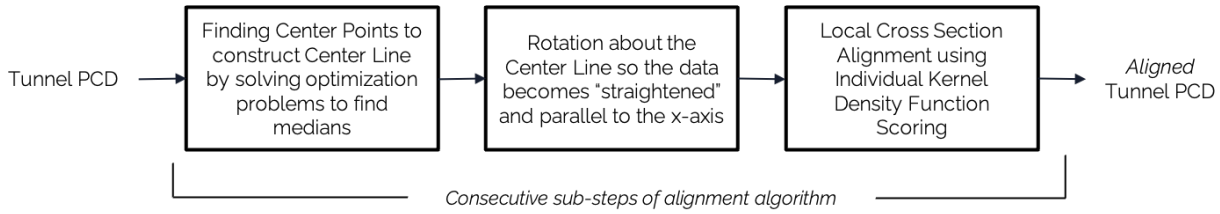

Figure 4: Pipeline of data and outline of each algorithm step.

3.1 Centre Points and Center Line

Given the PCD, the first part of the algorithm finds a set of points (each with a given step-size h apart) within the tunnel that aim to describe the tunnel's centre line.

To do this, at one end of the tunnel, a start value and a starting direction are selected, by hand. Points along the tunnel are then selected as along the central line by solving a series of optimisation problems. For some choice of constant integer K, we proceed by defining an objective function as follows

$$f(\mathbf{x}) = \text{median}(\mathbf{d_i}) \quad \text{where } \{\mathbf{d_i} = \|\mathbf{x} - \mathbf{x_i}\|_\mathbf{2} \text{ and } \mathbf{x_i} \text{ are the } K \text{ nearest points in the point cloud to location } \mathbf{x}\}. \tag{2}$$

We intend to maximise this function, finding the point maximally away from the point cloud. To ensure we continue to find points on the centre line we add the following additional constraints

$$h^2 - \|\mathbf{x} - \mathbf{o}\|_2^2 = 0, \tag{3}$$

$$\mathbf{n}.(\mathbf{x} - \mathbf{o}) \geq 0. \tag{4}$$

Here $\mathbf{o}$ is the previously found centre point, and $\mathbf{n}$ is the vector between the previous two determined centre points (at the initial step these are the manually chosen down-tunnel direction). These two conditions ensure, respectively, that the next proposed centre point is exactly on a distance h from the last, and that it is further down the tunnel.

To further decrease the effect of observation noise we fit a cubic spline to the centre points. We use a smoothness parameter to prioritise creating a smooth center line over interpolating through every center point exactly.

We have applied this algorithm to the synthetic data with $(0, 0, 0)$ as a start point, $(0.2, 0, 0)$ as a starting direction, $K = 10$ and $h = 0.2$ and the output can be visualised in Figure 5. These centre points can also be considered in the projected plane x-y, where it is expected that the points will follow the curve $y = 0.1\sin(x)$, and the x-z plane, where it is expected the points will follow the line $z = 0$. These visualisations can be seen in Figure 6.

For the synthetic data we calculated the RMSE (Root Mean Square Error) of the determined points normalised by the length of the tunnel curve, which was found to be 20.05. In the y direction this is 0.0212 and in the z direction 0.0215. The noise used to perturb the point cloud data had standard deviation 0.3162. As the y and z deviations are only 6.7% and 6.8% of the noise's standard deviation, respectively, the algorithm for determining centre point values appears robust to small amounts of observation noise in the point cloud.

We then fit a cubic spline to the synthetic data with smoothness parameter $s = 0.9$; splitting the tunnel into 200 pieces when fitting using piecewise cubic splines produces the output visualised in Figure 5. Again, to better understand how well this line fits the expected centre we can examine the projections of the line to the x-y, and x-z planes to compare the calculated centre line and the true centre line, both projections can be seen in Figure 6.

