

OSLO METROPOLITAN UNIVERSITY  
 STORBYUNIVERSITETET

Master's Degree in  
**Structural Engineering and Building Technology**  
 Department of Civil Engineering and Energy Technology

# MASTER THESIS

THESIS TITLE  Automatic Reverse Modelling of Box Girder Bridge from Point Cloud Data	DATE  14.06.20
	NUMBER OF PAGES  162 (139)
AUTHOR(S)  Torjus Berg Knutsen Fridtjof Vaksdal	SUPERVISOR(S)  Daguang Han

IN COLLABORATION WITH  -	CONTACT PERSON  -
--------------------------------	-------------------------

**SUMMARY**

Building Information Modelling (BIM) has over the last decades brought the physical- and digital worlds closer together. BIM implementation is now expected in construction projects, ensuring interdisciplinary cooperation from concept to demolition. In this thesis, a workflow for a reverse BIM modelling method specialized for as-built box girder bridges is presented. The method takes advantage of state-of-the-art software including both numerical and visual programming, parametric design, metrology tools, 3D laser scanning and BIM software, to create scripts, algorithms and digital data in order to digitally reproduce a physical bridge in its as-built condition. The results show that the automatically generated reverse BIM model accurately resembles the geometry of the studied box girder bridge. The created reverse modelling algorithms were found to struggle with accurately modelling quickly changing geometry and areas with much occlusion in the point cloud data. In areas with consistent geometry and complete point cloud data, the reverse BIM model is a highly accurate representation of the analysed bridge structure.

3 KEYWORDS
BIM
Reverse Modelling
Point Cloud



# Problem Statement

Building Information Modelling, or BIM, is a relatively new concept within the AEC-industry currently experiencing a massive rise in use. BIM implementation has over the last decades developed from being experimental and untested to being an essential part of most modern construction projects. Although BIM is now the norm, the vast majority of structures across the world were designed in an era where BIM integration in projects were an outlier. As a consequence, these structures does not have digital models which can be used as a reference for assessment and adjustment of the structure at hand.

With the steady shift towards a technological society, many new technologies have emerged. Some of these being laser scanning and parametric design. The emergence of these technologies in combination with an ever-increasing BIM implementation, presents the possibility of reverse modelling physical structures to digital, as-built versions of the scanned structure. Currently, reverse modelling existing structures based on laser scanning is a topic of lacking research and implementation within the AEC-industry. As of now, the reverse modelling process is largely being done by hand, a tedious approach with a substantial risk of human error. By automating this process, reverse modelling can be done more efficiently while also decreasing the risk of human error. This not only reduces the cost associated to reverse modelling, but can also increase the accuracy of the finished digital model. If sufficiently accurate, the generated model can be a viable source for, but not limited to, construction documentation, building information and structural inspection.

In this thesis, a process of reverse modelling a physical structure to a digital 3D model based on acquired scan data is created and investigated. The thesis aims to use state-of-the-art software to accurately generate an as-built reverse model of a typical box girder bridge. In order to evaluate the reverse model, the accuracy of the generated model in relation to the acquired scan data is to be analysed, indicating whether or not the proposed modelling method is of sufficient quality.

# Abstract

Building Information Modelling (BIM) has over the last decades brought the physical- and digital worlds closer together. BIM implementation is now expected in construction projects, ensuring interdisciplinary cooperation from concept to demolition.

In this thesis, a workflow for a reverse BIM modelling method specialized for as-built box girder bridges is presented. The method takes advantage of state-of-the-art software including both numerical and visual programming, parametric design, metrology tools, 3D laser scanning and BIM software, to create scripts, algorithms and digital data in order to digitally reproduce a physical bridge in its as-built condition.

The results show that the automatically generated reverse BIM model accurately resembles the geometry of the studied box girder bridge. The created reverse modelling algorithms were found to struggle with accurately modelling quickly changing geometry and areas with much occlusion in the point cloud data. In areas with consistent geometry and complete point cloud data, the reverse BIM model is a highly accurate representation of the analysed bridge structure.

The work presented in this thesis provides an innovative method of generating as-built BIM models of high accuracy. If of sufficient accuracy, the generated reverse model can efficiently provide viable, digital construction information to be used in applications such as structural assessments and geometrical inspections. With minor adjustments, the proposed method is not only suitable for use in box girder bridges, but can also be adapted to work for similar structural shapes.

**Keywords** – BIM, Reverse Modelling, Point Cloud

---

# Preface

This master's thesis is the concluding work of two students at the Structural Engineering and Building Technology program at Oslo Metropolitan University, written through the eventful spring of 2020.

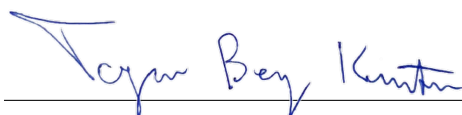
As part of an initially planned exchange program, the work is executed in collaboration with Chongqing Research Institute for Smart Cities and Sustainable Development, (SCSDA) and students at Chongqing Jiatong University, Chongqing, China. However, due to the outbreak of the infectious disease COVID-19, the exchange program was cancelled only days after we set foot in China, and the thesis was in full written in Norway. Although the large distance and vast time difference between the cooperating parties led to difficulties, we would like to extend a sincere thank you to everyone who has been supporting us in this period. Especially we would like to thank our supervisor Prof. Daguang Han, and the master degree students Guocheng Qin and Kaixin Hu. Their knowledge and guidance within this new and technical topic has been vital for the completion of our thesis.

We are glad we had the opportunity to get extensive knowledge of new and developing software and topics that will characterize the AEC-industry in the years to come. Reverse modelling, parametric design and 3D laser scanning are definitely areas that will have a future in the industry, and we are glad to help developing these with our research.



---

Fridtjof Vaksdal



---

Torjus Berg Knutsen

# Contents

<b>Problem Statement</b>	<b>i</b>
<b>Abstract</b>	<b>ii</b>
<b>Preface</b>	<b>iii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Research Background and Significance . . . . .	4
1.2 Research Status Worldwide and in Norway . . . . .	7
1.2.1 Research Status Worldwide . . . . .	7
1.2.1.1 BIM . . . . .	7
1.2.1.2 3D Laser Scanning . . . . .	10
1.2.1.3 Point Cloud Reverse BIM Modelling . . . . .	12
1.2.2 Research Status in Norway . . . . .	14
1.2.2.1 BIM . . . . .	14
1.2.2.2 3D Laser Scanning . . . . .	16
1.2.2.3 Point Cloud Reverse BIM Modelling . . . . .	18
1.3 Research Content and Technical Route . . . . .	19
1.3.1 Research Content . . . . .	20
1.3.2 Technical Route . . . . .	22
1.4 Applied Tools . . . . .	24
1.4.1 Hardware . . . . .	24
1.4.2 Software . . . . .	24
<b>2 Bridge Point Cloud Data Collection Technology</b>	<b>27</b>
2.1 3D Laser Scanner . . . . .	28
2.1.1 Type Introduction of Scanning Instrument . . . . .	28
2.1.2 Working Principle of Point Cloud Acquisition . . . . .	29
2.2 Measurement Accuracy Analysis of 3D Scanner . . . . .	30
2.2.1 3D Scanner Reliability . . . . .	31
2.2.2 Data Volume Analysis of Point Cloud . . . . .	32
2.3 Common Point Cloud Collection Methods . . . . .	34
<b>3 Collection and Processing of Bridge Point Cloud Data</b>	<b>36</b>
3.1 Bridge Point Cloud Data Collection . . . . .	36
3.1.1 Introduction of Scanning Instrument . . . . .	36
3.1.2 Scanning Scheme Design . . . . .	37
3.1.3 Bridge Point Cloud Scanning . . . . .	39
3.2 Pretreatment of Bridge Point Cloud . . . . .	41
3.2.1 Introduction to Point Cloud Preprocessing . . . . .	41
3.2.2 Bridge Point Cloud Registration . . . . .	43
3.2.3 Noise Reduction of Bridge Point Cloud . . . . .	44
3.3 Structural Segmentation of Bridge Point Cloud . . . . .	46
3.3.1 Introduction of Bridge Point Cloud Segmentation Software . . . . .	46
3.3.2 Superstructure Segmentation . . . . .	46
3.3.3 Substructure Segmentation . . . . .	47

---

<b>4</b>	<b>Feature Extraction of Bridge Point Cloud Data</b>	<b>48</b>
4.1	Feature Extraction of Box Girder . . . . .	48
4.1.1	Point Cloud Segmentation . . . . .	48
4.1.2	Feature Fitting of Point Cloud Segments . . . . .	50
4.1.3	Characteristic Parameters of Box Girder . . . . .	60
4.2	Feature Extraction of Pier Column . . . . .	63
4.2.1	Point Cloud Segmentation . . . . .	63
4.2.2	Feature Fitting of Point Cloud Segments . . . . .	64
4.2.2.1	Alternative Cutting Method . . . . .	72
4.2.2.2	Pier Point Cloud Sampling . . . . .	76
4.2.3	Characteristic Parameters of Pier Columns . . . . .	77
4.3	Feature Extraction of Pier Foundation . . . . .	80
4.3.1	Feature Fitting of Point Cloud Segment . . . . .	80
4.3.2	Characteristic Parameters of Pier Foundation . . . . .	83
<b>5</b>	<b>Reverse BIM Modelling of Bridge Point Cloud Data</b>	<b>84</b>
5.1	Introduction to BIM . . . . .	84
5.1.1	A Brief History of BIM . . . . .	86
5.2	Introduction to BIM Modelling Software . . . . .	88
5.2.1	Revit + Dynamo . . . . .	88
5.3	Reverse BIM Modelling . . . . .	90
5.3.1	Reverse BIM Model - Box Girder . . . . .	92
5.3.1.1	Main Part . . . . .	94
5.3.1.2	Extensions . . . . .	98
5.3.2	Reverse BIM Model - Pier Columns . . . . .	102
5.3.2.1	Main Part . . . . .	104
5.3.2.2	Extensions . . . . .	107
5.3.3	Reverse BIM Model - Pier Foundation . . . . .	111
5.3.4	Reverse BIM Model - Terrain . . . . .	115
5.3.4.1	Terrain Point Cloud Processing . . . . .	115
5.3.4.2	Reverse Modelling of Terrain Model . . . . .	117
<b>6</b>	<b>Accuracy Analysis of Reverse BIM Model</b>	<b>119</b>
6.1	Visual Comparison . . . . .	119
6.1.1	Box Girder . . . . .	120
6.1.2	Pier Column . . . . .	122
6.1.3	Pier Foundation . . . . .	127
6.1.4	Recurring Accuracy Errors . . . . .	128
6.2	Deviation Analysis . . . . .	131
6.2.1	Box Girder . . . . .	134
6.2.2	Pier Column . . . . .	141
6.2.3	Pier Foundation . . . . .	151
6.3	Summary of Accuracy Analysis . . . . .	154
<b>7</b>	<b>Conclusion and Prospect</b>	<b>155</b>
	<b>References</b>	<b>157</b>
	<b>Appendices</b>	<b>163</b>

# List of Figures

1.1	Difference in bridge type, (a) Monumental bridge, (b) unknown small bridge	1
1.2	Reverse model based on point cloud data [1]	3
1.3	Percentage of structurally deficient bridges in the U.S., U.K. and Norway [2], [3], [4]	5
1.4	Improved interoperability with openBIM	8
1.5	BIM adoption over time in the U.K. [5]	9
1.6	Laser scanning concept [6]	11
1.7	Digital Twin [7]	17
1.8	Point cloud of an oil platform [8]	18
1.9	Work productivity, relative growth 1992-2012 (SSB) [9]	19
1.10	Location of Chuxiong City, China	20
1.11	Pictures of the Dade bridge, Chuxiong City, Yunnan, China.	21
1.12	Section organisation	22
1.13	Technical route of the thesis	23
2.1	Example of placement of shared targets	30
2.2	Difference in point cloud density (low density: left, high density: right)	33
3.1	3D point cloud collection workflow	38
3.2	Placement of 3D laser scanners	38
3.3	Complete Point cloud	39
3.4	Scanning pictures	40
3.5	(a): A two-dimensional point cloud which has been divided using binary space partitioning. (b): The corresponding k-d tree [10]	42
3.6	Iterative Closest Point concept illustration	44
3.7	Cleaned point cloud	45
3.8	Explanation of box girder structure [11]	47
3.9	Highlighted superstructure of the Dade Bridge	47
3.10	Highlighted substructure of the Dade Bridge	47
4.1	How the box girder is sliced into 18 segments	49
4.2	Missing part at the one side of the box girder	50
4.3	Cross section of the Dade Bridge box girder with 9 intersection points	51
4.4	Concept of partitioning using thin horizontal slices	54
4.5	Illustration of flattening method and regression lines to find intersection points	57
4.6	Finding the missing intersection points	59
4.7	How the six pier columns are named	63
4.8	Geometry of the Dade Bridge pier column	64
4.9	Cross section of a Dade Bridge pier column with 4 intersection points	65
4.10	General concept of dividing the sides into separate parts by finding the length from the centre and out	68
4.11	Illustration of flattening method and regression lines to find intersection points	70
4.12	Graphical representation of changes in number of points for pier column 2	73
4.13	Development in number of points for Pier 2	74
4.14	Change in number of points for Pier 2	74
4.15	Figure of where the damages on the scan are located on pier column 6	75
4.16	The curvature of the change in the number of points for pier column 6	75



---

4.17	The point cloud of the pier foundation . . . . .	80
4.18	Least-squares method, fitted circle of the pier foundation point cloud . .	81
5.1	Dimension lines and simple force distribution of an arch bridge with varied support setups, generated and visualised in Sketchpad [12] . . . . .	86
5.2	Modelling process of a simple parametric cylindrical column in Dynamo and Revit . . . . .	89
5.3	Reverse modelling workflow . . . . .	91
5.4	The finished reverse model of the complete Dade Bridge in Revit . . . . .	92
5.5	The point cloud of the Dade Bridge with its box girder outlined in red .	93
5.6	Screengrab from Revit of the a) extensions, b) box girder segments and c) combined reverse model of the complete box girder . . . . .	93
5.7	Simplified conceptual algorithm for the box girder segments . . . . .	94
5.8	Regrouped intersection points of box girder 9, from Dynamo . . . . .	95
5.9	Closed polygons of box girder 9, from Dynamo . . . . .	96
5.10	The solid of the reverse model of box girder 9, visualised in Dynamo (left) and exported to Revit (right) . . . . .	96
5.11	Interval point distribution at intersection point line 5, from Dynamo . . .	97
5.12	Simplified conceptual algorithm for the box girder extensions . . . . .	98
5.13	Highlighted extension between box girder 8 and 9, visualised in Dynamo	99
5.14	Modelling process for extension between box girder 8 and 9, visualised in Dynamo . . . . .	100
5.15	Highlighted extension at start of box girder 1 . . . . .	101
5.16	Modelling process for extension before box girder 1 and after box girder 18	102
5.17	The solid of the finished extension model at start of box girder 1, visualised in Dynamo (left) and Revit (right) . . . . .	102
5.18	The point cloud of the Dade Bridge with its pier columns outlined in red	103
5.19	Screengrab from Revit of the a) extensions, b) cut pier columns and c) combined reverse model of the complete pier columns . . . . .	103
5.20	Simplified conceptual algorithm for the main part of the pier columns . .	104
5.21	Regrouped intersection points of pier 2, from Dynamo . . . . .	105
5.22	Reverse modelling process for the main part of pier 2 . . . . .	106
5.23	Simplified conceptual algorithm for the pier column extensions . . . . .	108
5.24	Modelling process for top and bottom extension for pier 2 . . . . .	110
5.25	The solid of the finished extension models of pier 2, visualised in Dynamo (left) and Revit (right) . . . . .	111
5.26	The point cloud of the Dade Bridge with its pier foundation outlined in red	112
5.27	Simplified conceptual algorithm for the pier foundation reverse modelling	113
5.28	Reverse modelling process for the pier foundation . . . . .	114
5.29	Point cloud of terrain model . . . . .	116
5.30	Supplemented terrain model . . . . .	116
5.31	Sampled terrain model . . . . .	117
5.32	Terrain model, contour lines at every meter . . . . .	118
6.1	Point cloud and BIM model comparison . . . . .	119
6.2	Comparison of box girders . . . . .	120
6.3	Visual comparison of box girder 9 . . . . .	120
6.4	Comparison of missing part in box 10-18 . . . . .	121
6.5	Comparison of the piers . . . . .	122
6.6	Visual comparison of pier column 2 . . . . .	123

6.7	Close-ups of how accurate the BIM model can be . . . . .	124
6.8	Visual comparison of pier column 3 and 4, short-side . . . . .	125
6.9	Visual comparison of pier column 3 and 4, wide-side . . . . .	126
6.10	Visual comparison of mid-span foundation . . . . .	127
6.11	Deviation in rounded corners . . . . .	129
6.12	Deviation at geometrical changes . . . . .	130
6.13	A noisy point cloud affecting the comparison accuracy of the pier foundation	131
6.14	Deviation map of the complete bridge . . . . .	134
6.15	Deviation map of the box girders . . . . .	135
6.16	Deviation map of box girder segment 5 . . . . .	137
6.17	Histogram of the variation in deviation for box girder segment 5, with standard deviation indicators . . . . .	138
6.18	2D analysis of box girder segment 5 . . . . .	140
6.19	Deviation map of the pier columns . . . . .	141
6.20	Deviation map of pier column 2 (left) and 5 (right) . . . . .	143
6.21	Histogram of the variation in deviation for pier column 2, with standard deviation indicators . . . . .	144
6.22	Position of 2D analysis for Pier 2 . . . . .	145
6.23	2D analysis of Pier 2 . . . . .	146
6.24	Deviation map of pier column 4 . . . . .	147
6.25	Histogram of the variation in deviation for pier column 4, with standard deviation indicators . . . . .	148
6.26	Position of 2D analysis for Pier 4 . . . . .	149
6.27	2D analysis of pier column 4 . . . . .	150
6.28	Deviation map of pier foundation . . . . .	151
6.29	Histogram of the variation in deviation for the pier foundation, with standard deviation indicators . . . . .	152
6.30	2D analysis of the pier foundation . . . . .	153

## List of Tables

1.1	Dade Bridge, Yunnan China, Bridge geometry . . . . .	21
3.1	Technical parameters of four scanning modes of Leica ScanStation P50 [13]	37
3.2	Amount of points at different stages of the noise reduction process . . . .	45
4.1	Typical cross-section of box girder bridges . . . . .	51
4.2	Table of length and number of slices for every box segment ((x)=after cutting)	61
4.3	Characteristic parameters (x-, y-, z- values [m]) of box girder 9 . . . . .	62
4.4	Geometry of the pier columns which the script is based on . . . . .	65
4.5	Table showing how the downsampling of pier column 1 affect the point cloud and its feature fitting . . . . .	77
4.6	Pier overview . . . . .	78
4.7	Characteristic parameters (x-, y-, z- values [m]) of pier column 2 . . . . .	79
4.8	Characteristic parameters [m] of pier foundation . . . . .	83
6.1	Overall structural tolerance [14] . . . . .	132
6.2	Explanation of statistics [15] . . . . .	133
6.3	Deviation statics of the complete bridge . . . . .	134
6.4	Deviation statics of the box girders . . . . .	135
6.5	Deviation statistics of box girder segment 5 . . . . .	137
6.6	Deviation statics of the pier columns . . . . .	142
6.7	Deviation statistics of pier column 2 and 5 . . . . .	143
6.8	Deviation statistics of pier column 4 . . . . .	148
6.9	Deviation statistics of pier foundation . . . . .	151

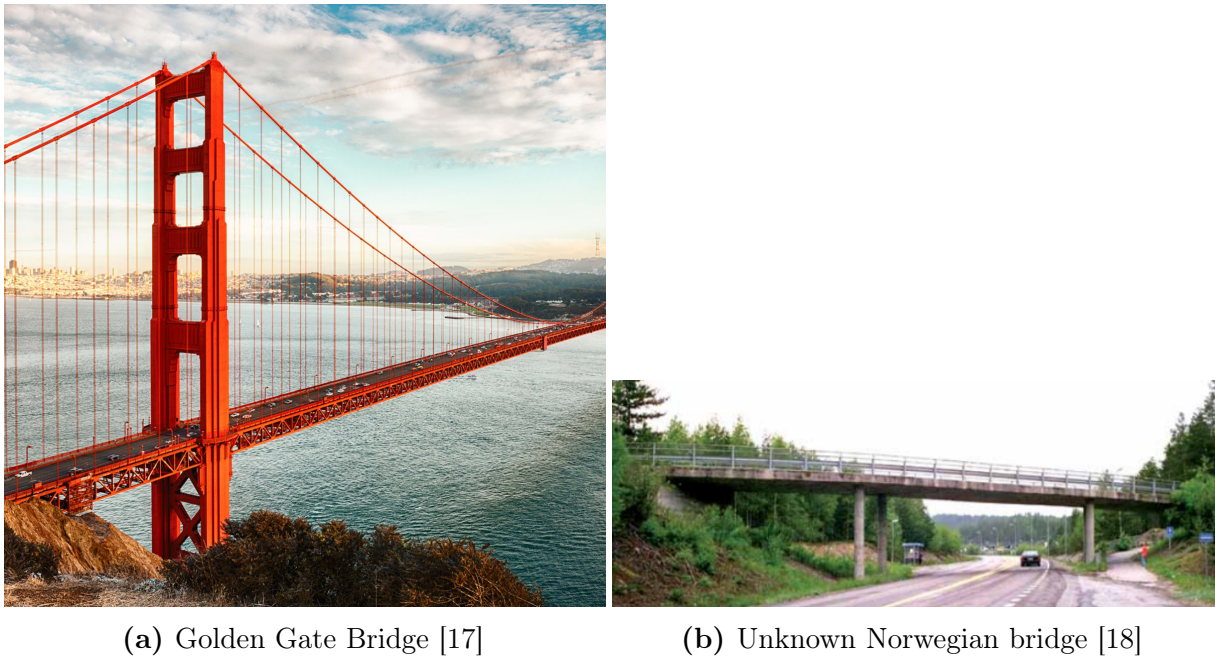
## List of Words

<b>Abbreviation</b>	<b>Definition</b>
ACSE	American Society of Civil Engineers
AEC	Architecture, Engineering and Construction
BIM	Building Information Modeling
CAD	Computer-Aided Design
CEN	European Committee for Standardization
DTM	Digital Terrain Model
ICP	Iterative Closest Point
IDM	Information Delivery manual
IFC	Industry Foundation Classes
IFD	International Framework for Dictionaries
LADAR	Laser Detection and Ranging
MEP	Mechanical, Electrical and Plumbing
NPRA	Norwegian Public Roads Administration
PCD	Point Cloud Data
TLS	Terrestrial Laser Scanning

# 1 Introduction

Since ancient times, infrastructure has played an important role in the economic growth of countries. The quality of its infrastructure is not only crucial for faster economic growth, but also to ensure inclusive growth. With inclusive growth, the benefits which stem from the economic growth is distributed across whole societies and creates opportunities for entire populations. A well-developed infrastructure links society together and in that way contribute to expanding the economy on a countrywide level [16].

Bridges are critical components in a nation's infrastructure [2]. There exist a huge variety of bridge types, all of them with the same task, creating a span over an obstacle. Undoubtedly, some bridges are iconic, monumental, signature structures known all over the world. However, the vast majority of bridges are smaller, unknown bridges simply serving its purpose, as seen in Figure 1.1.



**Figure 1.1:** Difference in bridge type, (a) Monumental bridge, (b) unknown small bridge

To preserve the road network and maintain the inclusive growth associated with high-quality infrastructure, thus also improving the economy of entire nations, it is important to extend the service life of existing bridges. Extending the lifetime of bridges is not only environmentally friendly as it prevents material waste, it also prevents major traffic congestion as roads experience less down-time. Due to the sheer amount of bridges

throughout the world, it is challenging to maintain and monitor them all, making it necessary to keep the focus on a selected number of bridges of critical importance, while smaller bridges are often neglected. This is especially the case for monumental bridges and bridges of historical heritage. Such bridges are often well maintained at the expense of smaller, ordinary bridges in remote areas. It is understandable that bridges of great importance which directly affects a large number of people are prioritised. However, by improving the accuracy and efficiency of bridge inspections, the amount of bridges which can be monitored and maintained increases, making it possible to increase the amount of bridges being kept in a structurally safe condition.

With the high number of bridges that already exist, there is a pressing need to lower the cost and effort required for modelling existing bridges. Reports from several countries show that a large number of their bridges are classified as structurally deficient and are missing required inspections [2, 3, 19]. To streamline the comprehensive inspection and maintenance work, there is a huge market demand for new technology that can efficiently boost bridge management productivity.

Many parameters of a bridge structure must be periodically inspected to determine its condition. Cracking, delamination, water infiltration, vehicular impact, biological attack, corrosion of reinforcement, spalling, blocked expansion joints, vandalism and/or geometrical deviation are all common conditions that decrease a bridges lifetime [20]. For most of them, geometry plays an essential part in the detection of damages. As a consequence, modern inspection techniques capable of quickly detecting geometrical deviation with high accuracy would be very useful.

Within the last couple of decades, the AEC-industry has experienced tremendous digital development, resulting in increased collaboration, shared knowledge and information exchange. Much because of the introduction of Building Information Modelling (BIM). A building information model, further referred to as a BIM model in this thesis, is in short a digital 3D-model with integrated information about the modelled structure. The growth of BIM implementation has increased a lot in recent years and is expected to keep growing in the coming years [21].

However, BIM is mainly used for as-designed models of new structures and not as-built models of existing structures, the issue being that as-designed models often do not

accurately portray the condition of the finished structure. As-built models can be used for many applications within the AEC-industry such as a reference for geometrical deviation inspection, Laser-scanning technology has changed this, making it possible to accurately collect large amounts of data of existing structures in order to model as-built. The collected data can be converted to a 3D point cloud data (PCD), which in turn can be used to create reverse 3D models representing the as-built condition of the scanned, physical structure. This concept, as shown in Figure 1.2, is currently being done manually, with the designer trying as best as possible to draw a 3D model by hand which fit the PCD, like a 3D blueprint.



**Figure 1.2:** Reverse model based on point cloud data [1]

Automating this process of reverse modelling will lower the cost and effort of creating as-built models of existing bridges. As-built BIM models of a bridge will ease the bridge detection process, while also reducing the possibility of human modelling error. Reverse modelling will have a significant impact on how maintenance and rehabilitation are planned and implemented, which can reduce the risk of structural failure of bridges on a global scale.

## 1.1 Research Background and Significance

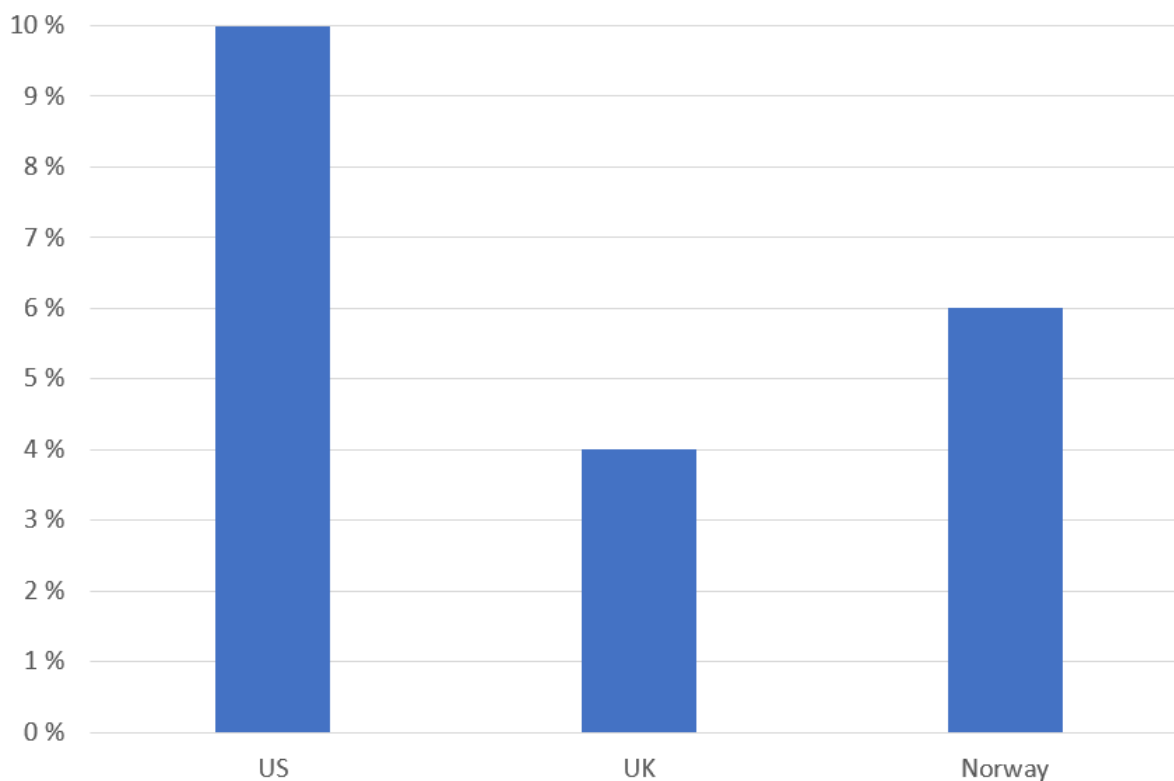
The U.S. has more than 600.000 highway bridges. According to the infrastructure report card from America's Society of Civil Engineers (ASCE), almost 10% of all bridges in the U.S. were structurally deficient in 2016. With four out of ten bridges being over 50 years old and a backlog of rehabilitating project estimated to \$123 Billion, there is a pressing need for new technologies and materials which can help engineers to both build, and improve the maintenance on bridges [2].

In the U.K. 4% of the approximately 88.000 bridges on the road network were considered substandard, i.e. not satisfying the set requirements, by the RAC Foundation. It is apparent that there also in the U.K. is a need to improve the standard and inspection method [3].

With a 2-year inspection cycle, as recommended in the Design Manual for Roads and Bridges used in the U.K. [22], there is a need for over 400.000 bridge inspections each decade only in the U.K.. The United States and the United Kingdom combined need to perform at least 340.000 inspections every year.

In Norway, much of the same problems exist. The Norwegian Public Roads Administration (NPRA) manages approximately 17 500 bridges and ferry docks all over the country, which has to be structurally safe [23]. In 2017, VG (a Norwegian newspaper) published an in-depth article about how the NPRA was violating the rules set in place to assure that the bridges of Norway are structurally safe for road users [4]. The conclusion was that 6% of the bridges in Norway were structurally deficient. In addition to this, in 2016 and 2017, the Norwegian Road Supervisory Authority conducted supervision of how bridge inspections were carried out in two Norwegian regions. Both of the reports revealed that many inspections were not planned and carried out in accordance to the set inspection requirements and that the faults found during the inspections were, in many cases, not followed up and corrected.





**Figure 1.3:** Percentage of structurally deficient bridges in the U.S., U.K. and Norway [2], [3], [4]

NPRA have developed manuals as a supplement to Eurocodes. These manuals contain supplementary provisions for calculation and design of bridges, ferry docks and other load-bearing structures. The N400 manual contains requirements for reliability and capacity, durability, traffic safety, accessibility and maintenance [24]. To ensure the condition of a bridge, inspections are carried out according to the bridge inspection manual V441 [18]. An inspection shall form the basis for operation and maintenance, which provides a safe and efficient road system with safety and good regularity for road users. These inspections form the basis for the preparation of preventive and corrective measures on the bridge body mass. An inspection plan must be available according to the needs and condition of the objects. As a standard, a bridge shall have a simple inspection every year, detecting visual defects that can have consequences for the bridge capacity, road safety, maintenance or affect the environment. In addition, a more thorough inspection shall be carried out every 5th year [18]. Although superficial, the time it takes to conduct annual inspections of all existing bridges quickly accumulates. Developing more time-efficient methods for inspections can have a huge impact on the AEC-industry in terms of both economic cost

being reduced due to time-efficiency, as well as increasing the amount of structurally safe bridges due to an increased amount of inspections.

To manage these constructions, a proprietary bridge management system called BRUTUS is developed. BRUTUS is NPRA's information and planning tool for the management, operation and maintenance of bridges and other road network structures in Norway [25].

In the aforementioned article from VG, access to BRUTUS showed that the yearly inspection was violated for over half of the Norwegian bridges. In Oslo, the capital of Norway, 96% of the bridges missed either the annual inspection, the main inspection, or both [4].

With this knowledge of the bridge status in the U.S., the U.K., Norway, and presumably many other countries across the world, it is apparent that an extensive upgrade to the current inspection practice with less labour-intensive documentation techniques is needed. Norway spent 100 million NOK on bridge administration in 2017 alone [19]. By streamlining and digitising the bridge administration work, it is possible to significantly reduce its cost, but also develop more accurate, extensive and secure inspection methods. This minimises the risk of human error due to subjective judgement, observational skills and personal experience.

One of the stated objectives of NPRA is to ensure effective development, use of new technology and digitisation of services and processes [26]. The work in this thesis applies modern, digital technology to improve current processes which limits the effectiveness of current inspection methods, falling well within the objectives set by NPRA.

## 1.2 Research Status Worldwide and in Norway

In this section, an overall overview of the current practice within BIM, 3D laser scanning and reverse modelling from point clouds across the world, as well as in Norway, is presented. Relevant standards, organisations and research documents are collected and discussed in relation to its connected research topic, with the aim of detailing the possibilities these topics can bring forth, and elements of potential improvements.

### 1.2.1 Research Status Worldwide

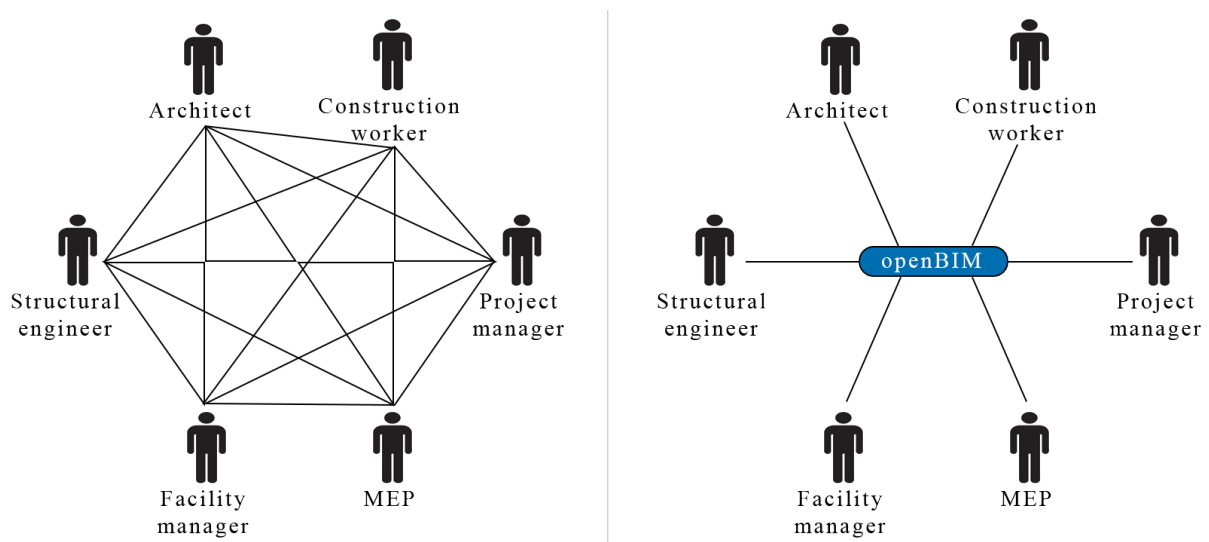
As the research presented in this master thesis is globally relevant, the worldwide research status on BIM, 3D laser scanning and reverse modelling based on point clouds are presented below focusing on the U.S., the U.K., and Asia as a whole.

#### 1.2.1.1 BIM

All around the world, massive urbanisation is happening. Modern concepts such as smart cities and digital twins are gaining momentum, and BIM gets more and more attention. The implementation of BIM in the AEC-industry is steadily increasing on a global scale. Collaboration, standardisation and governmental engagement are playing a big part in how each country is shifting towards a digitised industry with a common BIM language. The European Committee for Standardization (CEN), is an organisation officially recognised by the European Union which is "responsible for developing and defining voluntary standards at European level." [27]. CEN brings together the national standardisation bodies of 34 European countries.

Because of the rapid development and implementation of BIM in the European AEC-industry, CEN has started a program designed to develop new standards to support BIM processes. CEN also established a BIM-committee named "CEN/TC 442 - Building Information Modelling (BIM)" responsible for the standardisation of BIM processes in Europe. The scope of the committee is "standardisation in the field of structured semantic lifecycle information for the built environment. The committee will develop a structured set of standards, specifications and reports which specify methodologies to define, describe, exchange, monitor, record and securely handle asset data, semantics and processes with

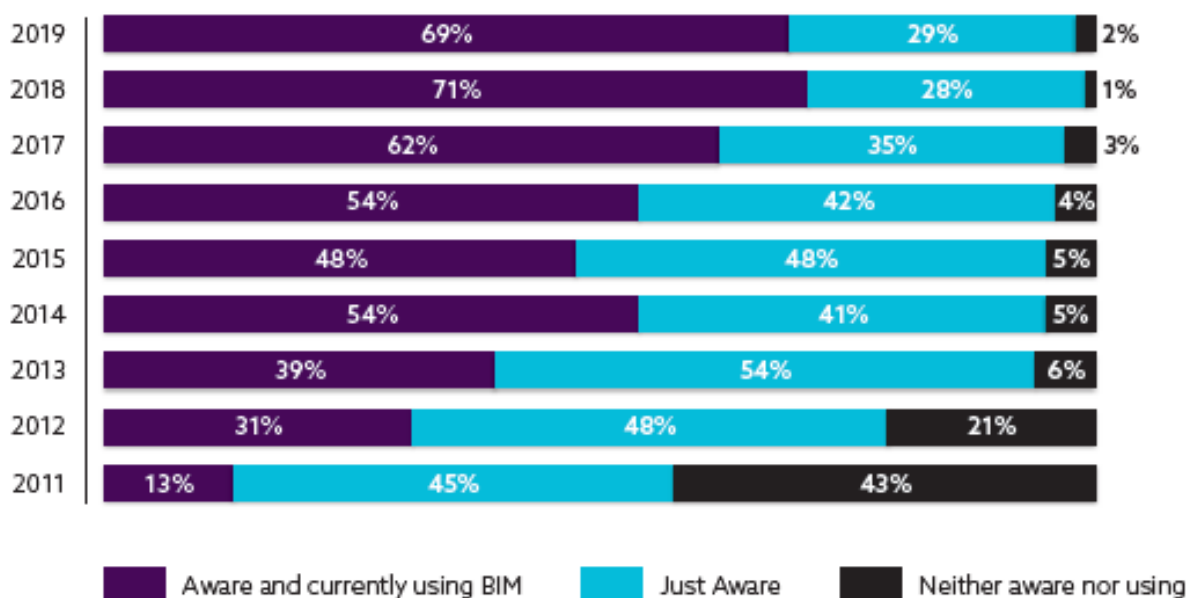
links to geospatial and other external data." [28]. Implementing openBIM, which in short, is a collaborative process within the AEC-industry which aims to promote interoperability between fields by taking use of open standards, workflows and file formats to openly share information with all participants in a project at any given time (read more about openBIM in [29]), as well as adopting the ISO-standards to European standards are important steps in their work. Amongst the more relevant standards for this thesis work are the three buildingSMART- standards IFC [30], IFD [31] and IDM [32], as well as the information management standard ISO 19650 [33, 34]. The increased collaboration achievable through openBIM is illustrated in Figure 1.4.



**Figure 1.4:** Improved interoperability with openBIM

ISO 19650 is a series of international standards, based on the U.K. 1192 series. The standards were finished in late 2018 and deal with information management using BIM.

The U.K. has one of the most extensive BIM strategies in the world. The BIM development is divided into four levels of maturity, from level 0 to 3, all with its objective, tasks and achievement indicators. The U.K. government now require that all projects funded by the government must be delivered with a fully collaborative 3D BIM. All government projects now require a level 2 compliant. According to the National Building Specification from the U.K., the overall trends of BIM awareness and adoption in the U.K. has increased from approximately 13% in 2011 to around 69% in 2019 [5].



**Figure 1.5:** BIM adoption over time in the U.K. [5]

The U.S. started using BIM early, already in the 1970s. But the implementation of BIM slowed down, to this date, there is no governmental mandate to use BIM. However, it does not mean that they do not use it. Most construction projects today utilise BIM, and some states have already put mandates in place. The U.S. has many different federal departments working on BIM standards. However, a lack of collaboration between the departments has led to confusion in the industry, having different methods with every new client and project. The lack of a standardised BIM method makes it difficult to implement BIM in the AEC-industry efficiently. Even so, the U.S. is gradually making its way back as a BIM pioneer, taking more responsibility in implementing BIM properly and having one of the world strongest technological communities.

In Asia, BIM has experienced a boom in the AEC-industry. Governments, associations, companies, and research institutes have seen the importance of BIM, with the application of BIM technology being promoted from the design to the construction phase.

Asia's largest country, China, recently started using BIM. Even though BIM is new, its development has been very fast, according to Bernstein et al. [35] one of fastest-growing BIM implementation in the world. The fast-growing BIM implementation reflects how the country economy increases. China's economy is among the fastest-growing in the world and is now the world's second-largest economy.

With the aforementioned economic boost and simultaneous technological investment, many Chinese workers are going from farming to industrial and technological jobs. Such jobs are mostly located in the cities, leading to increased urbanisation. Hundreds of millions of people are moving into the rapidly-growing megacities. To accommodate this enormous migration, the country is investing massively in upgrading its infrastructure and are hugely benefiting from the application of BIM.

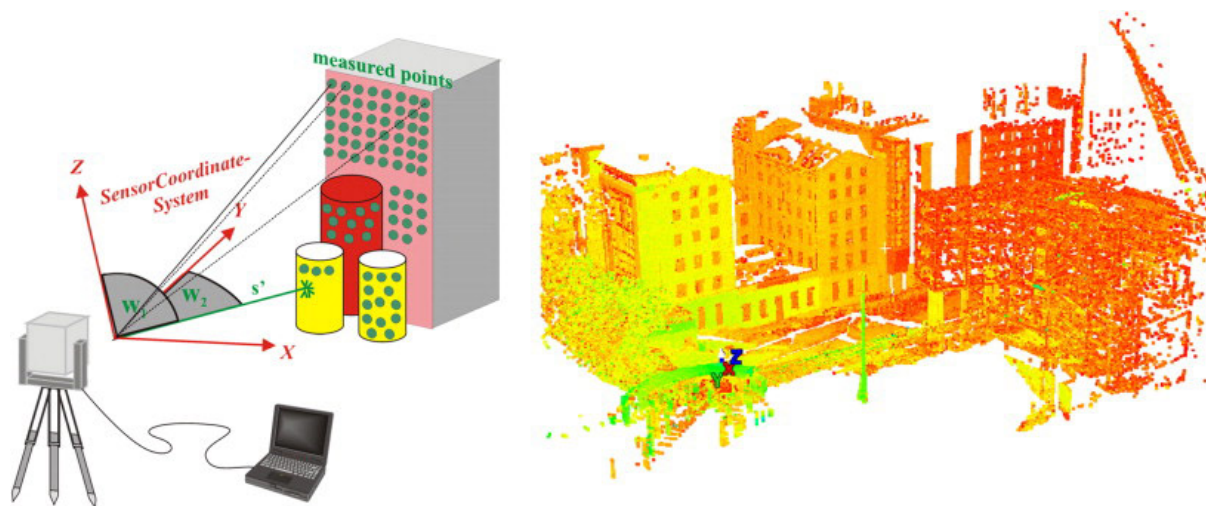
Now, China has set a national mandatory requirement with more than 90% use of BIM in major projects by 2020. Furthermore, more than 10 BIM standards are published in the last five years.

Overall, it is apparent that BIM is a subject of rapid development and research across the world. Especially in recent years, as the AEC-industry and governments alike have been recognising the opportunities which arise from the implementation of BIM.

#### **1.2.1.2 3D Laser Scanning**

The development of surveying and mapping technology is constantly being updated along with the continuous development in the field of science and technology. The methods have developed from using traditional flat maps to total stations and GPS, to the rapid development of high precision instruments in the field of 3D measurement, using tools like 3D laser scanners.

Due to technological advances, 3D terrestrial laser scanning, photogrammetry and other long-range, non-destructive imaging tools, has made it possible to perform architectural surveys with high accuracy and precision. 3D laser scanning technology is a technology which is used to obtain a 3D surface data of a target object, as illustrated in Figure 1.6. The 3D laser scanner can quickly obtain massive point clouds in a digital format, which is very convenient for post-processing in software.



**Figure 1.6:** Laser scanning concept [6]

3D laser scanning technology is a relatively new technology and is still in its early beginnings around the world. The value of the technology is demonstrated in the rapid growth of laser scanning hardware and supporting software. Laser scanning is considered to be the best available technology to capture 3D information of an object.

Even though laser scanning can have a significant impact on the industry, it has limiting factors. The instruments are very expensive, making it difficult for smaller companies to make use of the technology. The technology is still developing, application accuracy will continue to improve, and the integration between systems will gradually be strengthened. There are no technical standards developed on the subject, and data processing methods are quite time-consuming.

3D laser scanning has a huge potential in a wide variety of fields. It can be used for 3D reconstruction, numerical calculation (measurement, volume calculation, etc.), and deformation analysis. Some application fields are; Mining, surveying and mapping, structural surveying, reverse engineering, digital city, digital factory, cultural relic protection, virtual reality, medicine, entertainment, etc.

Within the AEC-industry, a promising field of research is automating the process of using PCD for structural evaluation. Terrestrial laser scanner used for developing processes and algorithms for: as-built modelling [36, 37, 38, 39], quality assessment of existing infrastructure and construction sites [40, 41, 42, 43], progress tracking [44, 45, 46, 47], and structural health monitoring [48, 49, 50, 12].

### 1.2.1.3 Point Cloud Reverse BIM Modelling

In the AEC-industry, there is a growing interest in how to integrate BIM for existing structures. Now that the industry is experiencing the benefits of BIM for new designs, they also want to implement as-built BIM. As-built BIM is used in a wide variety of ways, including documentation, maintenance and quality control. Common for most as-built BIM projects, is that they are based on 3D point clouds gathered from terrestrial- or mobile laser scanners.

A BIM model created during the design of a structure will most certainly vary significantly from the actual, as-built condition. Errors during construction, undocumented design changes or renovations will all lead to differences between the designed BIM model and the built structure.

As described in the previous section, growth of laser scanning hardware and software have had an exponential growth the last decade, and now a big part of development is focusing on the interface between the scanned data and the BIM model. With the use of 3D reconstruction, BIM tools can be used to incorporate the existing structure and make calculations of design alternatives, cost estimates, material quantification, data management, as-built documentation, constructive state analysis, execution plans and more [51]. Using 3D reconstruction tools, such as laser scanning, in combination with reverse modelling software, it is possible to significantly reduce tedious, repetitive modelling by automating the process.

In Europe, approximately 80% of all buildings were constructed before 1990. Effectively, these buildings do not have accessible BIM models, meaning that renovations, structural assessment, rehabilitation, city planning and similar, becomes challenging and inaccurate. Much of the same situation can be seen for old bridges which lack sufficient documentation. By using 3D reconstruction tools such as laser scanners which can be used for reverse modelling, as-built BIM models can be generated either manually, semi-automatically or automatically. This is an aspect of as-built BIM which is currently gaining in popularity and usage [52].

An as-built BIM enables a component level analysis and manipulation instead of changes at the individual point level of PCD. However, the current generation of as-built BIM



modelling methods are labour-intensive and prone to errors. The geometric modelling of the existing structures depends on dense spatial measurement, typically acquired by 3D laser scanners or photogrammetry. The procedure of converting a 3D point cloud to a 3D BIM model is referred to as Scan-to-BIM.

The creation of BIM models of existing structures is very limited due to the lack of research and experience within the AEC-industry to automatically create as-built BIM models from PCD. Thus, a realistic approach to introduce the concept of reverse modelling of physical structures to as-built BIM models within the AEC-industry is to gradually implement the technology starting from a simple, mostly manual method while working towards a fully automatic reverse modelling process. Either way, such a process often consist of three main consecutive steps:: Segmentation, classification and reconstruction.

Much research has been done on how to segment a geometric model. Segmentation can be divided into three parts: Geometry based segmentation (using local geometric features), topology-based segmentation (dealing with shapes, bodies, spaces, surfaces, curves) and interactive segmentation which is common in software. A software of such capabilities is Geomagic®Control X™(Geomagic), a 3D inspection and metrology software also used for reverse engineering, creating CAD models of 3D scans and more. The program has an auto segmentation tool for feature recognition [53]. Mejia et al. [54] propose an automatic hybrid method, using both the geometrical and topological approach. Qi et al. [55] uses local concavities as an indicator for part boundaries, and Arbeláez et al. [56] propose a method for detecting regions at an object.

For buildings, reconstruction of walls, floors, ceilings and openings (doors and windows) is the main target. Much research is done on this and the problems associated [57, 58]. [36] also use 3D point cloud as an input and automatically identifies and models floor, walls, ceilings and openings and produce a semantically rich 3D model. The approach does not rely on hand-coded rules to identify surfaces and is therefore not a subject to noisy measurement and cluttered environment. To avoid the problems associated with the aforementioned automatic segmentation methods used to reconstruct physical building elements to digital 3D models, while also maintaining better control of the segmentation process, manual segmentation with software such as Geomagic is an option.

## 1.2.2 Research Status in Norway

In the following sections about research status in Norway, a summary of reports and research done in the different fields; BIM, 3D laser point cloud and point cloud reverse modelling is presented. The section is also based on an interview done with Sigmund Reinsborg Log, Head of control and validation at NPRA. The statements are not documented but are considered by the authors as interesting points and considerations.

### 1.2.2.1 BIM

Norway is a small country, with a small AEC-industry in comparison to international powerhouses like the U.S., U.K. and China. On a national scale however, the AEC-industry is the largest mainland industry in Norway. Due to the cold and harsh environment of Norway, structures often have high requirements in terms of durability - resulting in projects and structures which are overall more complicated and expensive than what would be obtainable in more forgiving climates. A useful method to handle an increased project complication is to implement BIM in the projects. Norway has a tradition of standardisation which is an important part of streamlining the BIM process. Some advantages of standardisation is that it makes it easier to collaborate between fields, understand the work of others and re-visit old projects. A BIM concept highly integrated in the Norwegian AEC-industry is openBIM, as defined in Section 1.2.1.1. openBIM has contributed to placing Norway in a front-running position in the gradual change towards a digitised AEC-industry. Now, many Norwegian construction projects implement BIM requirements within the project contracts, due to the benefits which well-integrated BIM entail.

Norway is a pioneer in BIM. BIM is no longer anything new, but a well developed and valuable tool. It is being used in a lot of construction projects and is now a contract requirement for many builders and contractors.

A common digital roadmap for the AEC-industry shows an overall framework of how the industry must work to become a fully digital, competitive, sustainable and serious industry in 2025. In the roadmap, BIM plays a major part in reaching the goal to reduce the greenhouse gas emissions by 50%, have 50% faster project execution and a considerable reduction in cost. This goal is drawn from U.K.'s strategy "Construction 2025" [59].

Digital tools will be important in the future, but to make these systems work, the information given to the database is crucial. This information includes information about building objects and products. All this information is based on stored data. The data is stored in defined export formats. To improve interoperability and have information that is usable by several users and software in the industry, as shown in Figure 1.4, the data format should be open.

The construction industry is constantly moving closer to all information being stored in open formats. An example of an open file format which is commonly used in Norway is IFC (Industry Foundation Classes). Norwegian state actors, such as Statsbygg, is at the forefront of implementing this system. The IFC format provides information and properties of objects and is the preferred format when buildings and bridges are being modelled. For geographically dependent elements, such as roads and transport, GML (Geography Markup Language), another open format, is the most common. In addition to these two open formats, the Norwegian Mapping Authority uses a special Norwegian open format, .SOSI, a text format representing points, lines, areas, along with other geographical information.

BuildingSmart and Open GIS Consortium are two organisations responsible for respectively IFC and GML. Today, a cooperation between the two is lacking, and an open format usable for both buildings and roads are unlikely. However, the Norwegian Mapping Authority and the Norwegian Building Authority have taken up roles as project leaders in an international project with a goal to improve cooperation between the two organisations.

Statsbyggs position as Norway's largest builder, as well as being state-owned, gives it an important role to set the course for BIM-development and usage in Norway. Statsbygg is following a project model which implements BIM for all their projects. This model is being used in all phases of the projects to secure a uniform and proper project from the start to the end. The project model shall strengthen the planning, management and implementation of the project. One of the parts of doing this is Digibyg.

Digibyg is an executive project that aims at promoting the use of digitisation and smart technology in Statsbygg's business areas, throughout the whole value chain from idea and plan to operation. Digibyg is working on meeting the requirements and goals set by the Building Society's National Association (BNL), published in the "digital roadmap towards

2025” with cost reduction, reduced completion time and climate emissions as its main topics [60].

Statsbygg is working to change the AEC-industry by increasing the digitisation of the buildings entire lifecycle, from planning and design, during the construction phase, usage phase and up to the end-of-life. They are setting requirements that all the design must be done in the BIM-model and that the building site has to be paper-free.

The use of BIM is not new in Norway, but is highly dependant on the requirements from the customer and owner. Often, this results in projects being delivered with paper-based construction drawings made with traditional design methods, with a BIM model being attached as a supplementary delivery. A proper BIM-model has the ability to replace paper-based drawings, meaning that delivering both a digital BIM-model as well as physical drawings, is an inefficient design method both in terms of necessary work-hours as well as increased project complexity. This is what Statsbygg wants to improve by making the BIM- model the “master” information source. Paper drawings will be replaced with digital models. In the transition phase between paper and a paper-free construction site, the drawings should be made by the BIM-model and not trough individual, independent drawings. This way, drawings automatically updates in sync with project alterations.

In order to have a paper-free building site, the workers have to use digital tools such as smartphones, tablets and BIM-booths to see the digital model of what they shall build. Such a change may be difficult for many workers, whom are not experienced in digital tools. Transitioning from physical drawings to digital models will, therefore, require investments in not only the design process, but also in the workers on-site which are actually implementing the changes.

### **1.2.2.2 3D Laser Scanning**

In the digital roadmap of Norway, private and public commissioners have a particular responsibility for making demands on digital deliveries, where "digital construction site" and "digital twin" are important products [59].

To get a digital twin, or at the very least a digital mirror model, laser scanning plays an important part. The difference is that a digital twin is a virtual representation of a real-world object with real-world data updating in real-time, see Figure 1.7. In contrast,

a digital mirror model is merely a geometric reconstruction of the object which does not update according to its environment and state.



**Figure 1.7:** Digital Twin [7]

In Norway, the use of laser scanning is an emerging concept within the AEC-industry. Laser scanning has become a practical and accurate tool for measuring and perform control of structures. Some examples of its use are control of bridge constructions [61], measurement of homogeneity on asphalt pavements [62] and scanning of buildings [63].

Due to a widespread oil and gas industry, Norway has been far ahead with the use of scanning technologies. Laser Detection and Ranging (LADAR) is a common scanning method which is used in the oil and gas industry to map bedrock surfaces, analysing geometries and structures to find information on how oil reservoirs are constructed.

The technology has also been used to survey the danger of collapse of mountainsides [64] and the consequences of environmental changes, such as monitoring changes in glaciers over time [65].

In an ongoing project called "Nasjonal detaljert høydemodell", or "National detailed elevation model", the Norwegian Mapping Authority are using laser scanners mounted on planes or helicopters to gather detailed elevation data and terrain models of the whole country. This data is free for everyone to use, and the fact that an entire country can be scanned is an excellent example of how efficient laser scanning can be [66].

Just outside the capital of Norway, Oslo, a new indoor ski hall named "SNØ" was built with the help of laser scanning technology on a drone to get accurate mass calculations.

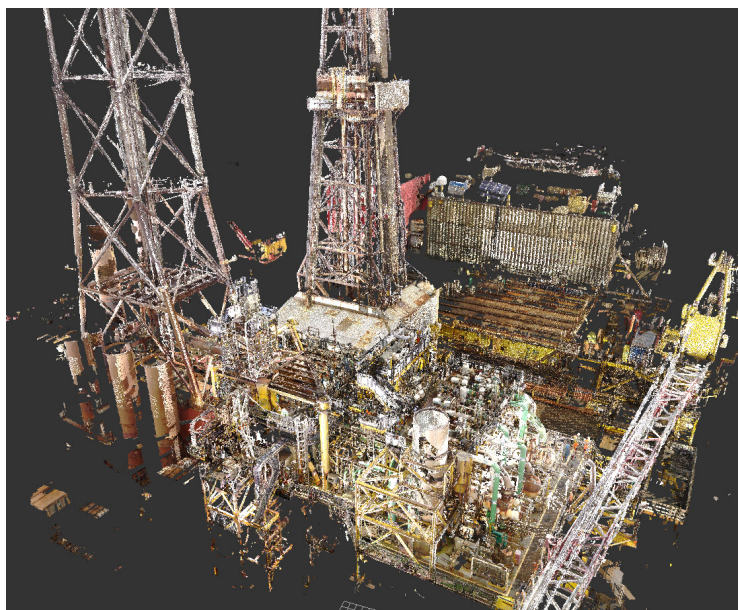
NPRA issued a report in 2019, where drones were used for bridge inspections [67]. In this report, although laser scanning was not used, the drones took images and videos. With

the use of such drones, it is possible to obtain more representative images that illustrate damage and/or the extent of needed repairs compared to inspection by people from the ground or when using any form of bridge inspection platform. Furthermore, using drones to access hazardous areas of a bridge increases safety on the work-site, and may even require less equipment and personnel than a standard bridge inspection. After this report, NPRA recognised the benefit of using drones to obtain a better representation of a bridge and started testing the use of also implementing 3D laser scanning with UAVs.

### 1.2.2.3 Point Cloud Reverse BIM Modelling

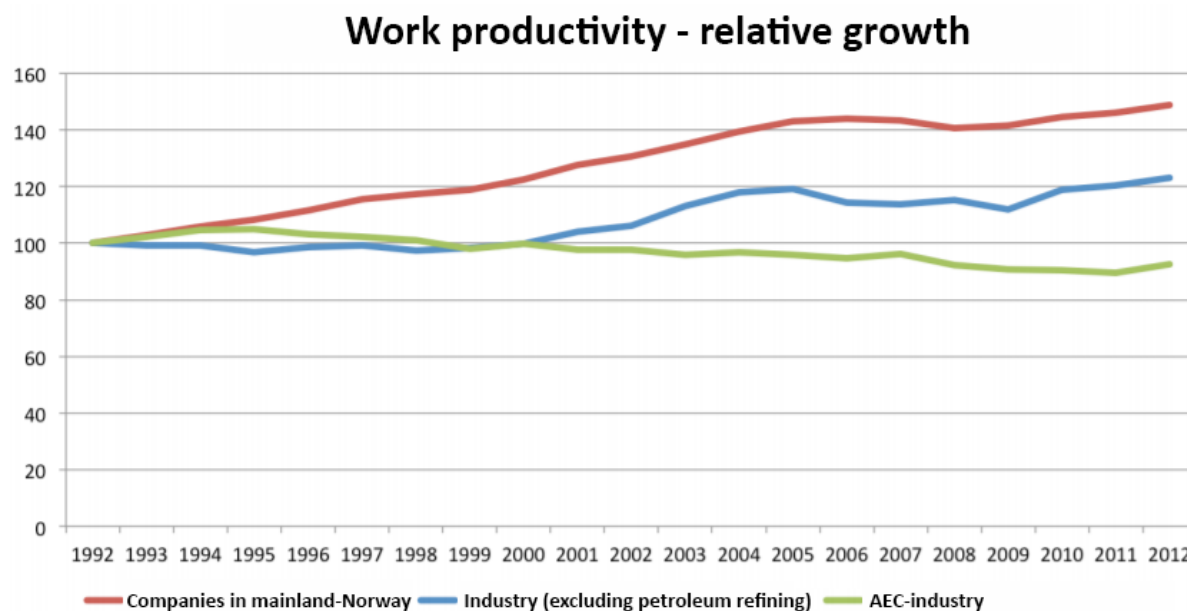
Reverse BIM modelling is being used more and more for rehabilitation work on buildings in Norway. However, the process commonly used is not an automatic process of recognising structural elements. The PCD is only used as a guideline when manually modelling the existing structure to a 3D BIM model. This can be a tedious process, and can in many cases be of insufficient accuracy.

The oil industry uses laser scanning and reverse modelling to update the model of offshore installations continually. The size and complexity of an oil platform combined with the distance to land and the safety of inspections mean that a fully updated model can be used by workers onshore. This dramatically simplifies work needed to be done at the platform. The oil and gas industry was early to implement 3D reconstruction technologies, which they are now benefiting from.



**Figure 1.8:** Point cloud of an oil platform [8]

The AEC-industry has not followed this trend and is lagging on the use as a result of this. While many industries have increased efficiency, the AEC-industry has decreased in efficiency over the last couple of decades. Statistics Norway (SSB) provides data, as seen in Figure 1.9, which show that over the last 20 years, the work productivity, i.e. the generated value in comparison to the resources invested, decreased between 1992 and 2012 [9]. A contributing factor to this negative development is the relatively slow implementation of new technologies in the AEC-industry in comparison to other fields. A consequence of the slow implementation of new technology in the Norwegian AEC-industry is that no bridges today have used a scan-to-BIM approach, and buildings have just started using PCD to create reverse BIM models.



**Figure 1.9:** Work productivity, relative growth 1992-2012 (SSB) [9]

### 1.3 Research Content and Technical Route

In this section, the research content of this master thesis is presented, with a subsequent technical route describing the methodology used in the thesis work. Firstly, the research content is presented together with information on the bridge, which is to be reverse modelled. Second, the technical route, methodology and reverse modelling workflow is presented. Lastly, the applied hardware and software tools used in the master thesis is presented in order of usage.

### 1.3.1 Research Content

With the steady increase in BIM implementation, laser scanning technology and 3D reconstruction in the AEC-industry, as detailed in Section 1.2, the approaches for object detection and recognition algorithms to process PCD are steadily increasing in quality.

As detailed in the problem statement, the aim of this thesis is to accurately reverse model a physical structure to a digital 3D model. The chosen structure is a box girder bridge called the "Dade Bridge".

The Dade Bridge is 15.000 ton concrete bridge at a railway and expressway conjuncture in Chuxiong City in the Yunnan Province in southwest China, shown in Figure 1.10. The bridge is a part of the Qinfeng-Heping section of an elevated expressway connecting Hangzhou in east China and Ruili in west, an airline distance of over 2.300 km. The bridge geometry is presented in table 1.1. Some pictures of the bridge can be seen in Figure 1.11



**Figure 1.10:** Location of Chuxiong City, China



**Table 1.1:** Dade Bridge, Yunnan China, Bridge geometry

Length	Height	Width
130m	19m	33,5m



(a)



(b)

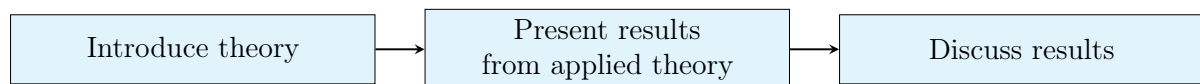


(c)

**Figure 1.11:** Pictures of the Dade bridge, Chuxiong City, Yunnan, China.

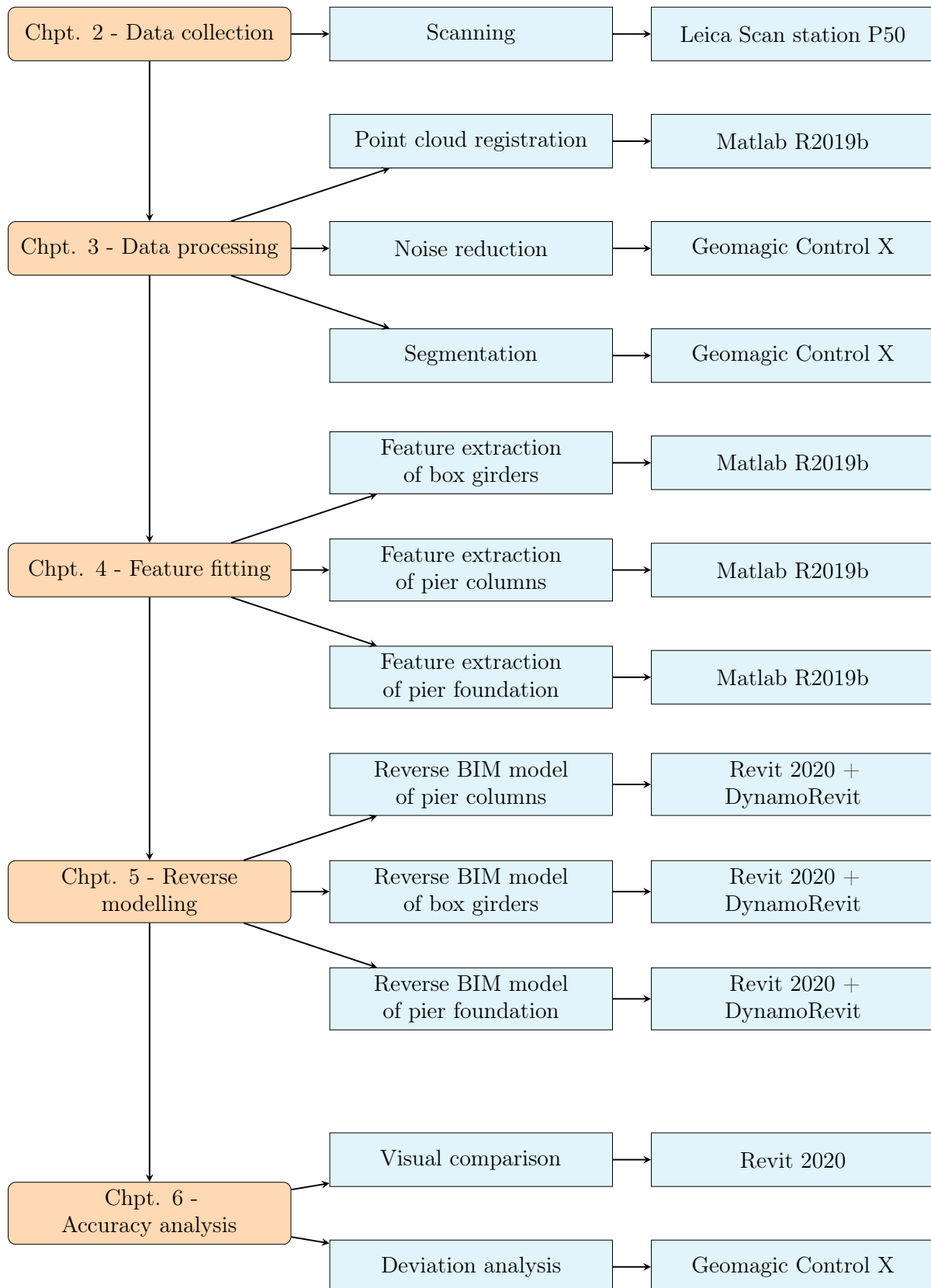
### 1.3.2 Technical Route

The methodology applied in this thesis varies depending on which task is to be solved. Section 1 provides the reader with relevant background information collected from various sources, with the intention of clarifying the significance of the thesis topic both in Norway as well as worldwide. Section 2 through 6 are similarly structured, as shown in Figure 1.12 by firstly introducing the technology and concepts which are to be applied in their respective sections. Subsequently, the relevant work with its corresponding results are presented and discussed throughout the main body of the section. In practice, this means that both theory and results are presented as it becomes relevant throughout the thesis. To end, in Section 7, key elements of the thesis are discussed and evaluated. Further work which can improve the presented thesis work are presented, and the results are evaluated in regard to the thesis problem statement.



**Figure 1.12:** Section organisation

In Figure 1.13, the main tasks within each section are presented together with the hardware and software used for the relevant task. The reverse model produced in Section 5 forms the foundation of which the problem statement is evaluated. To generate the reverse model, data from laser scans were collected, as expanded upon in Section 2. Before the reverse model could be generated in Section 5, the quality of the data was checked and processed as necessary in Section 3 and 4 by the authors using MATLAB R2019b (Matlab) and Geomagic Control X (Geomagic). In Section 6, to analyse the accuracy of the generated reverse model, both a visual comparison using Revit 2020 (Revit) and a deviation analysis using Geomagic was performed. The visual analysis in Revit is subjective, however, it does provide the reader with a general impression of the overall quality of the result. To counter the subjectivity of the visual analysis, an objective deviation analysis is necessary in order to provide accurate, detailed information of the results quality. A deviation analysis can highlight local areas where issues occur and provide objective, numerical results which indicate the quality of the reverse model. By using the attachments, software and methodology as presented in this thesis, the resulting work is reproducible for anyone with access to a sufficiently powerful computer containing the aforementioned software.



**Figure 1.13:** Technical route of the thesis

## 1.4 Applied Tools

The work presented in this thesis relies heavily on the use of advanced tools in order to collect, process, analyse and reverse model the gathered point clouds from 3D point data based on real-world structures to accurate 3D BIM-models. In the following sections, the main hardware- and software tools used to complete this work is presented in chronological order of usage.

### 1.4.1 Hardware

Although this thesis mainly revolves around the usage of software tools, top-of-the-line hardware is a necessity in order to collect accurate and trustable data to be implemented in the various software later in the process. For this thesis, a stationary 3D laser scanner was used to collect accurate point cloud data from real-world structures.

#### **Leica Scanstation P50**

Leica ScanStation P50, a long-range terrestrial laser scanner is used to capture the geometry of the Dade bridge. The high quality, together with ranges up to 1km makes it the perfect tool for gathering PCD of large infrastructure projects. More detailed information about the Leica Scanstation P50 can be found in Section 3.1.1.

### 1.4.2 Software

Usage of state-of-the-art software is the main part of this thesis work. The main use of the software is to combine, process, reverse model and analyse the gathered data from the 3D laser scanner. In the following text, a brief introduction to the used software is presented in order of usage.

#### **MATLAB R2019b**

Matlab is a programming platform by MathWorks, often utilised by scientists and engineers to develop algorithms, analyse data, visualise results and much, much more. When it comes to programming-based software, the range of application is mostly limited by the knowledge of the user, instead of the limitations of the software. The platform takes

use of its own matrix-based programming language, also named MATLAB, which in this thesis is used for two main tasks within the thesis; merging the laser scans as detailed in Section 3.2.2, and preparing the point cloud for reverse modelling in Revit and Dynamo, as detailed in Section 4.

### **Geomagic Control X**

Geomagic is an inspection software by 3D Systems, which in this thesis is used to further process the point cloud by denoising the data set. This processing is done to reduce the file size and necessary calculation time, which is especially important in large data sets such as a complete point cloud of a full bridge. Geomagic was also used to segment the bridge point cloud into its individual building parts for further processing. This process is detailed in Section 4.

Further, Geomagic is used to analyse and assess the accuracy of the reverse BIM model compared to the merged laser scans collected with the Leica Scanstation P50. The accuracy analysis can be reviewed in further detail in Section 6.

### **Revit 2020**

Revit is a powerful BIM software by Autodesk which can create intelligent, parametric, 3D models of structures with its corresponding construction drawings. Revit is widely used in the AEC-industry today, and its functionality can easily be extended with numerous interoperable software and plugins such as Dynamo. In this thesis, Revit is used to view and combine the generated reverse model of the Dade Bridge, as well as for the visual accuracy analysis. Revit is detailed further in Section 5.2.1.

### **DynamoRevit**

Dynamo is an open source visual programming tool by Autodesk, automatically installed as part of Revit since Revit 2020. It provides an alternate method of both customising and creating geometry in Revit which otherwise could be very difficult or even impossible to do manually. Its visual programming interface is well suited for inexperienced programmers, as its node-based language with lines connecting the different nodes is for many considered a more intuitive approach to programming than text-based languages such as MATLAB. The parametric capabilities and seamless integration of Dynamo in combination with

Revit, as illustrated in Section 5.2.1, makes Dynamo a suitable plugin for reverse modelling of point clouds to 3D BIM-models. The use of Dynamo in this thesis is detailed throughout Section 5.

### **ReCap Pro**

ReCap Pro (Recap) is a reality capture and 3D scanning software by Autodesk well suited to generate point clouds from laser scan data. Recap is used due to its synergy with Revit, both being produced by the same company, Autodesk. As Revit can only read point clouds with the file formats of \*.rcp (Point Cloud Projects) and \*.rcs (Point Clouds), both of which Geomagic does not support, Recap is used to convert the raw data from Geomagic to a file format supported by Revit. This step is necessary to combine the point cloud and the reverse BIM model in the same Revit-workspace, used for the accuracy analysis in Section 6.

---

## 2 Bridge Point Cloud Data Collection Technology

In order to create a reverse model from a physical structure, digital data representing the structure in question must first be collected. Doing this by hand with manual measuring devices is a tedious and inaccurate process, while at the same time making it severely difficult to gather data from parts of the bridge which are difficult to access. Furthermore, manual data collection is not directly translatable from physical measurement data to digital measurement data, resulting in an additional, time-consuming and labour-intensive step to convert the data which has to be added to the reverse modelling workflow. As efficiency is already an issue within the AEC-industry, reducing the amount of time-consuming labour in the workflow is of major importance.

With the rapid evolution of technology over the last couple of decades, traditional methods in industries across the world are shifting towards methods based on information technology. New technology is drastically changing industries and methods - including the measurement and data collection of bridges. Technologies used to create 3D models of a physical object is a subject of an increasing number of research reports, in fields of architecture, civil engineering, industry, to name a few [68, 36, 69]. The appropriate scanning approach depends on the accuracy required for the task at hand, and the structure type, which is to be scanned.

To gather 3D data of a bridge, terrestrial or mobile laser scanning and video or photogrammetry, are all valid approaches which can achieve a satisfactory level of detail. These methods are both non-contact scanning methods, feasible for scanning large objects. Both techniques have significantly evolved over the last few years, and both are capable of providing high accuracy point clouds which would otherwise not be achievable using traditional measuring devices.

By taking advantage of innovative laser scanning methods, the efficiency of inspection and analysis of existing structures can be significantly increased. The added detail and accuracy in combination with the easily accessible digital data which such scanning methods provides, makes it possible to use the gathered data to assess whether old,

degrading structures are in need of repairs or rehabilitation to extend their service life. Taking care of existing structures instead of demolishing and rebuilding without a second thought, is an important step towards a more sustainable future. Furthermore, the use of laser scanners greatly reduces the risk of accidents on-site, as the scanning devices can easily collect data at inaccessible and hazardous places.

Other techniques using single point coordinate precision or a small number of discrete points, such as total stations and inclinometers, fail to cover a full area of a structure.

In the following sections, 3D laser scanning will be in focus. Specifically, Terrestrial Laser Scanner (TLS), which is the approach matching the tasks in this report, producing a dense point cloud data of the bridge.

## 2.1 3D Laser Scanner

3D laser scanning is the method of choice to gather 3D digital data of the Dade Bridge. In the following sections, common 3D laser scan technology is presented along with the working principle of point cloud acquisition.

### 2.1.1 Type Introduction of Scanning Instrument

Laser Detection And Ranging (LADAR) technology is an automatic measurement technique which makes use of a sensor of known location to obtain the spatial coordinates of each point on a target surface and construct a detailed 3D model of the target from the collected measurement data. 3D Laser scanners produce an accurate representation of objects, represented with a point cloud making it an effective and non-destructive way of collecting the wanted data. LADARs depend on two information types, range (distance) and intensity (the strength of the return signal) [70].

A 3D laser scanner is the ultimate tool for solving complex measurement tasks. The measuring instrument creates a photorealistic, three-dimensional image of real-world objects. The basic method and principle of a 3D laser scanner are similar to that of a total station. The scanner has a rotating laser that is used to measure the distance to the object, together with the device rotation angle data. By measuring the time of the laser flight from the scanner with known coordinates, to the object and back to the



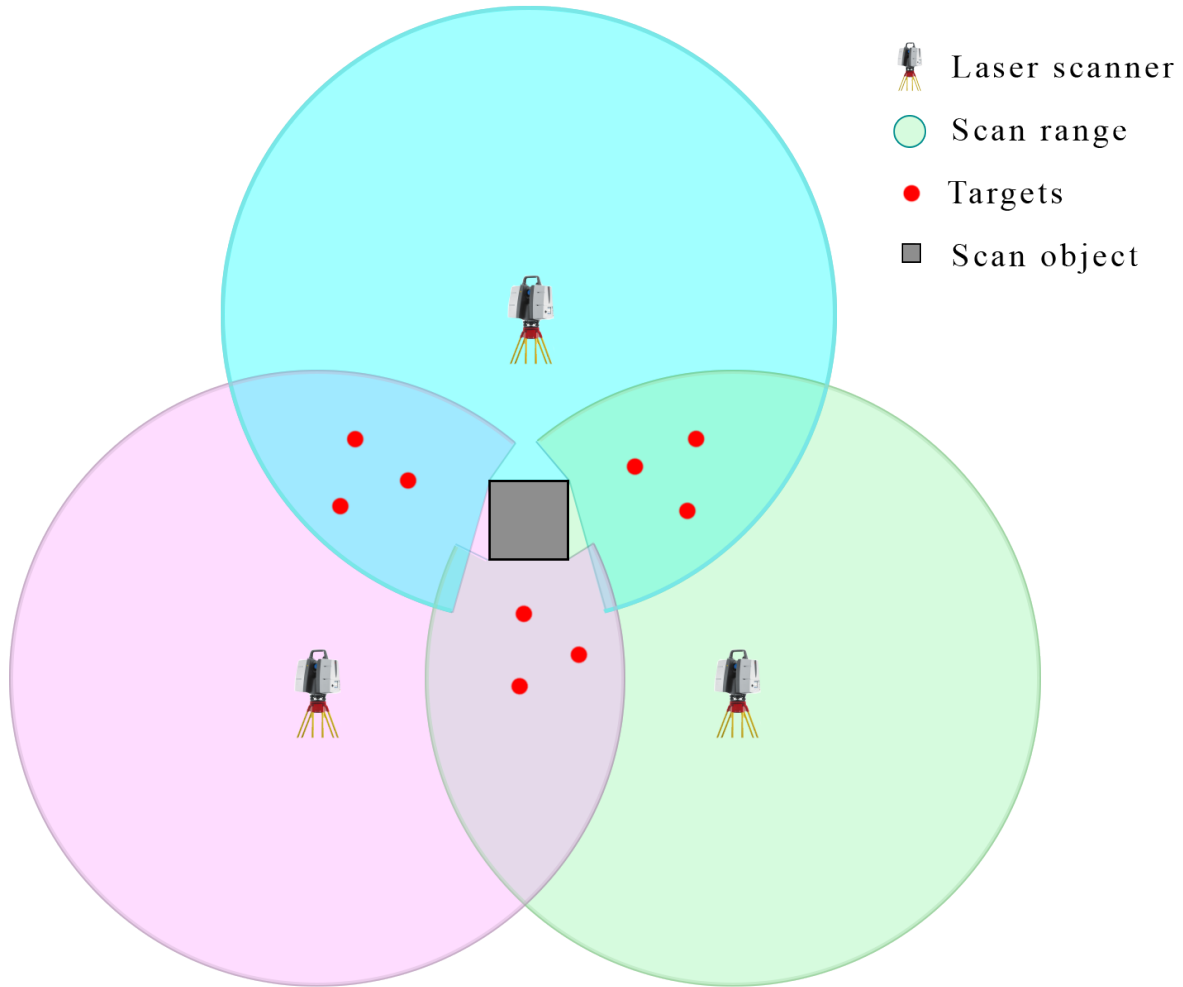
scanner, the coordinates in three-dimensional space of each scanned point on an object is established. The collection of several millions of such 3D-coordinates results in a complete three-dimensional expression of the scanners surroundings called a point cloud. The geometrical properties are captured at high speed, with extreme precision and detail. The point cloud data obtained by the 3D laser can reach millimetre-level sampling intervals, which can be applied in ways such as topographic surveys, cultural relics protection, ground site disaster monitoring and deformation monitoring.

### 2.1.2 Working Principle of Point Cloud Acquisition

In order to make a complete three-dimensional representation of a structure, it is necessary to carry out a large number of scans from different positions. The scanning position needs to be carefully planned with some overlapping to have a complete overall result from all necessary angles, as occlusions such as vegetation and objects occluding each other can block the laser from reaching the target surface. The overlapping is used to collect common points between the scans, to improve the accuracy when merging the scan data.

Although overlapping scans can accurately merge scan data, another possibility is to include shared target points when scanning. The use of shared target points can increase the accuracy even further when merging scans. Although not deemed necessary in the data collection for this thesis, the method of shared targets is detailed below:

A shared target is a physical object that is placed on the scene of the scan. Several of these targets will be used as references to mark common points between each scan. It is common to use 3-5 shared targets between a scan at one scanning position, and its neighbouring scan, as illustrated in Figure 2.1. Software can use these common points to overlay the different scans on top of each other, and put them in the same coordinate system. This process registration can be performed either manually or automatically using processing software. Well planned placement of the shared targets is important as it will result in effective collection of point cloud data and provide a sufficient base of which the scans can use to overlap each other. However, both when using shared targets as well as when scanning in general, it is important to keep in mind occluding elements such as vegetation, terrain and other physical objects, which can reduce the quality of the point cloud as well as block overlapping points between scans.



**Figure 2.1:** Example of placement of shared targets

The result from either a regular scanning process, or a scanning process using shared targets, is a highly accurate point cloud of the scanned object which can be used as a basis for further reverse modelling into a complete BIM model.

## 2.2 Measurement Accuracy Analysis of 3D Scanner

The reliability of the generated 3D point cloud, thus also the subsequent 3D BIM model which is to be reverse modelled, depends on the accuracy and precision of the laser scanner. Accuracy is the degree to which a measurement expresses the true value. Precision is defined as the scatter or variation in repeated measurements.

### 2.2.1 3D Scanner Reliability

TLS are subject to a variety of interference factors, including instrument error, error related to the measured object, environmental error and algorithm error [71, 72, 73]. Vertical index error, collimation error, eccentricity error are examples of instrument errors. Errors related to the scanned object most often refer to the size of the target, its colour, material, roughness or distance. Environmental errors mainly include temperature, pressure, humidity, etc. Algorithm errors are errors from point cloud registration.

For accurate results, the laser scanner needs to be properly calibrated and given suited settings for its intended use. Resolution and quality are two settings on a laser scanner affecting the scan duration, scan size (vertical and horizontal scan points), and million points (MPts), i.e. the number of points you get from the scan. Also, the point density will be determined by the quality parameter. For a typical laser scanner, the measurement of point distance is [mm/10m], i.e. a measurement from a 10 meter distance, should result in a point cloud of scan points with 1 millimetre distance between each other. The accuracy for the scanner used to capture the data of the Dade Bridge is presented in 3.

In 2013, Brown and Hugenholtz researched "Quantifying the effects of terrestrial laser scanner settings and survey configuration on land surface roughness measurement.", demonstrating how the user-defined TLS settings and scan geometry influence the accuracy [74]. Key findings from the article important to keep in mind when preparing a scan using TLS is that:

- The laser point spacing distance should be considerably smaller than the smallest element of interest.
- In order to reduce occlusions, TLS should be performed from at least three vantage points around the object.
- Composite point clouds can cause a positional error.

The resolution and quality vary from instrument to instrument and are generally programmable. These parameter settings can be changed in accordance with the target object, its environment and the requirements for data quality and accuracy. The quality can be reduced to accelerate the scanning process.

The quality of the digital model depends on its application. For applications associated with quality control and reverse modelling, geometric accuracy can be the most crucial factor. For other applications, such as virtual walk through, the photometric accuracy can be more important.

Research has shown that 3D laser scanning has high geometric accuracy [75, 76]. The geometric accuracy is crucial when performing a scan-to-BIM process and will, therefore, be the accuracy of importance in this thesis. For bridges that need to be inspected periodically, the geometric data is the most relevant, making it possible to determine settling or other movements.

### 2.2.2 Data Volume Analysis of Point Cloud

In addition to instrument errors, also the data processing contributes to the model accuracy. The model accuracy is affected by the data volume, i.e. the number of points, in the point cloud. In a point cloud based on two or more scans from different locations, the method of scan registration in a common reference frame and the method of noise filtering, both affect the point cloud accuracy.

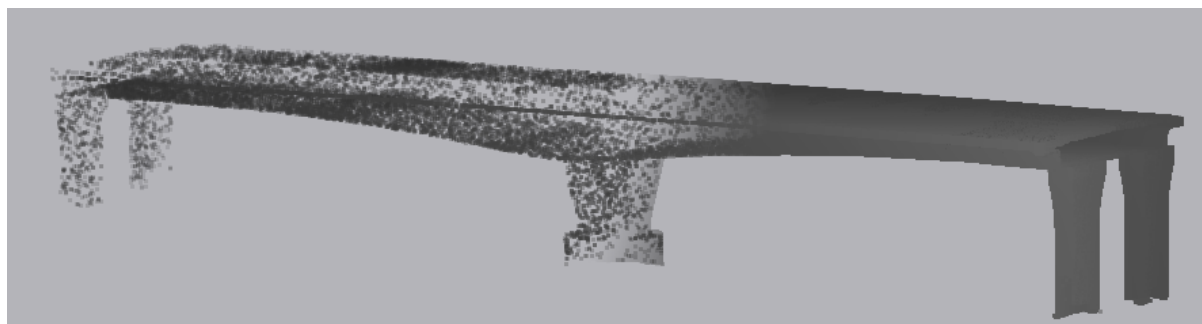
The output from the laser scanning technology is highly dependent on the scanner positions. The scanning range is limited, and distance can affect the accuracy of the point cloud. 3D laser scanners are also known to produce errors at edges [75].

Accuracy and point cloud density are two important parameters for evaluating the point cloud quality. Errors or imperfections can originate from different sources; however, common for all laser scanners is the laser itself. Laser scanners calculate the distance from the sensors to the target using the time-of-flight of the laser beam, which relation is described as  $d = \frac{t \cdot c}{2}$  with a distance,  $d$ , time,  $t$ , and speed of light,  $c$ , affecting the accuracy of measurement [74].

The signal strength of the returned signal is mainly dependent on the distance between the scanning instrument and the target object, as well as the reflectance of the scanned material. The reflectivity of an object is a function of its colour, texture and specularity. Furthermore, distance, colour and angle of incidence all affect the range accuracy. In a study on calibration of laser scanners, Cheok et al. found that the colour of the target is

of little effect. However, less reflective targets give less accurate results from longer ranges. The angle of incidence has more impact on the range accuracy, especially combined with highly reflective targets [70].

In 2017, Rui and Liu [77] published an article on "Research on Applied Talents Training of Urban and Rural Planning Major Based on 3D Laser Scanning Technology" in the Journal of Beijing City University which confirms that the colour and roughness of the target object does not have a significant effect on the accuracy of the point cloud. An important factor, however, affecting the accuracy of the point clouds is the number of points that make up the point cloud. The research shows that the number of points is inversely proportional to the scanning distance. The laser emitted by the scanner is continuously enlarged due to the divergence of light as the distance increases. This effect can make an object with 2000 points on a range of 5 meters, get as little as 50 points from a 30 meter distance. Naturally, a dense point cloud will be a more accurate representation of the target object and should be strived for.



**Figure 2.2:** Difference in point cloud density (low density: left, high density: right)

Within the accepted range of the instrument, the number of points is negatively related to the distance. When using a 3D laser scanner to model a target, it should be as close to the target as possible in order to obtain more points and avoid modelling errors from data holes.

Distance also has the effect that size and complexity of the object has to be taken into account together with the accuracy needed when evaluating how many scans will be required to get a satisfactory result. A shadowing effect can have a big impact on the composite point clouds, and thereby the number of scans needed. If the instrument is placed close to the object, the shadowing effect gets large with a high degree of obstruction.

For complex structures, it means that the number of scans needed to get a point cloud without data holes, will be high.

## 2.3 Common Point Cloud Collection Methods

To collect point cloud data, there are several different scanning methods available. The two most common methods being 3D laser scanning and processing photographs.

For 3D laser scanning, there are a wide variety of scanning instruments available for use today, all providing high resolution scans with accurate data collection. Some of the different types are introduced in this section.

Airborne LADAR mounted on planes, helicopters or UAV (unmanned aerial vehicle) is one kind of laser scanning. Using an airborne laser will result in fast readings over a large area. However, the point density is relatively low. This method is most suited to get an overhead view of large areas, often a technique used for capturing the earth surface, topographic height and environmental data [78].

Mobile LADAR can be mounted on moving objects, such as vehicles, backpacks, or drones. This method has a higher point density but a shorter range than that of airborne mounted LADAR. This method is most common in road inspections, where the mounted laser scanner can collect data on-the-go, which in turn can be used to analyse the condition of the road which the mounted-vehicle drives on [50].

A TLS is a stationary, ground-based LADAR. These scanners are often used for as-built modelling of structures or terrain mapping within the range of 50-300 m. However, long-range laser scanners with a range, up to 1 km are also available on the market. When using TLS, the stationary instrument should be moved several times to get a complete scan of the target object due to occlusions. Many of these scanners take high definition pictures along with the laser scan, meaning that pictures can be overlaid on top of the collected point cloud. This is especially practical for situations where a point cloud is to be manually processed, i.e. analysed, filtered, segmented and similar.

A TLS has high accuracy, high data acquisition rate and high data density [79], making it suitable for use on structures to be analysed.

Photogrammetry is using a series of photographs to create a point cloud. Matching

points are found in the different photographs, which the point cloud is derived from. Photogrammetric systems are advantageous in the sense that high-resolution images are rich in information about texture and colour. In addition, the collection method is relatively fast. However, the method is sensitive to different weather conditions and lighting, thus making it more suitable for controlled environments such as an indoor studio.

To collect the 3D point cloud of the Dade Bridge, the Leica ScanStation P50 TLS was used. This laser scanner is an intuitive measurement tool for long-range scanning. With the P50-model, the scanning of inaccessible places can be executed in a fast and safe manner. An integrated HDR-camera captures detailed imagery with colour overlaying the scan data. It is a mobile instrument for fast, secure and reliable scanning [80].

## 3 Collection and Processing of Bridge Point Cloud Data

In this section, the point cloud collection method and processing of the point cloud data of the Dade Bridge is presented. Firstly, the point cloud data collection is described by introducing the scanning instrument and how the scanning itself was executed. Subsequent to this, the pretreatment of the acquired bridge point cloud is described in detail from preprocessing algorithms to point cloud filtering. Further, the segmentation process of the merged point cloud of the Dade Bridge is detailed.

### 3.1 Bridge Point Cloud Data Collection

In order to generate an accurate reverse model, the initial point cloud data has to be as true to the built condition of the scanned structure as possible. To achieve this, it is important to know how the applied hardware should be used. In this section, the laser scanning instrument used in this master thesis is presented, along with a general workflow to collect 3D point clouds with laser scanners. Lastly, the scanned point cloud is presented and discussed in its completion.

#### 3.1.1 Introduction of Scanning Instrument

Leica ScanStation P50 - Long Range 3D Terrestrial Laser Scanner was used to 3D capture the Dade Bridge. Leica P50 delivers high-quality point cloud and HDR imaging with a scan rate of 1 million points per second at ranges  $>1$  km. A long-range scanner can provide a fast and safe scanning procedure of a large structure [80].

The longer range allows measuring of unreachable sites from a remote and safe location, less field time due to a reduced number of setups, higher flexibility finding a suitable scanning position and more object details from each scanner position. The time and cost savings with these properties are significant. Fewer setups reduce not only the time spent in the field, but also the required office time due to easier processing [80].

The safety aspect is one of the main reasons for using Leica P50. There will always be certain safety risks on the scene of scanning large infrastructures. Increasing the range



gives the opportunity to scan dangerous and inaccessible sites from a safe distance. The increased range can also provide a higher sensitivity at shorter distances, and difficult surfaces are easier to scan at short range [80].

When performing a long-range scan of a large structure, like a large infrastructure, the use of targets is not necessary. The scanning positions are too far apart for the targets to produce any effect on the accuracy.

The Leica P50 can be used for a complete 360° overview scan as well as for detailed scans of selected areas [80]. When scanning a specific object, using a reduced field of view instead of a full 360° overview, will result in a dense point cloud of the target object with less noise from the surrounding area. Scanning with a reduced field of view is used when scanning the Dade Bridge in order to collect data from the actual bridge and not unnecessary surrounding terrain. This also reduces the file size of the point cloud since fewer data points are stored.

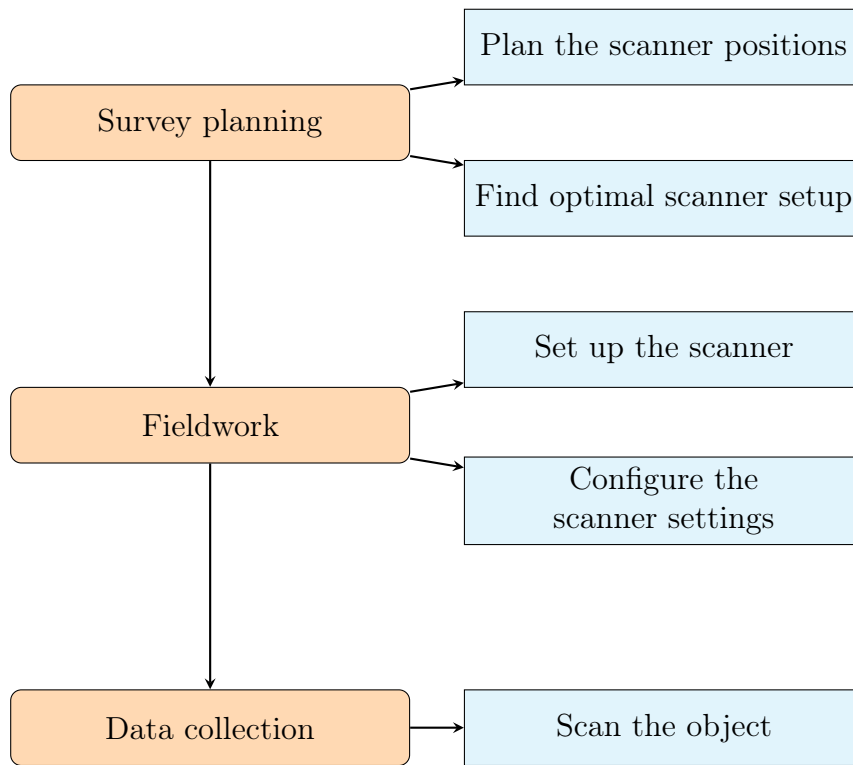
The scan parameters of the Leica P50 scanner are distance, range accuracy and reflectivity. Some standard modes used when scanning are presented in table 3.1. For the Dade Bridge, the parameters of Mode 3 matched the desired quality and size of the scanning area.

**Table 3.1:** Technical parameters of four scanning modes of Leica ScanStation P50 [13]

Scanning mode	distance (m)	reflectivity	Range accuracy
Mode 1	>1000	80%	3mm+10ppm
Mode 2	570	60%	3mm+10ppm
Mode 3	270	34%	1.2mm+10ppm
Mode 4	120	8%	1.2mm+10ppm

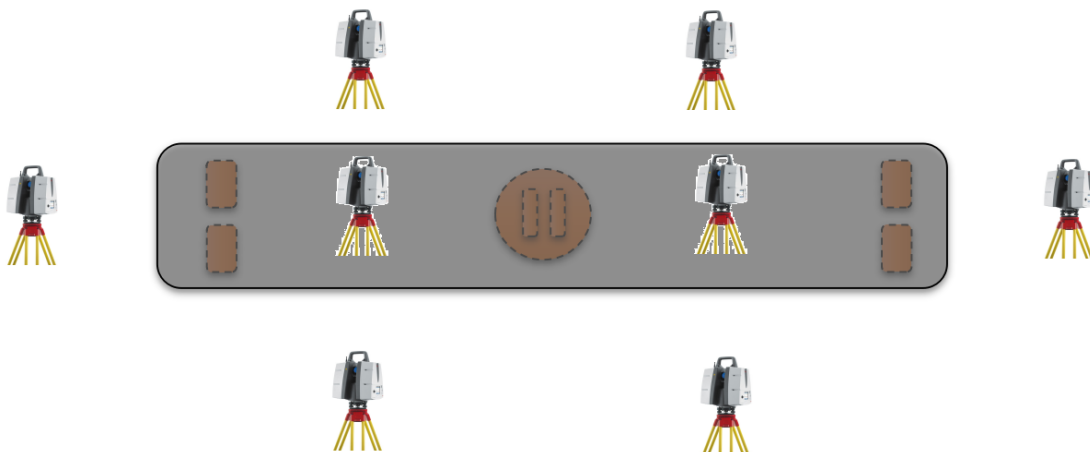
### 3.1.2 Scanning Scheme Design

There are no standards on how to perform a 3D laser scanning of a structure. However, a sequence of tasks certainly needs to be carried out to get a satisfying result. The scanning workflow used in this thesis is presented below, see Figure 3.1. Some of the processes are almost automatic, while some need manual work.



**Figure 3.1:** 3D point cloud collection workflow

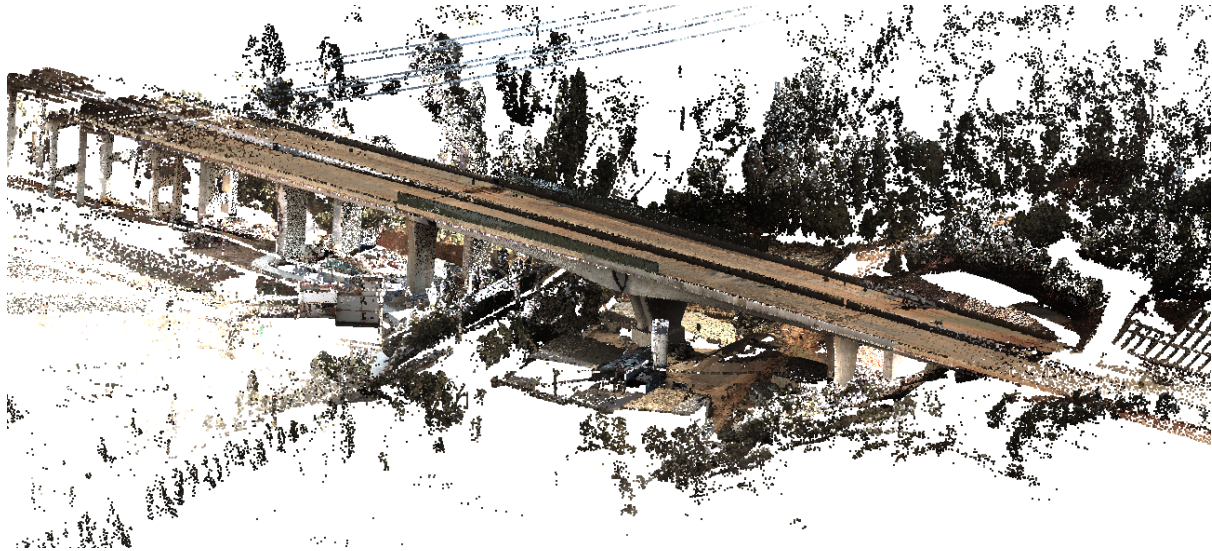
A total of eight scans were performed. Six scans from ground level around the bridge and two on top of the bridge deck. A rough layout of the scanning position can be seen in Figure 3.2. The scanner was positioned at approximately these positions, with slight variations due to the terrain of the site.



**Figure 3.2:** Placement of 3D laser scanners

### 3.1.3 Bridge Point Cloud Scanning

The complete point cloud consisting of all eight scans had a total of more than 150 million individual points. The unfiltered scans resulted in the point cloud seen in Figure 3.3. From the images, it is apparent that the point cloud is quite dense around the bridge part, while the points for the surrounding terrain further away from the bridge are scattered. This is intentional, as the bridge and its immediate area is the focus of this thesis.



(a) Overview 1



(b) Overview 2

**Figure 3.3:** Complete Point cloud

Some images of the scanning procedure of the Dade Bridge can be seen in Figure 3.4. Here, the TLS is placed at several different locations in order to generate a complete 3D point cloud representing the as-built Dade Bridge.



(a)



(b)



(c)



(d)

**Figure 3.4:** Scanning pictures

## 3.2 Pretreatment of Bridge Point Cloud

Before the point cloud data can be exported to the feature fitting process, the bridge point cloud must be pretreated. Firstly, the point cloud preprocessing is presented along with its theory. Second, the theory is applied during the point cloud registration, preparing it for the last pretreatment step: noise reduction.