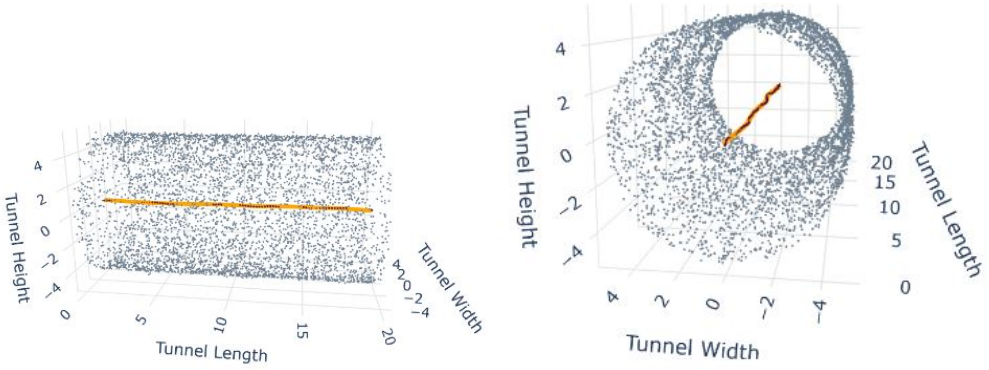

Figure 5: Visualisation of the Centre Line (Orange) and Centre Points (Purple) inside the Simulated Data Set (Grey)

In order to quantify the results achieved by this algorithm, the deviation was calculated for both y and z components by computing the RMSE and normalising as above. This results in errors of 0.0009 in both the y and z directions. This means that there was a 0.625% and 1.01% reduction in the y and z deviations respectively from that calculated simply. On comparison with the noise of in the Synthetic Cylinder data, the y and z deviations were equal to 0.295% and 0.27% of the noise's standard deviation. There still exists small deviations from the true solution which is to be expected as it would be difficult to completely reduce the effect of the noise. Additionally, the Synthetic Cylinder centre line is a lot smoother than the centre points, which will be advantageous later on when implementing the next step, the rotation algorithm, in Section 3.2.

Next, we ran the algorithm on the tunnel data with initial point $(1, 8.5, 8)$, direction $(0.2, 0, 0)$, $K = 10$ and $h = 0.2$. The determined centre points are shown in figure 6 with the projections of the points onto x-y the plane. We have omitted the x-z plane since no noise was added there. The effect of observation noise can be seen in the variability of the determined centres.

The spline fit was applied with smoothness $s = 0.5$ and again split into 200 sections; a visualisation can be seen on the same Figure 6. As the true centre of the tunnel point cloud is unknown, the projections onto the x-y and x-z planes can be calculated and compared to the equivalent projections shown earlier for the determined data points. These projections can be seen in Figure 7 where the effectiveness of the spline smoothing can be seen in the reduction in the variability of the line due to noise.

The motivation behind using a spline fit is the production of a smooth curve which provides a better basis for the next algorithm step.

3.2 Rotation of tunnel to a new axis

In this section, we introduce an algorithm that will use the fitted centre line to transform a tunnel's point cloud to a new axis. In this new axis, the x-axis will represent travelling down the tunnel, the y-axis will represent travelling across the tunnel and the z-axis will represent travelling vertically within the tunnel. Additionally, the transformation will result in the centre line of the tunnel laying on the x-axis.

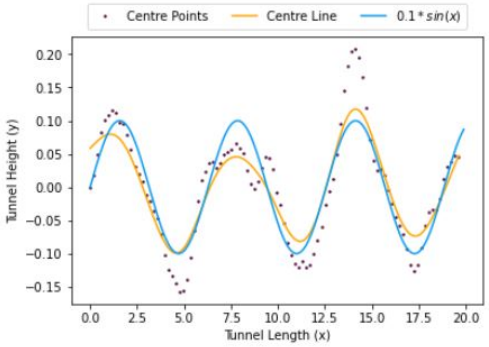

Figure 6: Projections of the Centre Points (Purple) and Centre Line (Orange) of the Simulated Data compared to the Centre Line used to generate the data on the x-y plane.

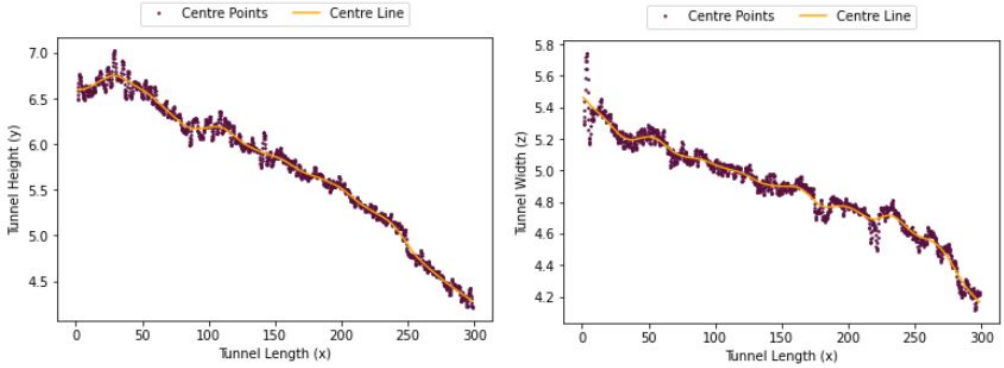

Figure 7: Projections of the Centre Points (Purple) of the Tunnel Data compared to Tunnel's Centre Line (Orange) on the x-y plane (Left) and x-z plane (Right)

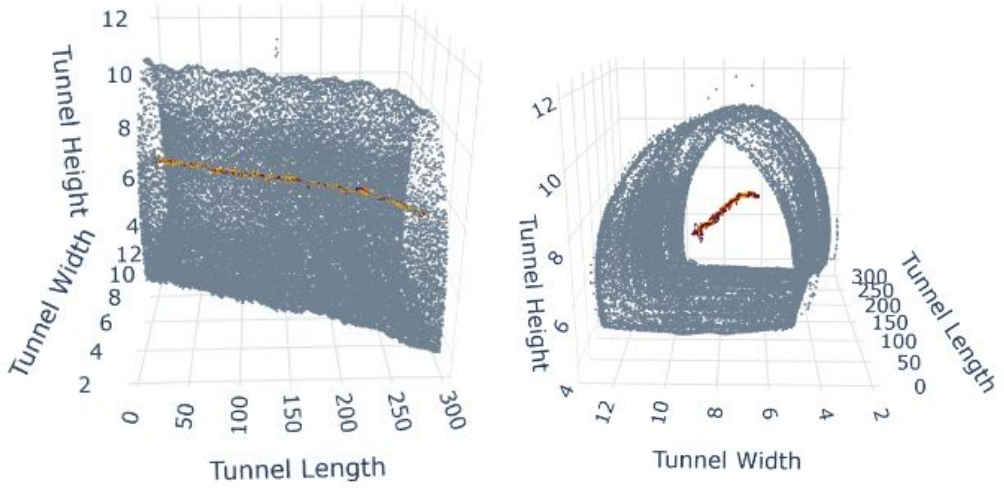

Figure 8: Visualisation of the Centre Line (Orange) and Centre Points (Purple) inside the Tunnel Data Set (Grey)

To achieve this, the vectors between points sampled from the spline fitted in Section 3.1 will be calculated and then used to define regions along the tunnel. Each region can be defined as between two planes, each sharing a normal equal to the vector joining the sample points mentioned before, and containing the sample point. A 2D visualisation of such a region can be seen in Figure 9, where the vertical lines represent the boundary planes and the horizontal line represents the region's associated vector. After the algorithm assigns each datapoint to a region, each region is transformed such

that the beginning point of the region's associated vector is mapped to the origin. Then, a further transformation is applied to the region's points that rotate all the points such that the region's associated vector lies on the x-axis. One last transformation is carried out such that the points are moved back to the correct location along the x-axis. This final transformation is achieved by summing the length of all the previous region's associated vectors and then shifting the x component of the current region's data points by this amount.

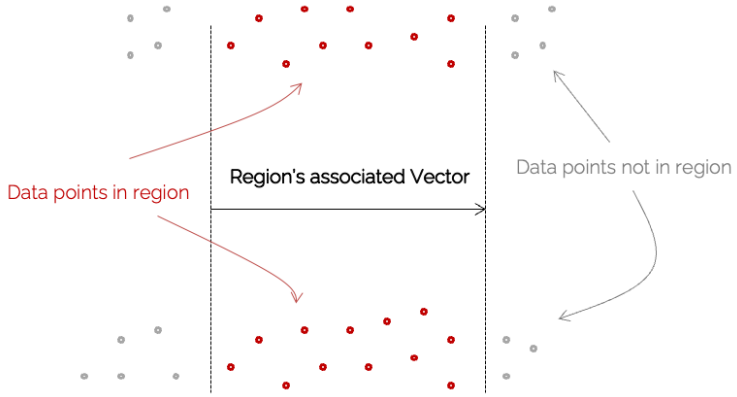

Figure 9: 2D visualisation of region selection method of the rotation algorithm between two boundary planes.