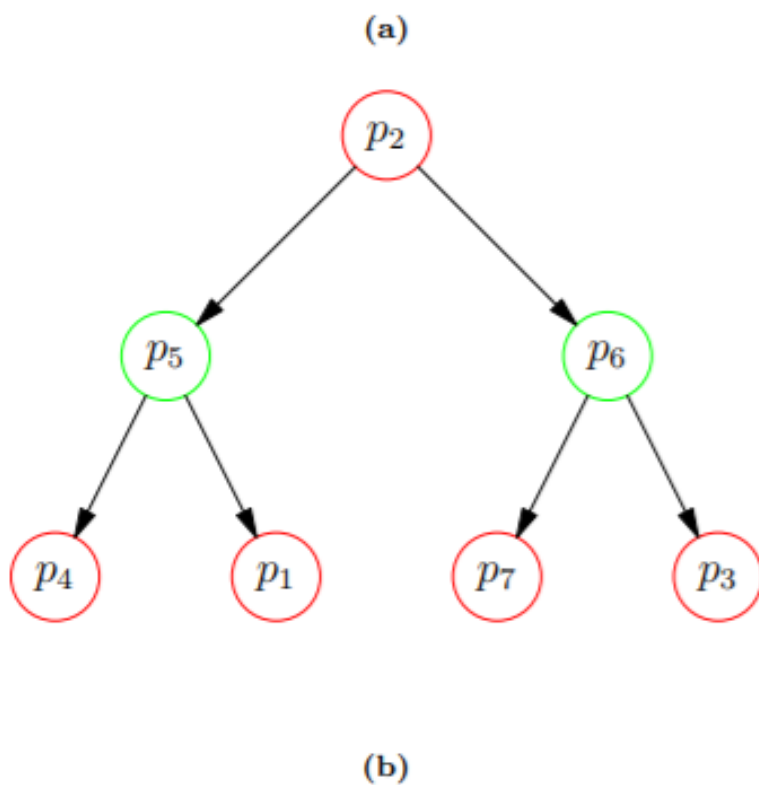
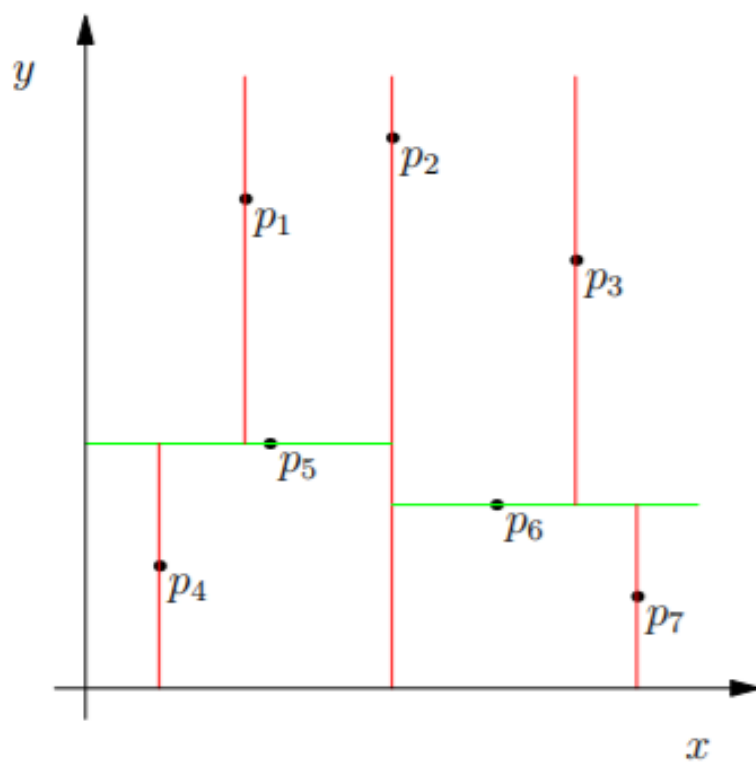
### 3.2.1 Introduction to Point Cloud Preprocessing

Preprocessing a large set of 3D laser scans is demanding. The scans must be merged to form a single 3D representation of the scanned object. This can be done with software such as Autodesk Recap. However, it is also possible to use the Iterative Closest Point algorithm (ICP), an algorithm which has become the dominant method for geometric alignment of three-dimensional models. The ICP algorithm can register point clouds generated from different viewpoints at the same scene in the same coordinate system. ICP estimates the best alignment of two point clouds [81].

With multiple scans overlapping each other, the ICP algorithm is minimising the distance between the corresponding point pairs. Many variations of the ICP-algorithm exist, all of which affect different elements within the algorithm, i.e. its process, selection, matching, minimisation and transformation. For data containing a large number of points, the registration speed and accuracy is affected. An ICP algorithm based on the k-d tree is used to increase efficiency and accuracy when merging the scans of the Dade Bridge to counter the large amount of points [81, 82].

The k-d tree is a data structure used in computer science. The process is a binary space partitioning creating a binary search tree used to find the point neighbour [10]. The k-d tree provides an accelerated search for neighbouring points by conditionally excluding points, which in turn reduces the number of distance calculations. Figure 3.5 illustrates the process for points in two dimensions.

The improved ICP algorithm combining the k-d tree and the traditional ICP algorithm shows great improvements on speed and accuracy [81, 83].



**Figure 3.5:** (a): A two-dimensional point cloud which has been divided using binary space partitioning. (b): The corresponding k-d tree [10]

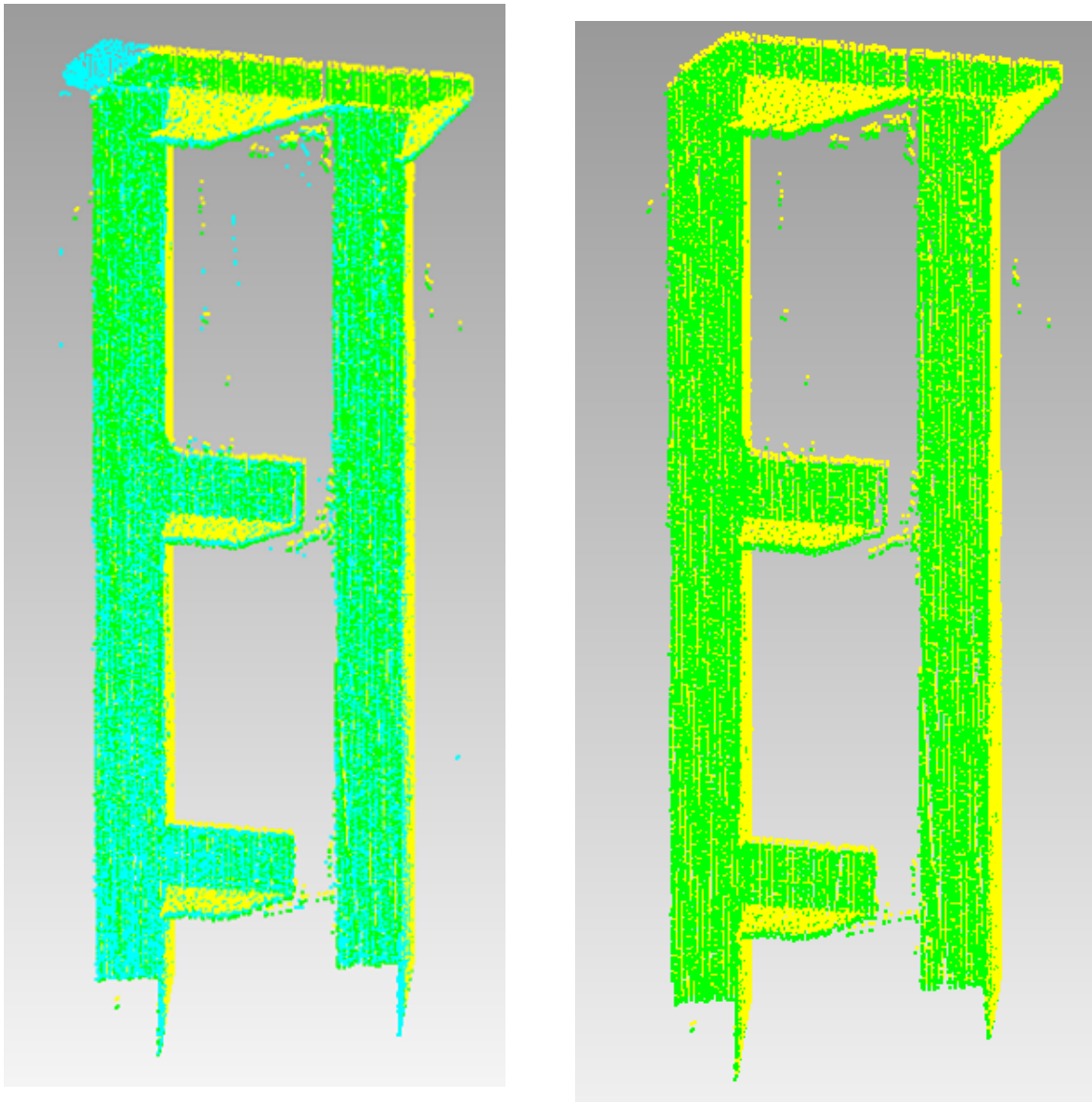
### 3.2.2 Bridge Point Cloud Registration

For a huge structure like the Dade Bridge, multiple scans have to be performed to get an accurate representation of the object. This result in a massive amount of points from each scan that must be merged. In this thesis work, the improved ICP algorithm implementing k-d trees is used to merge the scans into one.

For illustration purposes, a point cloud of a simple bridge pier is used to describe the improved ICP method in practice, instead of the complete Dade Bridge. This is done due to the complexity of the overlapping scans of the Dade Bridge. In Appendix A1 the ICP algorithm for the Dade Bridge is showed.

For this pier, the point cloud of two overlapping scans are merged, namely "A" and "B". Each point cloud, only showing the unprocessed pier, consist of approximately 36.000 individual points, each with its own position in 3D space. The goal is to move A, so that A and B properly connects and becomes a complete, merged point cloud. With the improved ICP algorithm, a new A, "A0", is generated and placed so that it connects with B with the use of spatial translation and rotation transformation of A. As seen in Figure 3.6, point cloud A is copied and moved to the position of A0, forming a complete, merged scan with B.

Figure 3.6 (a) shows the merged point cloud A0 (green) and B (blue). It also contains the original A (yellow) before it was moved. Figure 3.6 (b) shows how much A0 was moved from its starting position, A. The pictures both show how significant the move is, reaching a much better overlap between B (blue) and A0 (green). Precise possessing of the scans are crucial for the accuracy of the point cloud and the subsequent reverse modelling. Using the k-d tree in combination with the ICP algorithm can prove to have a significant impact on the results.



(a) The two scans A(Yellow) and B(blue) and the moved A0(green) (b) How much A(Yellow) has moved to A0(green)

**Figure 3.6:** Iterative Closest Point concept illustration

### 3.2.3 Noise Reduction of Bridge Point Cloud

The data provided by a 3D scan is often noisy, as seen in Figure 3.3. When creating a 3D model of the bridge, only the points representing the bridge are relevant. All noisy data cluttering the model should be removed for accurate results as well as to reduce the sheer size of the point cloud.

The reduction of noise is performed in the software Geomagic Control X. From the original point cloud with 150 million individual points, the noise reduction decreased the number of points to approximately 23 million points, i.e. a reduction of approximately 85%.



The noise is first removed automatically with the "filter noise" function in Geomagic. This function filters out noise clusters that have a maximum of a user-defined amount of points. Being the target of scanning, the bridge is the biggest cluster. Nevertheless, some smaller clusters can occur. Selecting a low number ensures that none of the points representing the bridge itself are removed. This automatic reduction removes a significant portion of noise, the rest is manually removed by simply selecting and deleting it. How the number of points in the point cloud is reduced from the original scan to the cleaned-up point cloud is presented in table 3.2, and the cleaned-up point cloud can be seen in Figure 3.7.

**Table 3.2:** Amount of points at different stages of the noise reduction process

Original scan	Post automatic reduction	Post manual removal
150 million	29 million	23 million



**Figure 3.7:** Cleaned point cloud

## 3.3 Structural Segmentation of Bridge Point Cloud

Although the laser scans are now merged into a single, combined point cloud, it is not yet ready for reverse modelling. First, the bridge must be segmented into its main structural parts, i.e. its superstructure and substructure. This section starts out by introducing the segmentation software used to segment the main structural parts, with a subsequent description of the super- and substructure to be segmented.

### 3.3.1 Introduction of Bridge Point Cloud Segmentation Software

Geomagic Control X (Geomagic) is a powerful inspection software from the American 3D Systems. The software can be used for processing, handling and analysis of massive 3D point clouds. It also offers CAD deviation analysis, excellent for metrology-level inspection.

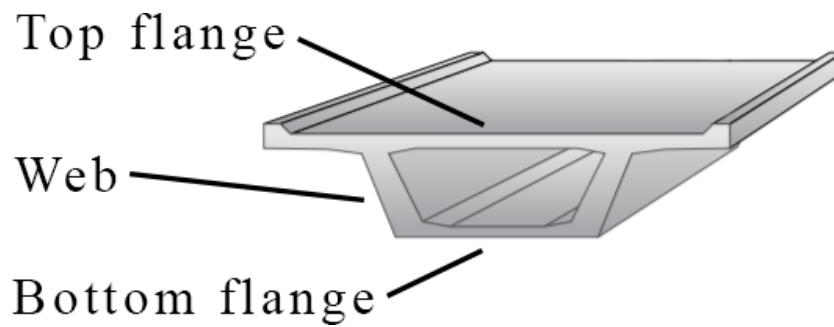
In addition to applications in civil engineering, Geomagic, is used in other fields such as manufacturing and medical research. It is the only software built specially built around the customer preference of design engineers, with fast and easy execution of inspection processing and CAD-like user interface.

### 3.3.2 Superstructure Segmentation

The bridge structure that directly receives the live load is referred to as the superstructure of a bridge. The portion of the bridge supports the deck and connects to the substructures.

To keep the bridge self-weight at a minimum, box girder bridges are a commonly used structure type that is indispensable for long-span bridges, often used for highway bridges.

Box girder bridges are normally built in concrete, but can also be made of composite of steel and concrete. A typical box girder structure can be seen in Figure 3.8, with an explanation of its structure. The webs often vary in height and can have a parabola-shaped variation in height for optimal adaption to the load effects.



**Figure 3.8:** Explanation of box girder structure [11]

A box girder made of concrete is the superstructure of the Dade Bridge. It has a parabola shaped variation of web height. With Geomagic, the superstructure is manually segmented from the rest of the bridge, resulting in the isolated point cloud segment, as shown in Figure 3.9.



**Figure 3.9:** Highlighted superstructure of the Dade Bridge

### 3.3.3 Substructure Segmentation

The substructure supports the superstructure and distributes the bridge loads to its foundations and the ground. The piers and foundation shown in Figure 3.10 are the isolated point clouds of the substructure of the Dade Bridge.



**Figure 3.10:** Highlighted substructure of the Dade Bridge

## 4 Feature Extraction of Bridge Point Cloud Data

The characteristic features of the super- and substructure of the Dade Bridge shall in this section be extracted with the intention of generating the intersection points which will eventually be used for reverse modelling. This section of the master thesis is structured in three main sections, each focusing on feature extraction of the box girder, pier columns and pier foundation respectively. Each main section is further split in three smaller sections, presenting the feature extraction process from point cloud segmentation to feature fitting and eventually a summary of the results from the applied process for each structural segment.

The characteristic features of each bridge part is extracted by creating comprehensive self-composed algorithms processing the point cloud data in Matlab. The complete scripts of the different bridge parts is attached in Appendices A.

### 4.1 Feature Extraction of Box Girder

Firstly, the features of the box girder is to be found. This section presents the process of segmenting the point cloud of the box girder into smaller, more workable segments in order to perform the feature fitting of the box girder point cloud segments. The feature fitting is a complicated procedure, with key elements being detailed throughout the section. To end, the characteristic parameters, i.e. the coordinates of the intersection points, of a randomly picked box girder segment is presented and discussed.

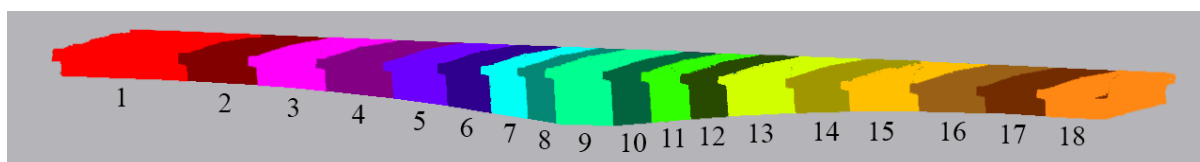
#### 4.1.1 Point Cloud Segmentation

The parabola-shaped geometrical variation of the web height over the bridge span, and the huge number of points (over 20 million points for the complete box girder) makes it necessary to slice the box girder into smaller segments due to geometric variations and limited computer processing power.

Each slice is done manually in Geomagic, trying to keep the change in cross-section at a minimum. This slicing also keeps the number of points in each segment at an acceptable

level, making the processing speed of the following tasks relatively fast.

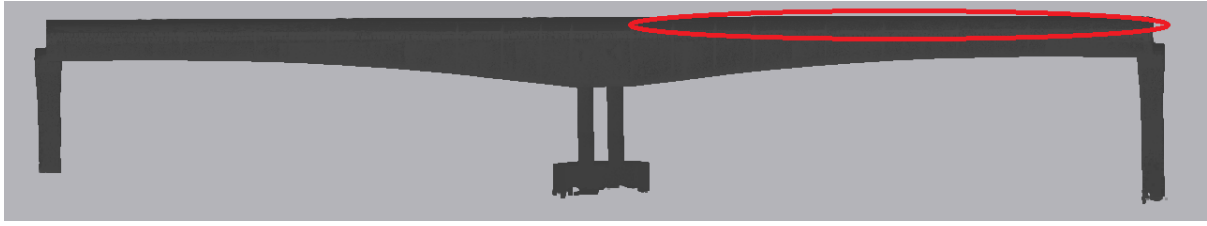
The box girder is divided into 18 segments of varying size. The segments are shorter where the web height varies the most, and longer where the web height is stable, as seen in Figure 4.1. This is necessary in order to reduce the risk of algorithm errors due to geometrical variations.



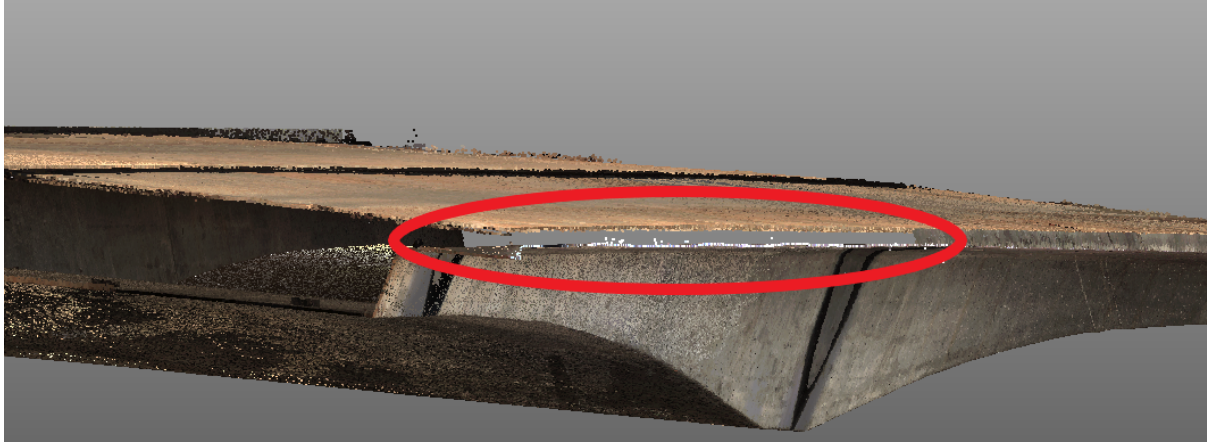
**Figure 4.1:** How the box girder is sliced into 18 segments

Further work with the feature extraction of the box girder shows that the method presented in the following sections is less prone to error with the box girder carefully segmented in accordance to geometrical variations. Another important factor to consider when performing the segmentation is the change in point cloud quality. If a part of the point cloud is missing, or if there is a change in the structure's geometry, thoughtful segmentation should be carried out.

The point cloud in this project had a small part of the box girder side missing for half of the bridge, as highlighted in Figure 4.2, likely due to inaccessible scanning position or material reflection. This missing part has a consequence for the accuracy of the feature fitting of the box girder, and is manually mended as detailed in Section 4.1.2.



(a) Overview of the bridge showing the missing part



(b) Side view of missing part

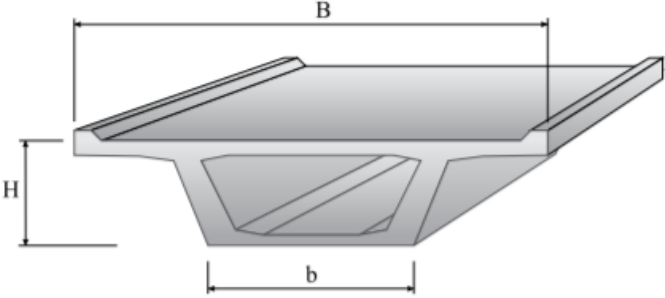
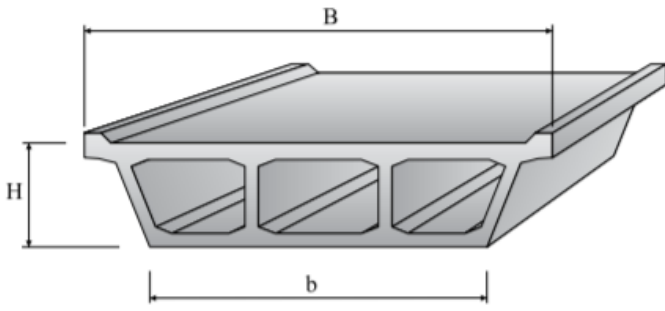

**Figure 4.2:** Missing part at the one side of the box girder

### 4.1.2 Feature Fitting of Point Cloud Segments

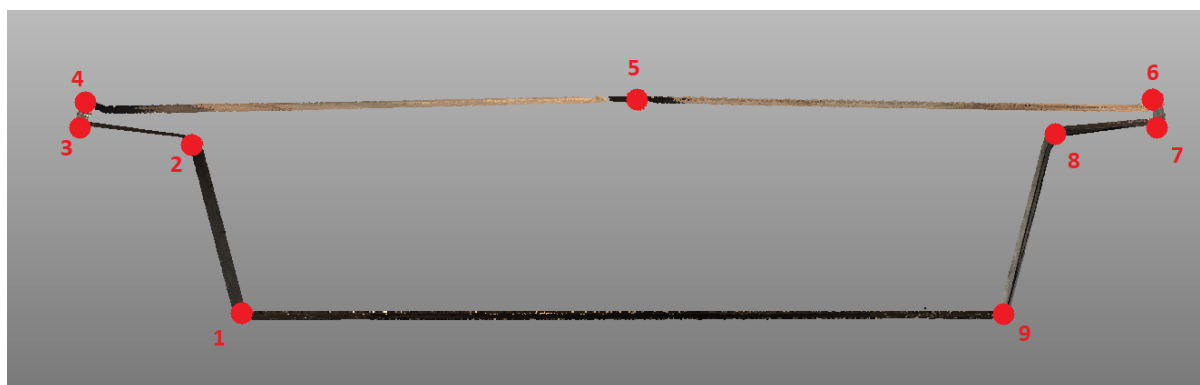
The feature fitting of the box girder segments is done by creating an extensive Matlab script for each segment of the box girder, attached in Appendix A2-A5. The target of these scripts is to find the characteristic parameters of the box girder, i.e. all of its intersection points.

Typical box girder structures are presented in Table 4.1. The most common box girder shape has an inclined exterior web. However, the webs can also be vertical. The difference between single-cell and multi-cell box girders is the number of interior webs. Although easily noticeable in a cross-section of the box girder, an exterior laser scan is unable to differentiate between a single cell- and multi-cell box girder. Some prerequisite knowledge is, therefore, necessary to produce the interior details of a structure based on external laser scans. In this thesis, only the exterior of the bridge is to be modelled. The exterior geometric similarities of the superstructure, be it single-cell or multi-cell box girders, makes the script written in this thesis adaptable for many different bridges.

**Table 4.1:** Typical cross-section of box girder bridges

Typical cross-section	Name
	Single cell box girder [11]
	Multi-cell box girder [11]
	Box girder with vertical web [84]

Simplified, these structures will have 8 or 9 intersection points, depending on the curve of the bridge deck, which when connected forms the shape of the box girder. Having a normal crown makes nine intersection points in total, as the case is for the Dade Bridge seen in Figure 4.3. The intersection point numbering shown in the figure is the same as will be used further on in this thesis.

**Figure 4.3:** Cross section of the Dade Bridge box girder with 9 intersection points

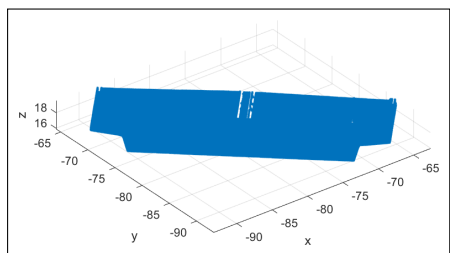
Because of the complexity of the Matlab script, only the main parts of the process will be described in the paper. The different steps of the process are showed like pictures and simplified codes in the following pages. The complete scripts is attached in Appendix A2-A5.

To find the intersection points of the box girder segments, a Matlab script was written and ran for each individual segment. Each segment contains between 300 thousand and 2 million 3D points with x-, y- and z-coordinates, depending on the size of the box girder segment.

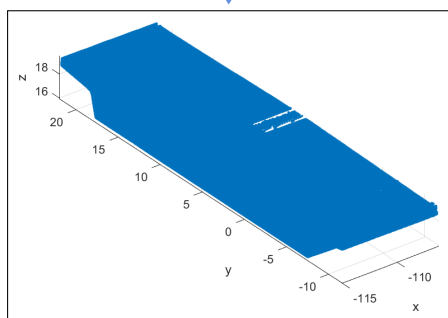
As it is highly unlikely that a scanned bridge is perfectly orientated with its span towards the x-axis, the first step of the process is to rotate the segments in the xy-plane so that the bridge spans along the x-axis. This was done using the rotation matrix around the z-axis. When the rotation is done, the span of the bridge is in the x-direction, width in the y-direction and height in the z-direction, making the dataset much easier to work with, as illustrated in **(B1-B2)** below.

Even though the box girder is initially segmented into smaller parts, each segment is further divided into vertical slices in the x-direction, as seen in **(B2-B3)**, taking small changes in the geometry into account, trying to get the results as accurate as possible. Each vertical slice, as shown in **(B3)**, forms the basis of a cross-section with 9 intersection points, as will be detailed in the coming steps.

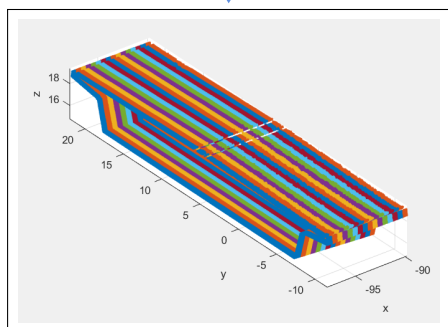




B1: Unrotated



B2: Rotate



B3: Slice

- 1: Read txt-file of point cloud
- 2: Extract 3D coordinates
- 3:  $x=(:,1)$
- 4:  $y=(:,2)$
- 5:  $z=(:,3)$

- 1: Vector between points=

$$\begin{bmatrix} \max X - X_{atMinY} & Y_{atMaxX} - \min Y \\ \end{bmatrix}$$

- 2: theta = angle of rotation

- 3: Rotation matrix(z-axis) =

$$\begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- 4: Rotate box girder

- 1: Range in x-direction= min : 0.25 : max
- 2: **for** Every slice **do**
- 3:   find x-values in range
- 4:   find corresponding coordinates
- 5: **end for**

The next step is the most complex, and most important part of the feature fitting. To find the intersection points, the box girder needs to be divided into smaller parts, facilitating for the use of intersecting regression lines which will indicate where the intersection points shall be placed.

As the geometry of the box girder varies considerably, this partitioning must be done in several steps, outlined below

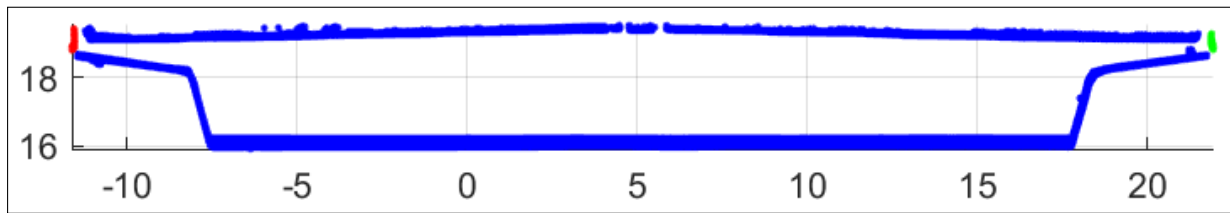
First of all, the outermost sides of the top flange are sectioned out, as shown in **(B4)**. This is done by splitting the points in numerous thin, vertical slices, which are then sectioned out by selecting a couple of the first and last rows with the largest number of points. Now that the two outermost sides have been sectioned out, further partitioning can be executed.

In the next step, **(B4-B5)**, the box girder is further partitioned in thin, horizontal slices, as illustrated in the conceptual Figure 4.4. It should be noted that the actual, thin horizontal slices are much thinner (50mm), however, for illustration purposes they are drawn quite large. From the figure in **(B5)**, it can be seen that horizontal slices containing the magenta, yellow and cyan point segments will have a large amount of points in each slice, while the red, green and blue point segments will lead to slices containing less points.



**Figure 4.4:** Concept of partitioning using thin horizontal slices

To partition the main horizontal parts, the change in amount of points from horizontal slice to horizontal slice is used. Since the amount of points vary from large amounts, to less amounts, back again to large, then less and lastly large again, the horizontal segments is cut at the point of change between max  $\rightarrow$  min  $\rightarrow$  max  $\rightarrow$  min  $\rightarrow$  max points, as shown in **(B5)**.

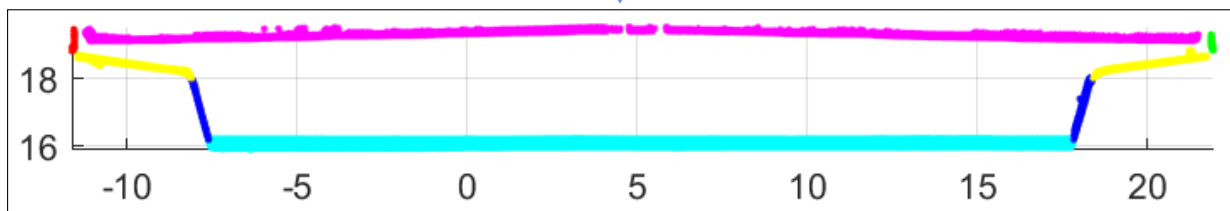


B4: Vertical cut to find outer parts

```

1: for Every slice do
2:   Split vertically  $(\max Y - \min Y) / 0.01$ 
3:   for every vertical slice do
4:     find points in range
5:   end for
6:   Combine a few slices from each end
7:   Remove top and bottom part
8: end for

```



B5: Horizontal cut

```

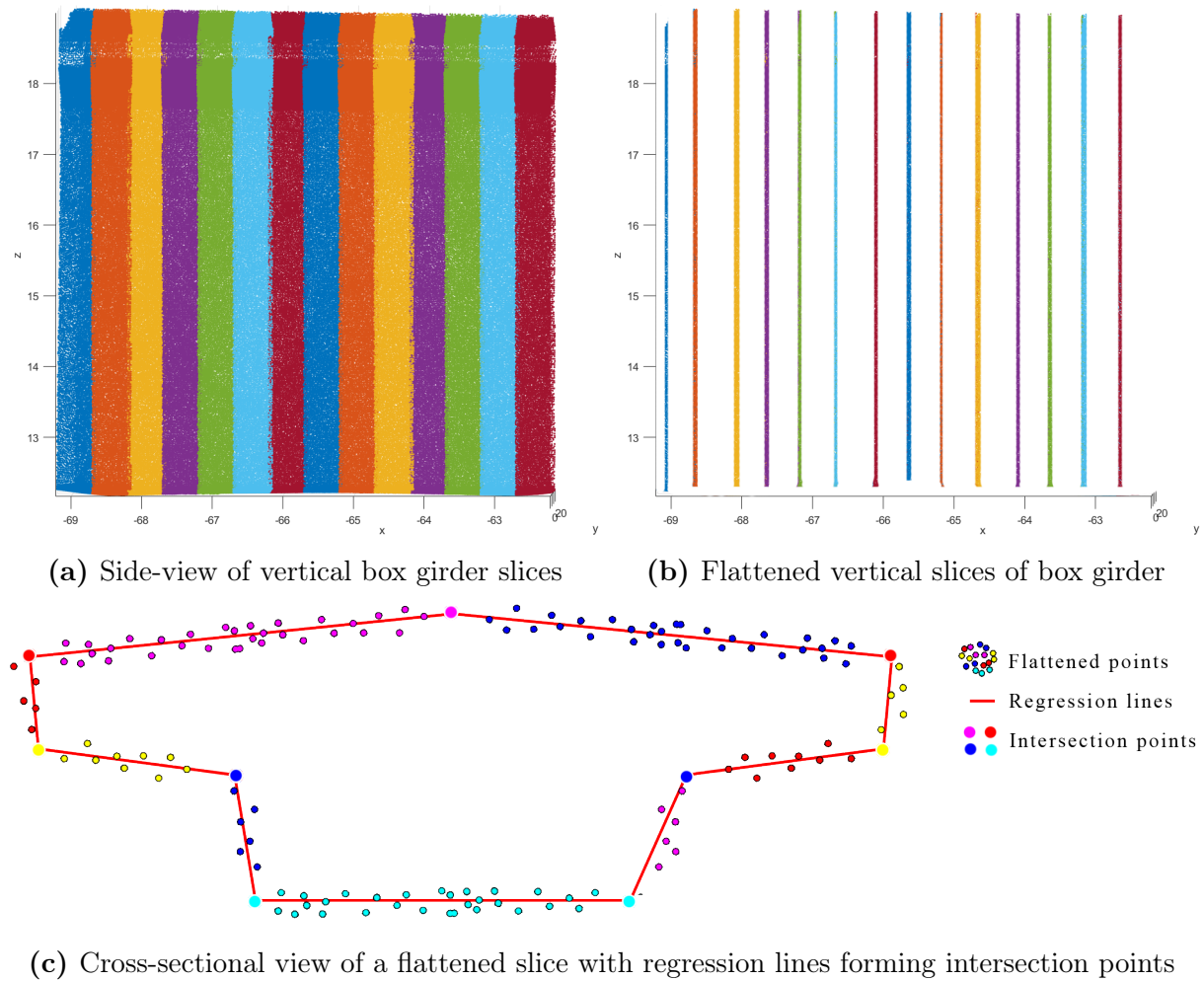
1: for Every slice do
2:   Split horizontally  $(\max Z - \min Z) / 0.05$ 
3:   for Every horizontal slice do
4:     find points in range
5:     Find change in amount of points
6:   end for
7:   Cut position =  $\max \rightarrow \min \rightarrow \max \rightarrow \min$ 
8:   Split parts at cut position
9: end for

```

Now that the box girder points are partitioned in four horizontal parts (magenta, yellow, blue and cyan), the splitting of the sides can be done easily by splitting the list of micro-segments in half, like seen in **(B5-B6)**. Since the bottom flange (cyan) is a single side it is not split in half.

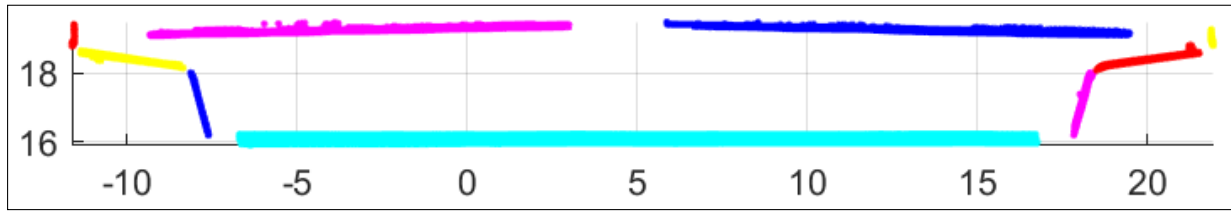
The box girder has now been divided into nine parts, visualised in **(B6)**, splitting the point cloud at every geometrical change. The geometrical transitions, i.e. the corners, are removed to compensate for inaccurate splitting with the added benefit of ensuring that the points in every part forms an as-straight line as possible. By having points in a relatively straight line, the linear regression will be more accurate in comparison to the reference points due to a tighter spread.

After the box girder segment is divided into nine parts with small gaps in-between, the intersection points can be located. The aforementioned vertical slices in the x-direction from **(B3)**, is being reused to divide the whole box girder segment into numerous smaller segments, which can then form as many intersection points as there are slices. As shown in Figure 4.5, all of the points within each vertical slice is flattened to the first x-value of each slice, placing all of the points in each slice at the same yz-plane. A regression line is subsequently generated through each of the nine point groups, and intersection points are placed at the coordinate of each neighbouring intersecting regression line, as shown in Figure 4.5 and **(B7)**.



**Figure 4.5:** Illustration of flattening method and regression lines to find intersection points

Although the intersection points are placed in 3D space, and the outline of the box girder is complete, the box girder is not yet ready for reverse modelling. Lastly, as shown in (B7-B8), the intersection points are rotated back to the original position of the box girder by multiplying the coordinates of each point with the inverse rotation matrix displayed in (B2). The coordinates of the intersection points of the box girder segment is lastly stored in an Excel-sheet, ready to be exported to a suited software for reverse modelling, in this case, Dynamo.

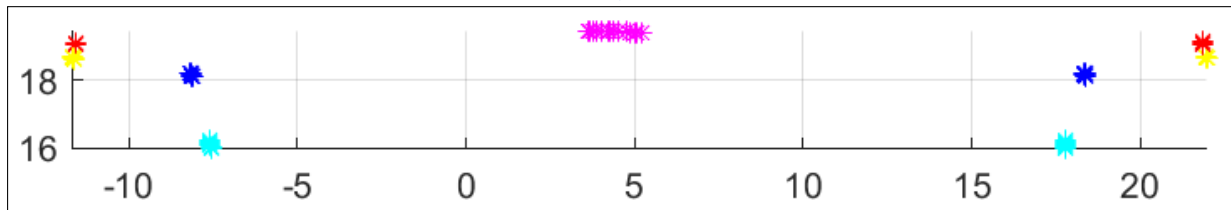


B6: Vertical cut

```

1: for Every slice do
2:   split the horizontal cuts in half, splitting the sides
3: end for
4: for each part do
5:   Split vertically → remove ends
6: end for

```

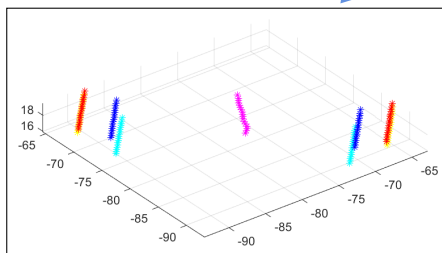


B7: Intersection points

```

1: for Every slice do
2:   Assign one x-value to all points
3:   for Every part do
4:     Make linear regression line between points
5:   end for
6:   Find intersection points at intersecting regression lines
7: end for

```



B8: Rotate Intersection points

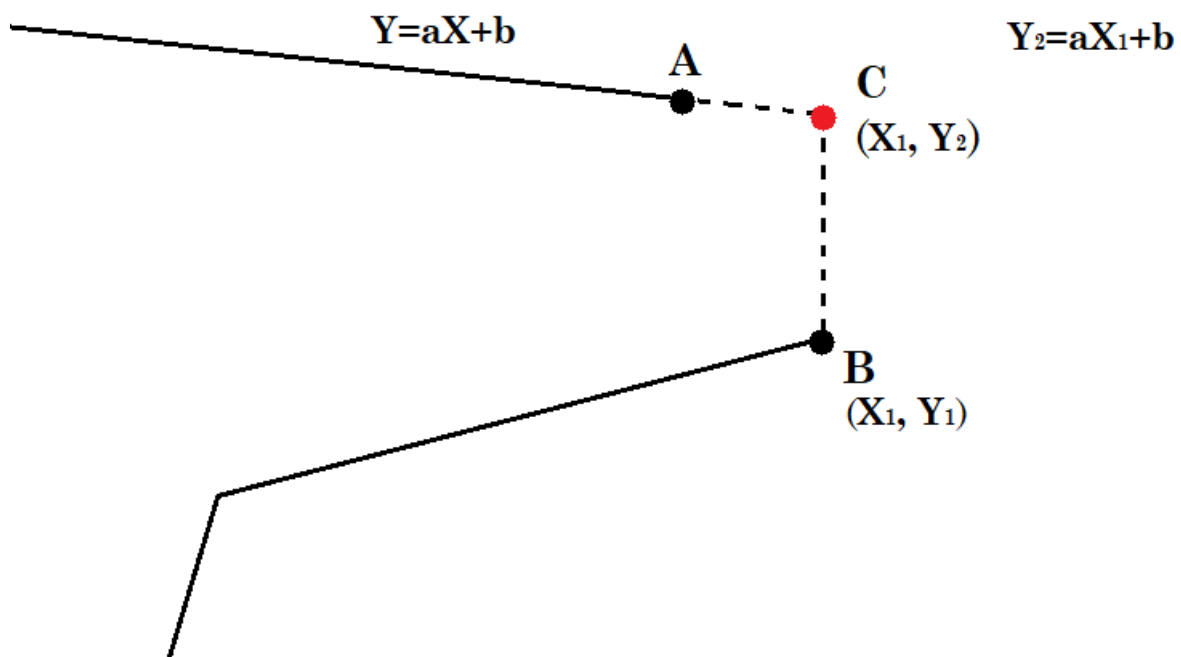
```

1: Rotate intersection points back to
   original position

```

The Matlab code contains almost the same steps for all of the box girder segments. However, there are some alterations made between the first nine box girder segments, and the last nine.

As described in the previous section, Section 4.1.1, the second half of the bridge is missing a side from the scan. When not taking this missing side into account, the Matlab script crashes. In order to address this issue, a separate script was made for the last nine box girder segments where the missing side is located. The script contains almost all the same steps as the first nine, only omitting the codes from the one part which is missing. Without this part, two intersection points are lost. To correct the error, the two missing intersection points need to be created using the concept shown in Figure 4.6, taking advantage of the existing groups of points to manually generate new intersection points approximately at the position where the top flange can be expected to end. The concept of finding these intersection points is shown in Figure 4.6, described in 2D with x- and y-axis. "B" is the end point of the point cloud, the point with the highest x-value. "A" is the endpoint of the top flange point cloud. By taking the x-value of "B" and by extending the linear function of the top flange through "A", intersection point "C" is found on the intersecting lines.



**Figure 4.6:** Finding the missing intersection points

For the first and last box, i.e. box girder segment 1 and 18, the point cloud is incomplete. The original scanning does not have a clean cut at the bridge ends, and can therefore not produce intersection points at all corners of the box girder. Missing intersection points makes it impossible to accurately reverse model the box girder segment, and must subsequently be avoided.

To easily correct the problem, the incomplete part is removed by performing a straight, vertical cut in Matlab at a suited length from the end, found manually with measuring tools in Geomagic. Cutting in Matlab produces a cleaner cut than by doing it manually in software such as Geomagic. Naturally, cutting the point cloud will result in a reverse model which is shorter than the original point cloud. To avoid this, box girder extensions are modelled using data from the original, uncut point cloud, as detailed within the reverse modelling of the box girders in Section 5.3.1.

### **4.1.3 Characteristic Parameters of Box Girder**

The result from the Matlab script has nine intersection points for each slice of the box girder segments. The amount of slices each box segment consists of depends on the length of the sections they are divided into. The length and number of slices in each box segment is presented in Table 4.2. The box girder has one slice at every half meter, i.e. box girder segment 2 with a length of 8,71m is made up of 17 slices.



**Table 4.2:** Table of length and number of slices for every box segment ((x)=after cutting)

Number	Length (m)	Slices
Box 1	16,23 (14,23)	33 (28)
Box 2	8,81	17
Box 3	8,71	17
Box 4	8,61	17
Box 5	6,92	13
Box 6	5,56	11
Box 7	4,82	9
Box 8	3,83	7
Box 9	7,42	14
Box 10	5,03	10
Box 11	5,35	10
Box 12	5,01	10
Box 13	8,70	17
Box 14	7,18	14
Box 15	8,96	17
Box 16	8,72	17
Box 17	7,63	15
Box 18	7,14 (5,14)	15 (10)

In Table 4.3, the intersection point of box girder segment 9 is presented. This box is divided into 14 slices and show the x-,y- and z-coordinates for each of the nine intersection points within each slice. The 9 intersection points correspond to the number sequence previously seen in Figure 4.3.

The intersection points displayed in the table, and also in the last part of the Matlab script, seen in **(B8)** shows promising results. Most of the points form straight lines. Only the 5th intersection point, the one on the top flange separating the two sides of the road, have trouble creating an exact crossing point. These irregularities are caused by the small angle between the two parts, increasing the inaccuracy of their intersection points. Small bumps or a rough road can be enough to cause such changes in the intersection points.

**Table 4.3:** Characteristic parameters (x-, y-, z- values [m]) of box girder 9

		Intersection Points								
		1	2	3	4	5	6	7	8	9
1	X	-58.98	-60.05	-62.66	-62.59	-49.67	-37.72	-37.59	-40.24	-41.34
	Y	-39.94	-38.98	-36.65	-36.71	-48.27	-58.97	-59.08	-56.71	-55.73
	Z	12.24	17.82	18.26	18.60	18.92	18.61	18.22	17.79	12.23
2	X	-58.72	-59.80	-62.40	-62.33	-49.47	-37.46	-37.33	-39.98	-41.08
	Y	-39.66	-38.69	-36.36	-36.42	-47.94	-58.68	-58.79	-56.43	-55.44
	Z	12.20	17.82	18.25	18.59	18.92	18.59	18.21	17.79	12.18
3	X	-58.39	-59.46	-62.07	-62.00	-49.27	-37.12	-37.01	-39.65	-40.74
	Y	-39.28	-38.32	-35.99	-36.05	-47.45	-58.31	-58.41	-56.05	-55.07
	Z	12.19	17.82	18.25	18.59	18.91	18.59	18.21	17.78	12.18
4	X	-58.06	-59.14	-61.71	-61.66	-49.22	-36.79	-36.68	-39.31	-40.40
	Y	-38.90	-37.94	-35.63	-35.68	-46.81	-57.94	-58.03	-55.68	-54.71
	Z	12.19	17.82	18.24	18.58	18.91	18.58	18.21	17.78	12.19
5	X	-57.73	-58.80	-61.39	-61.33	-49.01	-36.46	-36.36	-38.98	-40.06
	Y	-38.53	-37.57	-35.26	-35.31	-46.34	-57.57	-57.66	-55.31	-54.35
	Z	12.19	17.78	18.26	18.57	18.90	18.59	18.22	17.73	12.19
S L I C E	X	-57.40	-58.47	-61.04	-60.99	-48.62	-36.12	-36.02	-38.63	-39.72
	Y	-38.15	-37.19	-34.88	-34.93	-46.00	-57.19	-57.27	-54.94	-53.97
	Z	12.18	17.78	18.26	18.57	18.89	18.58	18.21	17.73	12.19
N U M B E R	X	-57.08	-58.15	-60.72	-60.66	-48.16	-35.79	-35.70	-38.31	-39.39
	Y	-37.78	-36.82	-34.52	-34.57	-45.76	-56.82	-56.91	-54.57	-53.60
	Z	12.18	17.78	18.25	18.57	18.89	18.57	18.21	17.71	12.20
8	X	-56.74	-57.82	-60.38	-60.33	-47.71	-35.46	-35.35	-37.96	-39.05
	Y	-37.40	-36.44	-34.14	-34.19	-45.48	-56.45	-56.54	-54.21	-53.23
	Z	12.19	17.79	18.24	18.57	18.88	18.57	18.20	17.71	12.20
9	X	-56.41	-57.50	-60.05	-59.99	-47.54	-35.12	-35.02	-37.62	-38.71
	Y	-37.02	-36.04	-33.76	-33.82	-44.96	-56.07	-56.16	-53.84	-52.86
	Z	12.19	17.84	18.21	18.56	18.88	18.57	18.17	17.73	12.21
10	X	-56.09	-57.17	-59.72	-59.66	-47.38	-34.79	-34.70	-37.30	-38.37
	Y	-36.65	-35.68	-33.40	-33.45	-44.43	-55.70	-55.78	-53.46	-52.50
	Z	12.18	17.79	18.22	18.55	18.86	18.56	18.17	17.69	12.21
11	X	-55.75	-56.85	-59.38	-59.32	-47.11	-34.46	-34.37	-36.96	-38.03
	Y	-36.27	-35.28	-33.02	-33.08	-44.00	-55.33	-55.41	-53.09	-52.13
	Z	12.18	17.83	18.20	18.55	18.85	18.56	18.16	17.70	12.20
12	X	-55.43	-56.53	-59.04	-58.99	-46.65	-34.13	-34.02	-36.62	-37.70
	Y	-35.90	-34.91	-32.66	-32.71	-43.75	-54.96	-55.05	-52.73	-51.76
	Z	12.19	17.83	18.20	18.54	18.84	18.56	18.15	17.69	12.20
13	X	-55.09	-56.20	-58.71	-58.65	-46.30	-33.79	-33.67	-36.28	-37.36
	Y	-35.52	-34.53	-32.28	-32.33	-43.39	-54.58	-54.68	-52.35	-51.39
	Z	12.18	17.83	18.19	18.53	18.83	18.55	18.15	17.68	12.19
14	X	-54.75	-55.87	-58.38	-58.31	-45.92	-33.45	-33.34	-35.95	-37.02
	Y	-35.14	-34.15	-31.90	-31.96	-43.05	-54.21	-54.31	-51.97	-51.02
	Z	12.17	17.83	18.19	18.52	18.83	18.54	18.15	17.67	12.19

## 4.2 Feature Extraction of Pier Column

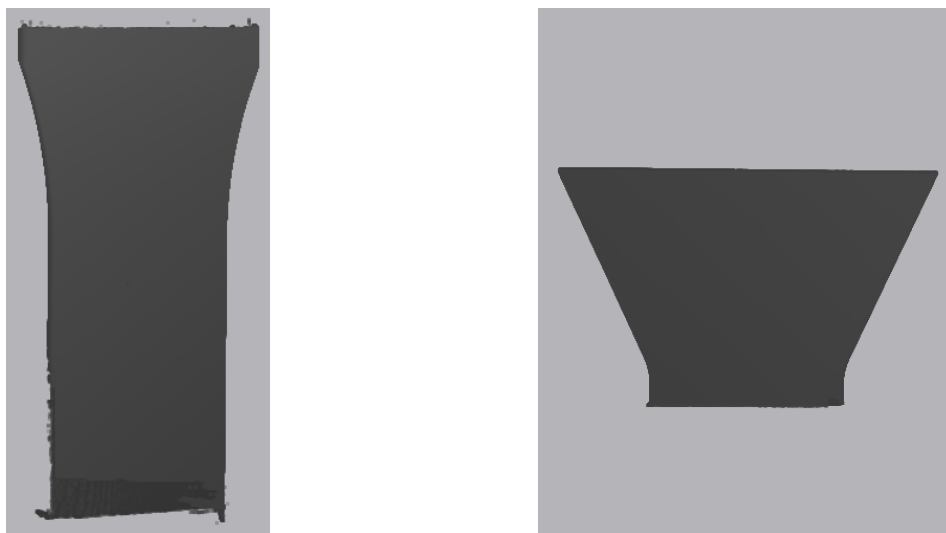
The feature extraction process for the pier columns is similar to that of the box girder, with some alterations due to the different geometry. The segmentation of the pier columns is first presented, and key details for the feature fitting of the different piers is presented step-by-step using a combination of illustrations, simplified codes and textual explanations and discussions with potential improvements to the scripts. To end, the characteristic parameters, i.e. the coordinates of the intersection points, of a randomly picked pier column is presented and discussed. The complete scripts for the pier columns can be seen in Appendix A6-A7

### 4.2.1 Point Cloud Segmentation

After dividing the substructure from the superstructure. The substructure is further divided so that each pier has a separate point cloud segment, as seen in Figure 4.7. Each of the six pier point clouds are separated from the complete point cloud of the Dade Bridge using Geomagic. Of the six piers, there are two in each end and two at the mid-span of the bridge. The two mid-span piers are identical, but have a different geometrical shape than the outer piers. The general geometry of the pier columns are presented in Figure 4.8. Notably, pier column 3 and 4 have a more severe change in cross-section than pier column 1, 2, 5 and 6.