The rotation algorithm was applied to the real tunnel data set alongside the determined tunnel centre line. A plot of the resulting rotated tunnel can be seen in Figure 10. In this figure, we can see the successful rotation with the tunnel retaining its general shape whilst being transformed such that the tunnel's centre line lies on the x-axis. A minimal deviation away from both $y = 0$ and $z = 0$ can be seen for all values of x. It should also be noted that the transformation has retained the two tunnel features highlighted in Section 2 as the indents in the tunnel walls as well as the air-shaft in the tunnel's ceiling are still visible in the figure 10.

We have omitted the visual results when applying this to the synthetic data as the data was already rotated.

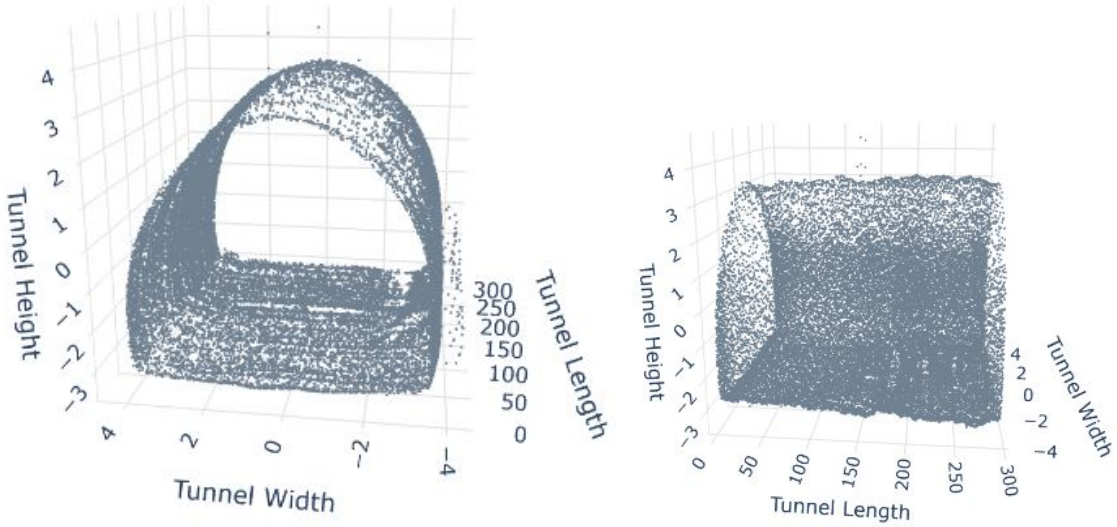

Figure 10: Visualisation of the rotated tunnel point cloud from two differing angles

3.3 Alignment of Cross sections

As discussed in Section 3.2, the output of the rotation algorithm has aligned a tunnel based on the estimation of the tunnel's centre line. However, if on examination of a rotated tunnel it is decided that further work is required to align the tunnel, there is one further step which can be taken to further fine-tune the alignment. Since the tunnel is now rotated such that it is approximately centred on the x-axis, the cross-sections can be defined by splitting a tunnel's data set into

a given number of sections along the x-axis. For a given section, its data points can be projected onto the y-z plane where a smooth function may be constructed using Kernel Density Estimation (KDE). After the smooth function is built for each cross-section, neighbouring sections can be aligned by simply scoring the points of one section using the function from their neighbour. The required translation to align the y and z components of the data points can be found using an optimisation algorithm to maximise this score. This alignment can then be repeated for all the defined cross-sections of the tunnel.

The left image of Figure 11 shows points from a cross-section of tunnel in blue that is centred on the origin point (0,0), and points from a cross-section which has been adjusted such that it is centred on the point (1,1) shown in orange, with both cross-sections having 100 data points. The output of the proposed alignment algorithm is then shown in the right image of the figure. To calculate the reliability of the algorithm, the process was repeated 20 times for different orange and blue cross-sections with random perturbations to the points used. These random perturbations were added to the cross-section points by choosing a random direction from a uniform distribution between 0 and 2π and random distance drawn from a normal distribution with mean 0 and standard deviation of 0.1. The mean change in the y coordinates for the orange sections was equal to -0.985 and the mean change in the z coordinate was equal to -1.030. The expected change was equal to -1 for both coordinates, which highlights there was an error of 0.115 and 0.03 for the y and z changes respectively. This means that there is an improvement of 88.5% in the deviation for the y-co-ordinate and a 97% improvement in the deviation for the z-co-ordinate. The proposed algorithm is therefore successful at aligning cross-sections with the remaining small errors being explained by the random noise used to perturb the cross-sections.