**Figure 4.7:** How the six pier columns are named



(a) Shape of end pier column 1, 2, 5 and 6      (b) Shape of mid-span pier column 3 and 4

**Figure 4.8:** Geometry of the Dade Bridge pier column

Due to the fact that piers go down to the ground, the chance of having disturbance in the scanning is high. Trees, shrubs, cars, and other occlusions often results in incomplete 3D scans. This is also the case for the Dade bridge, as described in the noise removal section, Section 3.2.3.

The geometry of the piers are much simpler than that of the box girder, so unlike the box girder, the piers are not further sliced into smaller parts within each segment.

#### 4.2.2 Feature Fitting of Point Cloud Segments

The feature fitting of the point cloud is done in a similar way as with the box girder, creating an extensive Matlab script and finding the intersection points between the sides of the pier.

A Matlab script that can be used for different pier geometry in general will be very useful in the preparation of a BIM 3D-model. The variety of geometry used for bridge piers is large. This script, however, will work for most square, four-sided piers, just performing some small alterations to the code. Most of the script was made using a quadratic pier of another bridge, and was subsequently adjusted so that it works for both the quadratic pier as well as the six piers of the Dade Bridge. The geometry of the four end-piers, mid-piers and the original quadratic pier which the script is based on, is presented in Table 4.4. The Pier Column intersection point numbering used throughout this thesis is presented in

Figure 4.9

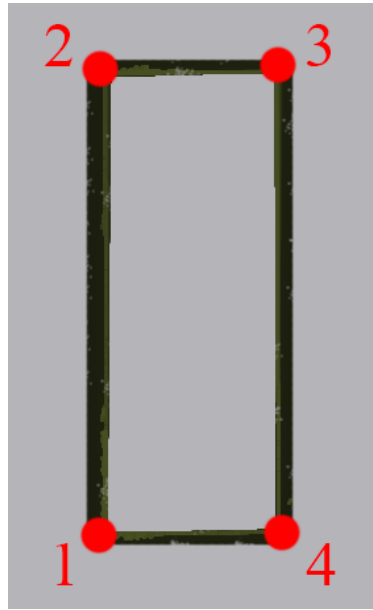
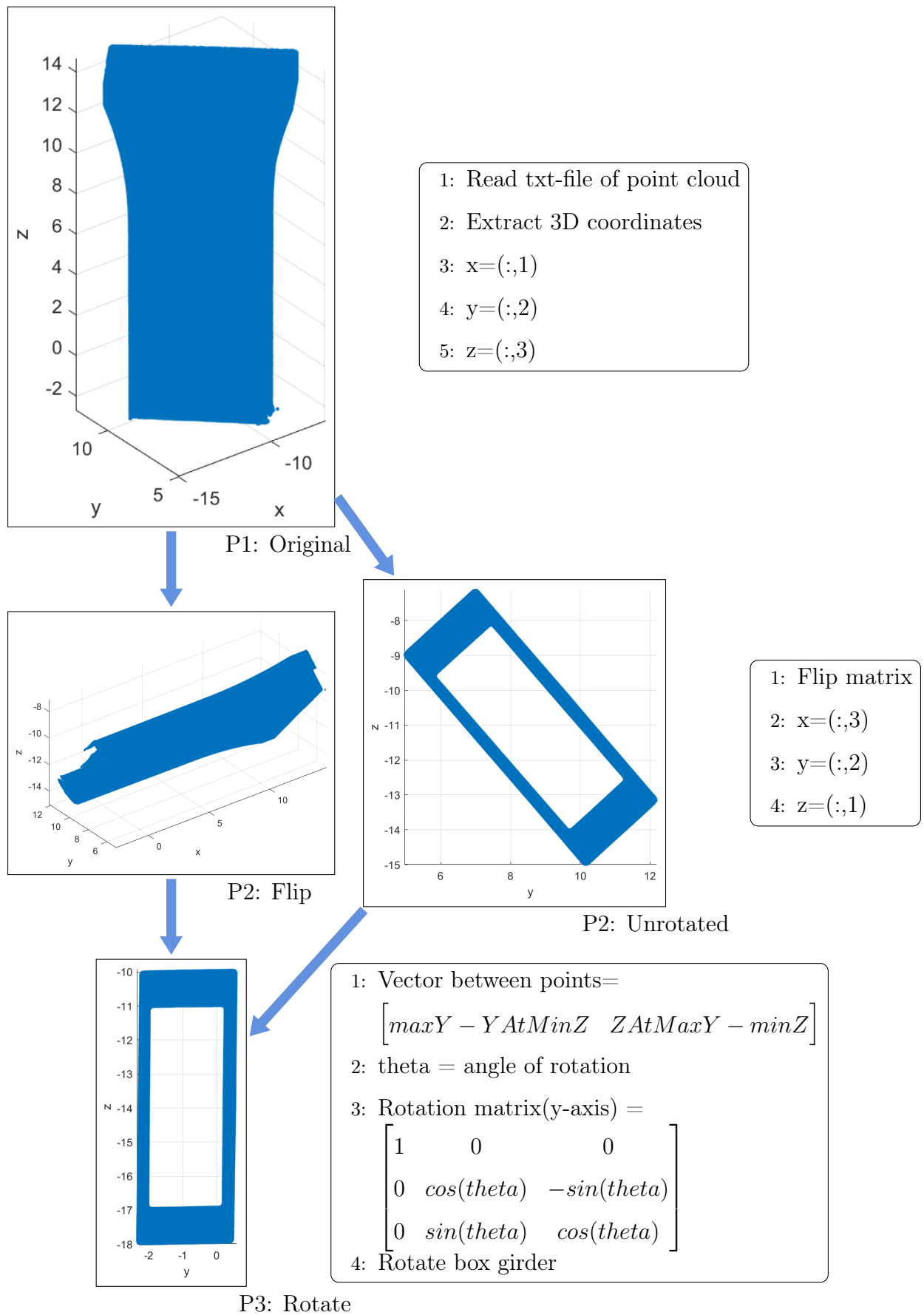


Figure 4.9: Cross section of a Dade Bridge pier column with 4 intersection points

Table 4.4: Geometry of the pier columns which the script is based on

End pier column	Midspan pier column	Alternative quadratic pier column

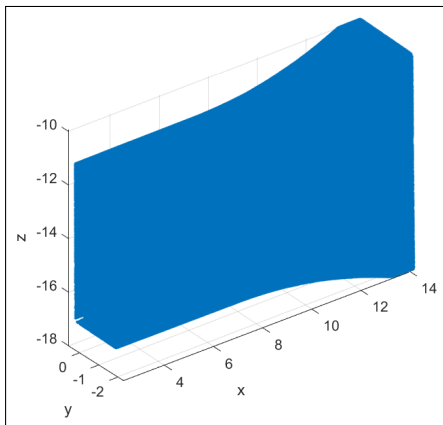
As for the box girder, the first task is to make the point cloud easy to work with. In Matlab, the pier is flipped so that the original height is laid along the x-direction as seen in (P1-P2), and subsequently rotated in the yz-plane using the rotation matrix shown in (P2-P3). By doing so, the pier lies flat on the y-axis with its height going in the x-direction, as shown in the cross section of (P3).



With the pier lying in this way, it can easily be divided into vertical slices along the x-axis, as was done with the box girder segments, seen in (P5) with each colour representing a slice.

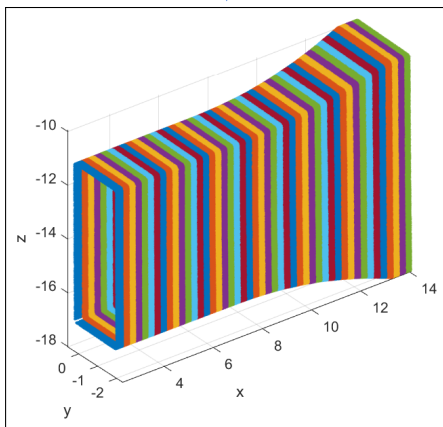
A good scan of an object will result in a complete point cloud. However, this is not always the case. Because of the described disturbance caused by occluding objects, a complete point cloud can be difficult to obtain. Since this occlusion also occurs in the pier columns, the first and last couple of slices are manually removed with Matlab when necessary, i.e. when the slice lacks points at a whole side, making it impossible to later generate intersecting regression lines.

The damaged part can be seen by comparing the bottom part of illustration (P2) and (P4), and is removed by performing a cut at a specified x-value, resulting in the shortened pier column shown in (P4).



P4: Cut

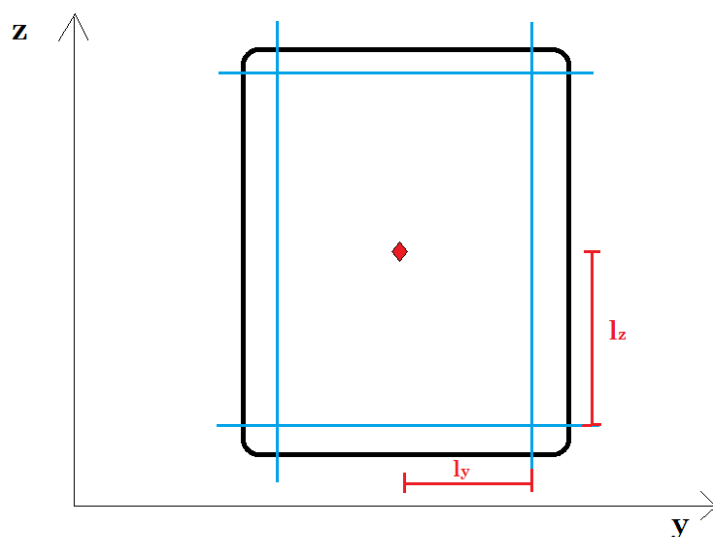
1: Delete damaged part of the pier in X-direction



P5: Slice

1: Range in X-direction= min : 0.25 : max  
 2: **for** Every slice **do**  
 3:   find x-values in range  
 4:   find corresponding coordinates  
 5: **end for**

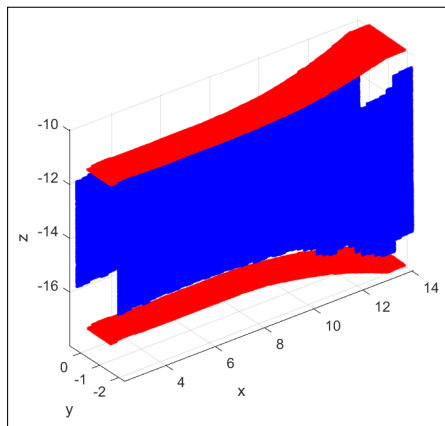
With the pier column point cloud now consisting of only complete slices, the sides of the piers need to be divided into separate parts. This splitting is done by finding the centre of the pier and then finding the appropriate length from the centre and out to where the corner starts. Figure 4.10 shows a cross-section illustrating the general concept for finding these lengths for a general, four-sided pier column. For the vertical sides this means that the point cloud is cut at the centerpoint  $\pm l_z$ , while the horizontal sides of the point cloud is cut at  $\pm l_y$ . To isolate the vertical sides, the points above and below the blue horizontal lines are removed, while the horizontal sides are found by removing the right and left side of the vertical blue lines. By doing this, the corners are also removed, making it easy to perform linear regression through the points as will be detailed in the coming steps.



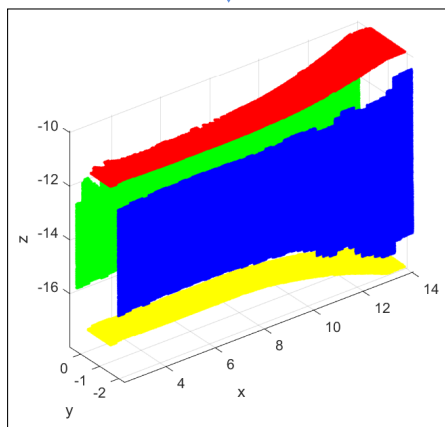
**Figure 4.10:** General concept of dividing the sides into separate parts by finding the length from the centre and out

When the horizontal and vertical sides are divided and the corners are cut out, as seen in (P6), the pier column is further partitioned into a total of four separate parts, two horizontal- and two vertical point clusters for each side. As with the box girder, this is done by partitioning the whole point cloud in thin slices as shown the previous Figure 4.4 and subsequently count the amount of points within each slice. However, different to the box girder, in this case the thin slices are made both horizontally and vertically, and the change in amount of points from slice to slice is used to decide where to partition the points. This method is illustrated in (P6-P7).





P6: parts



P7: Sides

```

1: for Every slice do
2:   Length from center Z-axis= $(\max-\min)/2-1$ 
3:   Length from center Y-axis= $(\max-\min)/2-0.5$ 
4:   Find points in range  $\rightarrow$  Y-matrix
5:   Find points in range  $\rightarrow$  Z-matrix
6: end for

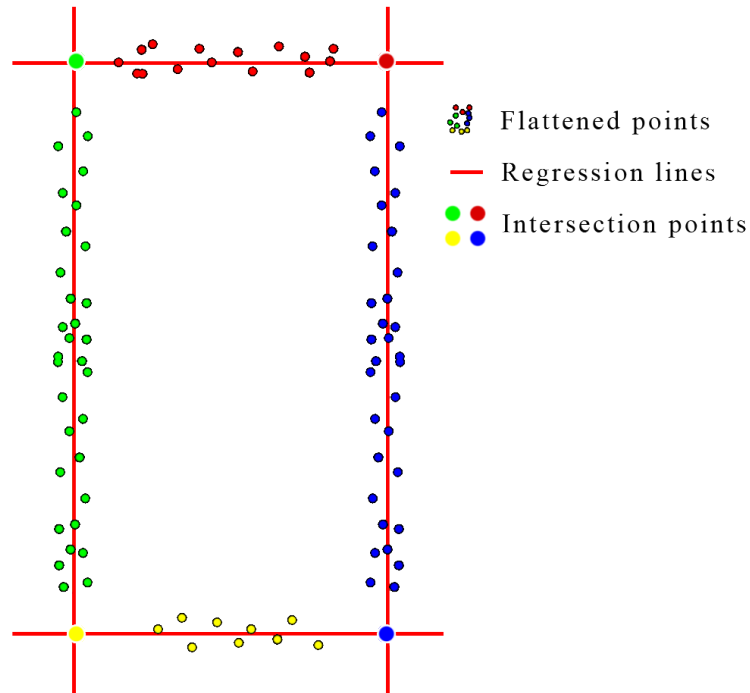
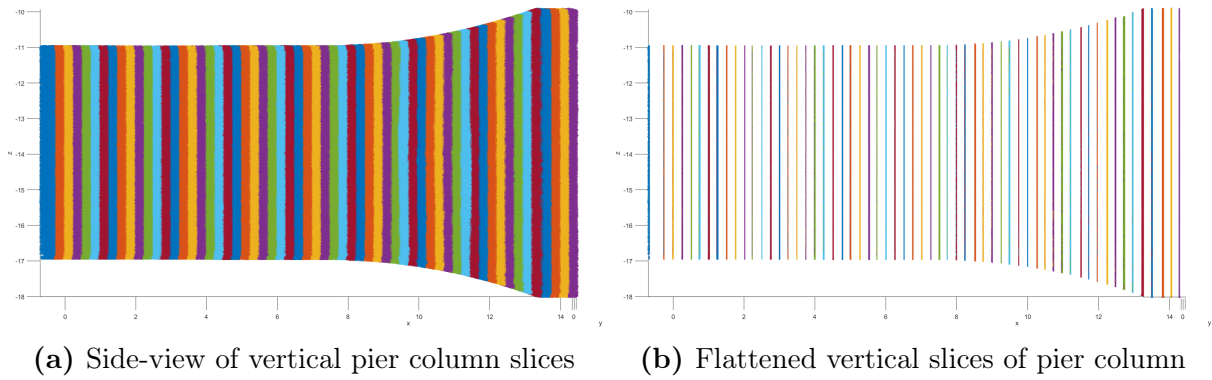
```

```

1: for Every slice do
2:   Split horizontally  $(\max Z-\min Z)/0.005$ 
3:   Split vertically  $(\max Y-\min Y)/0.005$ 
4:   for Every horizontal slice do
5:     Find points in range
6:     Find change in amount of points
7:   end for
8:   Split Y-matrix  $\rightarrow$  Find sides
9:   Split Z-matrix  $\rightarrow$  Find sides
10: end for

```

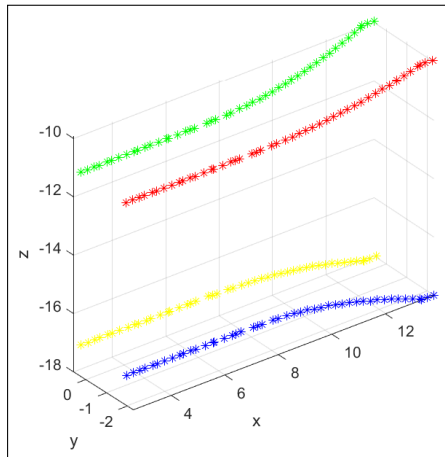
Now that the pier column point cloud is separated in four sides of points, the rest of the script has a similar procedure as for the box girder. As illustrated in 4.11, all of the points within each vertical slice of the pier column segment is now flattened to the start of each slice, placing all of the points from each slice on the same yz-plane. Regression lines are then dragged through the four sides, and intersection points are found where the regression lines intersect.



(c) Cross-sectional view of flattened slices with regression lines forming intersection points

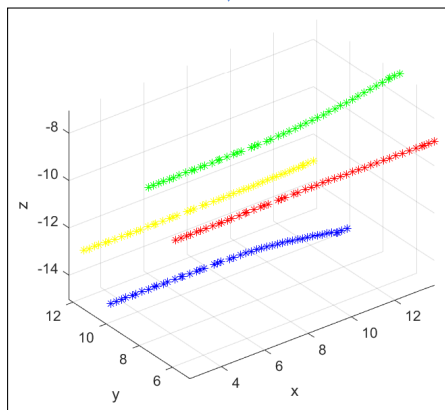
**Figure 4.11:** Illustration of flattening method and regression lines to find intersection points

The intersection points of the pier column can now be seen in 3D space, as visualised in (P8), however, the points must be rotated back to its original position before it can be reverse modelled. This is simply done by multiplying each point with the inverse rotation matrix previously used to rotate the pier to lay flat on the y-axis, and subsequently flip the pier back to an upwards position, seen in (P9-P10).



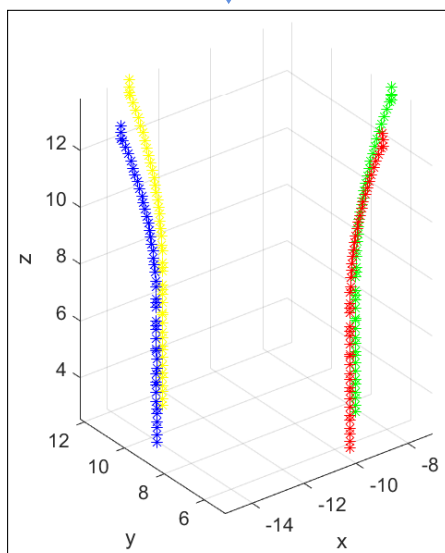
P8:Intersection Points

- 1: **for** Every slice **do**
- 2:   Assign one X-value to all points
- 3:   **for** Every side **do**
- 4:     Make linear regression line between points
- 5:   **end for**
- 6:   Find intersection points at crossing regression lines
- 7: **end for**



P9:Rotated Back

Rotate intersection points back to original position



P10:Flipped Back

Flip intersection points back to original position

This Matlab script is used for all of the Dade Bridge piers. The only part that needs to be adjusted in the algorithm is the removal of the damaged part for each pier column, as some columns are less complete than others, and vice versa.

A solution to automate this process was developed and is presented in the next section, Section 4.2.2.1. However, because of incomplete point cloud scans, this method did not give a satisfactory result for the majority of the piers and was not used.

As described throughout the thesis, the level of noise will affect the feature fitting of the bridge parts, therefore, the noise are carefully removed. For pier column 1, the noise removal was not good enough and some points were causing a disruption in the intersection points. How this was sorted out is explained in Section 4.2.2.2

#### **4.2.2.1 Alternative Cutting Method**

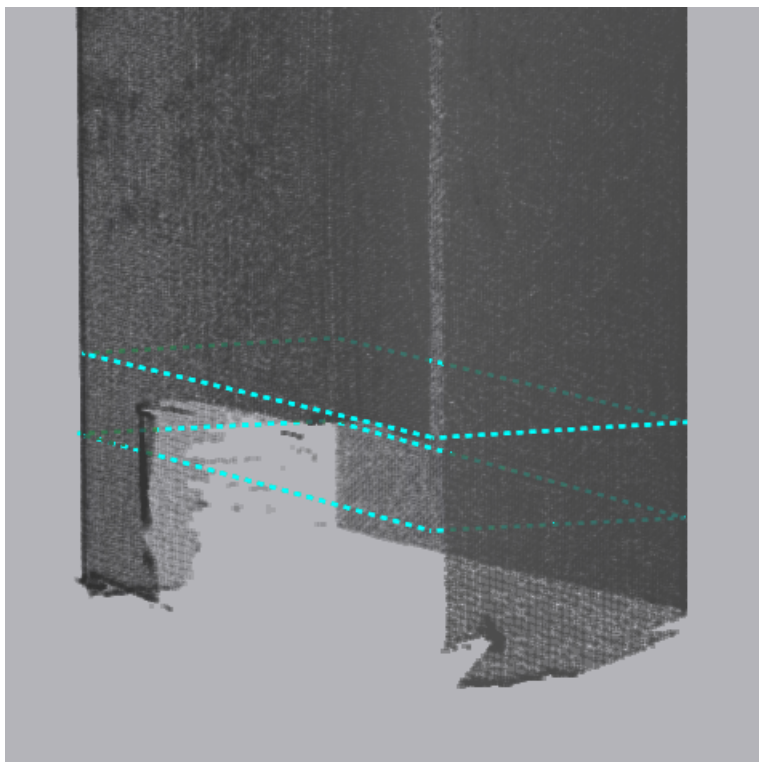
For some piers, the removal of slices could have been done by finding the place where the change in the number of points is at its largest. This method is based on the fact that incomplete, i.e. "damaged", slices will contain considerably less points than that of a complete neighboring slice. This indicates the position where the scan transitions from "damaged" to complete. This method is dependent on what is causing the damage and to what extent the point cloud is damaged.

The scanning of the piers done in this project does not have one clear line for where the damage starts, in fact, many of the piers show damage which is too big to perform an automatic cutting with the method described.

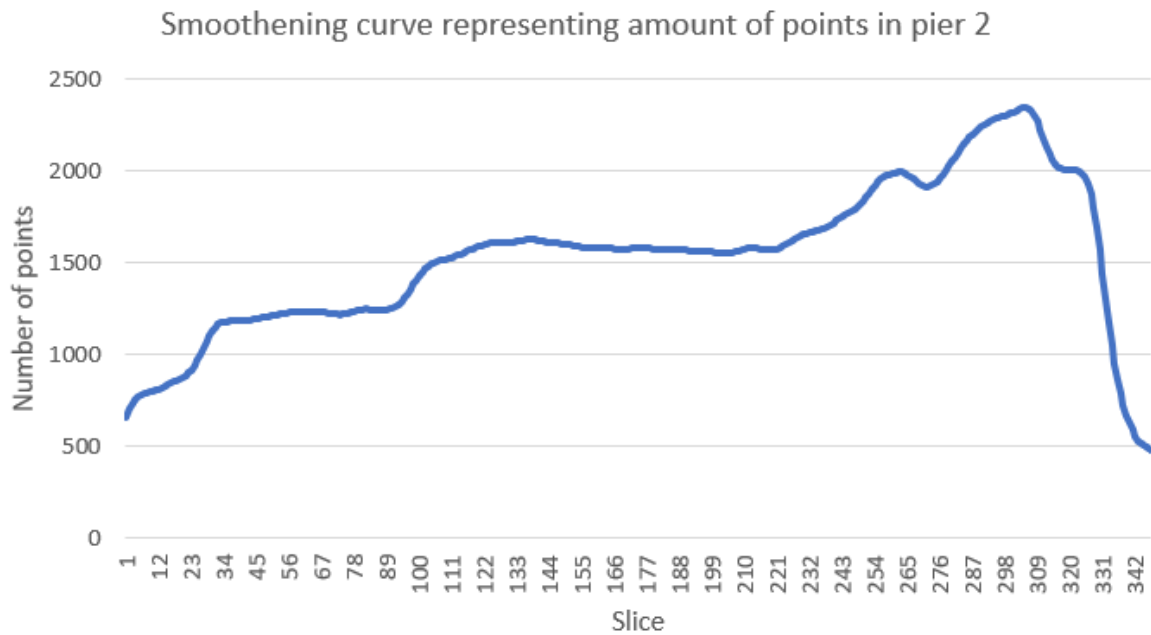
For one pier, pier column 2, the method can be used. The pier only has smaller damage, and the lines caused by the damage are quite straight, forming a clearer change in the point cloud. The two lines in Figure 4.12 show the lines between damage and complete point cloud.

To automate the cutting for this pier, first, the pier is divided into much smaller slices. Every slice is having a great variation in number of points. To smooth out the variation, the average over a number of slices is taken. By doing this several times, the variation shows a quite smooth curve, as seen in Figure 4.13. Now it is possible to see where the biggest leap in the number of points is. A curve that better shows this leap is presented

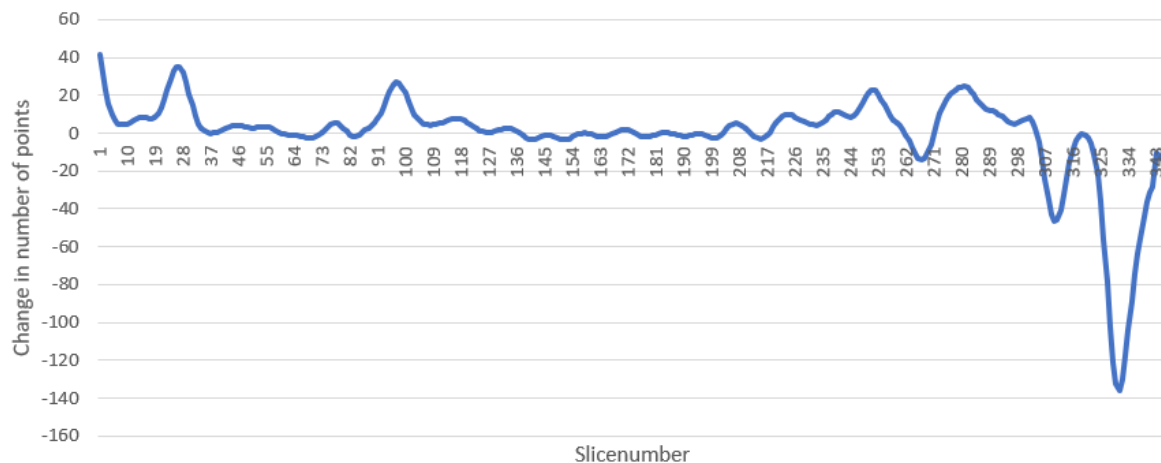
in Figure 4.14. This figure shows the change in the number of points over the entire pier. Two peaks stand out, one at approximately slice 25 and one at slice 100. In addition, there are two dips at the end of the curve, where the biggest is near slice 330 and the next is near slice 310. By cutting at the last peak and the first dip, this method will be cutting the pier at the exact right location.



**Figure 4.12:** Graphical representation of changes in number of points for pier column 2



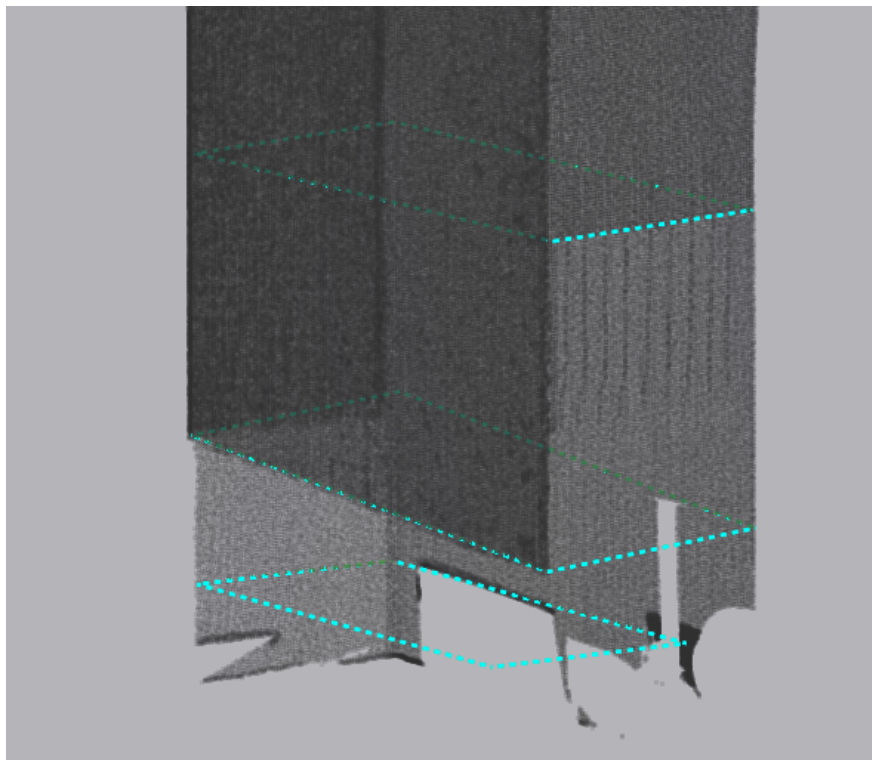
**Figure 4.13:** Development in number of points for Pier 2



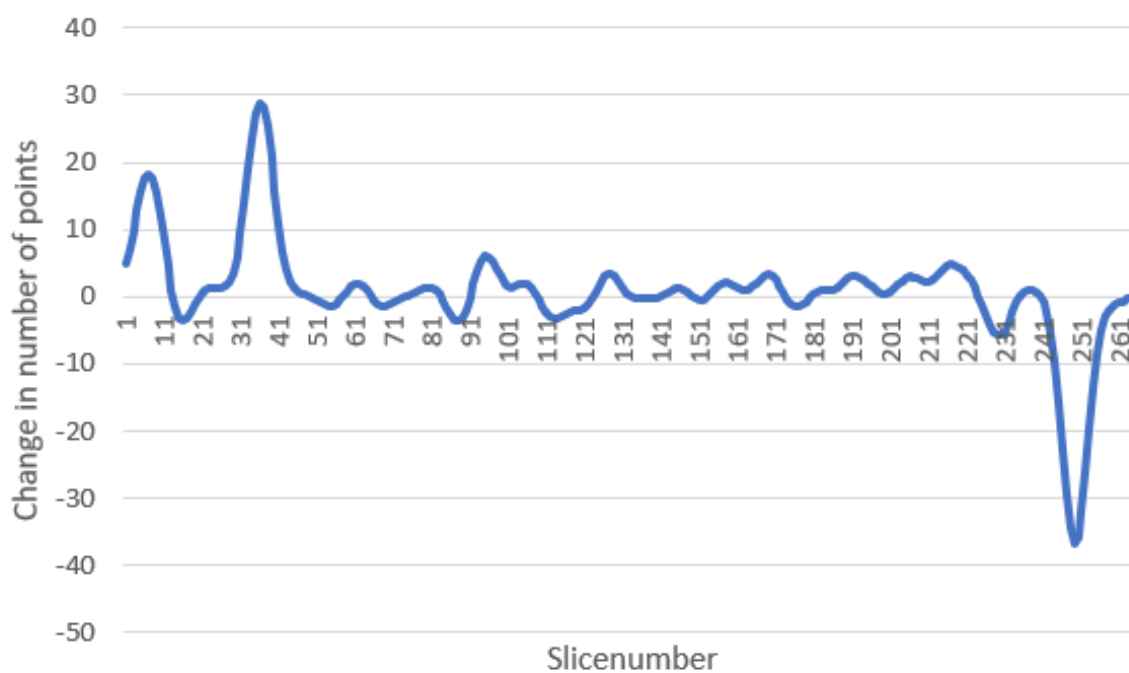
**Figure 4.14:** Change in number of points for Pier 2

If the pier has larger damage, the variation can be harder to find. Figure 4.15 illustrates how the case is for another pier point cloud (pier column 6) in this project. In this figure, there are not two big transition between damaged and complete scan but three, and the place of cutting is not at the biggest difference in points. The curvature of point change for this pier is shown in Figure 4.16. The two first peaks represent the first two lines marked in Figure 4.15. But the third, and correct, place to cut is more difficult to spot. Doing this for all the piers, finding no similarities in curvature between the point cloud

differences and the correct cutting position, the solution is manual cutting, measuring the length of the damaged part in Geomagic and inserting the length-value to Matlab to perform the cut. Almost all of the piers of the Dade Bridge scan needs manual cutting.



**Figure 4.15:** Figure of where the damages on the scan are located on pier column 6



**Figure 4.16:** The curvature of the change in the number of points for pier column 6

With a more complete point cloud of a bridge pier, the proposed method would have given a good result, automatically cutting away the damaged scan pieces.

In any case, the incomplete parts must be removed from the scan data in order to create intersection points to be used for reverse modelling. As with the box girder, the cut-away part must also be modeled by extending the now cut-short pier columns. The reverse modelling method of the pier column extensions is detailed in Section 5.3.2.2.

#### 4.2.2.2 Pier Point Cloud Sampling

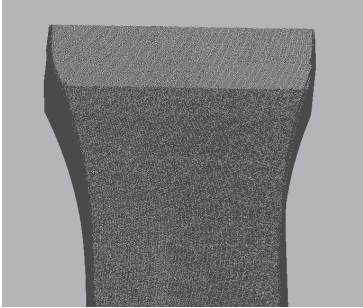
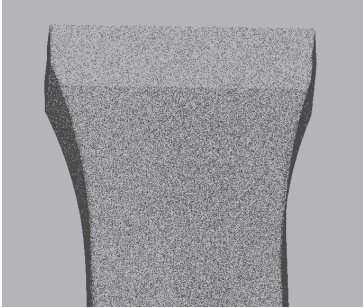
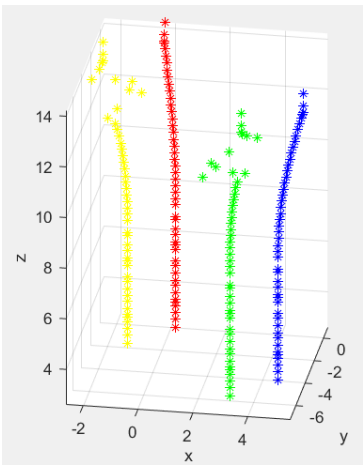
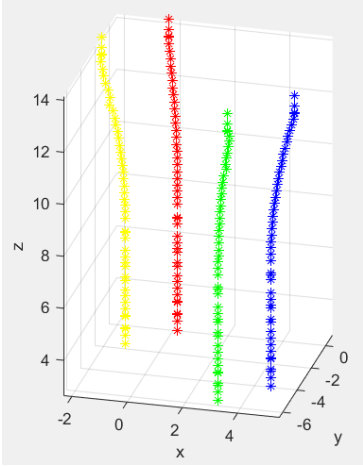
A method of removing noisy data that is not obvious noise, is downsampling. Downsampling decreases the amount of data within a data set, while maintaining the general shape of the data. For point clouds, the data size is reduced and less obvious noisy points can be removed.

Downsampling can also give less accurate data, removing important points. In this project, accuracy is crucial. Thus, downsampling is avoided as much as possible. Precise removal of noise without the use of downsampling, will in most cases give satisfactory results. This is the case for almost all parts of the bridge. However, for the point cloud of pier column 1, inaccurate intersection points were found and downsampling was carried out to counter this.

The original point cloud of pier column 1 shows some intersection points not representative for the pier geometry. Not finding any obvious noise, a downsampling was performed using a sampling ratio of 30%, reducing the number of points from almost 578 thousand to 173 thousand. Geomagic was used to perform the downsampling. Table 4.5 show how the downsampling affect the point cloud and how it changes the feature fitting, i.e. the generated intersection points.



**Table 4.5:** Table showing how the downsampling of pier column 1 affect the point cloud and its feature fitting

	Original point cloud	Downsampled point cloud
Number of points	578.118	173.435
Point cloud		
Intersection points		

### 4.2.3 Characteristic Parameters of Pier Columns

The pier has four intersection points, one at each corner. Every intersection point contains a coordinate with x-,y and z-values.

Because of the bridge height, each pier is sliced many times as can be seen in Table 4.6. Each of the pier columns have a been sliced at every 25 cm. The presented intersection points in Table 4.7 show the coordinates for the intersection points at slice 20 to 33 to present how the data output from the Matlab scripts look. This concept is the same for all of the intersection points at each pier column, and forms the outline of which the pier columns will be reverse modelled from.

**Table 4.6:** Pier overview

	Pier 1	Pier 2	Pier 3	Pier 4	Pier 5	Pier 6
Height (original) (m)	16,92	17,36	9,87	9,87	13,32	13,41
Height (after cutting) (m)	11,50	14,94	8,42	9,58	10,51	8,84
Number of slices	47	60	35	39	43	36

From the values in the table, one can see some consistency.

- The height, Z-axis, gets higher for every slice.
- The Z-coordinate is equal for all intersection points in each slice.
- The X- and Y-coordinates at one intersection point is quite similar for every slice.

**Table 4.7:** Characteristic parameters (x-, y-, z- values [m]) of pier column 2

		Intersection Points			
		1	2	3	4
		<b>1-19</b>			
20	X	-12.56	-8.09	-9.58	-14.04
	Y	11.32	7.44	5.81	9.69
	Z	3.96	3.96	3.96	3.96
21	X	-12.56	-8.09	-9.58	-14.05
	Y	11.31	7.44	5.81	9.69
	Z	4.27	4.27	4.27	4.27
22	X	-12.56	-8.09	-9.58	-14.05
	Y	11.31	7.44	5.81	9.69
	Z	4.54	4.54	4.54	4.54
23	X	-12.57	-8.09	-9.59	-14.05
	Y	11.31	7.44	5.81	9.69
	Z	4.66	4.66	4.66	4.66
24	X	-12.57	-8.09	-9.59	-14.05
	Y	11.30	7.44	5.81	9.69
	Z	4.95	4.95	4.95	4.95
S L I C E	X	-12.57	-8.09	-9.59	-14.05
	Y	11.30	7.44	5.81	9.68
	Z	5.13	5.13	5.13	5.13
26	X	-12.57	-8.09	-9.58	-14.05
	Y	11.30	7.44	5.81	9.68
	Z	5.48	5.48	5.48	5.48
N U M B E R	X	-12.57	-8.09	-9.58	-14.05
	Y	11.30	7.44	5.81	9.68
	Z	5.77	5.77	5.77	5.77
28	X	-12.57	-8.09	-9.58	-14.05
	Y	11.30	7.44	5.81	9.68
	Z	5.82	5.82	5.82	5.82
29	X	-12.57	-8.09	-9.58	-14.05
	Y	11.30	7.44	5.82	9.68
	Z	6.18	6.18	6.18	6.18
30	X	-12.56	-8.09	-9.58	-14.05
	Y	11.30	7.44	5.82	9.68
	Z	6.49	6.49	6.49	6.49
31	X	-12.56	-8.09	-9.58	-14.05
	Y	11.30	7.44	5.82	9.68
	Z	6.63	6.63	6.63	6.63
32	X	-12.56	-8.09	-9.58	-14.05
	Y	11.30	7.45	5.82	9.68
	Z	6.82	6.82	6.82	6.82
33	X	-12.56	-8.09	-9.57	-14.06
	Y	11.30	7.45	5.82	9.68
	Z	7.29	7.29	7.29	7.29
		<b>33-60</b>			

## 4.3 Feature Extraction of Pier Foundation

The feature extraction process for the pier foundation is different than that of the box girder and pier columns. Firstly, as there is only one visible foundation in the Dade Bridge point cloud, segmentation beyond isolating the pier foundation from the rest of the bridge point cloud is not needed as it was in the aforementioned structural elements. Further, the feature fitting is not reliant on intersecting regression lines, as the foundation does not have any intersecting lines. In this section, the feature fitting of the pier foundation will be detailed with illustrations and simplified code, with a subsequent presentation of selected characteristic parameters of the pier foundation. The complete script for pier foundation is seen in Appendix 8.



**Figure 4.17:** The point cloud of the pier foundation

### 4.3.1 Feature Fitting of Point Cloud Segment

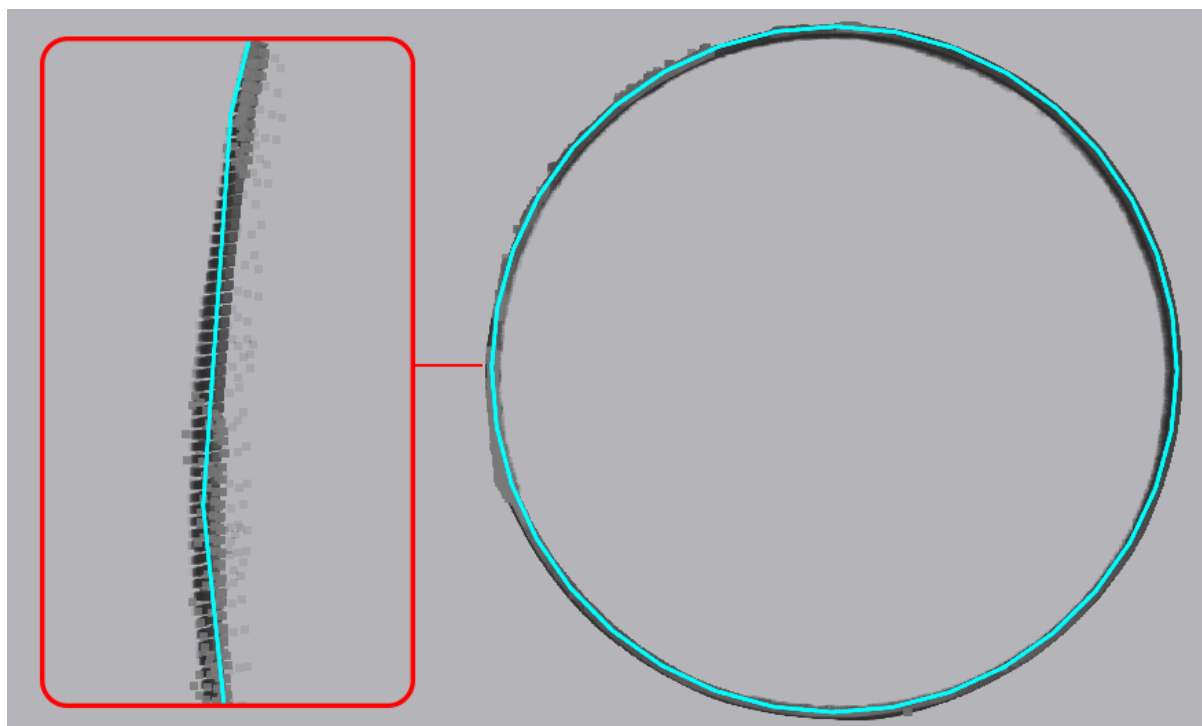
There is only one visible foundation for the Dade bridge. This is under the two midspan piers and is actually used to turn the whole bridge after construction. In that way the construction caused a minimal delay for the heavily trafficked trainline going under it [85].

The foundation has a cylindrical form, and the method of feature fitting presented can be used for all cylindrical structures, such as cylindrical piers.

The process is much simpler than the other bridge parts presented. It uses the least-squares method to form a circle best fitting the points in the xy-plane. The cylinder is segmented in several horizontal slices, and the method is used to find the centre and radius of each

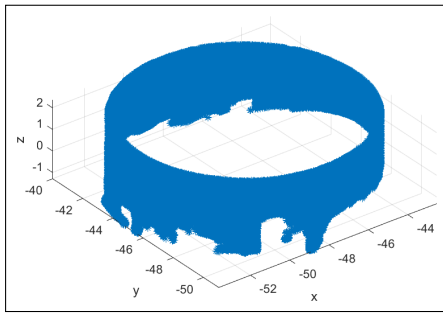
circle around each slice centerpoint [86].

Ordinary least-squares method is an important tool for curve fitting. The method finds the parameters that best minimise the total squared difference between the measured values and the value of the function. Figure 4.18 shows how the fitted circle matches up to the point cloud of the pier foundation.



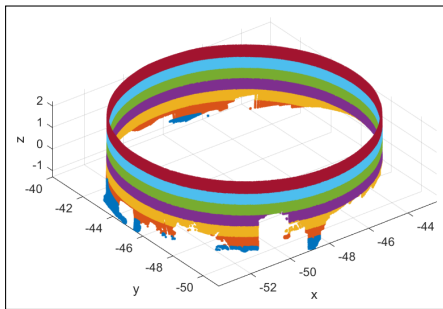
**Figure 4.18:** Least-squares method, fitted circle of the pier foundation point cloud

As shown in (F1-F3), the least-squares method is used on each of the horizontal slices, finding the centerpoint and radius of each slice. Further, the maximum and minimum z-value of the original pier foundation scan is stored to be used in the reverse modelling process later on.



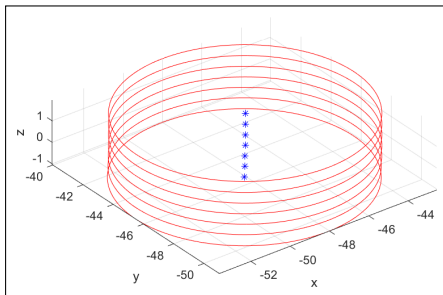
F1: Original

- 1: Read txt-file of point cloud
- 2: Extract 3D coordinates
- 3:  $x=(:,1)$
- 4:  $y=(:,2)$
- 5:  $z=(:,3)$



F2: Slice

- 1: Range in Z-direction= min : 0.5 : max
- 2: **for** Every slice **do**
- 3: find x-values in range
- 4: find corresponding coordinates
- 5: **end for**



F3: Circlefit

- 1: **for** Every slice **do**
- 2: Least square method  $\rightarrow$  Center  $\rightarrow$  Radius
- 3: **end for**
- 4: Find maxZ and minZ

### 4.3.2 Characteristic Parameters of Pier Foundation

The method above gives the characteristic parameters of the pier foundation. The data is shown in Table 4.8 and shows the centrepoint of each slice and its corresponding radius. For the position in z-direction, the max and min value of the original scan data is gathered from Matlab and evenly distributed between the two values, matching the amount of intervals to the amount of slices. How each circle is placed in z-direction is further described in the reverse BIM model section for the pier foundation, Section 5.3.3.

**Table 4.8:** Characteristic parameters [m] of pier foundation

	X	Y	Radius	
	1	-48.09	-45.48	5.52
S	2	-48.08	-45.50	5.51
L	3	-48.06	-45.50	5.52
I	4	-48.06	-45.50	5.51
C	5	-48.06	-45.50	5.51
E	6	-48.06	-45.50	5.51
	7	-48.06	-45.50	5.51

## 5 Reverse BIM Modelling of Bridge Point Cloud Data

In this section, the process of reverse modelling a physical structure - in our case, the Dade bridge - to a digital 3D BIM model is presented. The resulting work in this section is dependent on the data collection, processing and feature extraction detailed in Section 3 and 4. The aforementioned work can be considered to be the input of which the reverse model (output) is based on. In order to end up with a reverse BIM model of high quality, it is crucial that the input is of a high standard.

As the applied technology needed to reverse model a physical structure is fairly new in the AEC-industry, the following sections provides the reader with an introduction to the concept of BIM, some of its applications and the historic development of BIM. Subsequent to this brief introduction, selected BIM software are presented before the process of reverse modelling the Dade Bridge and its immediate terrain is detailed.

### 5.1 Introduction to BIM

In recent years, the concept of BIM and digital construction projects, in general, has experienced an increase in popularity in the AEC-industry. The main reason for this is the rapid development in the field of computer technology, bringing the physical and virtual worlds closer together than ever before.

BIM, or Building Information Modelling, presents a very wide set of possibilities and therefore also a wide variety of definitions. The basic idea of BIM is that different stakeholders work on the same intelligent building information model to add dynamic and intelligent, virtual building components to a model that consists of a large quantity of information about the identity and geometry of the model. Modern construction projects are, more often than not, heavily reliant on multi-disciplinary collaboration work. With a BIM-model, a change from either the architect, engineer or contractor will automatically be processed and updated in the model, meaning that all involved parties can collaborate in real-time on the same construction project. This assures consistency in the model throughout all of its phases, from planning to demolition.



Global trends are consistently making AEC-projects more and more complex, and with BIM technology the industry can increase its efficiency, which in Norway, as mentioned in section 1, has declined the last 20 years. The data integrated in a BIM model defines the design elements and establishes behaviour and relationships between model components. When an element in the model is changed, every connected view is updated with the new change appearing in section, elevation and sheet views. The information in the model can be used to improve the design of a structure before it is built, and in this way retain model intelligence all the way from concept to construction, minimising change orders, reducing coordination issues, providing extra documentation and ultimately contribute to automating the construction process.

BIM provides insights into a design's constructability, but also a better understanding of the constructions future operation and maintenance. Owners can use BIM for predictive maintenance, asset tracking and facilities management, and for future renovation. Working with BIM can result in reduced project risk, improved timelines and cost savings, and better project outcomes.

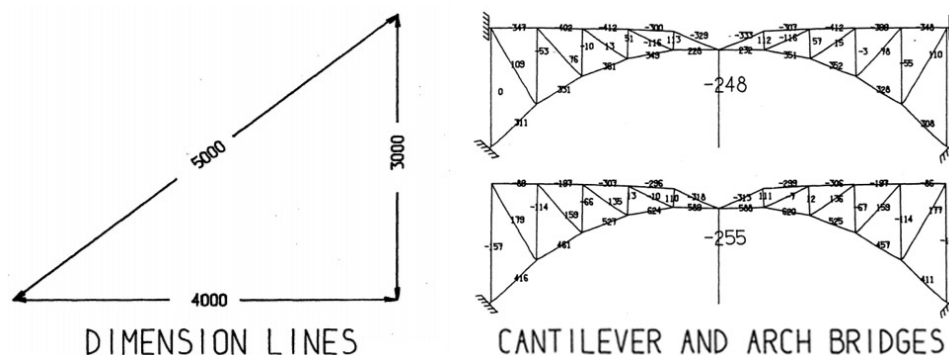
The power of BIM is now growing with the use of cloud-connected technologies, virtual reality, augmented reality, laser scanning, sensors and much more. Although the term "Building Information Modelling" implies that BIM is used for buildings only, it is in fact a general term used for a variety of structures such as roads and bridges. For bridges, some have started using the term BrIM, or Bridge Information Modelling. As with BIM, also BrIM is an important topic in the industry. BrIM is a subcategory of BIM, specialising in the design and construction of bridges. Much of the BIM development is designed for vertical structures like buildings and not horizontal structures such as bridges. Information modelling for bridges has seen an increasing popularity, and the infrastructure industry is constantly developing technology specially designed for bridge structures to increase productivity and cope with tight deadlines and sustainability issues.

As information modelling of bridges is increasing in popularity, the term BrIM is still not a well known abbreviation. Therefore, BIM will be the term used in this thesis, also because reverse modelling as a concept is not limited to bridges, although the method presented in this thesis is designed for box girder bridges.

### 5.1.1 A Brief History of BIM

BIM is a computer-based 3D modelling process that has drastically changed the AEC-industry since its early beginnings in the 1960s. At the time, specialised teams of drafters would meticulously draw two-dimensional construction plans by hand. Naturally, this was a time-consuming process with substantial limitations compared to what would soon become available to the industry due to an increased global digitisation.

The start of computer-aided design, CAD, is often attributed to Ivan Sutherland's graphically interfaced CAD-program from 1963, Sketchpad, which made it possible to create 2D polygons on a graphical surface [12]. As seen in Figure 5.1, Sketchpad made it possible to create drawings with dimensions and execute simple truss calculations. However, due to the expensive hardware required to take use of CAD technology at the time, its use was limited to only a selected few companies such as General Motors and Boeing of the United States. Although the uses of CAD was limited due to its 2D approach with simple geometric capabilities, Sutherland's CAD-software can be seen as the seed of which the modern BIM-process emerged from.



**Figure 5.1:** Dimension lines and simple force distribution of an arch bridge with varied support setups, generated and visualised in Sketchpad [12]

Over the next decade, cost of computers decreased while their processing power steadily increased. Although still in its early phase, CAD experienced a boom in use in the early 1980s when Autodesk released the publicly available CAD-software AutoCAD - originally created for mechanical engineers but quickly adopted by other AEC-branches such as structural engineers and architects. Onward, CAD-software developed quickly and eventually reached somewhat of a plateau in the early 2000s due to limitations within the concept of 2D CAD such as, but not limited to the:

- unintuitive approach of conceptualising 3D models with 2D drawings
- drawings being independent of each other without automatically updating with project changes
- lack of integration between project collaborators
- limited scope of application - 2D CAD does not describe the full life cycle of a structure

As it became apparent that the fundamental approach of CAD was insufficient to evolve with an increasingly digital society with complex projects, the emergence of BIM-technology quickly replaced CAD-software. In the early 2000s, BIM-technology boomed, with Revit being a front-running software for architects, engineers and designers. Not only capable of creating 3D BIM models containing information of individual element properties connected to automatically updating 2D drafting sheets, BIM-software such as Revit introduced collaborative projects and 4D (addition of the dimension of time) capabilities making it possible for all the involved fields to simultaneously plan and track the whole life-cycle of a structure from concept to demolition.

Furthermore, parametric design emerged in the late 2000s, paving the way for an even more efficient modelling method well suited for a construction industry with increasingly complex geometries of large scale. In short, parametric design is a highly customisable design method where complete digital models can be adjusted and optimised dependent on given design goals by tinkering with set input parameters. Although basic parametric capabilities are implemented in BIM-software such as Revit, specialised software for parametric design have the potential to dramatically increase both the geometric complexity of models as well as the modelling efficiency in regard to time spent. A common software of this nature is Dynamo, a visual programming tool which last year (April 2019) became an integrated extension of Revit. The uses of Dynamo and its integration with Revit is further detailed in section 5.2.1.

## 5.2 Introduction to BIM Modelling Software

When it comes to BIM modelling software, there exist a variety of options. Although many of these software has similar areas of application, the functions of each software often differ to some extent, making certain software better suited for certain applications, and vice versa. In this section, the BIM software Revit, and its integration with the visual programming software Dynamo is detailed.

### 5.2.1 Revit + Dynamo

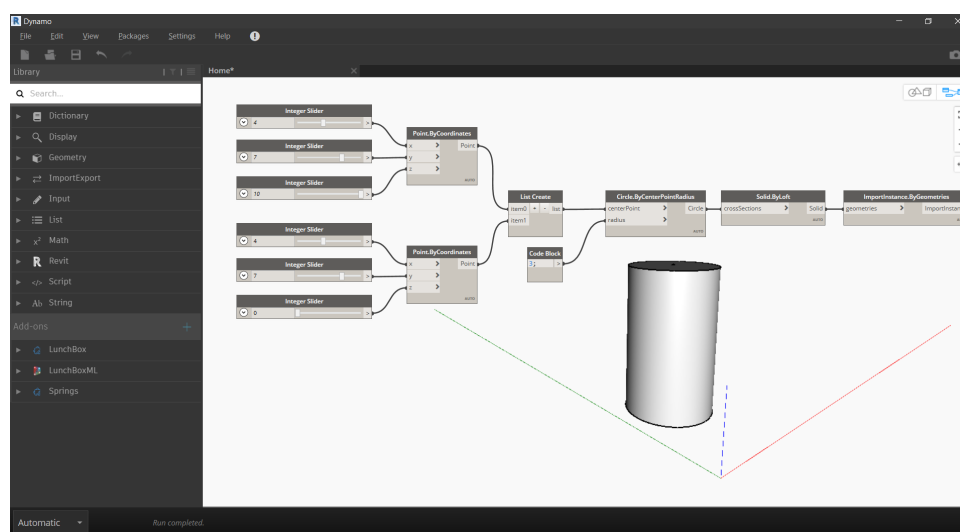
Revit is a Building Information Modelling (BIM) software originally developed by Charles River Software in 1997, bought by Autodesk in 2002. Revit provides the user (mostly designers, architects and structural engineers) with the ability to model structures in 3D, using parametric, object-based elements. In BIM-models made in Revit, it is possible to integrate various parameters to each instance of an element, including but not limited to unit cost, material use, construction phase, placement and volumes. The designer can choose to model in 3D, or create 2D views such as floor plans and section views and model in those instead. A change in one view, changes all views within the project. It is also possible to collaborate online with other users, making it a great software to use in multi-disciplinary projects where it is advantageous for multiple branches such as architects, MEP-engineers and structural engineers at different locations to work together simultaneously. The option to export 3D BIM-models made in Revit, to open formats such as IFC, also contributes to this interdisciplinary interaction.

Although a very powerful BIM software, there are cases where the core functionalities of Revit are insufficient, such as the semi-automatic reverse modelling based on point clouds which is to be done in this thesis. However, Revit supports add-ins which can extend the functionalities of Revit beyond its core applications.

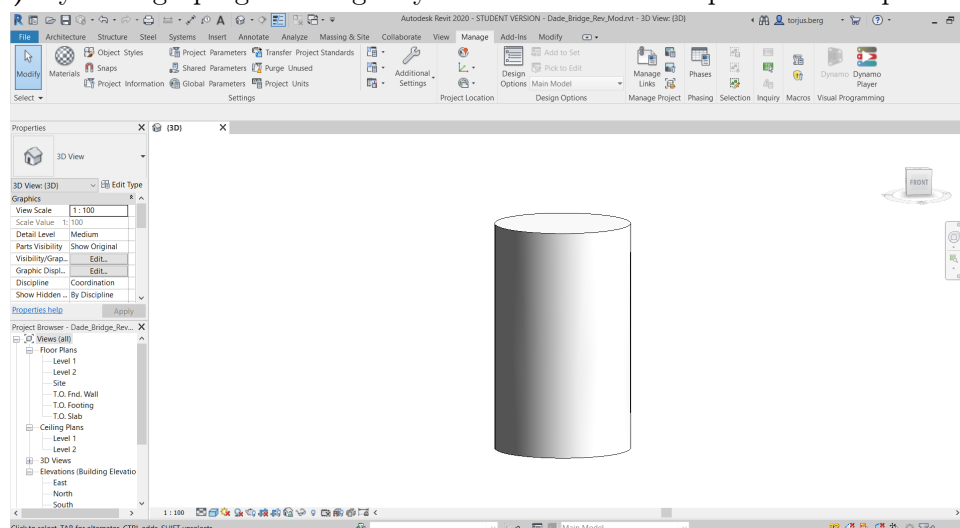
One of these add-ins is Dynamo. Dynamo is an open source graphical algorithm editor, which in April 2019 became an integrated add-in within Revit version 2020. Dynamo is based on visual programming with lines being dragged between nodes, making it an intuitive tool for many users whom are inexperienced with regular programming. The integration between Dynamo and Revit makes it possible to easily select, modify and

transfer elements between the software in ways that stand-alone Revit cannot.

In Figure 5.2, a simple cylindrical column with parametric capabilities is generated using nodes in Dynamo and exported to the Revit workspace as a solid geometry. The nodes which define the characteristic properties of the column, i.e. the bottom centerpoint, the radius and the column height, can be easily adjusted using connected integer sliders, subsequently changing the box component in accordance with the input adjustments. Both the Dynamo- and Revit-workspace is continually updated in sync with the adjustments made to the nodes which together amounts to what is called a Dynamo-graph.



(a) Dynamo-graph generating a cylindrical column with parametric capabilities



(b) Cylindrical column exported to Revit as a solid element

**Figure 5.2:** Modelling process of a simple parametric cylindrical column in Dynamo and Revit

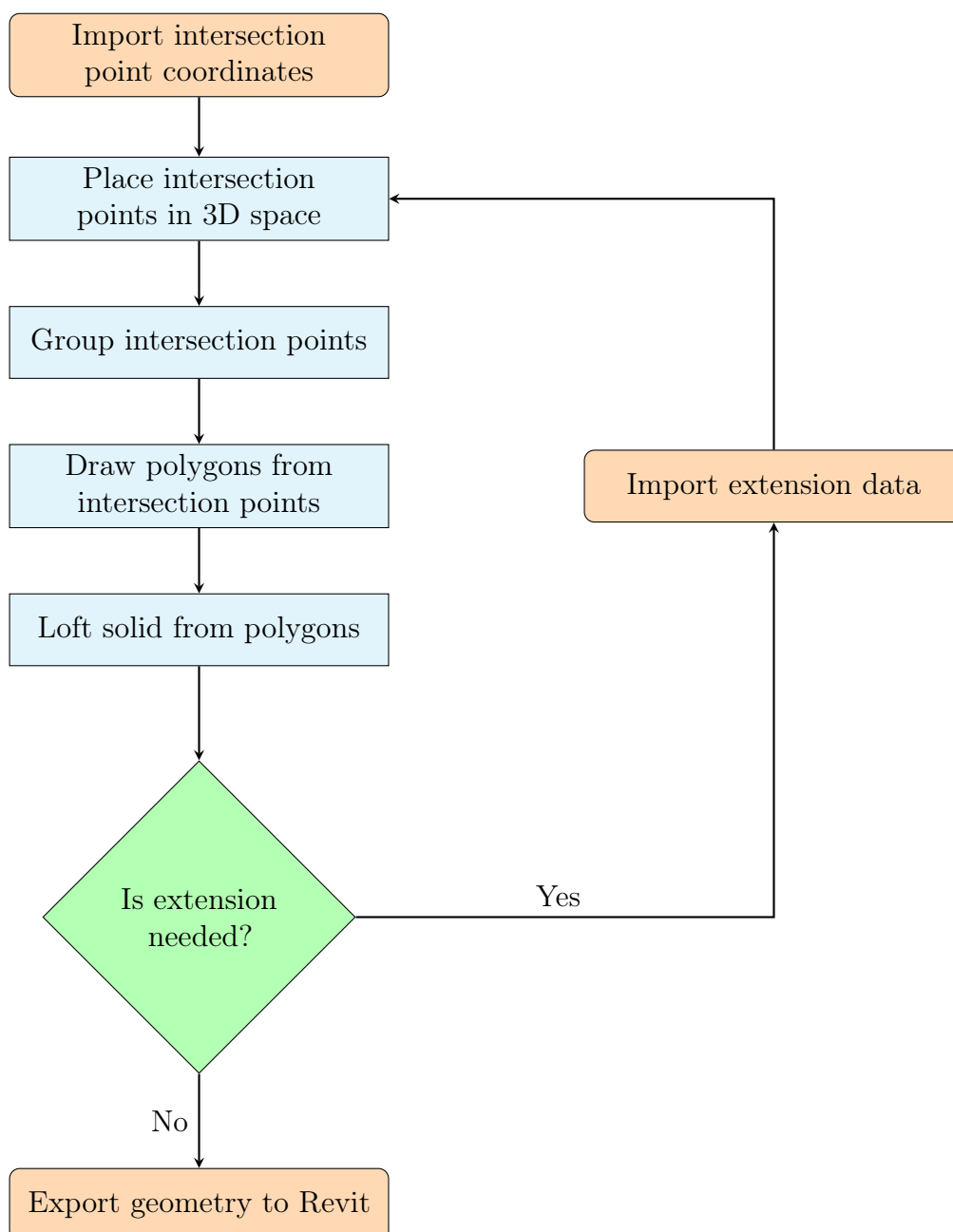
## 5.3 Reverse BIM Modelling

From the calculated intersection points of the box girder, pier columns and pier foundation from Section 4, a reverse BIM model is to be made. The coordinates of the intersection points are collected in matrices within Excel-sheets, and need to be exported to a suited software for reverse modelling. At this point, the 3D coordinates are simple number values which must be further processed and converted to solid 3D elements to accurately represent the geometry of the different bridge parts, i.e. the box girder, pier columns and pier foundation.

As the points must be converted from coordinate values to solid 3D elements, a software which can further process and modify the calculated intersection points is needed. The visual programming interface of Dynamo and its synergy with Revit, as detailed in Section 5.2.1 is well suited for this work, and is thus the software of choice for this task.

The concept of reverse modelling is quite similar for all of the bridge elements. As shown in Figure 5.3, the 3D coordinates of the intersection points, collected in matrices like seen in Table 4.3, 4.7 and 4.8, are first imported in Dynamo. Then, these coordinates are sorted and converted to 3D points in space, which are further grouped into segments along the span length of the element to be made. A polygon is made of each group along the span length, contouring the shape of the element. A solid is then lofted from polygon to polygon, and subsequently exported to Revit where all the individual elements are combined in the same workspace.

In cases where the point cloud of the element is occluded, thus activating the cutting algorithm in Matlab, the solid element in Dynamo has to be extended to represent the original geometry as closely as possible. Although the extension method differ slightly depending on whether the element is a box girder, pier column or a pier foundation, similar for all methods is that extension data is collected with Matlab from the uncut point cloud data and is used to simulate new intersection points at the beginning and end of the original span length of the element. From these new intersection points, polygons are made, and a solid is lofted from the extreme values of the existing solid to the extreme values of the new polygons. The new solid consisting of both the originally generated solid and the extended part of the solid is then exported to Revit.

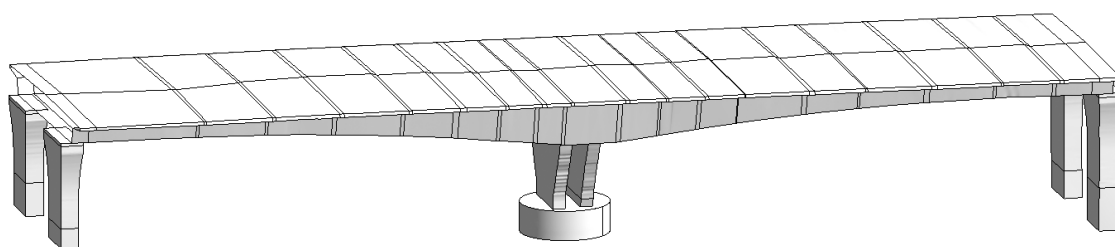


**Figure 5.3:** Reverse modelling workflow

In theory, all of the bridge elements could be reverse modeled within the same Dynamo-script. However, due to limited processing power of the author's PC, and the ongoing lock-down at Oslo Metropolitan University as a preventative measure against the spread of COVID-19 as this thesis was written, the access to better hardware was prevented, and the Dynamo-scripts needed to be split into three files: pier columns and pier foundation in one, box girder 1 to 9 in one, and box girder 10 to 18 in one. Naturally, dividing the script in three parts instead of combining everything in one, will be slightly more

time-consuming to execute as the scripts must be ran one at a time. However, the quality of the end result is not affected by this.

In the following sections, the process of reverse modelling the complete bridge is detailed. Due to slight differences in the method for reverse modelling the box girder, pier columns and pier foundation, individual sections are dedicated to each bridge element category. Firstly, the modelling process for the box girder is detailed, then the pier columns and lastly the pier foundation. Each Dynamo-script is connected to a shared Revit workspace where all of the reverse modelled elements are combined, as shown in Figure 5.4.



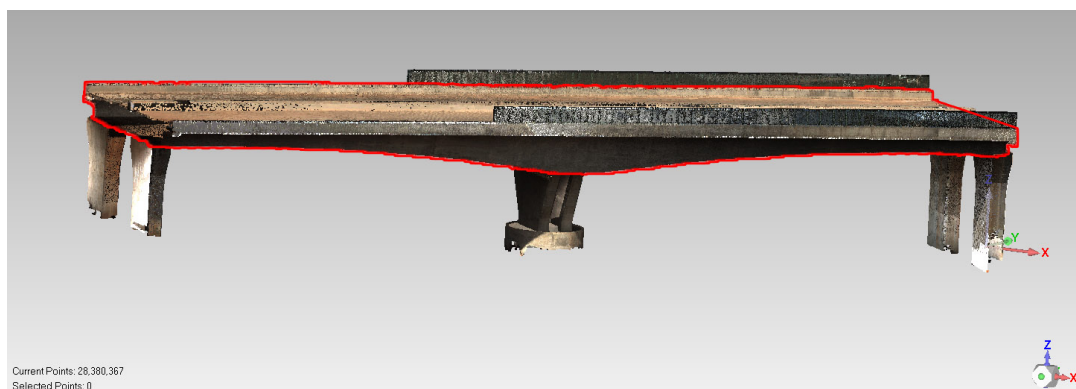
**Figure 5.4:** The finished reverse model of the complete Dade Bridge in Revit

As the Dynamo-graphs are quite large and complex, it is of no use to simply paste the nodes from the Dynamo-graphs into the thesis. Instead, to illustrate the work done in Dynamo, a simplified conceptual algorithm for each reverse modelling method is presented throughout the text. For the complete, original Dynamo-graphs for each reverse modelling process, see Appendix B.

### 5.3.1 Reverse BIM Model - Box Girder

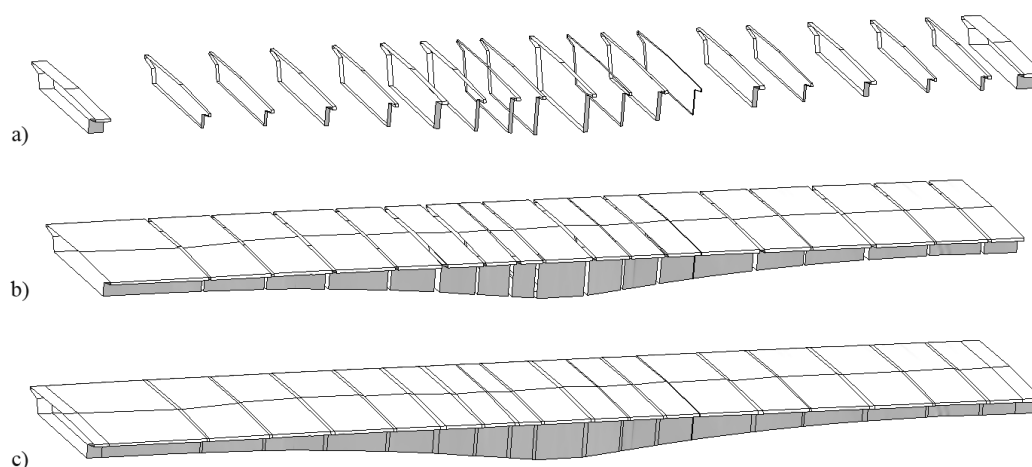
The box girder, as outlined in Figure 5.5, was previously split into 18 segments along its span length. From each of the 18 boxes, intersection points were calculated in Matlab and exported to the Dynamo workspace. Even though the Dynamo-scripts for the box girders are split in two, from box girder segment 1 to segment 9 and box girder segment 10 to segment 18, the concept remains the same for both scripts, i.e. the modelling method for box girder segment 5 is the same as the method for box girder segment 15.





**Figure 5.5:** The point cloud of the Dade Bridge with its box girder outlined in red

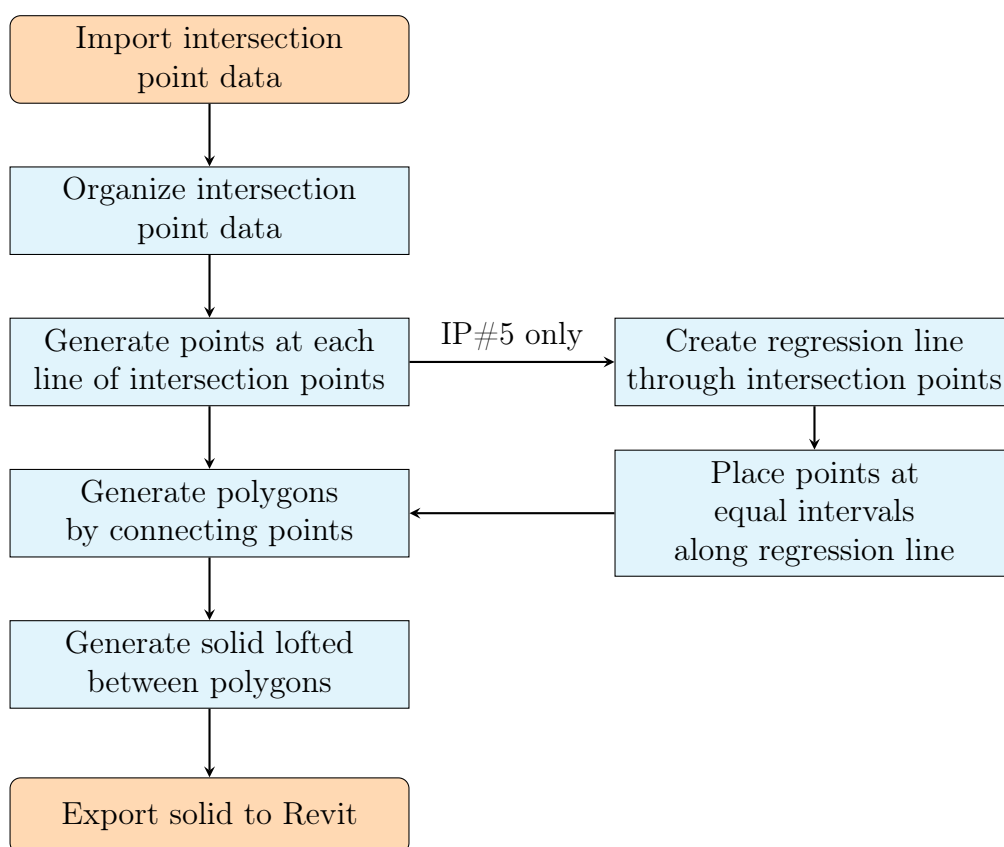
Due to the flattening algorithm which was executed in Matlab (read more about this in Section 4), the data gathered from Matlab is missing a small part of the point cloud at the end of each box girder segment, creating an empty gap between each segment. Further, the cutting algorithm in Section 4 cuts off the start of the point cloud for box girder segment 1, and the end of box girder segment 18. The reverse modelling process can thus be divided into two main tasks. First, reverse modelling the main parts from the intersection points found in Matlab, then, filling in extensions in between the box girders as well as the start and end where the Matlab algorithm cuts the point cloud due to lack of quality. The generated reverse model of the box girders from the intersection points, the infilling extensions and the combined reverse model of the box girder with infillings and extensions, is shown in Figure 5.6. In the following section, the reverse modelling processes for the box girder and its extensions are detailed.



**Figure 5.6:** Screenshot from Revit of the a) extensions, b) box girder segments and c) combined reverse model of the complete box girder

### 5.3.1.1 Main Part

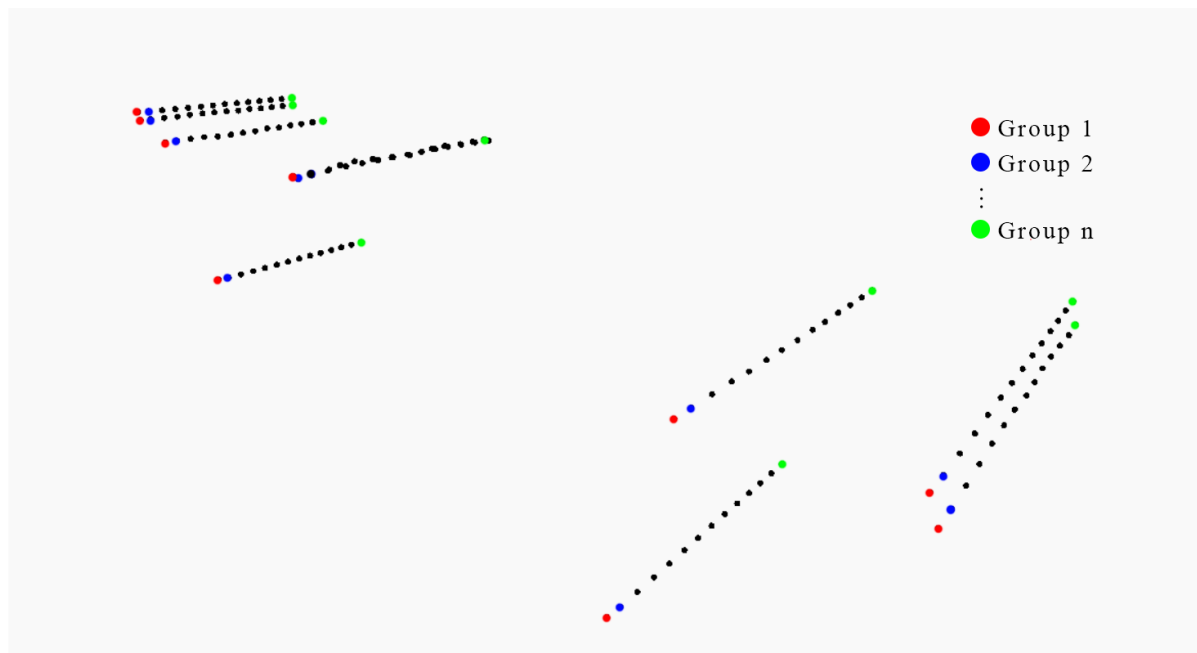
The simplified conceptual algorithm for the main part of each box girder segment is presented in Figure 5.7. The algorithm is applicable to all of the box girder segments, and is for all practical purposes a simple copy-paste from segment to segment. The complete Dynamo-script for the box girders can be found in Appendix B1, while some key details of the script crucial for the reverse modelling of the box girders, such as the special case of the intersection point line 5, will be presented in further detail throughout this section.



**Figure 5.7:** Simplified conceptual algorithm for the box girder segments

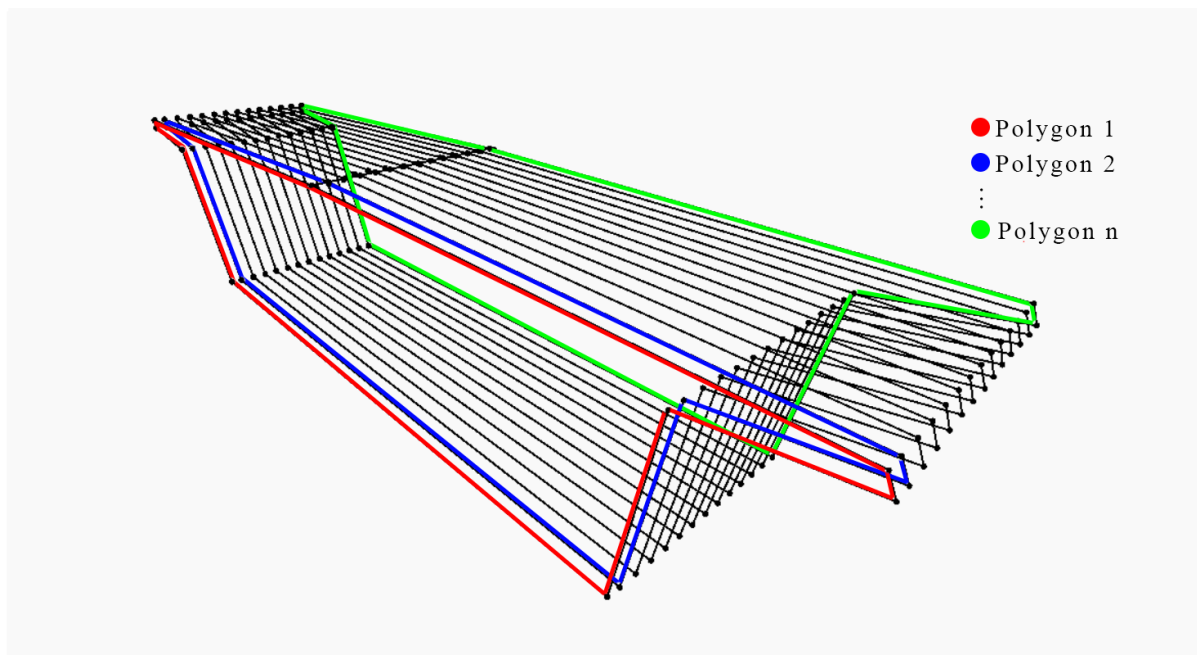
The general outline of the box girders are first seen when importing the Excel-sheets filled with the intersection point coordinates for each box girder. Each point is represented in the sheets by a row with an x-, y- and z-value, which the Dynamo graph gathers and creates 3D points in space from, based on the given coordinates. These points are subsequently sorted into groups of 9 in the vertical plane along the span length of the box girder, as illustrated in Figure 5.8. Since the size of the box girders varies from segment to segment, the Dynamo-graph is generalised to automatically account for differing amount of intersection points, and consequently organises the points into the correct amount of

groups based on the length of the box girder. As can be seen in the Dynamo-graph in Appendix B1, the script can create everything from 1 to thousands upon thousands of point groups, depending on the imported intersection point data size.

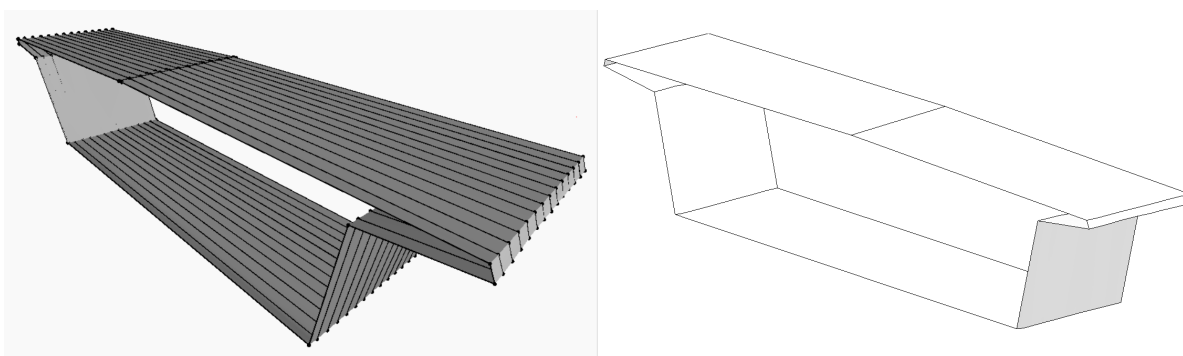


**Figure 5.8:** Regrouped intersection points of box girder 9, from Dynamo

After the intersection points are grouped, the Dynamo-graph generates closed polygons from each group of intersection points along the box girder, as illustrated in Figure 5.9. A solid is then lofted from polygon to polygon within each box girder, and eventually exported to the Revit-workspace as a family instance, as illustrated in Figure 5.10. This process is done for each box girder, and each girder family is named in accordance to its segment, i.e. the box girder generated from segment 9, is named "Box Girder 9", 10 is named "Box Girder 10" and so on. Important to note, is that even though the reverse model of the box girder appears to be shell structures due to the openings at each side, as seen in Figure 5.10, each box girder segment is actually modeled as a solid, and both exported and read as solids in Revit.



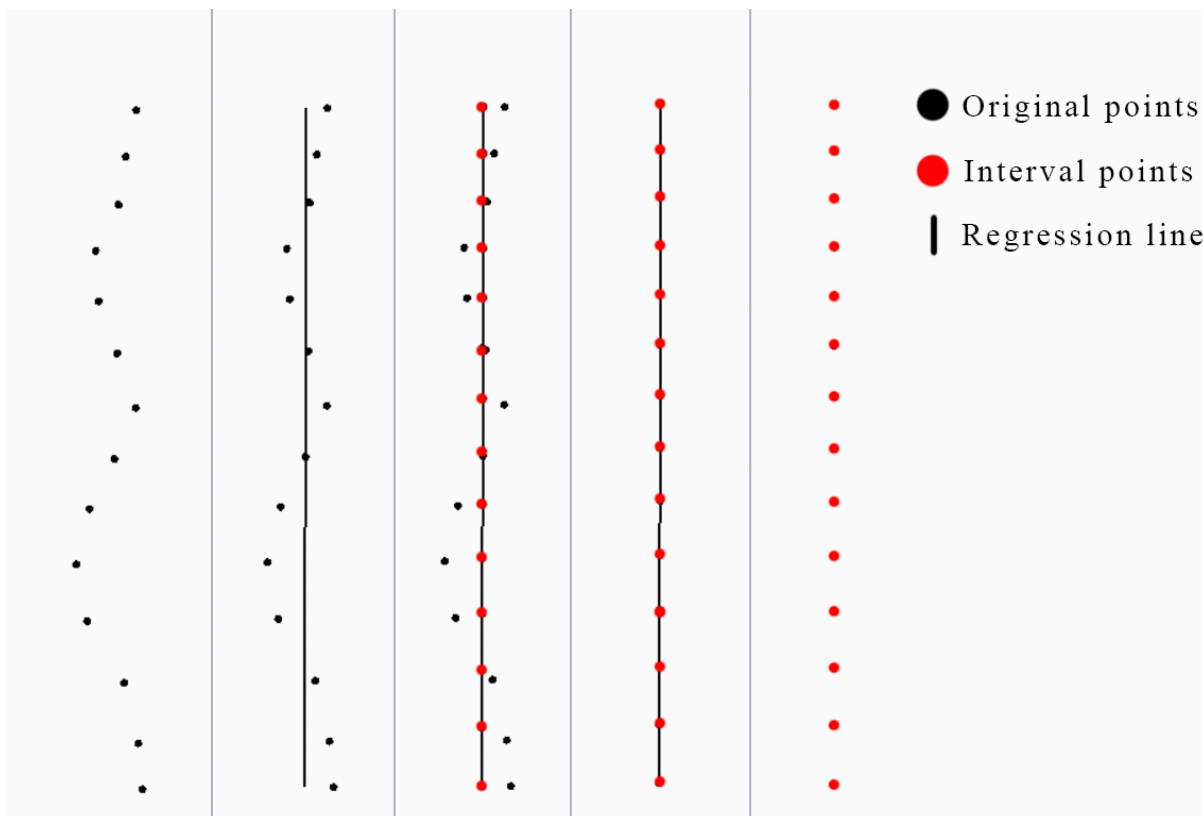
**Figure 5.9:** Closed polygons of box girder 9, from Dynamo



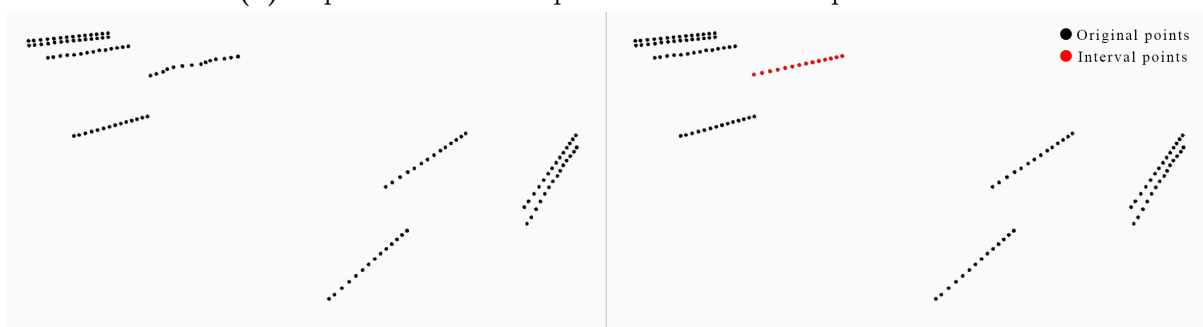
**Figure 5.10:** The solid of the reverse model of box girder 9, visualised in Dynamo (left) and exported to Revit (right)

Although the Dynamo-graph is quite straight forward for most of the intersection points, at intersection point line 5, i.e. the intersection points marking the road crown, varies considerably in the xy-plane due to the slope from the road centerline to the road edge (see more in Section 4.1.3). To even out the would-be jagged line between each of these intersection points, a regression line from the first coordinate to the last coordinate of the intersection point line 5-group is made, as illustrated in Figure 5.11. New 3D points are then placed at equal intervals along this line, matching the amount of intersection points of each box girder segment. This is an important step to generalise the script to work for all box girder segments, as the box girder segments differ in size, meaning that the amount of intersection points also vary from segment to segment. The created

algorithm is dependant on the generation of polygons along the span length of each segment, making it crucial to match the amount of new 3D points along the regression line with the original amount of intersection points. If the amount of interval points does not equal the imported intersection points, it is impossible to generate the polygons of which the solids are eventually lofted from.



(a) Top-view of interval points at intersection point line 5

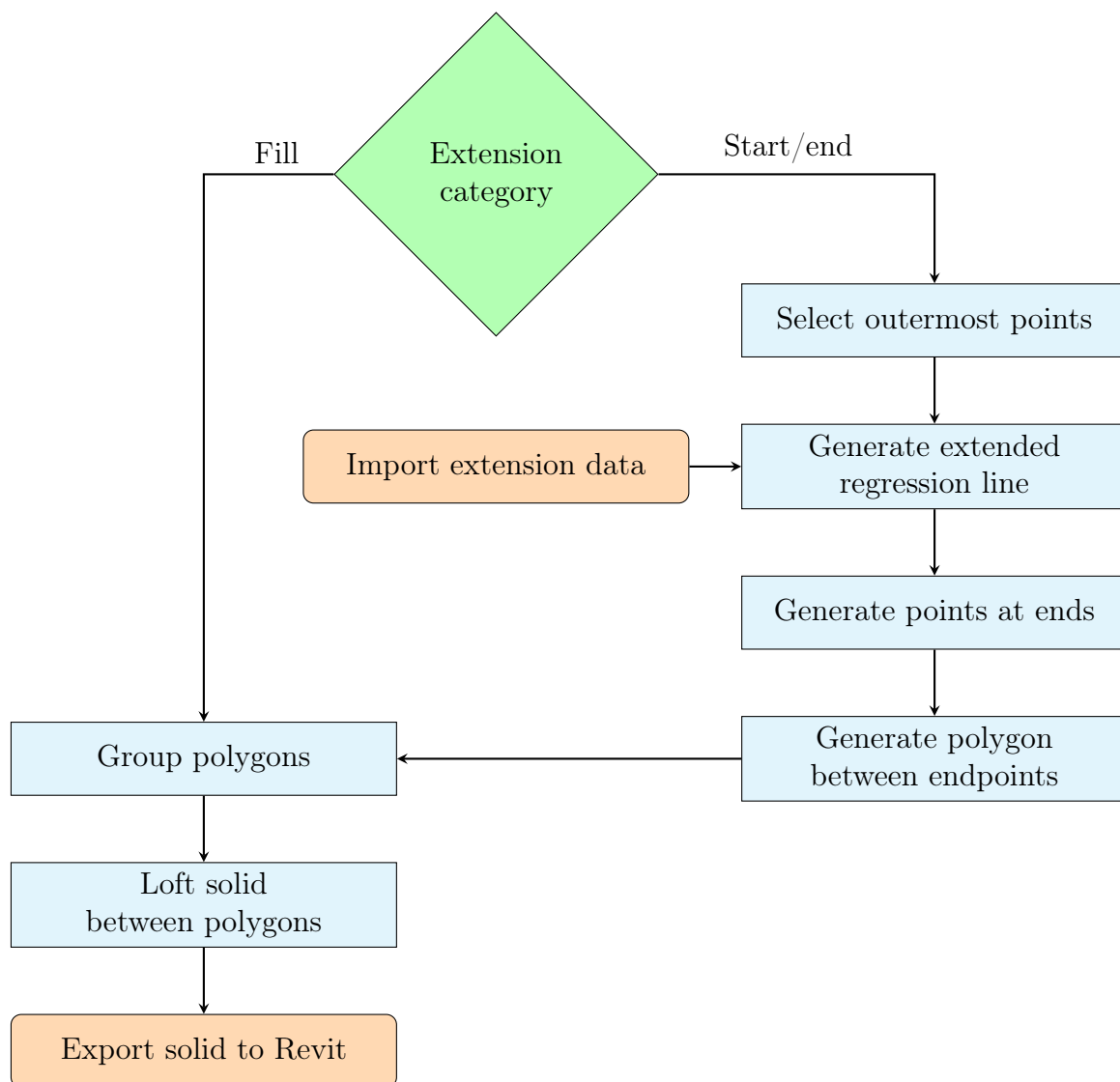


(b) 3D view with difference between original points (left) and added interval points (right) at intersection point line 5

**Figure 5.11:** Interval point distribution at intersection point line 5, from Dynamo

### 5.3.1.2 Extensions

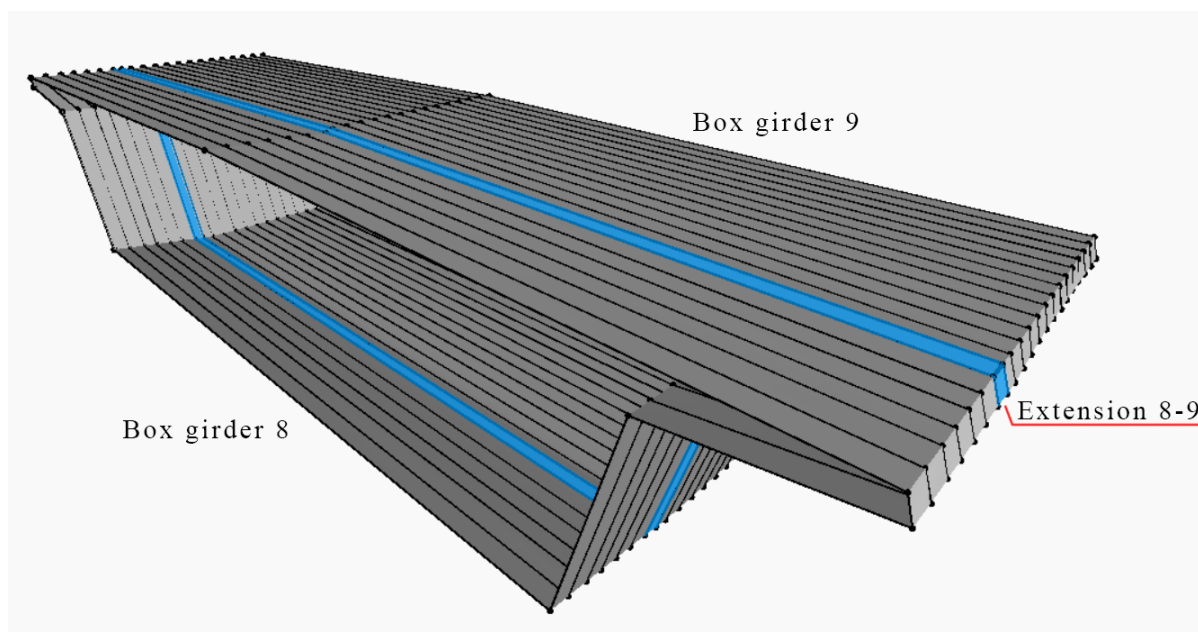
The simplified conceptual algorithm for the box girder extensions is presented in Figure 5.12. The created Dynamo-graph is integrated within the graph for the main part of the box girder segments, and can be seen in its complete state in Appendix B1. The modelling process for the extensions varies somewhat depending on whether an infilling extension or a start- or end-extension is to be modeled, all of which will be detailed within this section.



**Figure 5.12:** Simplified conceptual algorithm for the box girder extensions

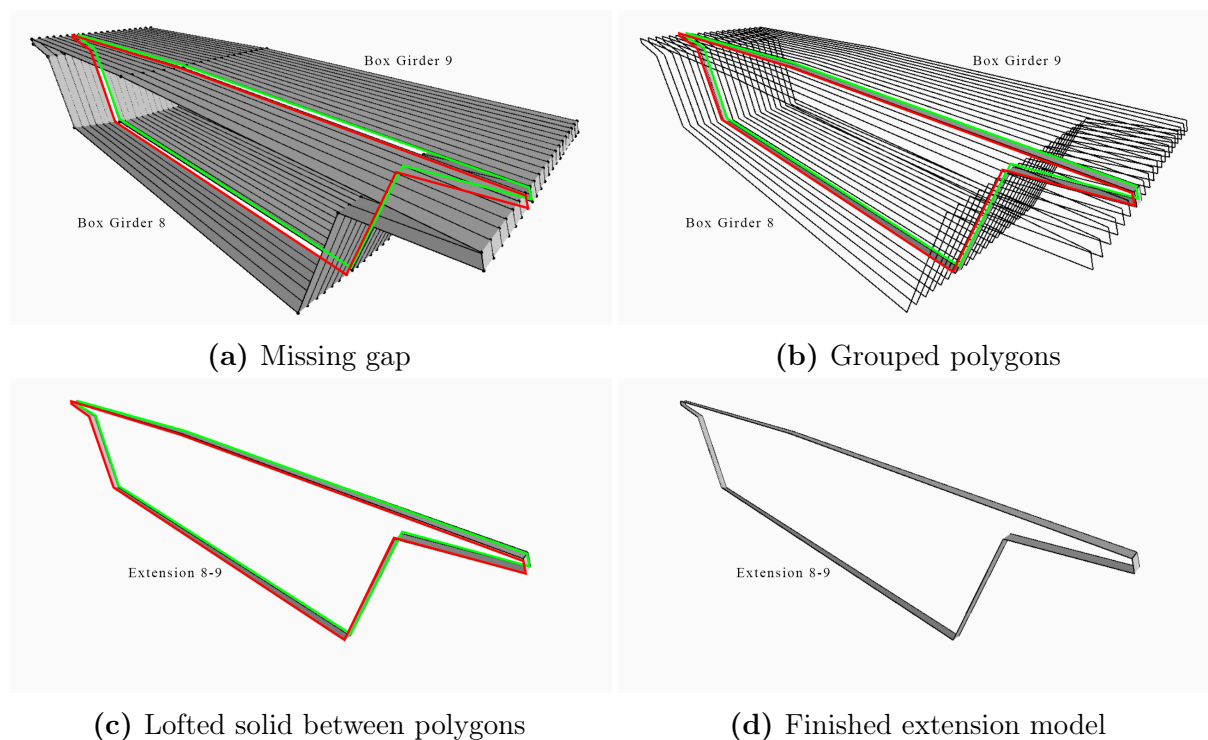
### Fill Extensions

Up until this point, only the main part of the box girders have been modeled. Due to the flattening of each segment slice, the last intersection point of one box girder does not equal the first intersection point of the next box girder. Consequently, small gaps are found in between the main part of each box girder, all of which have to be filled, as shown in Figure 5.13.



**Figure 5.13:** Highlighted extension between box girder 8 and 9, visualised in Dynamo

To fill these gaps, the Dynamo-graph collects the last polygon of one box girder and the first polygon of the next box girder into a combined list, as illustrated in Figure 5.14. From these polygons, a solid is lofted and subsequently named and exported to the shared Revit-workspace in the same manner as previously done with the main part of the box girders.



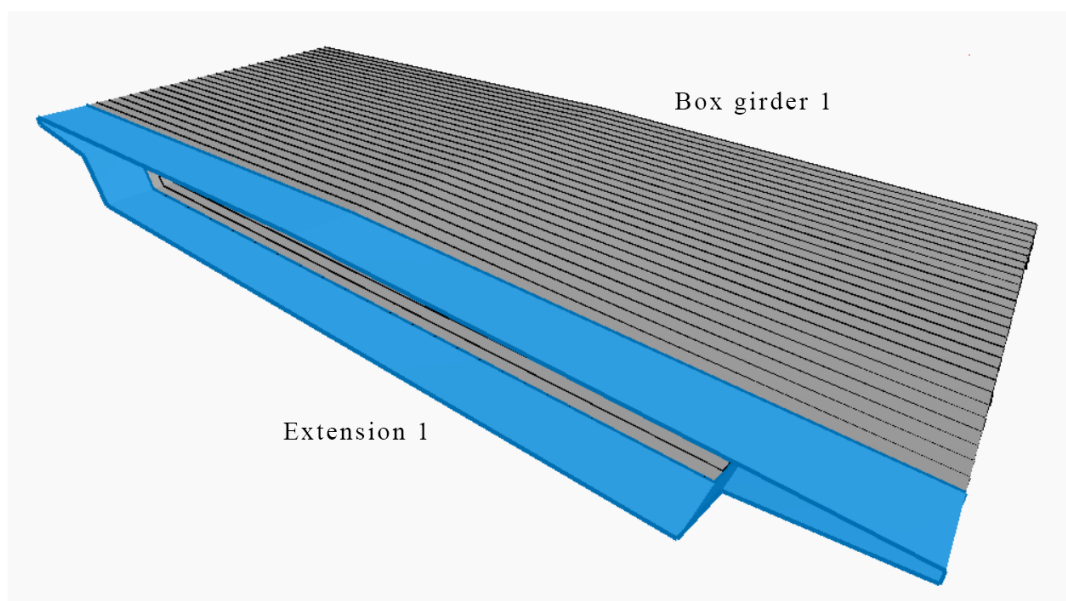
**Figure 5.14:** Modelling process for extension between box girder 8 and 9, visualised in Dynamo

As the Dynamo-graphs are divided into two files of box girder segments 1-9 and box girder segments 10-18, a small addition to the script for box girder segments 1-9 has to be made in order to generate the extension between box girder segment 9 and box girder segment 10. The intersection points of box girder segment 10 are imported to the Dynamo-file for segments 1-9, however, the main part of box girder 10 is not modeled as a solid. Only its first polygon is collected, and afterwards put into a list with the last polygon of box girder 9. A solid is then lofted between these two polygons, creating the infilling extension between box girder segment 9 and 10.