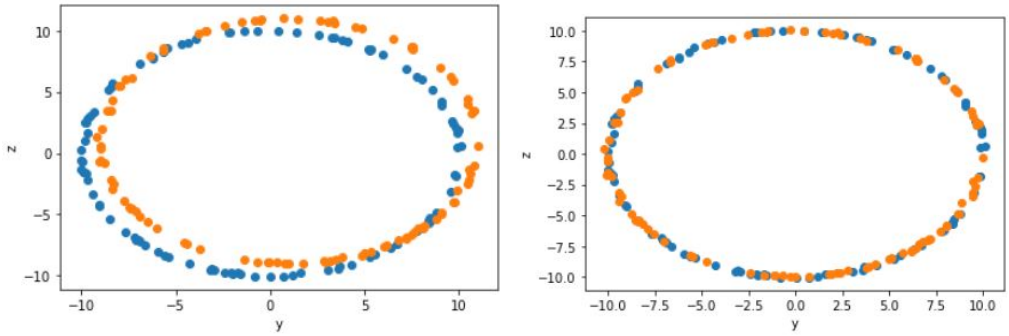

Figure 11: Visualisation of two misaligned cross sections of a circular tunnel (left), then aligned using the proposed algorithm (right).

Due to the adjustments made by the alignment algorithm being small, it is difficult to visualise its effect on the two point clouds. However, Figure 11 highlights the success of the algorithm, even in the extreme circumstances used in that image.

4 Experiments

As we are working with (a) unlabelled data that does not have an aligned ground truth to compare with and because (b) we have not found any preexisting algorithm for our use case we have tested the algorithm in two ways:

1. In terms of processing speed which we depends on the total number of points we work with.
2. By doing a comparative analysis between our algorithm and a baseline method for central line extraction (using straightforward regression) and calculating the error with the true center points.

4.1 Speed

We have tested the algorithm in terms of speed by varying the data size, i.e., the number of points. All experiments have been run on a Apple M1 (2020) Chip with 8 cores and have been averaged out from 1000 runs.

Points	10^4	10^5	10^6	50^6	10^8
Speed	30s	4min	7min	10min	15min

4.2 Comparative analysis

For these experiments the goal is to compare a baseline approach: regression in 2D spaces (x-y and x-z planes) and our alignment algorithm for the extraction of the central line.

In order to quantify the errors we have created synthetic data by creating different center lines which we consider as the ground truth. The center lines are randomly created Bezier Curves using Blender[2] and Python. Some examples, along with the results of our algorithm and the baseline regression, can be seen in Figure 12. A key observation is that regression is less accurate in examples where the direction of the tunnel changes rapidly, i.e., where the tunnel bends more quickly Figure 12. On average (on $x-y$ and $x-z$ planes), out of 1000 random center lines, the regression models shows an RMSE of 0.1 compared to our algorithm, which shows an RMSE of 0.046. However, as can be seen from Figure 13, the regression algorithm proves to be more robust to fully straight tunnels.

5 Application to other tunnel data

To test the reliability of the algorithm we tested it on unseen data from a real tunnel. The was done with the whole unprocessed tunnel which is 1115m long and consist of 122.65 million points. We show in Figure 14 the unprocessed tunnel, which has relatively high curvature compared to the data used in Section 3.

In Figure 15 we show the aligned tunnel with the algorithm run on an Intel(R) Xeon(R) Gold 6248R CPU with 32 cores which, for the tunnel downsampled by 100 times, ran for roughly 10 minutes to find the center line and rotate all of the cross sections. Note that the two "distorted" parts that can be seen on the top left side of the tunnel are not errors of the algorithm – the original data contained those features.