### Start/end Extensions

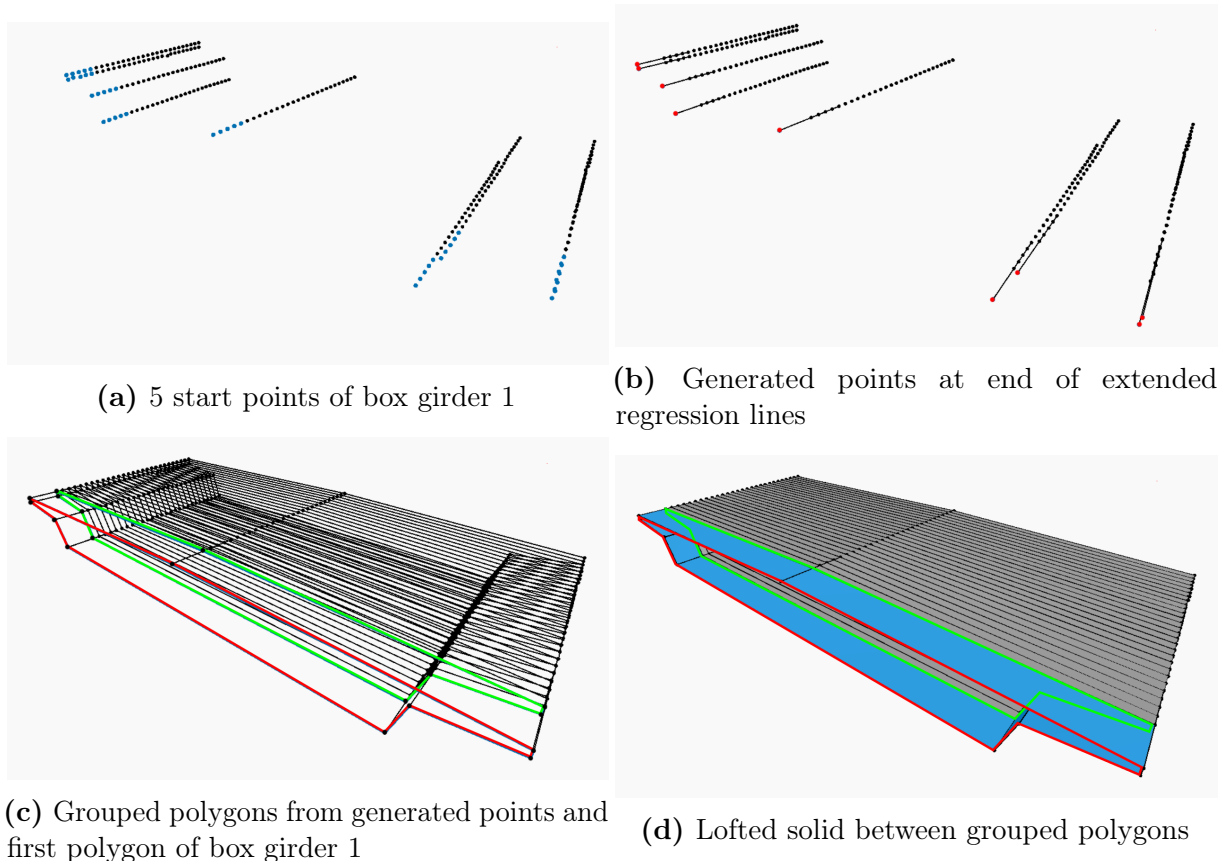
Box girder segment 1 and 18 was previously cut off due to insufficient manual cutting . Generating extensions, as highlighted in Figure 5.15, for these box girders is therefore necessary in order to accurately portray the real Dade Bridge. For the extensions of box girder segment 1 and 18, i.e. the first and last box girder segment, a different modelling method is applied. As there are no adjacent box girders to gather extension data from, the data must be collected from the original scanned 3D point cloud, before the cutting algorithm is executed in Matlab.



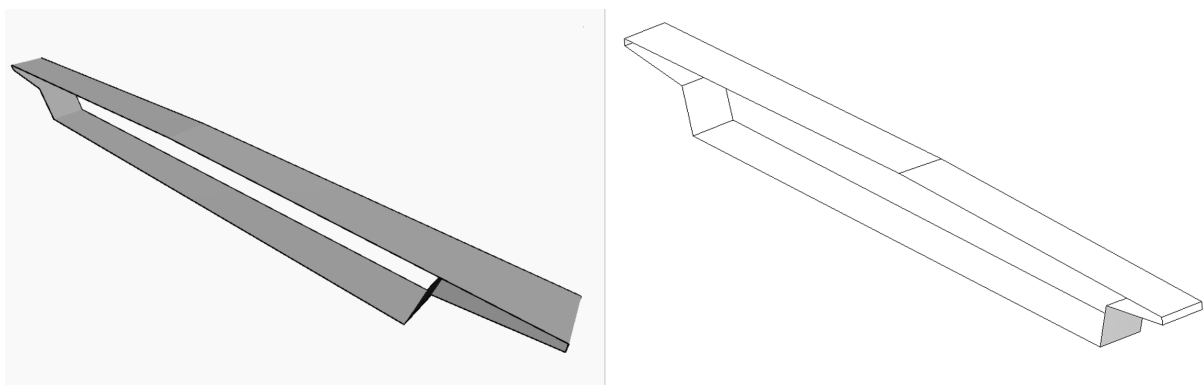


**Figure 5.15:** Highlighted extension at start of box girder 1

In order to extend the model as accurately as possible, while also replicating the shape of the box girder to represent its actual condition, 9 new points are generated in the Dynamo-graph based on the existing groups of intersection points. As illustrated in Figure 5.16, a straight regression line is generated based on the 5 points closest to the extension to be made. This is done for all of the 9 intersection points. The lines are then extended by the distance between the outermost point of the original point cloud, and the outermost point of the imported intersection points. This value is calculated in the Matlab-script for both box girder 1 as well as box girder 18 from Section 4. The formula collects the extreme x-value of the original, uncut and rotated box girder point cloud (maximum or minimum, depending on whether the box girder is for segment 1 or 18). The absolute value of the difference between this extreme value, and the closest x-value of intersection point 1, is then exported to an Excel-sheet which in turn is imported into the Dynamo-graph and used as the distance of which the generated regression lines are extended. A point is then placed on the end of the 9 generated regression lines, and subsequently grouped. From this group of points, a polygon is generated. A list is then created consisting of the generated polygon and its closest polygon (either the first polygon of box girder segment 1, or the last of 18). A solid is lofted between these two polygons, named and exported to the Revit-workspace as a family instance. This is done for both the start- and end-extensions, and ends up looking like shown in Figure 5.17.



**Figure 5.16:** Modelling process for extension before box girder 1 and after box girder 18



**Figure 5.17:** The solid of the finished extension model at start of box girder 1, visualised in Dynamo (left) and Revit (right)

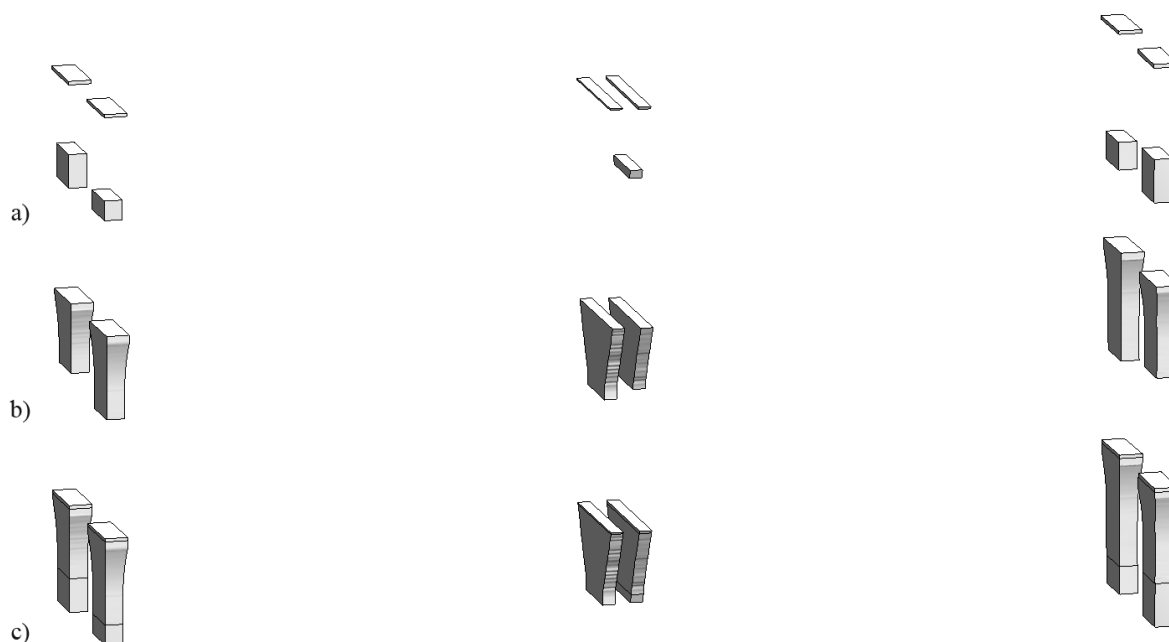
### 5.3.2 Reverse BIM Model - Pier Columns

Figure 5.18 outlines the six pier columns which are to be reverse modelled with the use of Dynamo graphs in this section. A similar concept to that of the reverse modelling for the box girders is also applied for the pier columns.



**Figure 5.18:** The point cloud of the Dade Bridge with its pier columns outlined in red

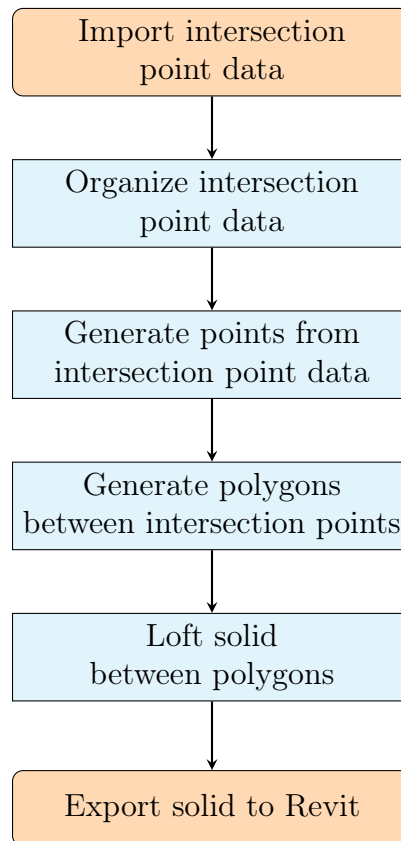
As with the box girder segments, also the pier columns are missing some data necessary to accurately portray the actual Dade Bridge in its as-built state. In Section 4, the point cloud data for the pier columns were cut due to occlusion causing incomplete point clouds of the piers. For this reason, also the reverse modelling process for the pier columns is split into two main graphs; the main part, and the top- and bottom- extensions due to cutting. The finished reverse model of the pier columns which are to be modeled in this section, along with an exploded view of the main part and the corresponding extensions, is shown in Figure 5.19.



**Figure 5.19:** Screenshot from Revit of the a) extensions, b) cut pier columns and c) combined reverse model of the complete pier columns

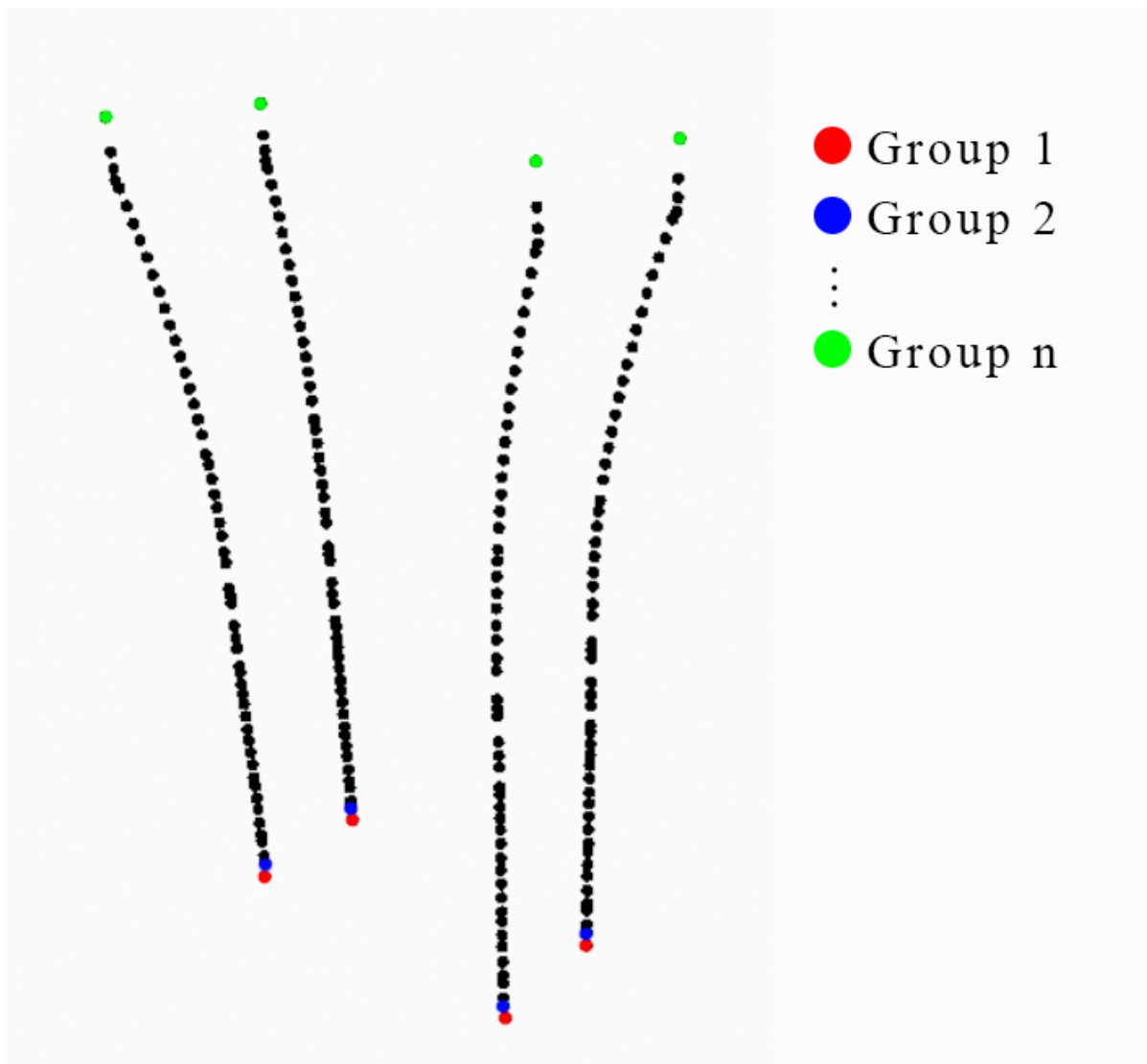
### 5.3.2.1 Main Part

The simplified conceptual algorithm for the main part of the pier columns is presented in Figure 5.20 below. The Dynamo-graph, as seen in full in Appendix B2, is applicable for all of the pier columns, and can in theory be copied and pasted as many times as necessary depending on the amount of four-sided pier columns to be modeled. The following section details the reverse modelling process of the main part of the pier columns.



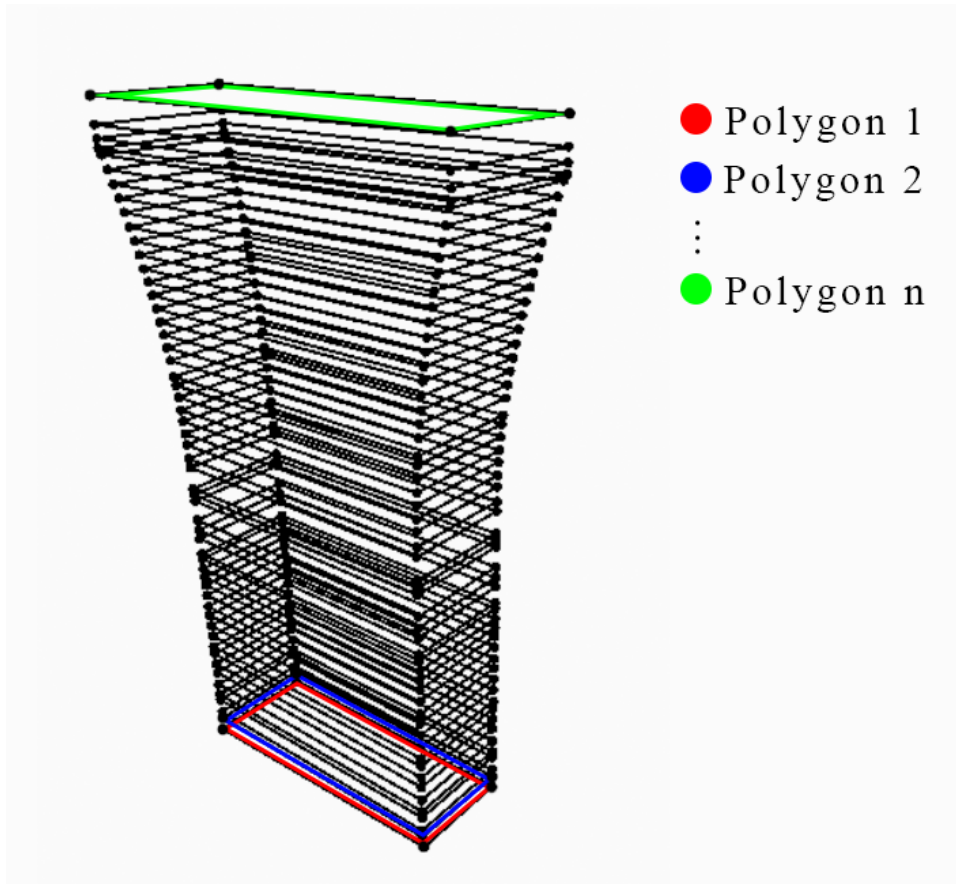
**Figure 5.20:** Simplified conceptual algorithm for the main part of the pier columns

As with the main part of the box girders, firstly, the Excel-sheets containing the coordinates of the calculated intersection points from Section 4 are imported into the Dynamo workspace. The pier columns are quite similar in shape, i.e. they are all four sided made up of four lines of intersection points, meaning that the graph for one pier column can be used for all of the six pier columns, the only difference being the imported input data. As illustrated in Figure 5.21, from the imported coordinates, 3D points in space are generated and grouped in the xy-plane along the z-axis, outlining the piers which are to be generated.

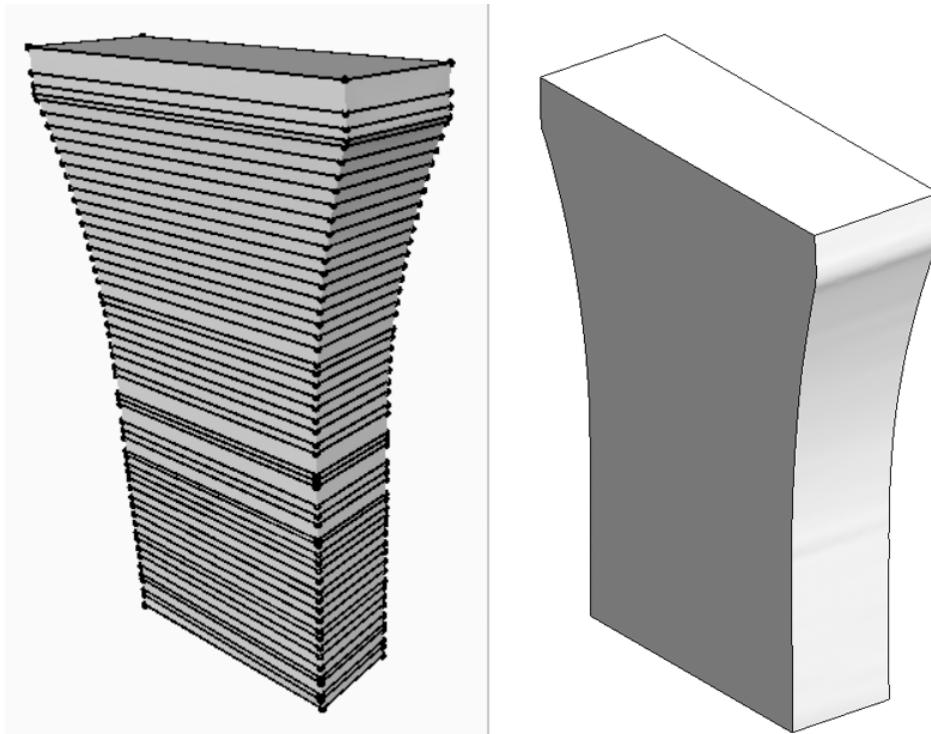


**Figure 5.21:** Regrouped intersection points of pier 2, from Dynamo

Subsequently, as seen in Figure 5.22, polygons connecting the points within each of the aforementioned groups are generated. A solid is then lofted from polygon to polygon along the z-axis, forming the main part of the pier column. Finally, the solid is named, a family is created and the generated model is exported to the Revit-workspace at its corresponding coordinates.



(a) Closed polygons of pier 2, visualised in Dynamo

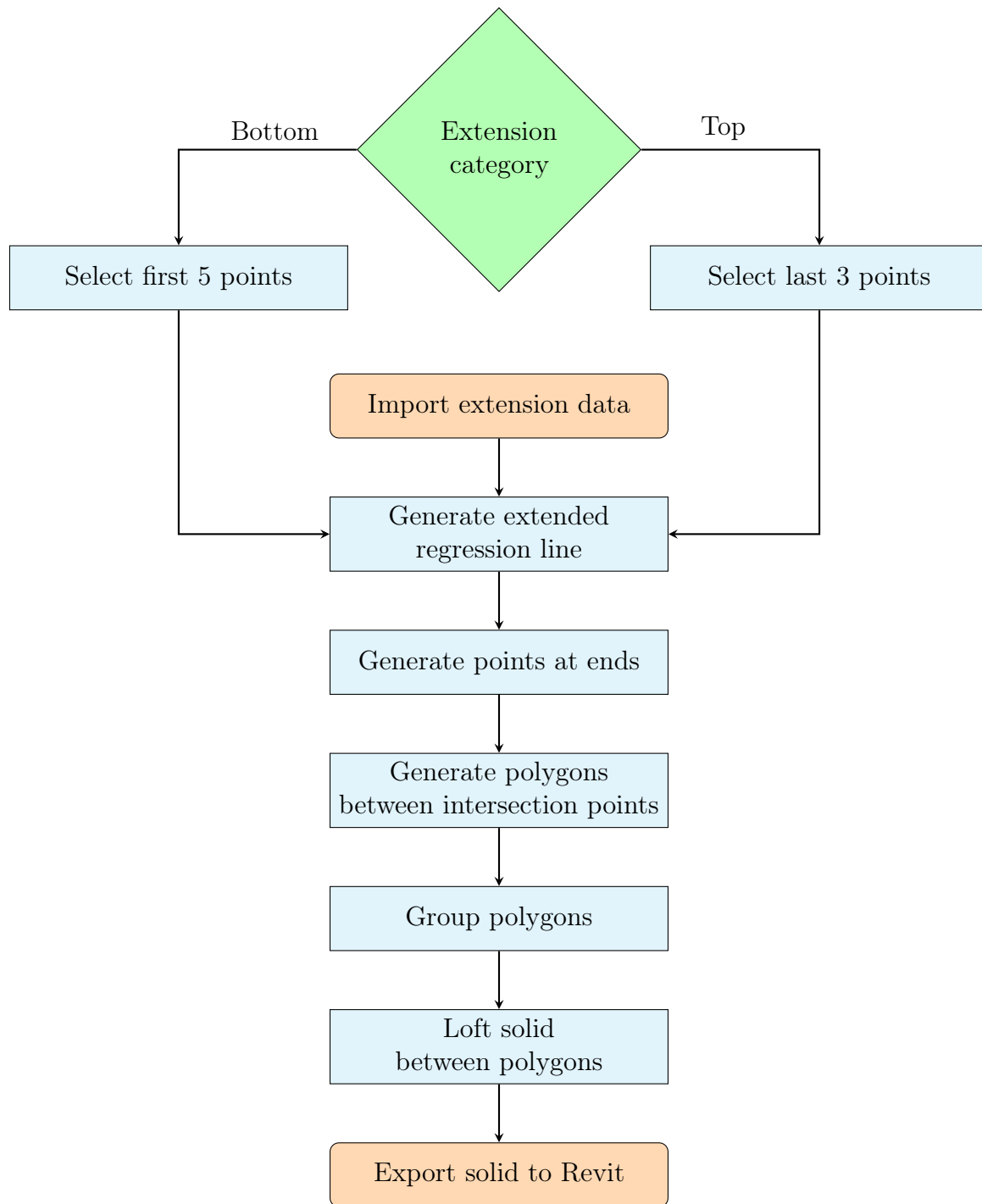


(b) The lofted solid of the reverse model of pier 2, visualised in Dynamo (left) and exported to Revit (right)

**Figure 5.22:** Reverse modelling process for the main part of pier 2

### 5.3.2.2 Extensions

The simplified conceptual algorithm for the pier column extensions shown in Figure 5.23 is almost the same for both the top- and bottom- extension, the difference being the amount of points forming the basis of the extended regression line. Due to the relatively large curvature in the top parts of the piers, fewer points (3) could be selected to form the basis for the creation of regression lines. The bottom part is straighter in shape, and can use more points to form the regression lines (5). This can be seen in the full Dynamo-graph in Appendix B2.



**Figure 5.23:** Simplified conceptual algorithm for the pier column extensions

In Section 4, the top of all six pier columns were cut due to inaccurate segmentation caused by incomplete point cloud data. The bottom of all of the piers, except for pier 4 which was good enough to keep, were also cut as a result of incomplete scanning data, likely due to occlusion. As with the box girder, including the cut parts in the reverse model is crucial in order to accurately create a reverse BIM model of the Dade Bridge in



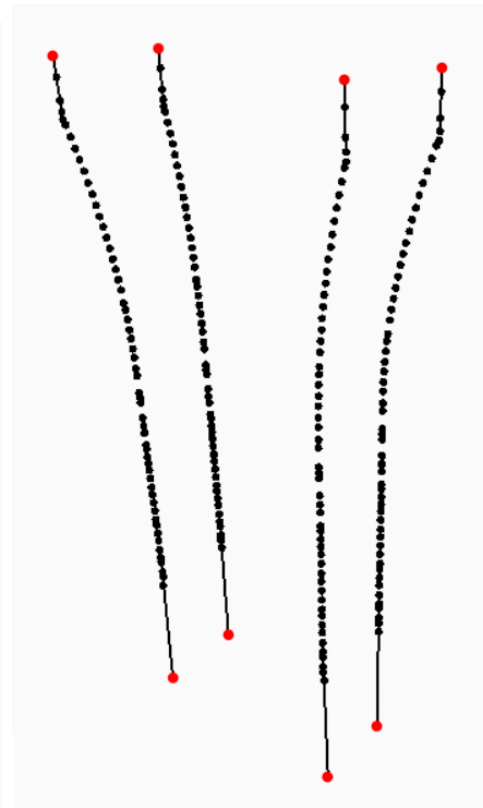
its actual state as it were when it was scanned with the 3D laser scanners. The modelling method for the top and bottom extensions of the piers are quite similar to that of the box girder, with some alterations to certain details of the Dynamo-graph.

As shown in Figure 5.24, the concept starts out by generating straight regression lines through the five lowest, and three highest intersection points at each line of intersection points for the bottom and top extension, respectively. The generation of regression lines results in four straight lines by best fit in both the top and the bottom part of each pier (except for the aforementioned pier 4 of which the bottom was not cut, thus not in need of extension). For the bottom extension, each of these lines are then extended by the distance between the lowest z-value of the original point cloud of the pier column in question, and the lowest z-value of the first intersection point, and vice versa for the top.

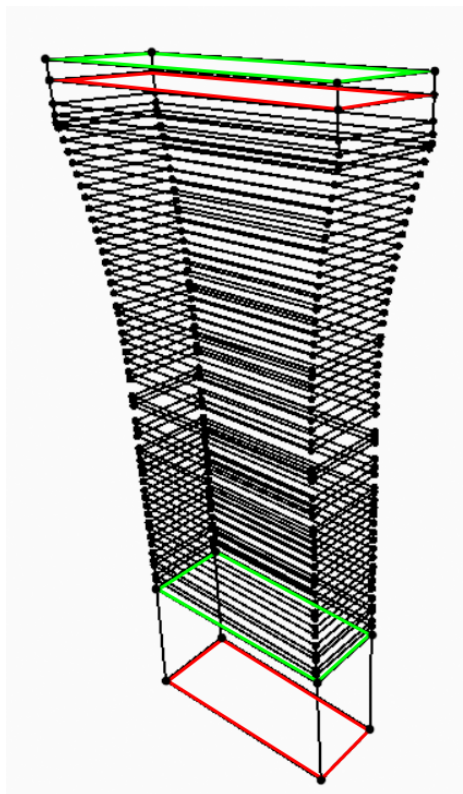
On the end of each extended regression line, an extension point is placed, forming the outline of a quadrilateral. The placed points are grouped, and four-sided polygons are drawn between them. As with the start- and end-extensions of the box girder, also here the closest polygon from the intersection points of the pier is grouped together with the newly generated polygon, as outlined in red and green in Figure 5.24. Between these polygons, a solid is lofted. This is done for both the top and the bottom extension of the piers, the only difference being that the closest polygon to the generated extension points is the lowest polygon for the bottom extension, and the upper polygon for the top extension.



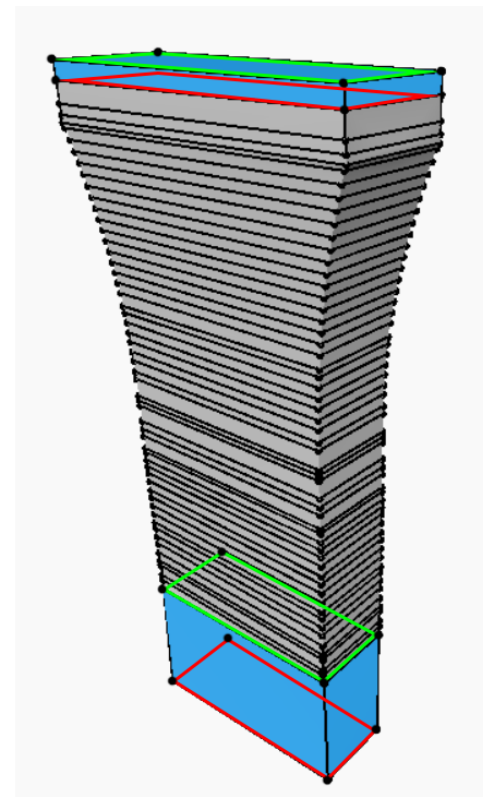
(a) Selected outermost points of pier 2



(b) Generated points at end of extended regression lines



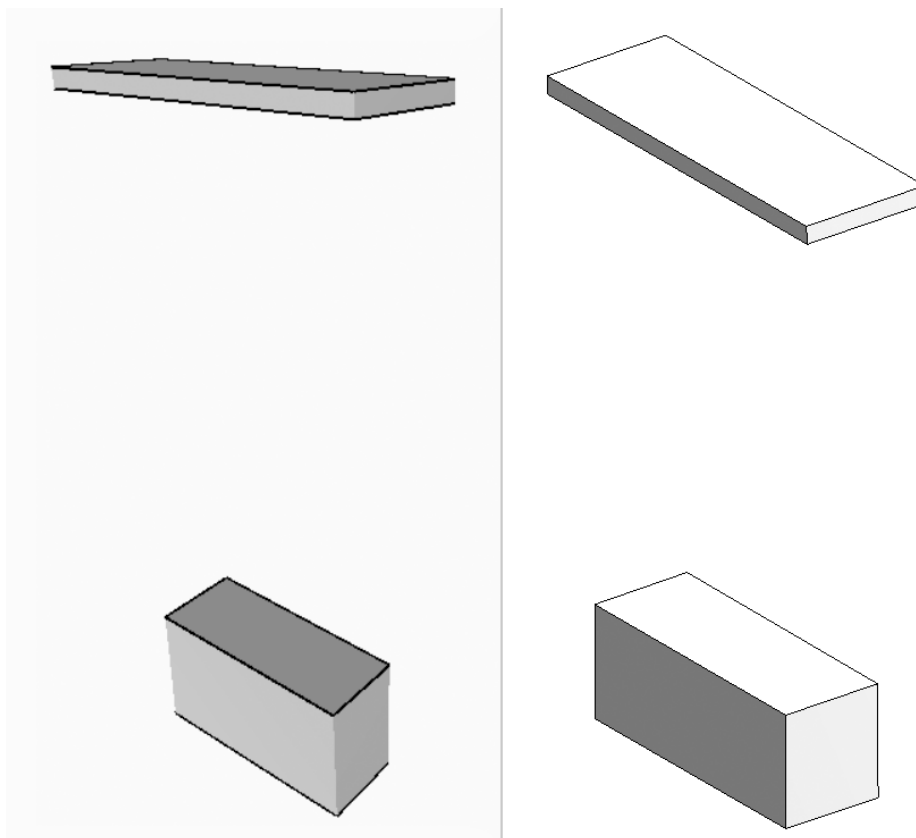
(c) Grouped polygons from generated points, and first and last polygon of pier 2



(d) Lofted solid between grouped polygons

**Figure 5.24:** Modelling process for top and bottom extension for pier 2

The solid for each pier column is subsequently named, and a family instance for each element is exported to the Revit-workspace at the given coordinates of the intersection points. Similar looking to the extensions of all pier columns, the finished reverse model of the pier column extensions of Pier 2 can be seen in Figure 5.25.



**Figure 5.25:** The solid of the finished extension models of pier 2, visualised in Dynamo (left) and Revit (right)

### 5.3.3 Reverse BIM Model - Pier Foundation

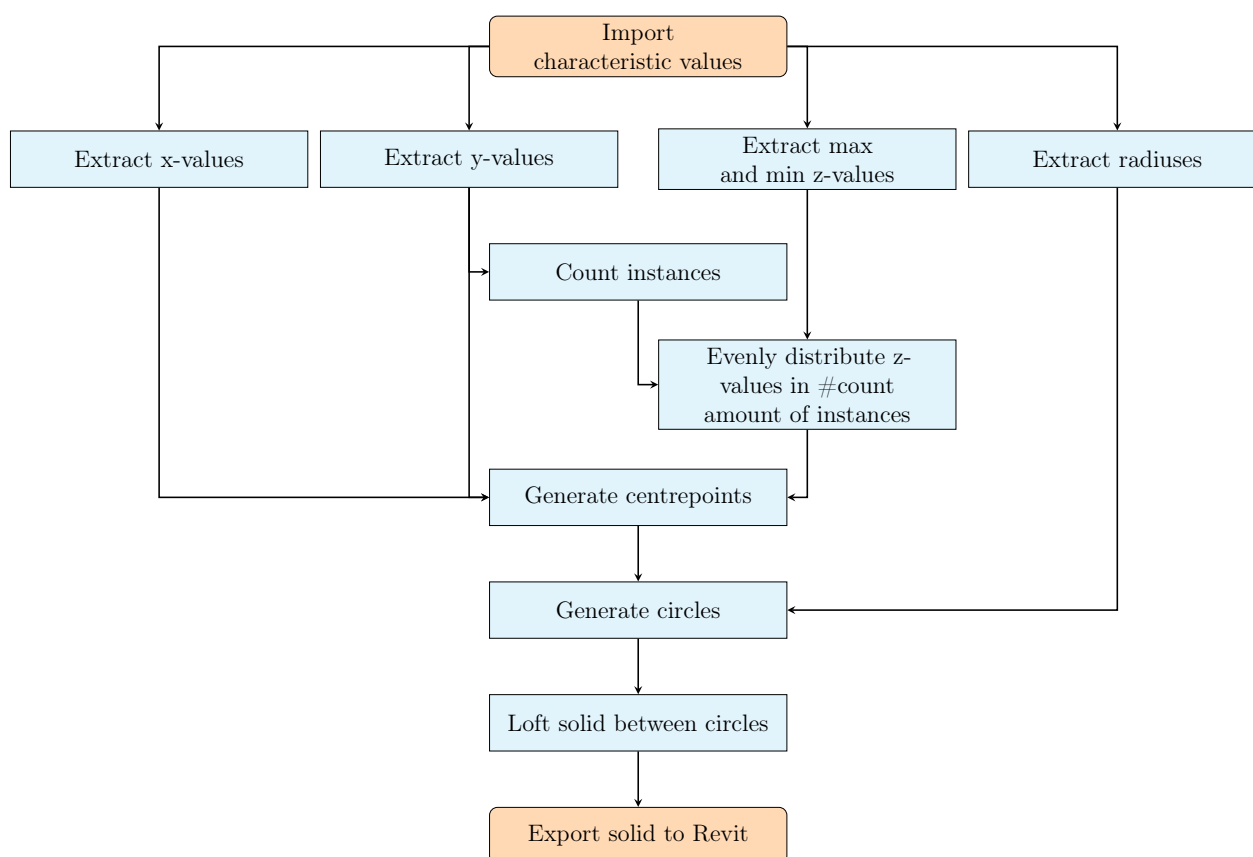
To complete the reverse model, the pier foundation must also be modeled. In the section of the Dade Bridge, which is analysed in this thesis, only one pier foundation can be found, as outlined in 5.26.



**Figure 5.26:** The point cloud of the Dade Bridge with its pier foundation outlined in red

The pier foundation is of a cylindrical shape, with its width being larger than its height. The geometry of the point cloud is fairly simple, reproducible by calculating some key values from Matlab, as detailed in Section 4.3.

Opposite to the method for reverse modelling the box girders and pier columns, the Dynamo-graph for the pier foundation is not based on intersection points due to its cylindrical shape. Instead, Matlab is used to divide the foundation into numerous horizontal slices along its z-axis, and subsequently locate the x- and y- values for the centerpoint of each slice. The centerpoints are exported to an Excel-sheet along with the corresponding radius of each slice, as shown in Table 4.8. Notably, the z-values of the centerpoints are not exported. Instead, the two z-values at the top and bottom of the original pier foundation point cloud segment are exported. By doing this, the small gap which would be created between the topmost centerpoint and the highest z-value of the pier foundation point cloud due to the centerpoint algorithm which flattens each pier foundation slice to its center, is avoided. Further, this solves the issue of the missing pier foundation data at the bottom, which is cut in Matlab with the cutting algorithm due to the incomplete point cloud. In Figure 5.27, a simplified algorithm of the Dynamo-graph for the reverse modelling of the pier foundation is presented. The full script can be seen in the Dynamo-graph in Appendix B3.



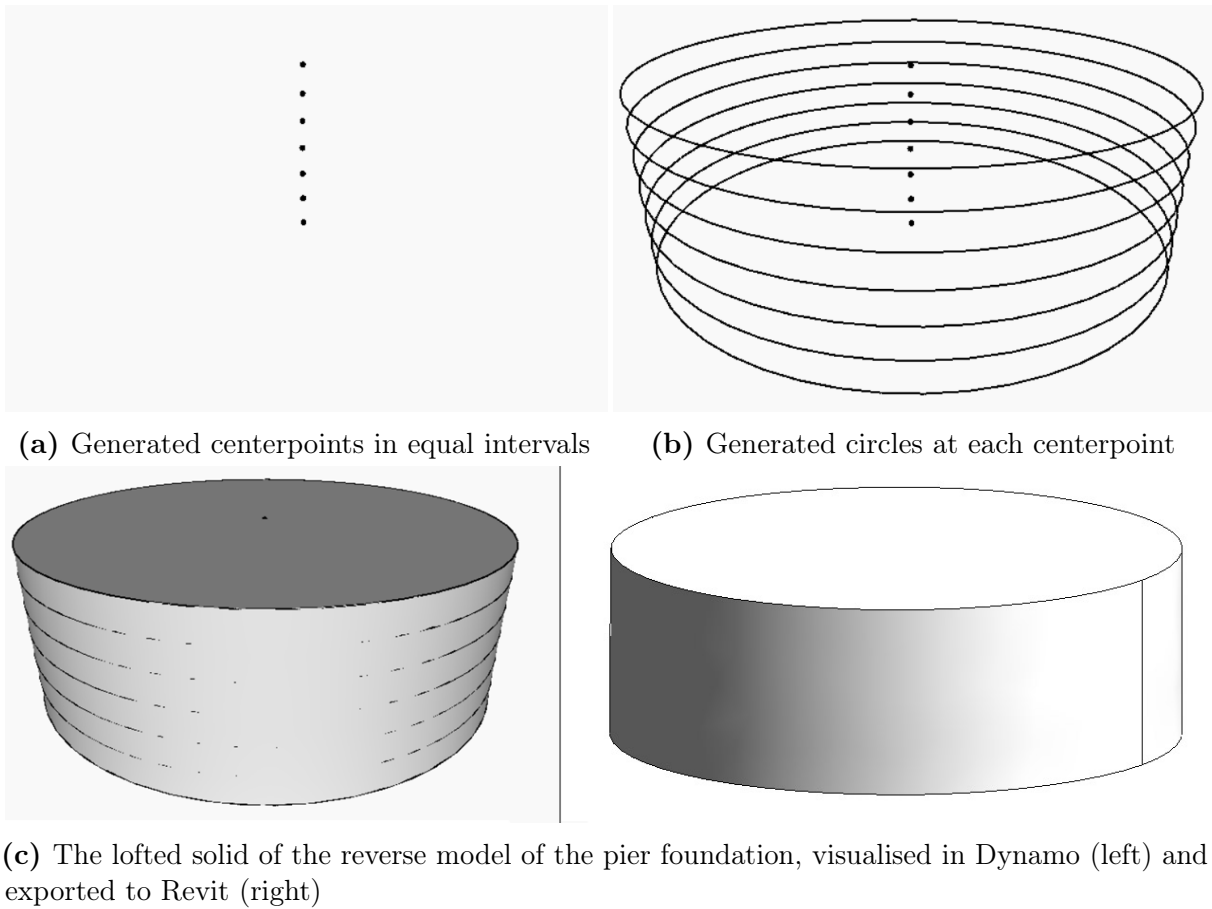
**Figure 5.27:** Simplified conceptual algorithm for the pier foundation reverse modelling

When assembling the pier foundation in the Dynamo-graph, the x- and y-values for each slice with the corresponding radius of the foundation at each centerpoint is imported. The amount of centerpoints are counted, and the difference between the extreme z-values are divided into points of equal intervals amounting to the counted value.

At each of the centerpoints, now with x-, y-, and z-values, as shown in Figure 5.28, a horizontal circle is generated using the calculated radiuses. The radiuses differ slightly from the lower part of the foundation to the top, with values of approximately 5511,8 mm to 5523,4 mm respectively. As with the polygons of the box girders and pier columns, a solid is lofted from circle to circle throughout the pier foundation, then exported to Revit as a named family element.

With this method, potential damages to the foundation affecting the geometry can be accurately modeled, as the cross-section can vary from slice to slice throughout the cylindrical foundation. The maximum difference in radiuses is only approximately 11,6 mm across the almost 3,5 m tall foundation, meaning that the cylindrical geometry of the

pier foundation is almost perfectly straight. More detailed analysis of each bridge part can be found in the following Section 6.



**Figure 5.28:** Reverse modelling process for the pier foundation

### 5.3.4 Reverse BIM Model - Terrain

Although the complete BIM model of the Dade Bridge provides an accurate representation of the bridge geometry, in order to form a better understanding of the overall situation on the site, the bridge should be placed in the correct terrain model. The terrain adjoining the bridge structure is an important part of a bridge inspection. Landslides, soil erosion, damage to slope protection, unintentional removal of embankments or fill, flood scouring and other ground movements can have catastrophic consequences for the bridge structure. These kind of ground movements are the most common cause of bridge failures or severe damages on bridge structures [87]. As such changes can be difficult to inspect with the naked eye, a Digital Terrain Model (DTM) will give an accurate representation of the ground around the bridge and the maintenance that must be performed as a result of any damage.

#### 5.3.4.1 Terrain Point Cloud Processing

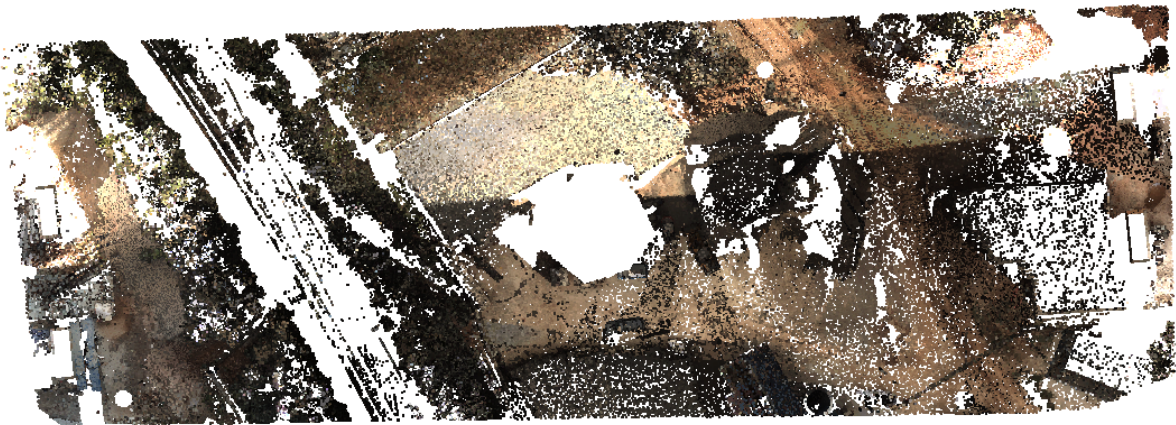
The terrain was earlier in this thesis classified as noise and subsequently filtered out of the full point cloud, leaving only the bridge point cloud remaining. In order to generate a DTM, the terrain point cloud of the bare earth needs to be utilised. The original merged scan was used to create the DTM under the bridge. Because of the use of a long-range, terrestrial laser scanner with a defined scanning area and a relatively small amount of executed scans, the point cloud of the ground was quite deficient, as can be seen in Figure 3.3. A comprehensive reconstruction of the terrain was needed in order to create a correct representation of the site. This would likely be avoided by taking advantage of UAV-mounted laser scanners instead of TLS to collect scan data of the terrain surrounding the bridge.

Geomagic was used for the processing of the terrain point cloud. From the complete point cloud of the bridge, the points that did not belong to the terrain was manually deleted. The terrain that is far away from the geometry of the bridge was also removed not only because it is irrelevant for inspection, but also because it was deficient and would not have given a good representation of the reality. After the initial processing, the bounded, cleaned up point cloud representing the terrain in the immediate area around the Dade Bridge is created, as shown in Figure 5.29.



**Figure 5.29:** Point cloud of terrain model

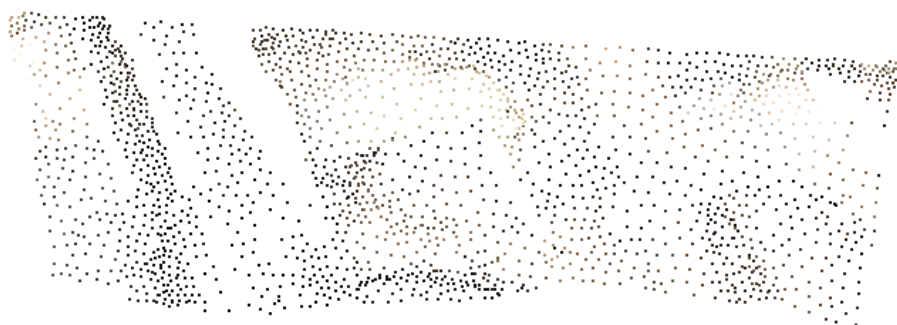
However, this point cloud does also have some areas which are deficient. Blank areas with missing points were filled with new points representing the ground. Figure 5.30 show the supplemented terrain point cloud.



**Figure 5.30:** Supplemented terrain model

The supplemented terrain point cloud consists of a large number of points. However, a small portion of the points are sufficient to create an accurate DTM. By downsampling, the point cloud of the terrain model is reduced to a more suitable amount of points for terrain regeneration. The downsampled terrain point cloud can be seen in Figure 5.31.





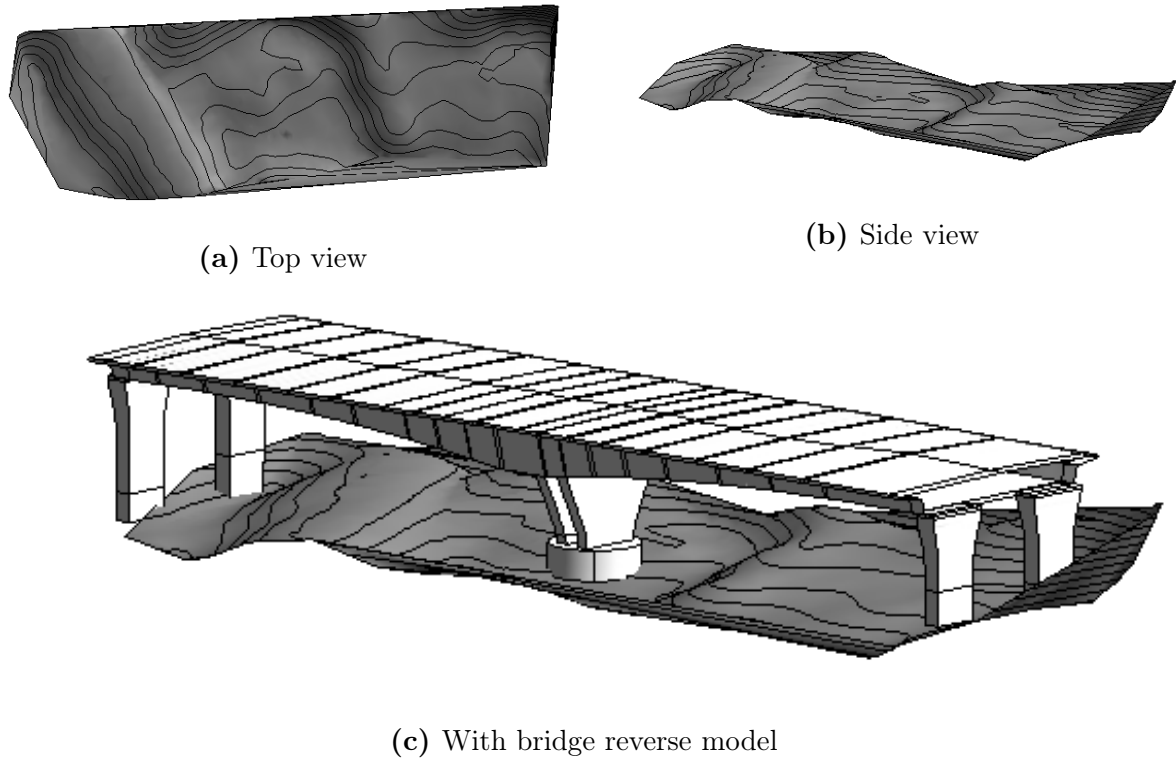
**Figure 5.31:** Sampled terrain model

#### 5.3.4.2 Reverse Modelling of Terrain Model

To make a DTM of the terrain under the bridge, the sampled point cloud is imported to Revit through Excel. In Revit, a topographic surface was generated by using the position and z-value of each point. The result is an accurate terrain model with contour lines, as seen in Figure 5.32. The contour lines show the difference in elevation, with each contour line representing a difference in elevation of one meter.

Elevations on existing terrain are basic data that form the basis for planning, design and building of infrastructure projects. The elevation data often consists of a quality point cloud from scanning or measurement data from total station. Terrain scanning can be done by UAV, plane, helicopter, car or TLS.

This short section describing the process of creating a reverse model of terrain data acquired from a TLS have shown how easily a laser scan can be used to generate a DTM. This DTM can be used to discover terrain changes critical to a bridge structure. To have a greater representation of the surrounding terrain, scanning from the skies, or performing a full 360° scan should be considered, in combination with more accurate planning of the scanning positions to fully capture the terrain and not only the bridge.



**Figure 5.32:** Terrain model, contour lines at every meter

## 6 Accuracy Analysis of Reverse BIM Model

Checking consistency in models is an important part of reverse modelling. Without knowing the consistency and accuracy of the reverse model, the model cannot be trusted for further use. A consistent model in this context is a model with a geometrical accuracy when it comes to shape, size, location and direction.

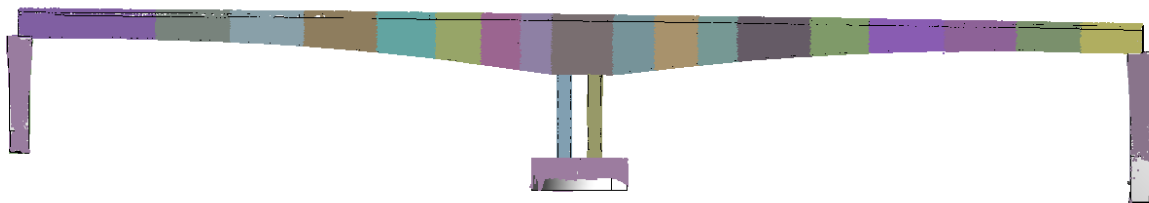
Two different comparison methods are used to find the accuracy of the bridge BIM model, one visual and one deviation analysis.

The next few sections will show and describe the similarities and differences between the reverse BIM model and the point cloud, first in a visual matter using Revit and then by putting numbers on their deviation performing a deviation analysis with Geomagic.

### 6.1 Visual Comparison

The visual check is performed in Revit. The point cloud is imported via Recap into Revit, reasoned in Section 1.4.2, and aligned with the BIM model.

The bridge point cloud and BIM model show significant similarity. The BIM model has the same size and geometry as the existing bridge, represented as the point cloud. The only place a difference is observed is at the bottom of the columns, where the extension is done as compensation for the poor scan quality in this area. The geometrical similarities of the complete bridge can be seen in Figure 6.1.

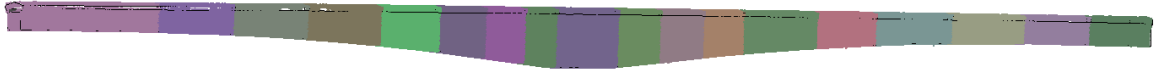


**Figure 6.1:** Point cloud and BIM model comparison

In order to perform a more thorough comparison, each part of the bridge needs to be closer examined. First, the overall accuracy will be presented for both the box girder, pier columns and pier foundation, and then selected areas with larger deviation will be analysed.

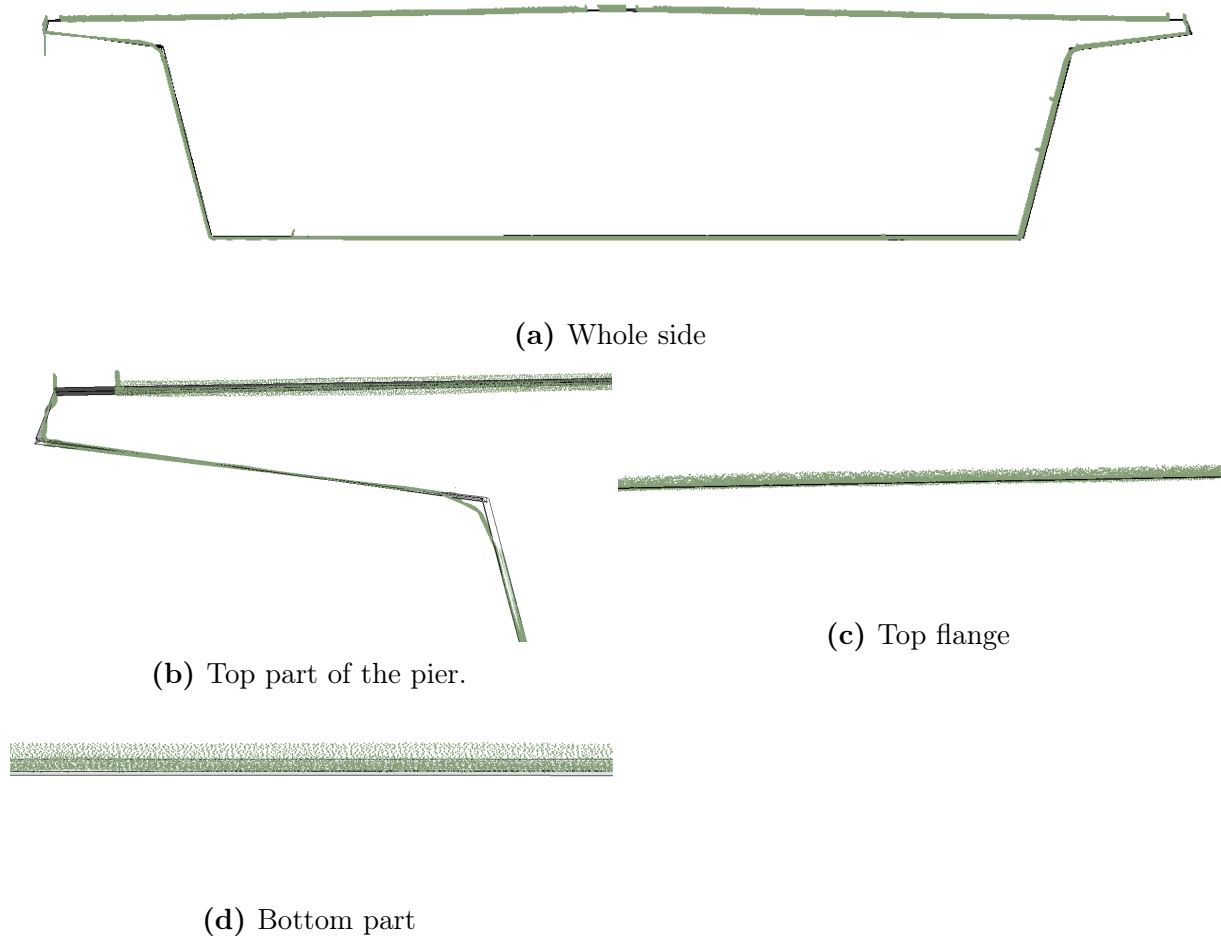
### 6.1.1 Box Girder

The reverse BIM model of the box girders is a good representation of its point cloud, seen in Figure 6.2.



**Figure 6.2:** Comparison of box girders

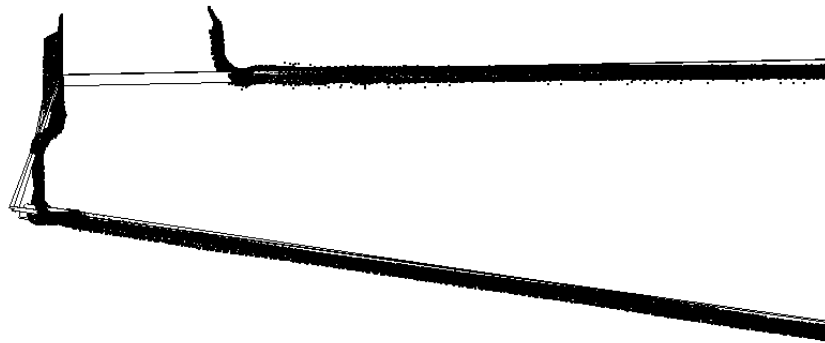
Having a total of 18 boxes, only some representative figures of the BIM model accuracy will be shown. The results from box girder 9 were presented earlier in this thesis and is the one being presented with a close-up comparison between the reverse BIM model and the point cloud, shown in Figure 6.3.



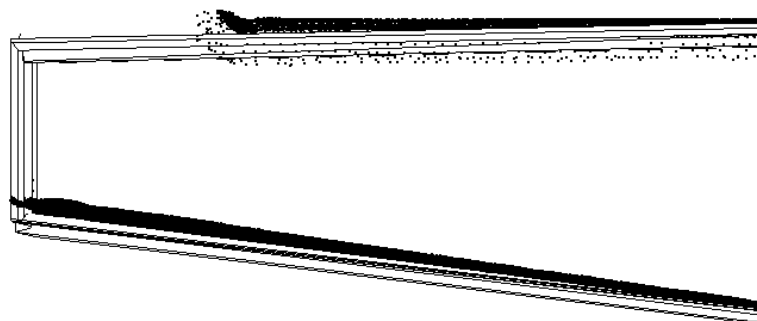
**Figure 6.3:** Visual comparison of box girder 9

All the box girder sides show quite good accuracy. Zooming in even more, show that all the corners and the outer part (intersection point 3,4 and 6,7, see Figure 4.3), have the biggest deviation from the point cloud. This will be further explained in Section 6.1.4.

Since one part of box girder 10-18 was missing from the scan, the intersection points 6 and 7 had to be established manually based on geometry, as previously detailed in 4.1.2. Figure 6.4 show how this missing side corresponds to the point cloud and how it looks compared to the side with a complete point cloud. The created side, show a good interpretation of the bridge, maybe even more correct than the side modelled based on the point cloud and the deviation at this part due to geometrical changes.



(a) Side with point cloud

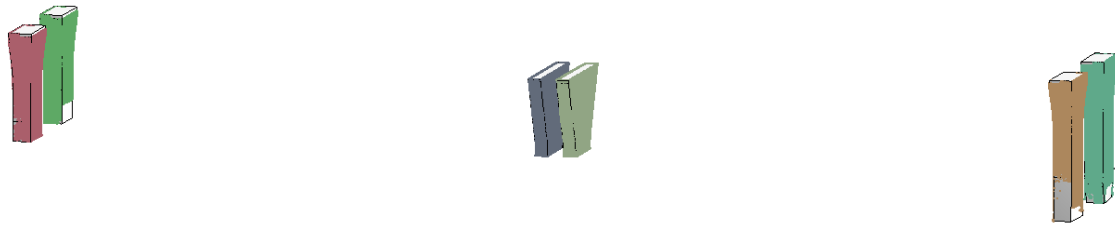


(b) Side with missing point cloud

**Figure 6.4:** Comparison of missing part in box 10-18

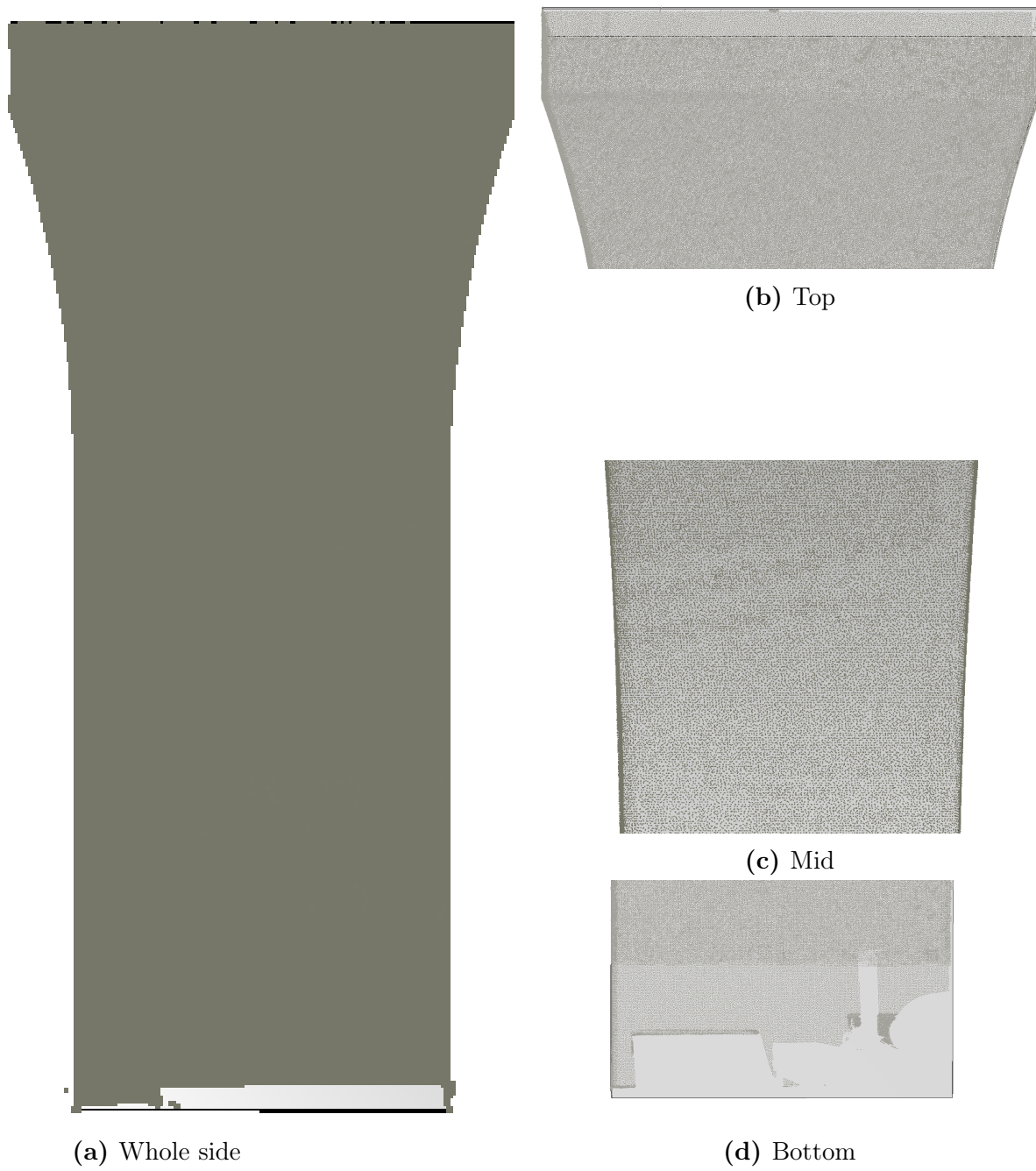
### 6.1.2 Pier Column

The substructure is examined in parts, firstly the pier columns, then the pier foundation. Each pier column is controlled, and some of the typical results will be presented. Since there are two different pier geometries, i.e. pier column 1, 2, 5, 6 and pier column 3, 4, each geometry will be presented by itself. Figure 6.5 show the visual comparison of all the piers.

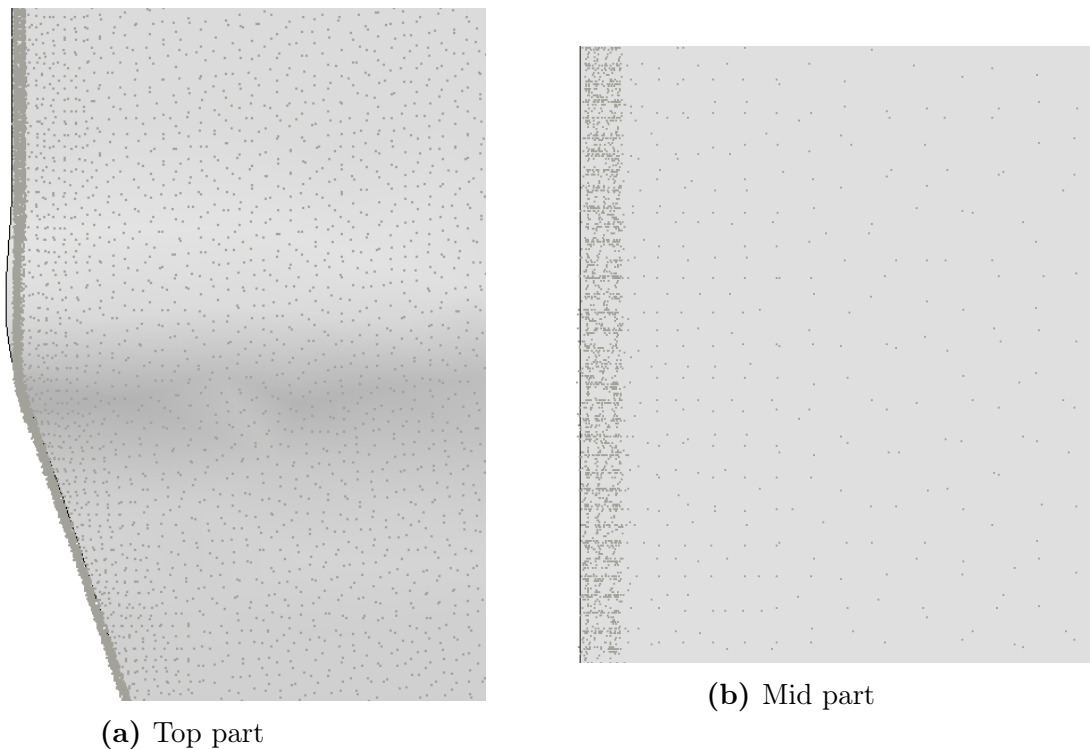


**Figure 6.5:** Comparison of the piers

For pier column 1,2,5 and 6, the comparisons show great similarities. Due to these similarities, only the visual comparison of pier column 2 will be closer examined in this section. The comparison of pier column 2 is seen in Figure 6.6. Some close-ups in Figure 6.7 also show how good the accuracy is for this pier geometry. The point cloud is laying right on the BIM models edge for all parts of the pier column, and the extension in the bottom part has the correct length.



**Figure 6.6:** Visual comparison of pier column 2



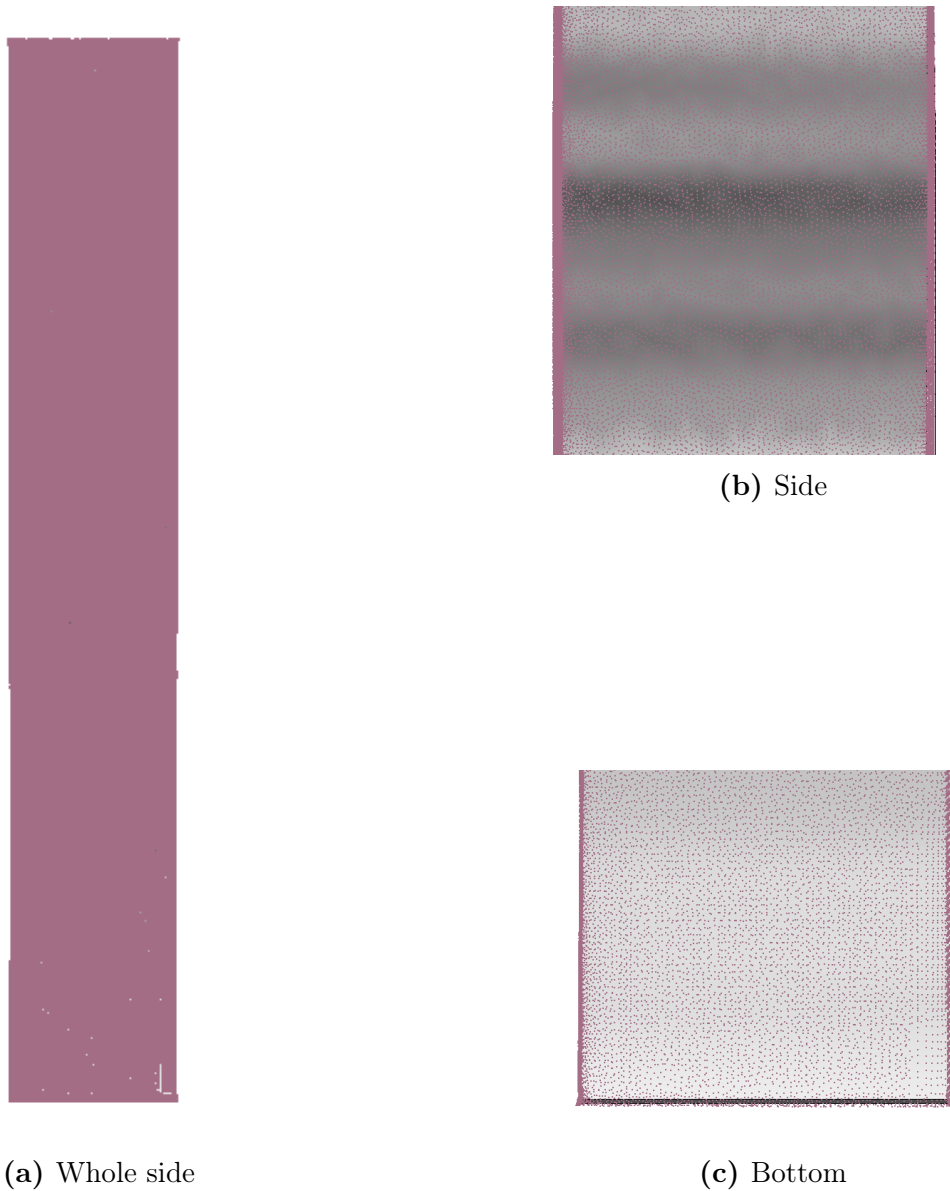
**Figure 6.7:** Close-ups of how accurate the BIM model can be

The mid-span piers, pier column 3 and 4, have a different geometry than the piers in the outer edge of the bridge. The geometry of these piers have a changing cross-section and are harder to perform reverse modelling on. Figure 6.8 shows the short side of pier column 4, and Figure 6.9 shows the wide side. The short side has no change in geometry, which cause the accuracy of the wide side of the BIM model having perfectly fitting edges with the point cloud. However, the accuracy on the short side of the mid-span piers are not satisfying. The increasing cross-section on the pier wide side is having an impact on the feature fitting and reverse modelling of this part. The intersection points are not accurate enough, and as a consequence, the line between them gets wavy, Figure 6.9 **b**.

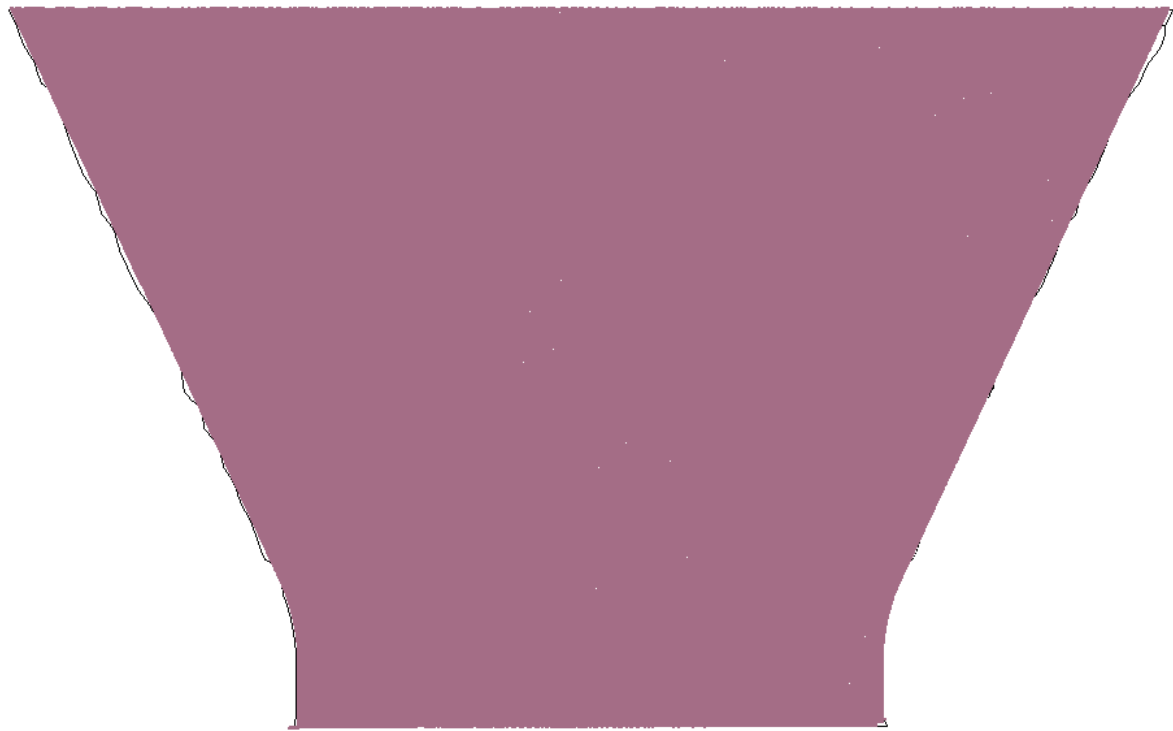
In order to fix the inaccuracy of the mid-span of pier columns 3 and 4, a couple of different measures could have been done. In Section 4 the pier could have been divided into smaller slices. Using smaller slices could have contributed to getting a smaller distance between the intersection points and in this way have the undulating sides smoothed out. Another method could have been to change the way the line between intersection points in Section 5.3 was created. It is possible to create regression lines between the intersection points, and then place new points along the generated regression line in equal intervals. This would guarantee smooth curves in the reverse BIM model. However, this method would



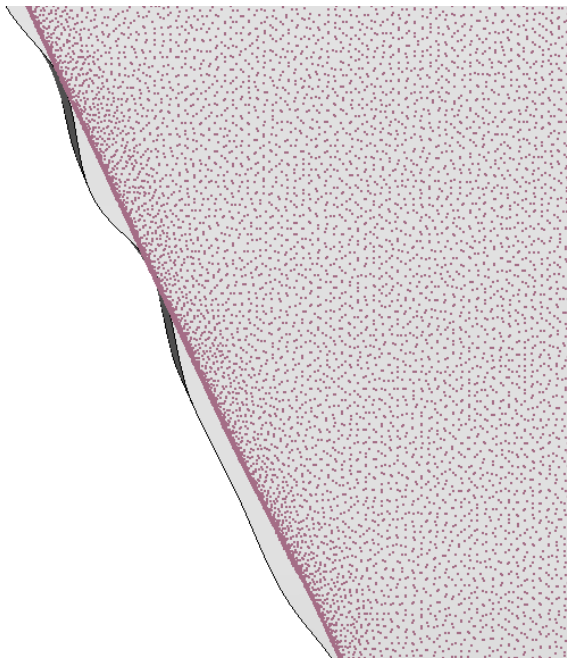
not accurately model possible geometrical errors, and would thus not be usable for bridge inspection. The wavy BIM model is therefore deemed to be a more usable result than a regression-based method.



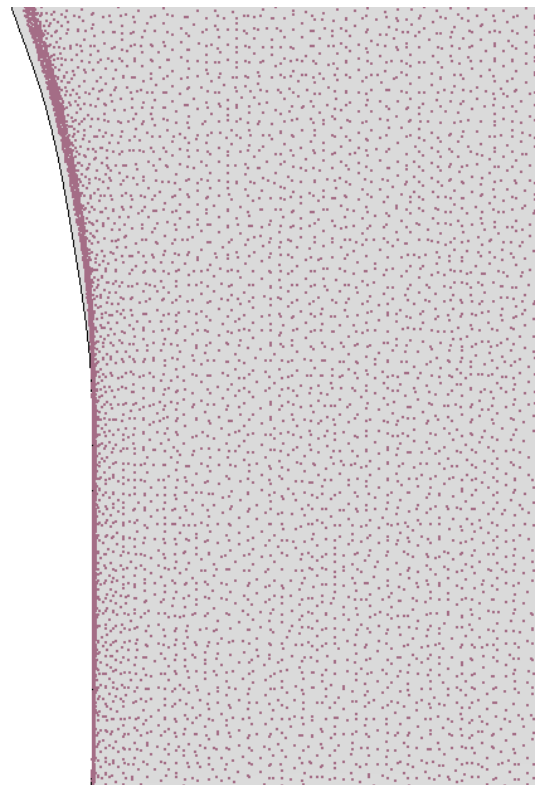
**Figure 6.8:** Visual comparison of pier column 3 and 4, short-side



(a) Whole pier



(b) Side

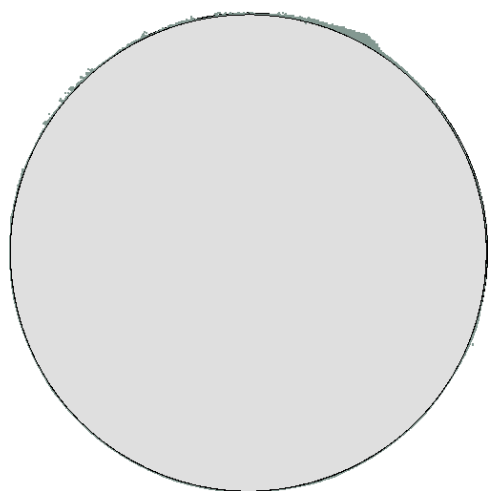


(c) Bottom

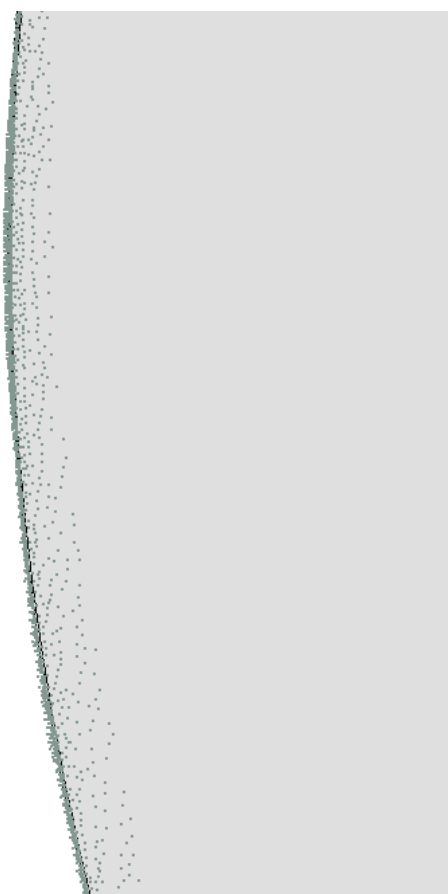
**Figure 6.9:** Visual comparison of pier column 3 and 4, wide-side

### 6.1.3 Pier Foundation

Other than the bridge pier columns, the foundation under the mid-span pier columns is also a part of the substructure. The accuracy of the mid-span foundation can be seen in Figure 6.10. Both the height and the circle have what looks like a perfect fit with a visual conception.



(a) Top view



(b) Close-up, top view



(c) Side view

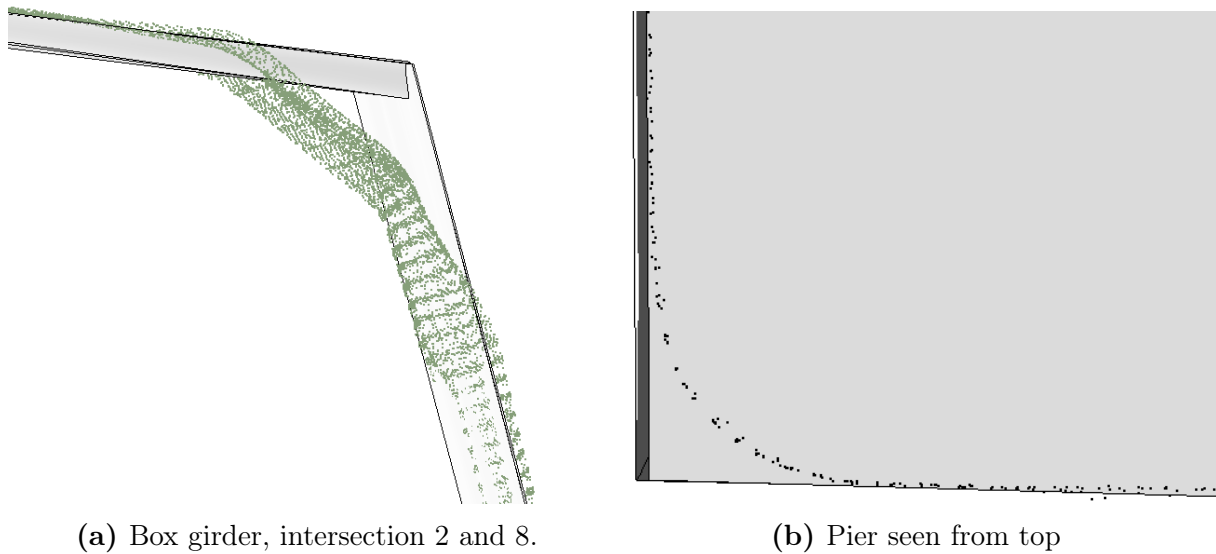
**Figure 6.10:** Visual comparison of mid-span foundation

### 6.1.4 Recurring Accuracy Errors

Some of the deviations between the BIM model and point cloud is caused by an insufficient method of feature fitting and 3D modelling, and are inaccuracies which are repeated several times. These deviations will be presented in this section.

Close-ups at the most vulnerable places can show errors in the reverse 3D BIM model. Although the following figures present areas where the reverse model is not perfectly accurate with the point cloud, these issues does not necessarily provide an overall representation of the model, as will be clarified in the following deviation analysis in Section 6.2. However, the figures can give a good indication of improvements in areas where the feature fitting and reverse modelling of the point cloud is insufficient.

The method used for reverse modelling of both box girder and pier columns are not able to model rounded edges. When creating the 3D BIM model, a straight line between intersection points make up the model, and this modelling method will only create sharp corners. Corners, where the point cloud have the largest radius, is where the deviation is at is biggest. In Figure 6.11, the deviation in a typical corner for a box girder can be seen in **(a)**, and in **(b)**, a corner of a pier column is seen from the top. It is apparent that the intersection points which are generated from regression lines through each side of the four-sided pier column, is a cause of inaccuracy due to the rounded edges of the pier column point cloud.

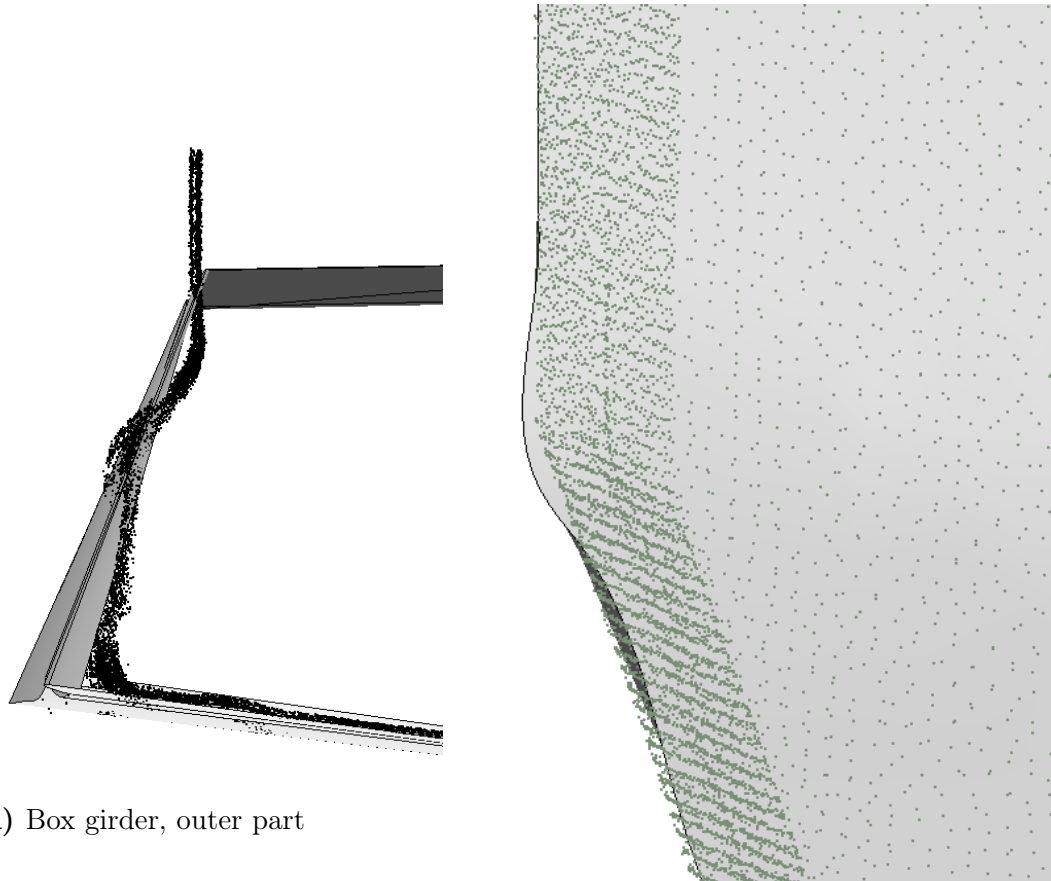


(a) Box girder, intersection 2 and 8.

(b) Pier seen from top

**Figure 6.11:** Deviation in rounded corners

Corners are geometrical changes where the BIM model accuracy is not at its best, but other geometrical changes can also be hard to perfectly remodel. In Figure 6.12, two geometrical changes can be seen, (a) shows the outer part of the box girder and (b) the top part of a pier. None of them are easily remodelled and will cause deviation in the BIM model. I.e. the section of the box girder presented in Figure 6.12(a) is impossible to accurately model using only two intersection points as was done in this reverse modelling process.

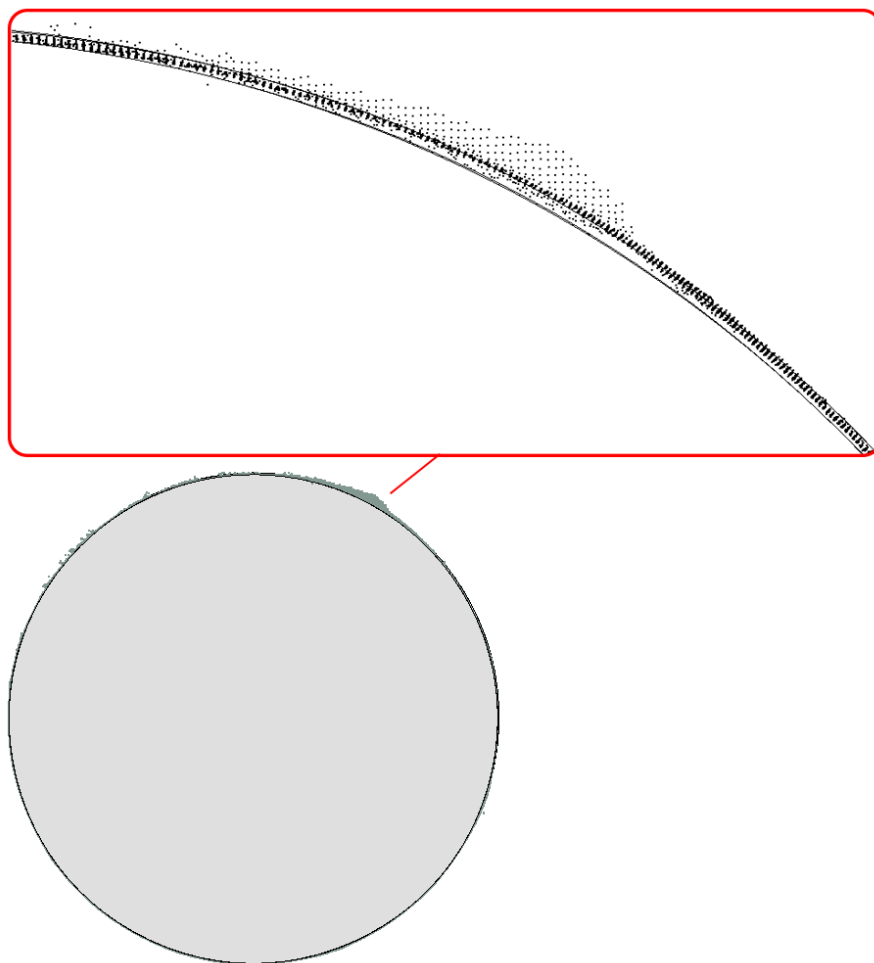


(a) Box girder, outer part

(b) Pier column, top part

**Figure 6.12:** Deviation at geometrical changes

Another cause of inaccuracies is a noisy point cloud. A noisy scan data will be seen as a visual deviation. A good example of this is a part of the pier foundation where noisy data are causing a deviation between the BIM model and the point cloud, see Figure 6.13. This uneven line of points looks like a deviation from a distance. However, zooming in reveals that the line consisting of the most points are actually laying closer to the BIM model edge. This kind of deviation is only an analysis problem and are not causing actual deviations in the BIM model.



**Figure 6.13:** A noisy point cloud affecting the comparison accuracy of the pier foundation

## 6.2 Deviation Analysis

A deviation analysis to compare the BIM model with the point cloud is executed in the inspection software Geomagic. The model is exported from Revit, using the CAD format, ACIS (.sat), and imported in as reference data in Geomagic, the point cloud is the measured data. In Geomagic a coloured deviation map is created using the function "3D compare", and 2D comparison is used to see the deviation within specific cross-sections. The complete deviation analysis report is attached in Appendix C.

A coloured deviation map is a graphical depiction of the deviation between two objects, in this case, the point cloud and the reverse BIM model of the Dade Bridge. The 3D compare function will measure how the surface of the parts differ between the reference and measured data by searching for the deviation between them. The coloured deviation map will display blue data which indicates measured data under or behind the reference

data, yellow to red areas indicate the measured data is above or in front of the reference data. Green data is the data inside the chosen tolerance range.

The as-built BIM model should be as close to the built structure as possible. The bridge design manual, N400 of NPRA, state that manufacturing and build tolerances is given in the manual, R762: Standard specification text for bridges and quays [24]. Chapter 84 of the R762 manual includes structural parts of concrete, a great base for setting tolerance level for concrete bridges. If the reverse BIM model is used for maintenance or remodelling in Norway, the BIM model must satisfy the tolerance level of this manual. The manual states that the geometrical tolerances are linked to the safety, durability, usage characteristics, and the structures aesthetic appearance [14].

The tolerance classes for different structural components are set to class **B** if there are no special project specifications. What this means for the different structural components are given in Table 6.1. Table 6.1 is a reproduction of relevant content in table 84-1 and 84-2 from R762 put together. These tolerance levels are used to perform the deviation control for the BIM model in this thesis.

**Table 6.1:** Overall structural tolerance [14]

<b>Structural component</b>	<b>Tolerance level</b>
Foundations	$\pm 100mm$
Piers	$\pm 30mm$
Box girder	$\pm 30mm$

The deviation map of the complete bridge BIM model can be seen in Figure 6.14. The tolerance level of the complete bridge is set to  $\pm 30mm$ , similar to the tolerance for the main parts of the bridge. With  $\pm 30mm$  tolerance level, most of the bridge display a green colour, indicating data inside the tolerance.

Table 6.3 contains the statistics of the 3D comparison. The two average values of positive and negative gap distance ( $\pm Avg$ ) are a good indication on the accuracy of the reverse BIM model.  $+Avg$  is the average deviation of the points in front or over the model, while  $-Avg$  is the average deviation of the points behind or under the reverse model. Thus, the average of these two values gives the overall average deviation. For the complete bridge the  $+Avg(12,2907mm)$  and the  $-Avg(-11,5196)$  results in an average gap distance of 11,9mm, i.e. far within the set tolerance level. Further, from the analysis it is found



that approximately 97% of the data is within the tolerance level of 30mm.

The deviation map and the statistics also reveals some parts of the model having a greater gap distance than the accepted tolerance, mainly at the pier extensions and at the missing side of half of the box girders. These deviations are for the most part connected to errors in the point cloud, and caused by the railing not being a part of the reverse modelling in this thesis. These errors interfere with the actual accuracy of the reverse model, as the missing points increases the average deviation values.

The percentage of measured data under or behind the average is approximately the same as the data laying over or in front of the model. This even distribution indicates that the alignment of the BIM model and the point cloud is accurate.

In the next few sections, each part of the bridge will be compared, and the results will be interpreted.

An explanation of the different statics are presented in Table 6.2:

**Table 6.2:** Explanation of statistics [15]

Avg	The arithmetic mean of all the gap distances.
+Avg	The mean of all the positive gap distances.
-Avg	The mean of all the negative gap distances.
Std.Dev	Standard deviation of all the gap distance values.
Within Tol. (%)	The percentage of points that have a gap distance within the defined tolerance.
Beyond Tol. (%)	The percentage of points that have a gap distance beyond the defined tolerance.
Over Avg (%)	The percentage of points within tolerance that are greater average.
Under Avg (%)	The percentage of points within tolerance that are less average.



**Figure 6.14:** Deviation map of the complete bridge

**Table 6.3:** Deviation statics of the complete bridge

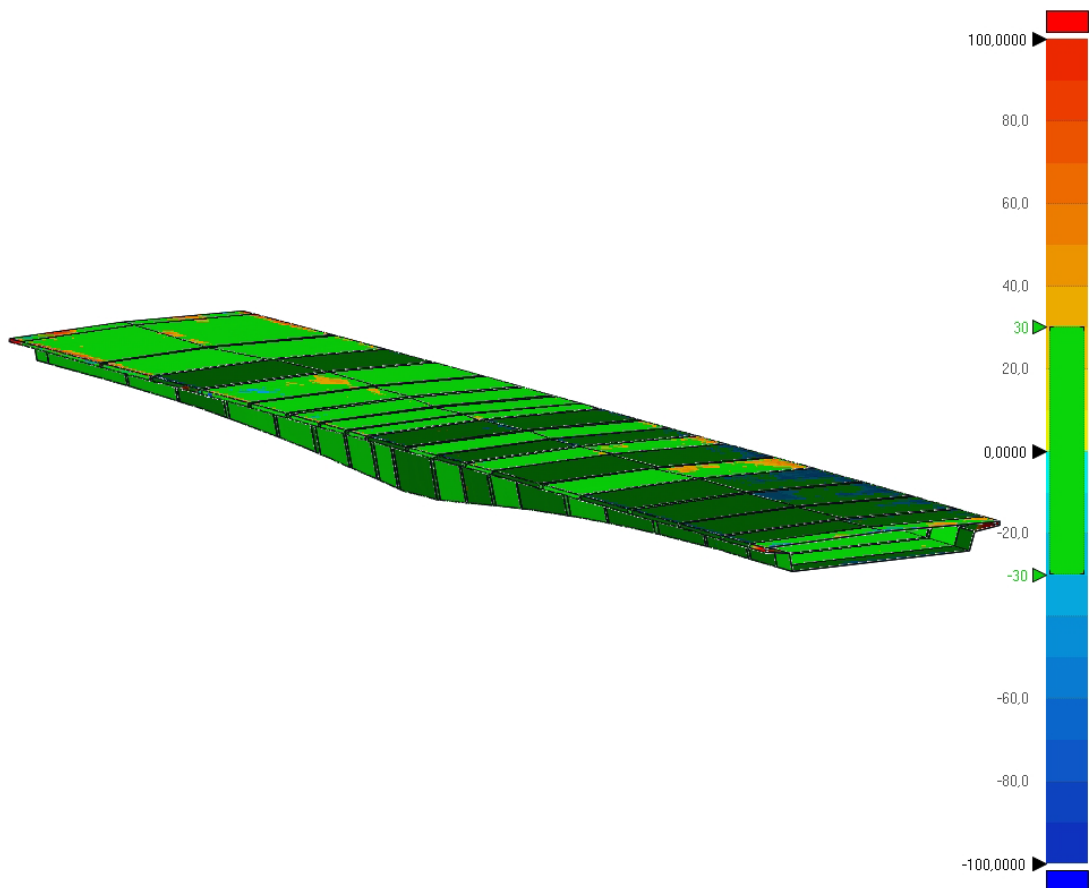
Avg.	0,5772 mm
+Avg.	12,2907 mm
-Avg.	-11,5196 mm
Std. Dev.	20,4472 mm
Within Tol.(%)	97,4283
Beyond Tol.(%)	2,5717
Over Avg(%)	47,9193
Under Avg(%)	49,5090

### 6.2.1 Box Girder

A 3D analysis of the superstructure, Figure 6.15, show the deviation map of the bridge box girders, and Table 6.4 its statistics. Most of the measured data lay within the tolerance level and are displayed in green. The average deviation is calculated to 1,3mm, and approximately 97% of the BIM model has an accepted gap distance between the

model and the point cloud. The average gap distance calculated from the  $\pm$ Avg values is 10,4283mm, i.e. well within the set tolerance level.

A low standard deviation indicates values close to the mean value and an accurate model. In Table 6.4 of the box girders, the standard deviation is 20mm, almost the same as for the complete bridge (20.45mm) and inside the tolerance level of 30mm.



**Figure 6.15:** Deviation map of the box girders

**Table 6.4:** Deviation statics of the box girders

Avg.	1,3340 mm
+Avg.	11,5368 mm
-Avg.	-9,3398 mm
Std. Dev.	19,9553 mm
Within Tol.(%)	96,7414
Beyond Tol.(%)	3,2586
Over Avg.(%)	47,2546
Under Avg.(%)	49,4868

In order to better estimate the tolerance level and analyse the impact that the deviations

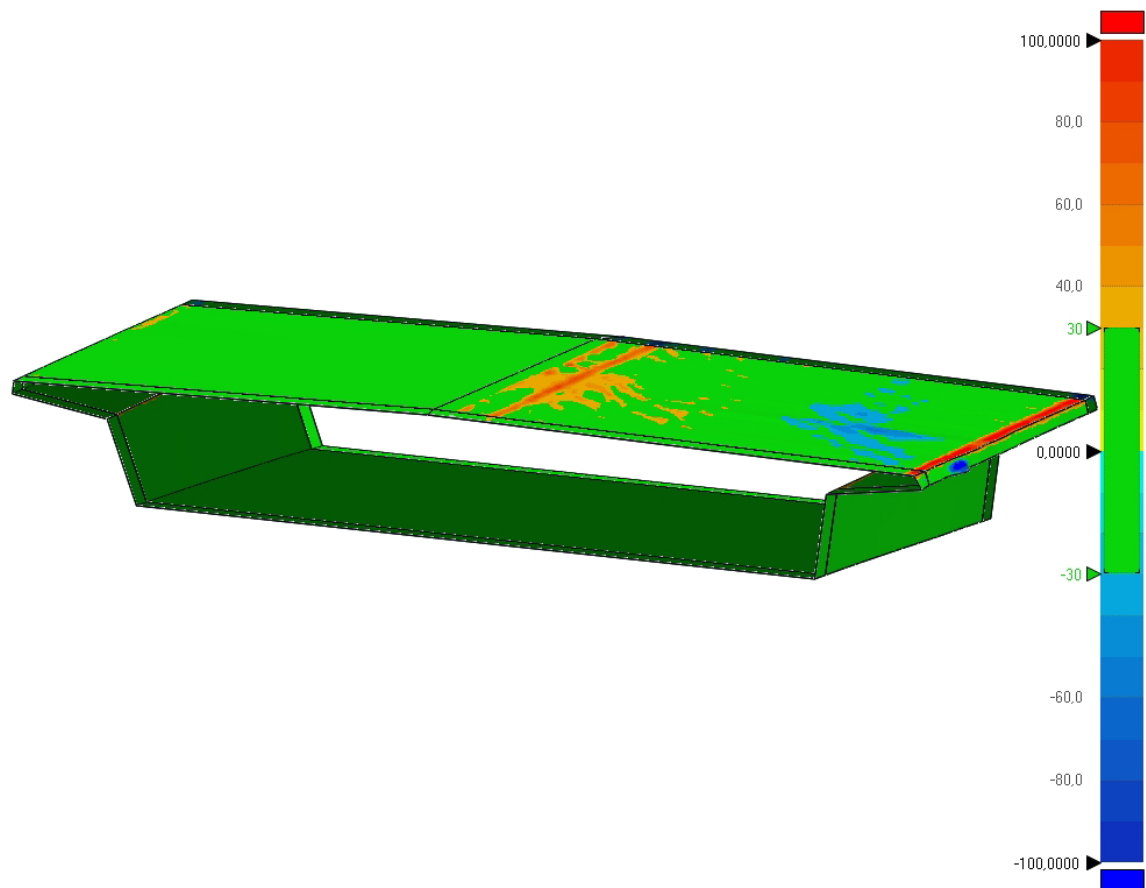
will cause, several comparisons of a single box girder must be made. A randomly picked box girder segment is analysed, and its result can be seen in the following figures and tables.

Figure 6.16 shows the deviation map of box segment 5. The analysed area is the part of the BIM model corresponding to the point cloud segment of Box 5, including the main part of the BIM box girder segment and part of the extensions on both sides. Most of the segment lies in the tolerance level of 30mm, displaying a green colour. The largest deviations can be seen in the middle and the outer edge of the top flange. Both of these areas are positions where a bridge railing was originally placed but was cut out of the point cloud during the early point cloud processing steps since the railing is not of structural importance. Thus, it is reasonable to consider these areas of the point cloud as irrelevant for the box girder comparison.