6 Limitations

As explained above the current algorithm depends on a user correctly choosing the right start point and start direction which means the goal of fully automating the aligning preprocessing step has not been fully accomplished. A user needs to correctly pick the start point and start direction otherwise the centre point finding fails in some cases. Figure 17 is a representative example of what happens in most cases when the direction is not set correctly. The algorithm starts to go in the opposite direction, away from the tunnel.

7 Conclusion

Point cloud data representing tunnels collected using TLS can result in misalignments making the task of studying tunnel deformation and tunnel surface degradation more difficult. In order to develop algorithms to study this alignment problem two data sets were introduced: a synthetic tunnel and a real tunnel scan with a known and unknown central line, respectively.

A sequence of algorithms were then developed and applied to these two data sets. Overall the algorithm can be seen to be effective for both data sets. In the case of the synthetic data set this can be quantified, and at each stage the RMSE involved were found to be not more than 0.0009 in y and z directions. For the real tunnel data set the final tunnel is clearly corrected to lie in line with a single axis, the additional features within the data (alcoves in the tunnel walls and the air vent in the tunnel ceiling) were also preserved. Moreover, comparing our algorithm with regression we found that it outperforms regression on tunnels with higher curvature.

Finally, we have presented the current limitations of the algorithm which happen when two starting parameters are wrongly chosen: starting point and direction. Future work will focus on automating the step of picking these parameters.

8 Acknowledgements

The current research was undertaken by Viapontica AI in association with COWI UK and the School of Mathematics at The University of Edinburgh under Innovate UK contract with Network Rail. We thank the programme sponsor Network Rail for providing access to a critical live rail environment and COWI UK for making available railway tunnel LiDAR datasets which have been instrumental for development and testing. The current work was funded to develop

[2] https://www.blender.org/

[3] `http://www.open3d.org/`

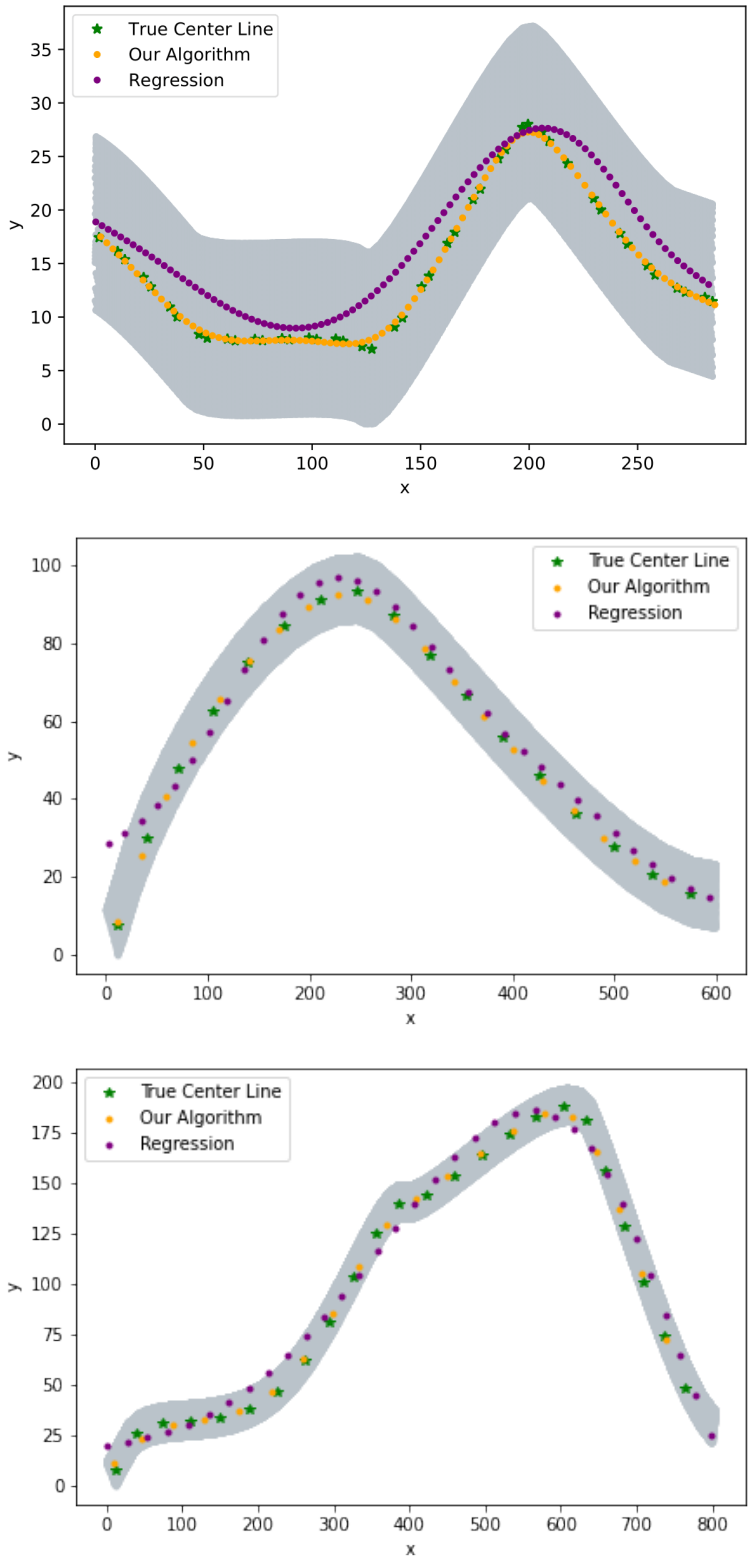

Figure 12: Examples of tunnels and the found center lines with regression and our algorithm.

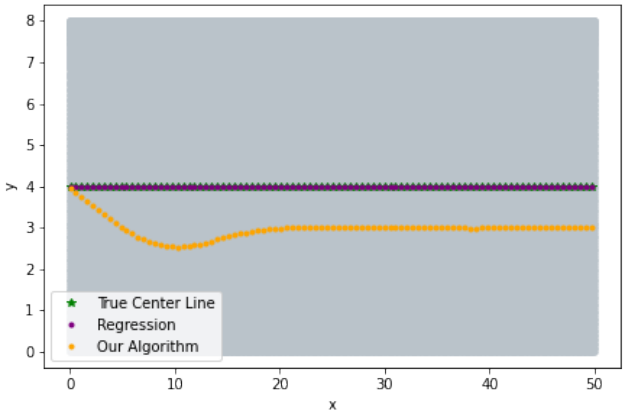

Figure 13: Comparison of algorithms on a fully straight tunnel.

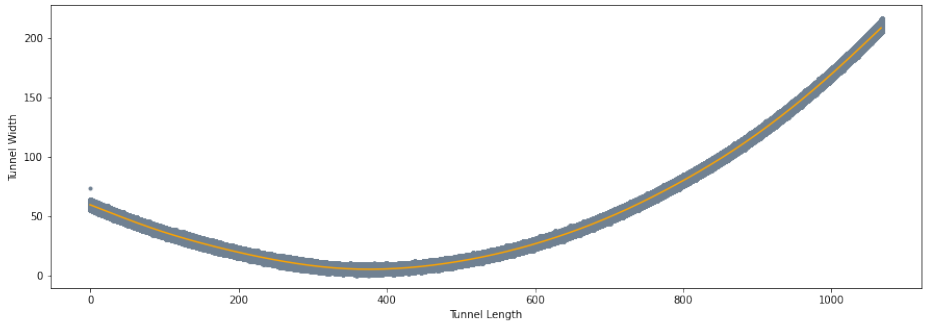

Figure 14: Visualisation of the original tunnel and the central line found by the algorithm in orange.

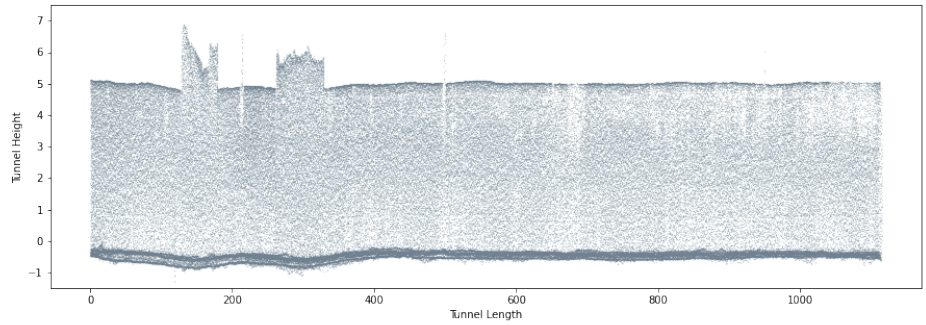

Figure 15: Visualisation of the aligned tunnel data set from x-y plane

new mobile mapping solutions which collect and process data to build accurate three-dimensional models and improve the accuracy, efficiency and safety of tunnel examinations.

References

[1] S. Gordon, D. Lichti, and M. Stewart. Application of a high-resolution, ground-based laser scanner for deformation measurements. In *Proceedings of 10th international FIG symposium on deformation measurements (pp. 23–32).*, 2001. `https://fig.net/resources/proceedings/2001/com6_orange_2001/pdf/Session%20I_Paper%204.pdf` (accessed 19/11/2022).

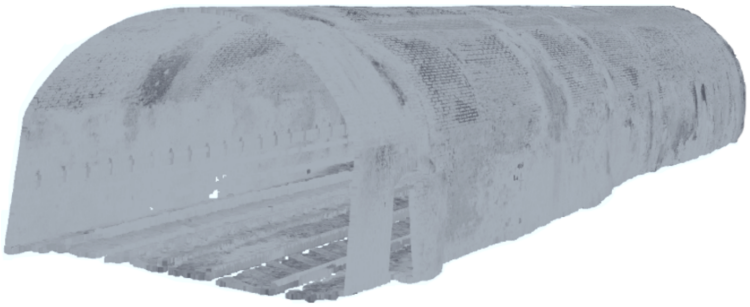

Figure 16: Visualisation of the aligned tunnel data set in 3D (made with Python's Open3D library[3]).

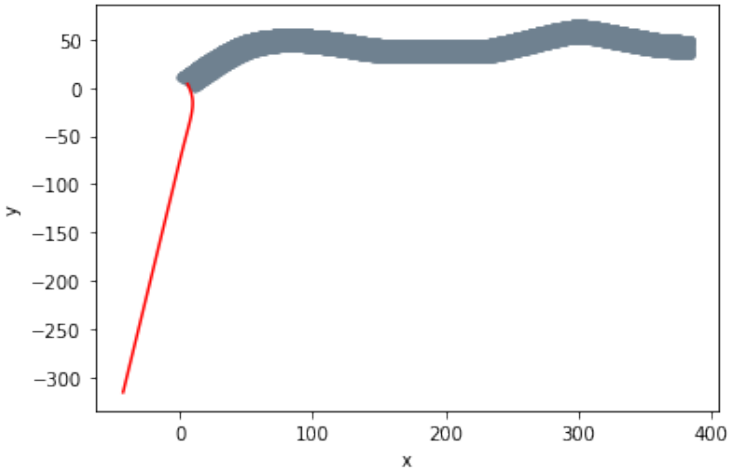

Figure 17: Examples of algorithm failure when inappropriate starting conditions are chosen.

[2] J Guo, X Zong, X Xie, L Wang, and J Zhai. Deformation monitoring of noncircular tunnels based on 3d laser scanning. *IOP Conference Series: Earth and Environmental Science*, 570:042003, 11 2020. https://doi.org/10.1088/1755-1315/570/4/042003.

[3] Yuriy Reshetyuk. Investigation and calibration of pulsed time-of-flight terrestrial laser scanners. *Royal Institute of Technology (KTH), Stockholm, Sweden.*, 2006. https://www.diva-portal.org/smash/get/diva2:10841/fulltext01.pdf (accessed 19/11/2022).

[4] Xiongyao Xie and Xiaozhi Lu. Development of a 3d modeling algorithm for tunnel deformation monitoring based on terrestrial laser scanning. *Underground Space*, 2(1):16–29, 2017. https://doi.org/10.1016/j.undsp.2017.02.001.

[5] Cheng Yi, Dening Lu, Qian Xie, Jinxuan Xu, and Jun Wang. Tunnel deformation inspection via global spatial axis extraction from 3d raw point cloud. *Sensors*, 20(23), 2020. https://doi.org/10.3390/s20236815.

[6] Wenting Zhang, Wenjie Qiu, Di Song, and Bin Xie. Automatic tunnel steel arches extraction algorithm based on 3d lidar point cloud. *Sensors*, 19(18), 2019. https://doi.org/10.3390/s19183972.