In Table 6.5, as well as in the histogram in Figure 6.17, key deviation statistics for the accuracy analysis of box girder segment 5 is presented.

The average gap distance calculated from the  $\pm$ Avg values is 6,1410mm, i.e. much lower than the similar value for the complete box girder. The standard deviation for box girder 5 is also lower than for the complete box girder, meaning that the deviations in general lies closer to the average and that the fit between the reference and measured data is better for this segments than some of the other segments.

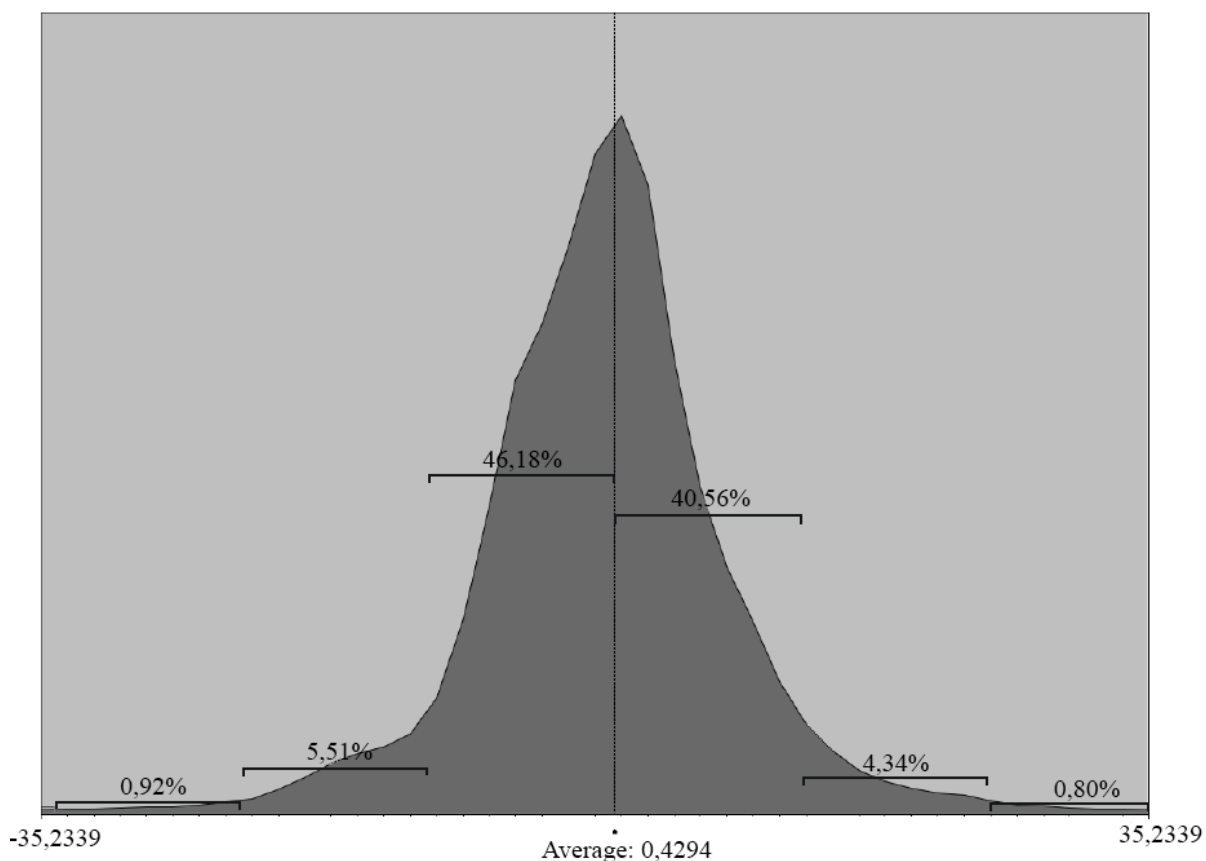
The histogram in Figure 6.17 is a normal distribution curve (Gauss' Bell curve) of the deviations, showing the average value in the centre, and 87% of the points laying within  $\pm 1$  standard deviation of 11,6mm. With a tolerance of 30mm, over 96% the measured data will have an accepted gap distance.



**Figure 6.16:** Deviation map of box girder segment 5

**Table 6.5:** Deviation statistics of box girder segment 5

Avg.	0,4294mm
+Avg.	6,9769mm
-Avg.	-5,3051mm
Std.Dev.	11,6015mm
Within Tol. (%)	95,2279



**Figure 6.17:** Histogram of the variation in deviation for box girder segment 5, with standard deviation indicators

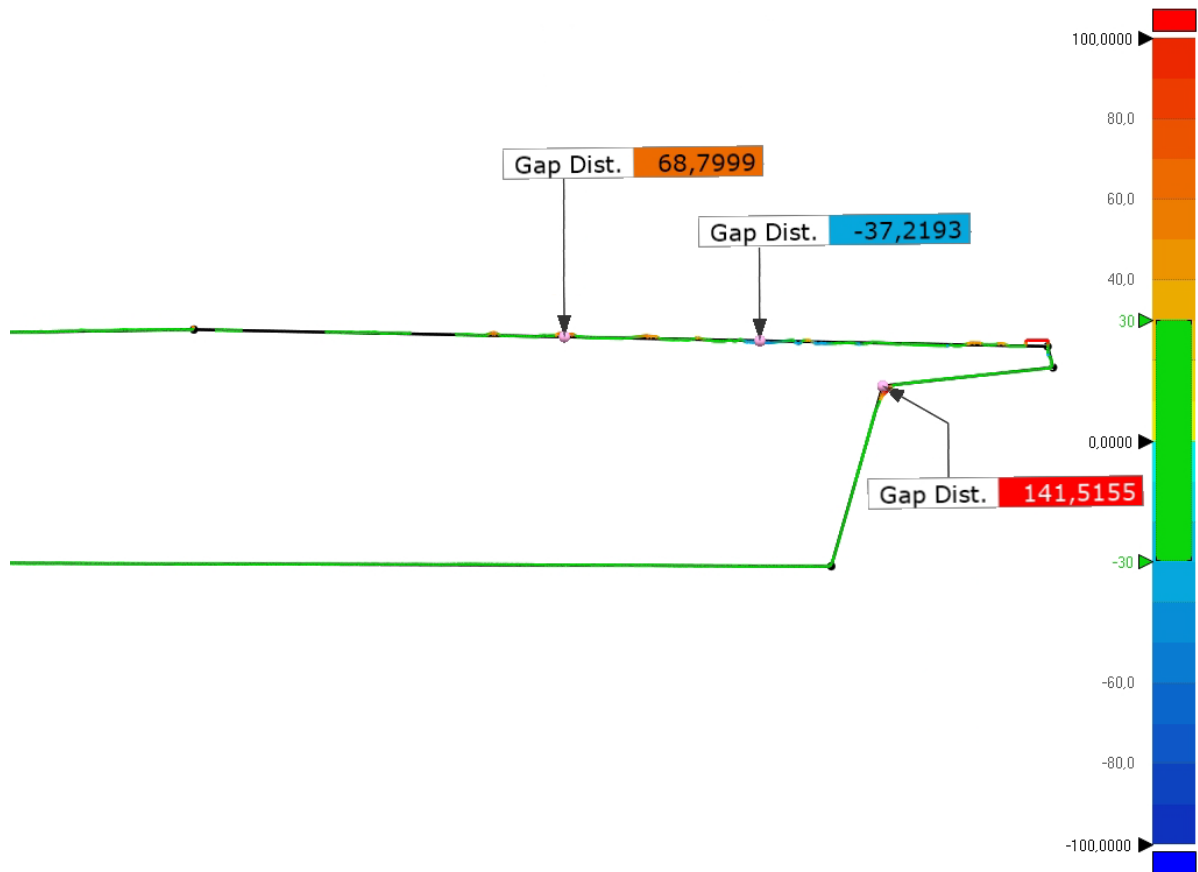
Figure 6.18 shows a 2D comparison of the one half of box girder 5 with the most deviations. **(a)** shows the real gap distance, while in **(b)** its corresponding distance indicators are multiplied by 10 to provide a better visualisation.

Other than deviations caused by noise coming from the railing point cloud, two of the three biggest gap distance lies on the top flange of the box girder. In general, the top flange is responsible for a big part of the points outside the tolerance level. The reason for this is a combination of less accurate scan data from uneven pavement and the slightly less accurate feature fitting caused by the small angle in intersection point 5, see Figure 4.3, both described in Section 4.

The biggest gap distance, except the position of railings, is at the rounded corners which are not perfectly modeled, as the generated intersection points create sharp corners instead of rounded, as remarked in the visual analysis previously done in Revit. A deviation of 141,5mm at intersection point 8 occurs for this reason. However, the deviation is confined

to a small area and will not be of a significant impact for the rest of the model.

It is important to note that the presented gap differences display the single biggest gap differences in the model, and that approximately 95% of the data is well within the accepted level of tolerance, see Table 6.5.



(a) actual deviation



(b) 10x actual deviation

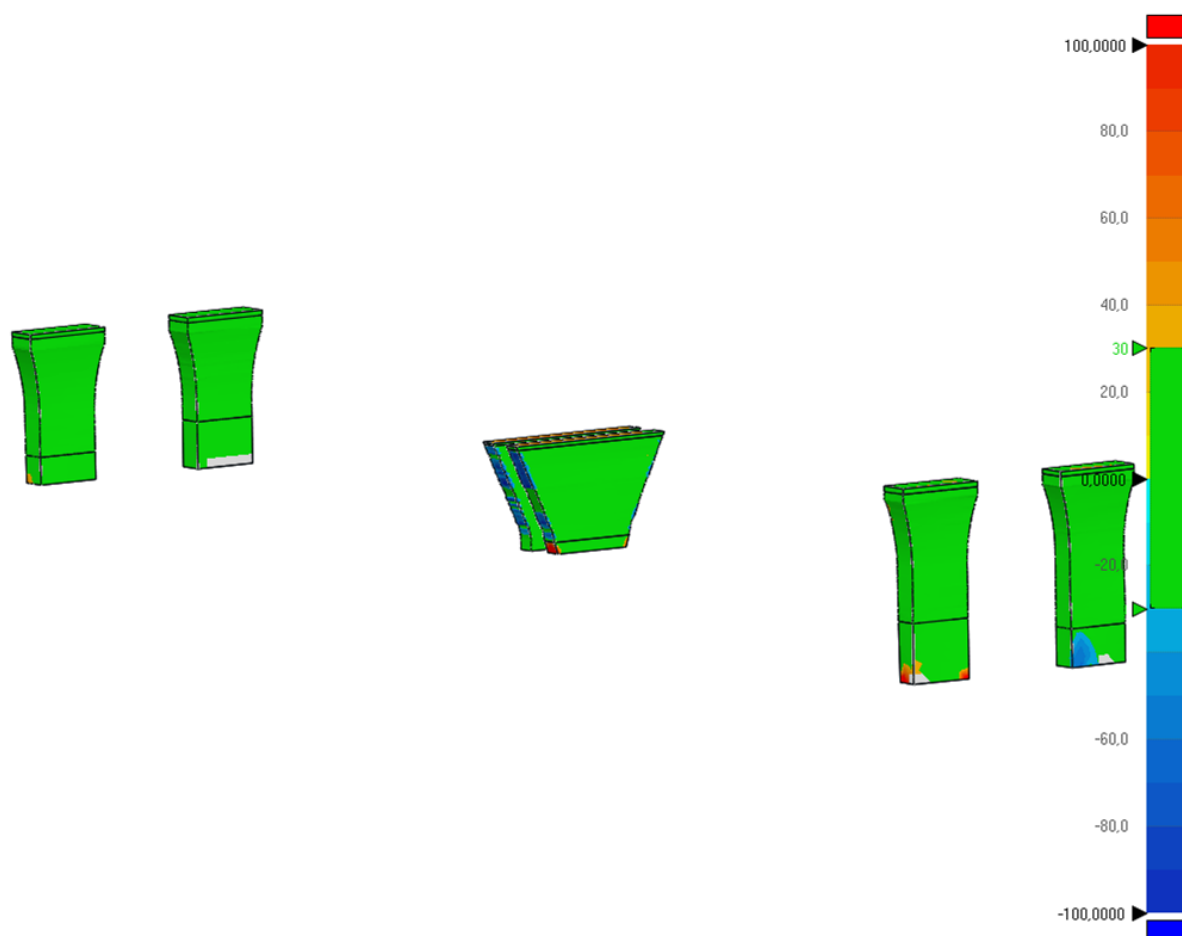
Figure 6.18: 2D analysis of box girder segment 5



### 6.2.2 Pier Column

The deviation map, Figure 6.19 and the deviation statics, Table 6.6, displays that the reverse BIM model of the pier columns generally lies inside the tolerance level of  $\pm 30\text{mm}$ , see Table 6.1. The average deviation of the pier column lies approximately 1mm under or behind the measured data, unlike the average deviation for the complete bridge and the box girder both having the average deviation laying over or beside the measured data. The averages are all close to zero and the different sign have little impact. Taking the calculated average from the positive and negative gap distances, the value 7,6032mm is indicating a higher overall accuracy for the piers than for the box girders.

7,6mm average deviation is considered a very good approximation, and with a standard deviation at 18,6mm, the reverse BIM model of the pier columns lies closer to the point cloud data than that of the box girder. The other statistics in Table 6.6 supports this statement, by generally having values closer to zero deviation.



**Figure 6.19:** Deviation map of the pier columns

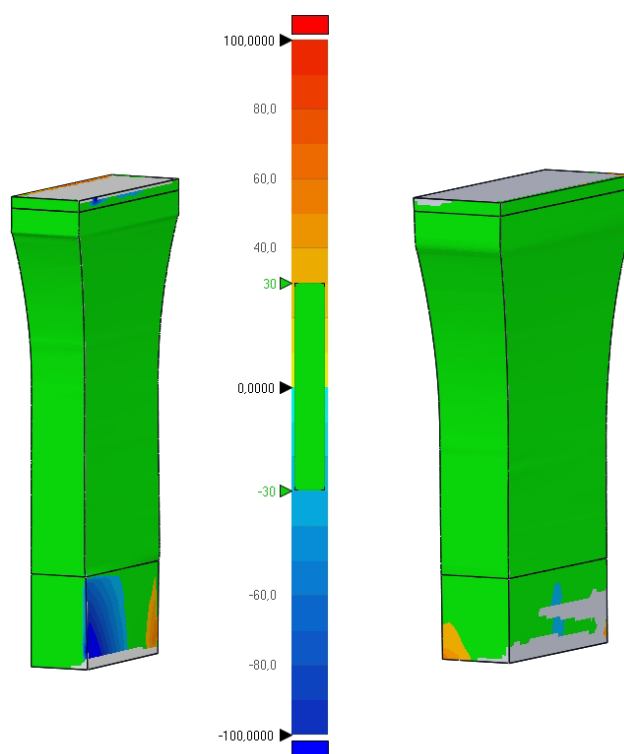
**Table 6.6:** Deviation statics of the pier columns

Avg.	-1,0483mm
+Avg.	6,5787mm
-Avg.	-8,6276mm
Std. Dev.	18,5816mm
Within Tol.(%)	92,4221
Beyond Tol.(%)	7,5779
Over Avg.(%)	46,3234
Under Avg.(%)	46,0987

A close-up of pier column 2 and 5, see Figure 6.20, reveals the bottom part of pier column 2 to be significantly worse than that of 5, i.e. the bottom extension of the pier columns are the most critical area of reverse modelling. This bottom extension is the only area of the slim pier columns (2, 3, 5, 6) where the data consistently lies outside the accepted tolerance level. In Figure 6.19, pier column 2 (to the right) is one of the two piers showing the largest deviations in the extension at the bottom of the piers.

Table 6.7 shows that pier column 2 has a significantly bigger standard deviation than pier column 5, likely caused by a more accurate lower extension. This extension is a trouble area for all the pier columns, and creates disruptions in the deviation statistics. The calculated average value of gap distance for pier column 2 is 9,5212mm, i.e it is increasing the overall average for all the piers combined, while pier column 5 has an average of 7,5564mm, decreasing the overall value.

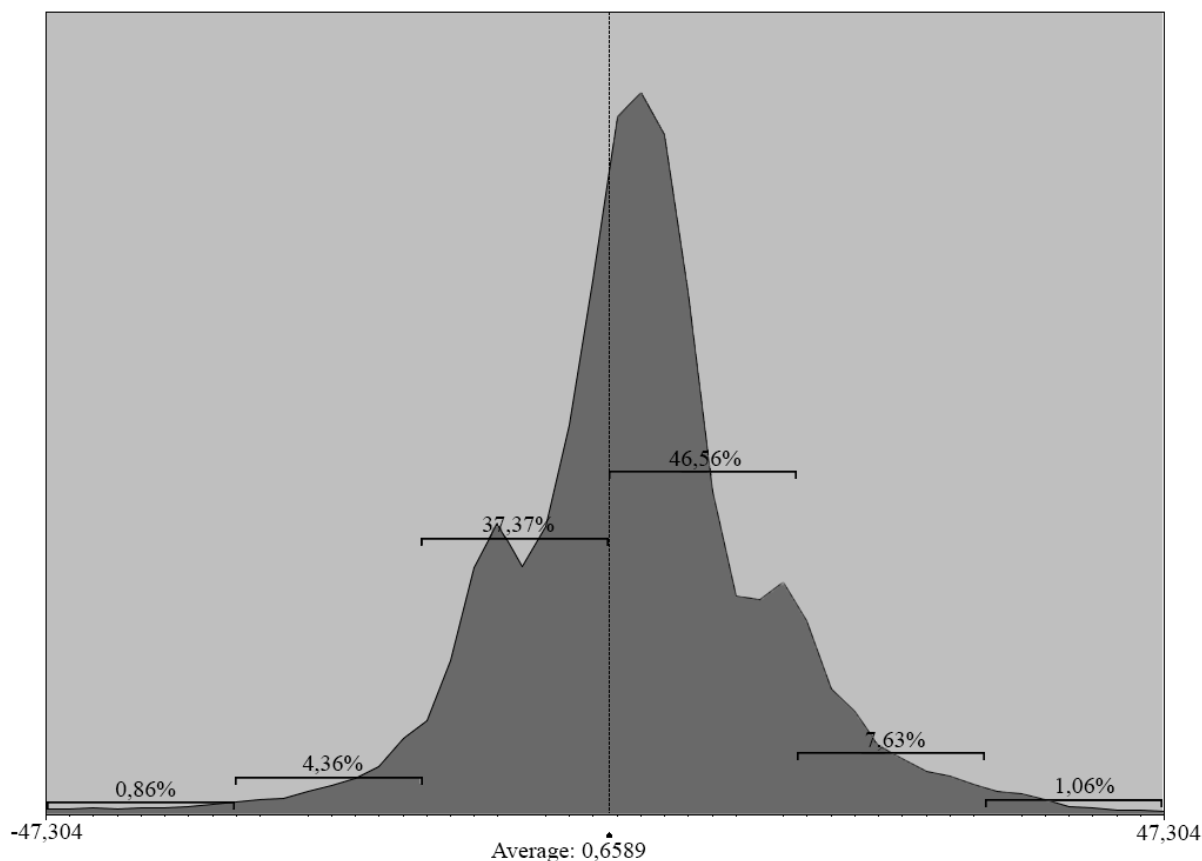
The histogram for pier column 2, presented in Figure 6.21, shows the distribution of gap distance. With the standard deviation at 15,6mm, approximately 84% of the points lay within  $\pm 1$  Std.Dev. and almost 97% of the compared points lay within the tolerance range. It is clear that most of the data have a good fit and that most of the remaining three percent outside the tolerance is caused by inaccuracies in the bottom part extension.



**Figure 6.20:** Deviation map of pier column 2 (left) and 5 (right)

**Table 6.7:** Deviation statistics of pier column 2 and 5

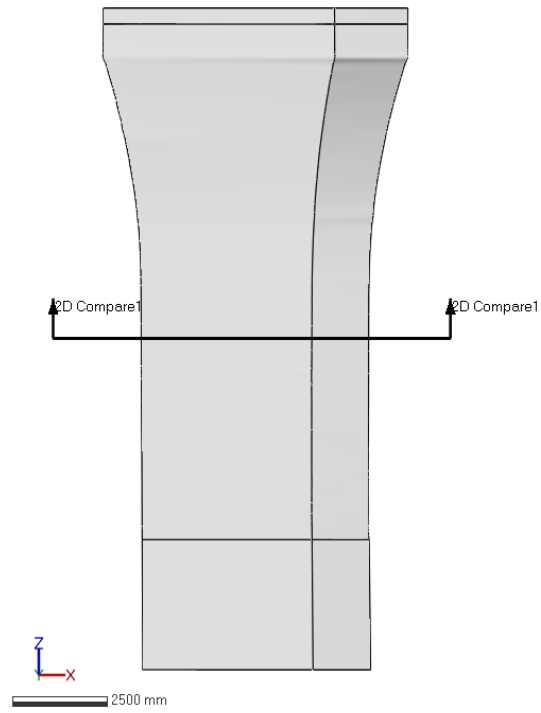
	Pier 2	Pier 5
Avg.	-0,6589mm	1,767mm
+Avg.	8,4657mm	8,3505mm
-Avg.	-10,5767mm	-6,7622mm
Std.Dev.	15,5623mm	11,2753mm
Within Tol. (%)	96,8634	96,8252



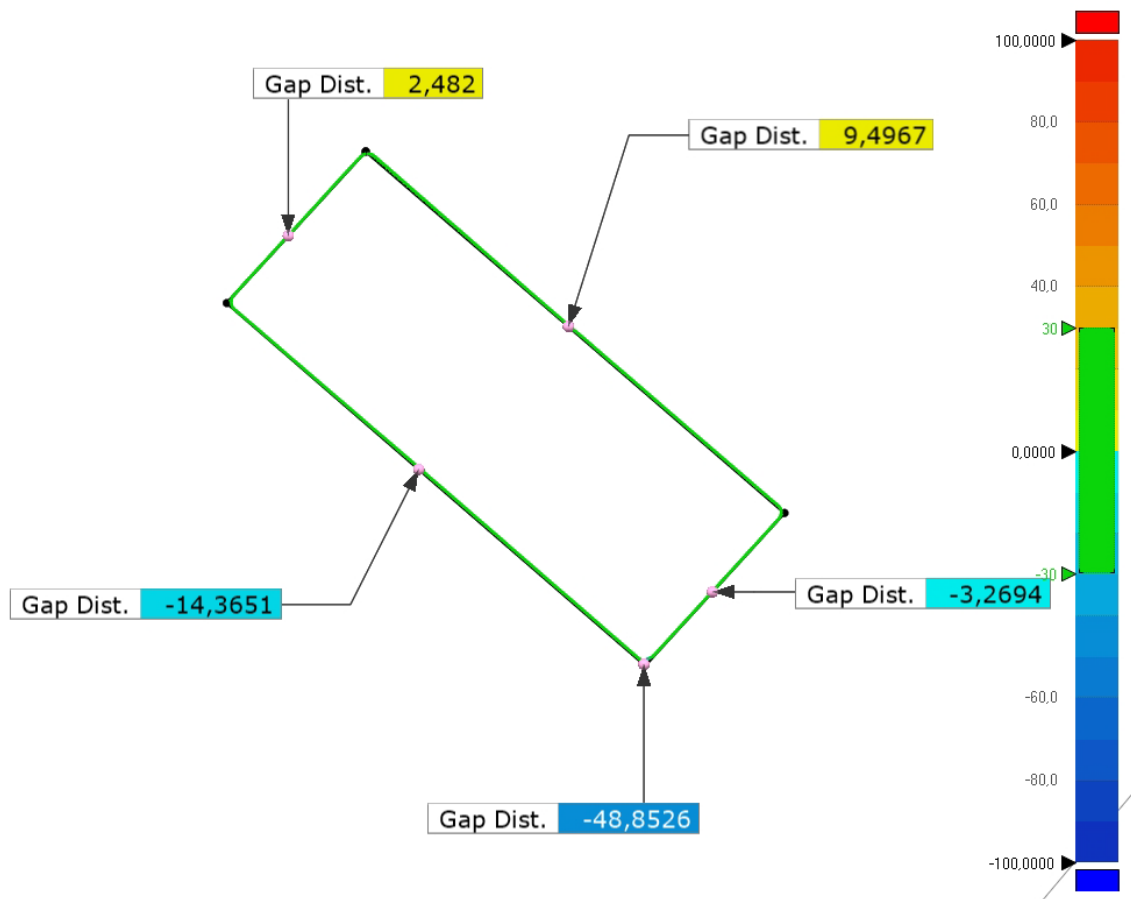
**Figure 6.21:** Histogram of the variation in deviation for pier column 2, with standard deviation indicators

A 2D comparison analysis for pier column 2 is carried out in xy-direction in the cross-section slice shown in Figure 6.22. The 2D comparison in Figure 6.23, show a pier that is well within the set tolerance range, except for the corners. As described in the visual comparison section, the method of reverse modelling in this thesis does not model the rounded corners. In the 2D analysis, the largest gap differences are found in the corners and are the only place where the 2D analysis of the pier BIM model does not satisfy the tolerance level of  $\pm 30mm$ . The gap distance in the corner is 48,85mm.

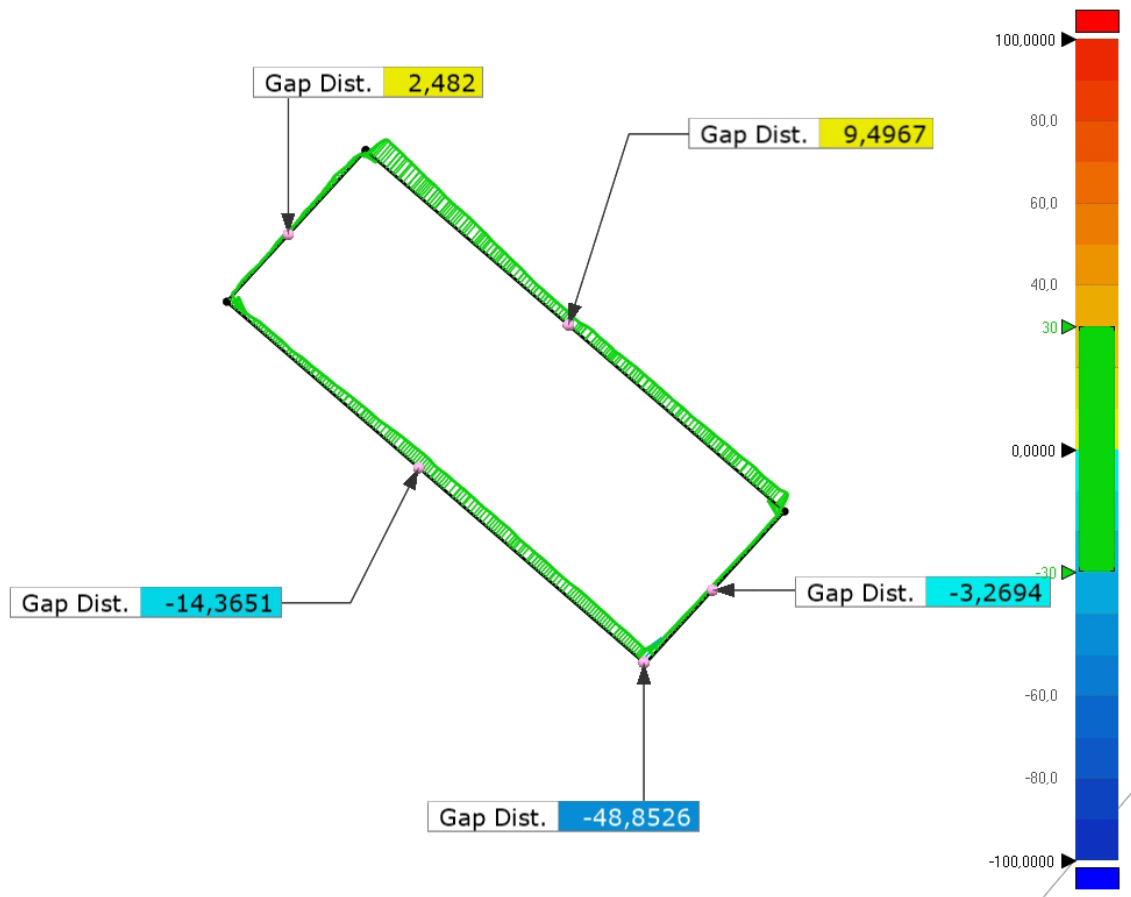
In addition to the largest gap distance in the corner, four more gap distances are displayed in the 2D comparison figures, one at each side of the pier column. These gap distances do not reveal any large local deviation, but are deviations caused by the alignment of the reference and measured data. The deviations on opposite sides almost cancel each other out. With a slightly different positioning at the assembly of the bridge, the fit would be almost perfect in these areas.



**Figure 6.22:** Position of 2D analysis for Pier 2



(a) Actual deviation

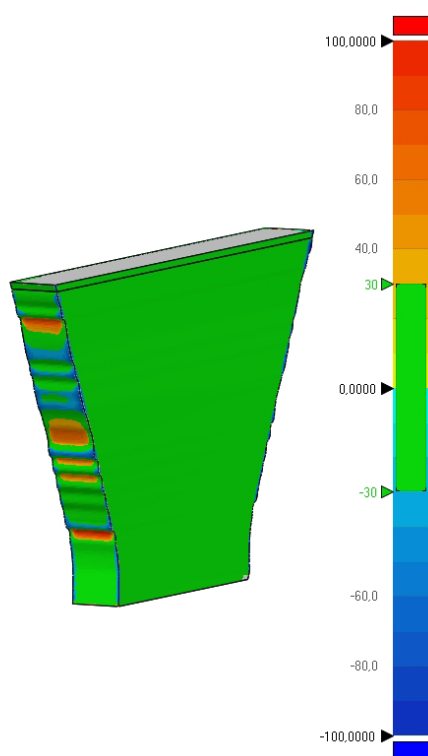


(b) 10x deviation

Figure 6.23: 2D analysis of Pier 2

As the Dade Bridge has two different pier column geometries, also the deviation analysis of the mid-span piers (pier column 3 and 4) is presented. The deviation map of pier column 4 in Figure 6.24, gives the same deviation result as discovered with the visual comparison. A large portion of the pier short-side has deviation outside the tolerance level. Deviation both over and under the tolerance level in such a big scale as seen in the deviation map, indicate the wavy side discussed in Section 6.1.2.

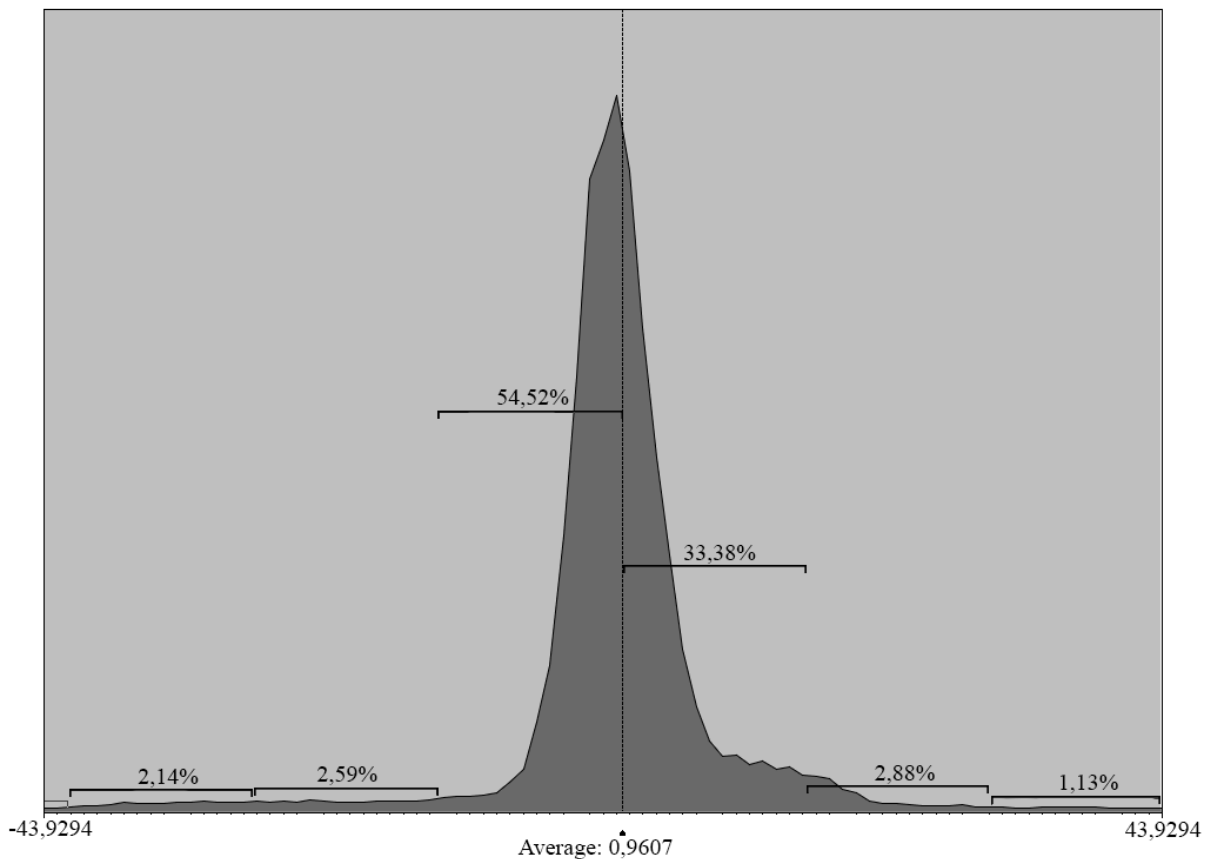
Despite the short side of pier column 4 showing large deviations, the statistics, Table 6.8 and histogram of Figure 6.25, show that a larger portion of the measured data lies close to the BIM model. The calculated average is 6,9437mm, and the standard deviation is 14,3mm, are smaller than for pier column 2, and the histogram shows a much higher collection of points close to the average deviation. The proportion of pier column 4 within  $\pm 1$  standard deviation is 87%, i.e. higher than that of Pier 2. However, the amount of points within the tolerance is slightly lower at 93,4%. This mean there are a higher number of points close to the average deviation, but that some large deviation is found to be outside the tolerance level.



**Figure 6.24:** Deviation map of pier column 4

**Table 6.8:** Deviation statistics of pier column 4

Avg.	0,9607mm
+Avg.	7,8885mm
-Avg.	-5,9988mm
Std.Dev.	14,3229mm
Within Tol. (%)	93,4015

**Figure 6.25:** Histogram of the variation in deviation for pier column 4, with standard deviation indicators

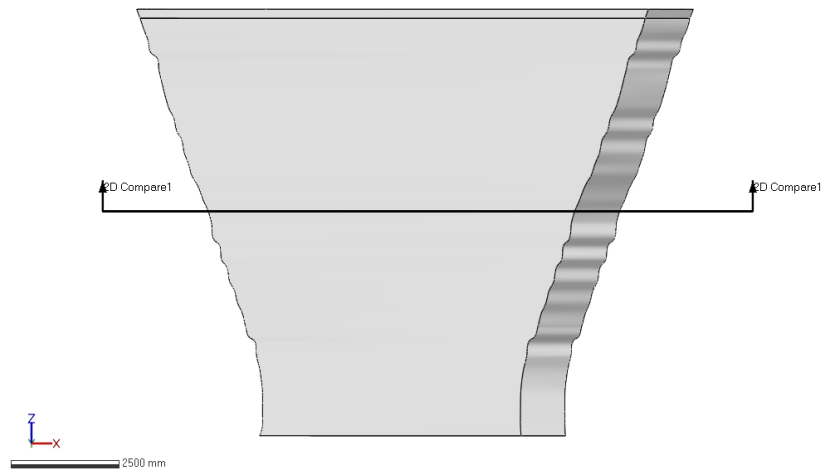
The 2D analysis of pier column 4 shows much of the same results as the 3D analysis did. The 2D analysis is performed at the cross-section highlighted in Figure 6.26. At this position, the measured data is laying on the outside of the BIM model. This is clearly shown in the 2D analysis in Figure 6.27. The two short sides of the pier column are displaying a deviation of approximately 80mm, over 2,5 times as large as the maximum accepted deviations of 30mm.

Because of the slight misalignment discussed earlier, the gap difference for the wide side of pier column 4 have the same problem as for pier column 2. A more accurate assembly

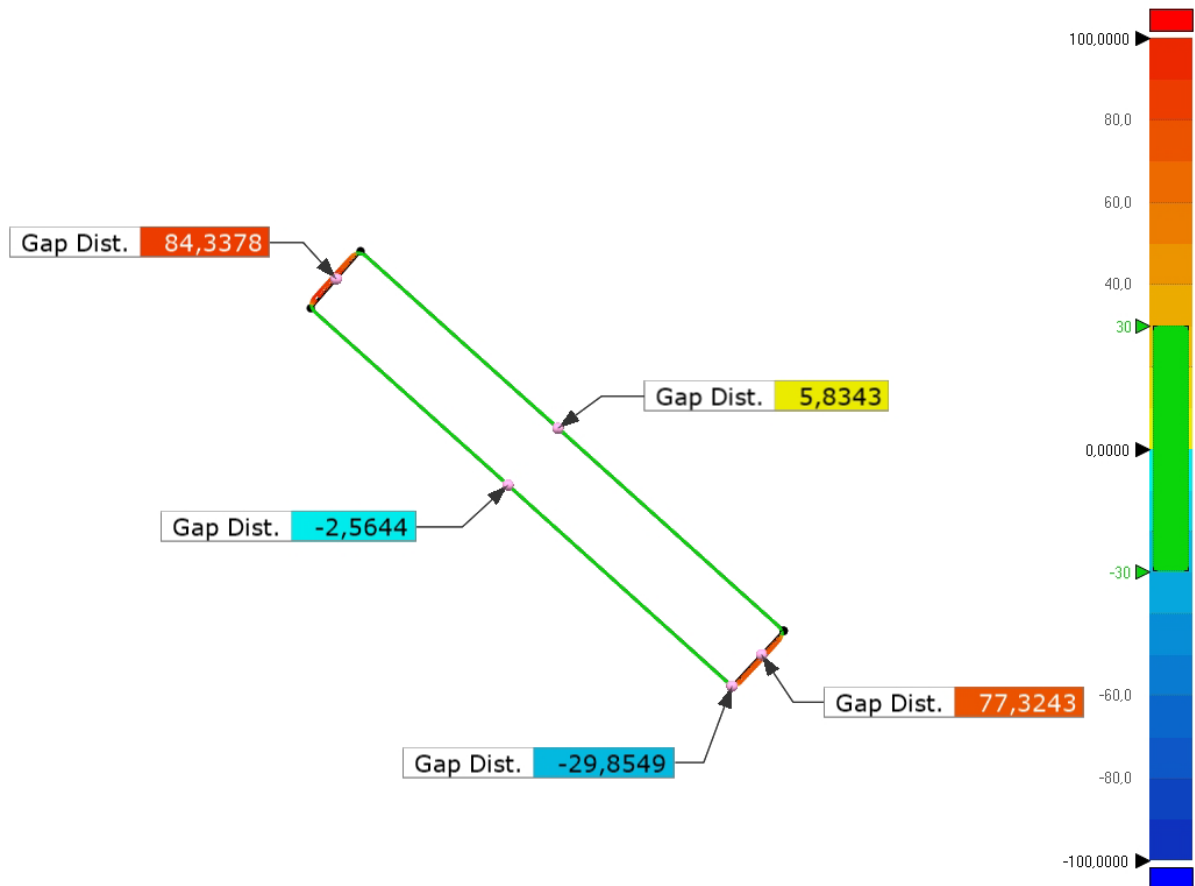


of the pier would have close to zeroed out the gap distance on the wide side.

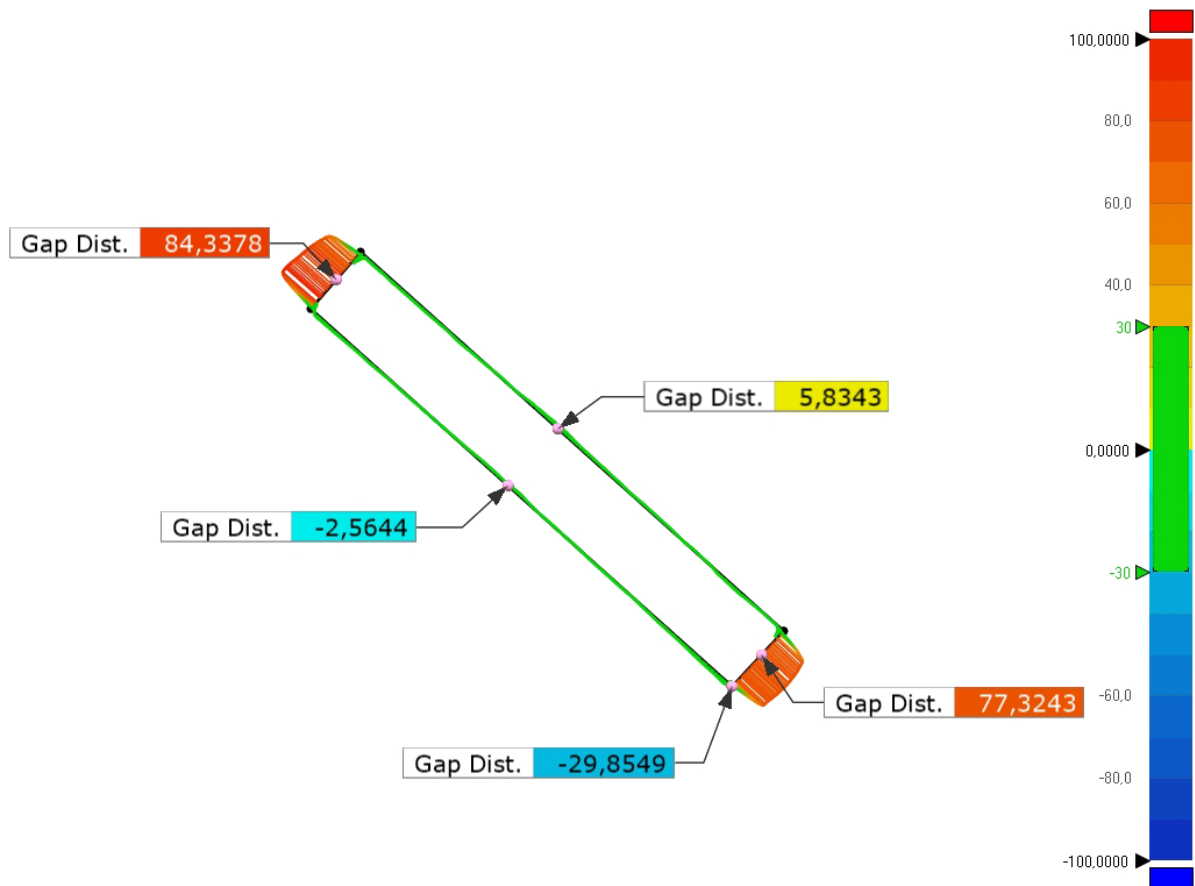
The gap distance in the corner indicates less rounded corner for these mid-span piers than for the end piers. This is the same problem, but a smaller radius on the rounded corner is resulting in a smaller gap distance between the reverse BIM model and the point cloud.



**Figure 6.26:** Position of 2D analysis for Pier 4



(a) Actual deviation

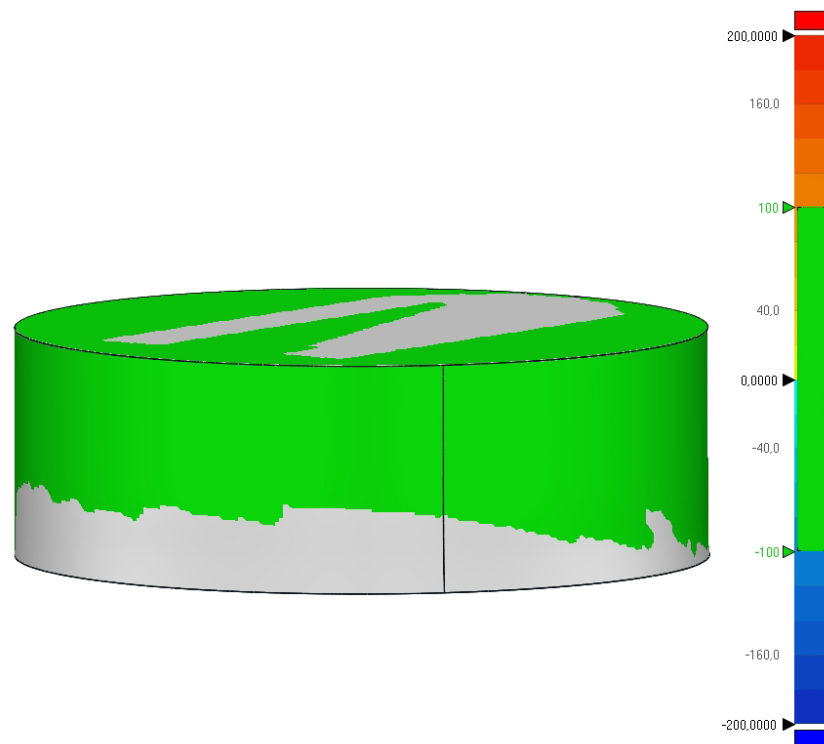


(b) 10x deviation

Figure 6.27: 2D analysis of pier column 4

### 6.2.3 Pier Foundation

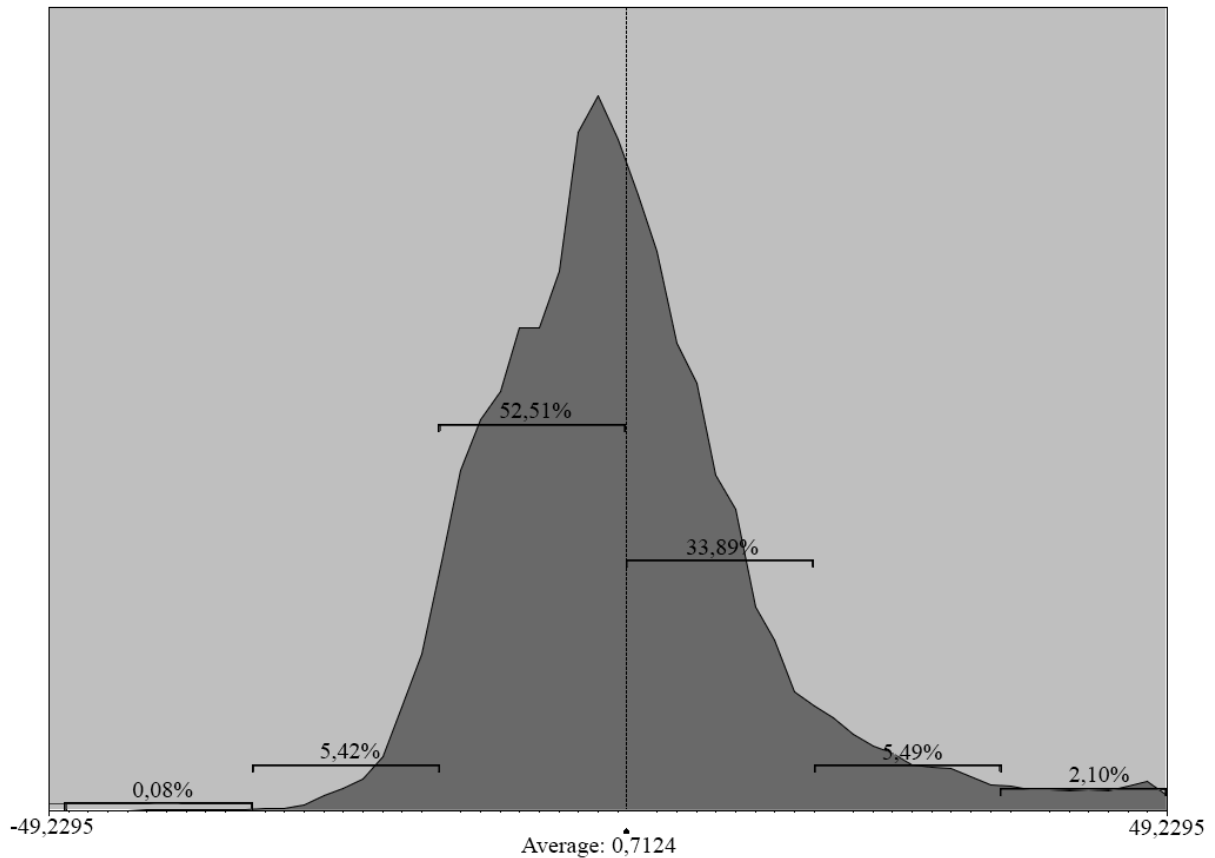
The pier foundation have a tolerance level at  $\pm 100\text{mm}$ . The results from the deviation map, Figure 6.28, and deviation statistics, Table 6.9, indicates that the accuracy level is far within this requirement with a calculated average gap distance of 9,4574mm. The results are showing quite similar statistics as for the other bridge parts, with a standard deviation at 16,2mm and approximately 97% of the points within a gap distance of 30mm. 99,8% of the measured data is within the tolerance level for foundations at  $\pm 100\text{mm}$ . The 0,2% of measured data outside of the tolerance range is likely a small point cluster of noise which was not removed in the preprocessing phase.



**Figure 6.28:** Deviation map of pier foundation

**Table 6.9:** Deviation statistics of pier foundation

Avg.	0,7124mm
+Avg.	11,1199mm
-Avg.	-7,7949mm
Std.Dev.	16,188mm
Within Tol. (%)	99,8

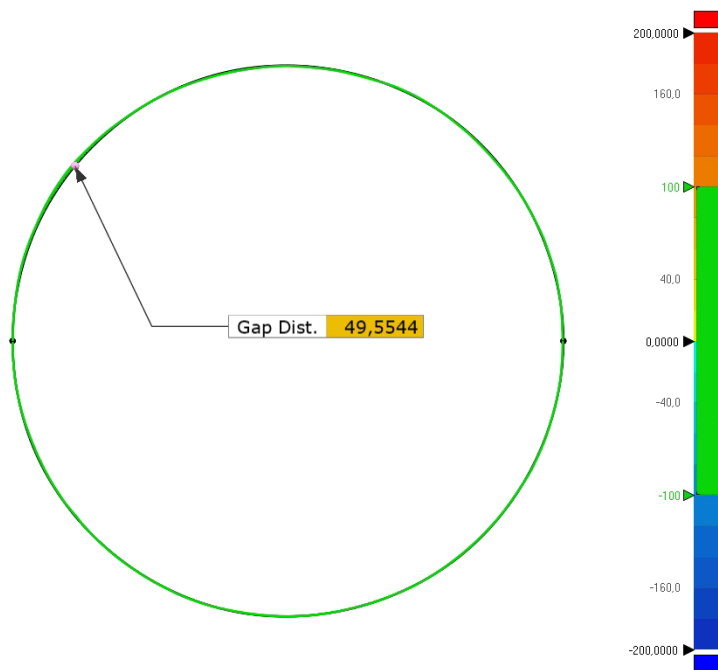


**Figure 6.29:** Histogram of the variation in deviation for the pier foundation, with standard deviation indicators

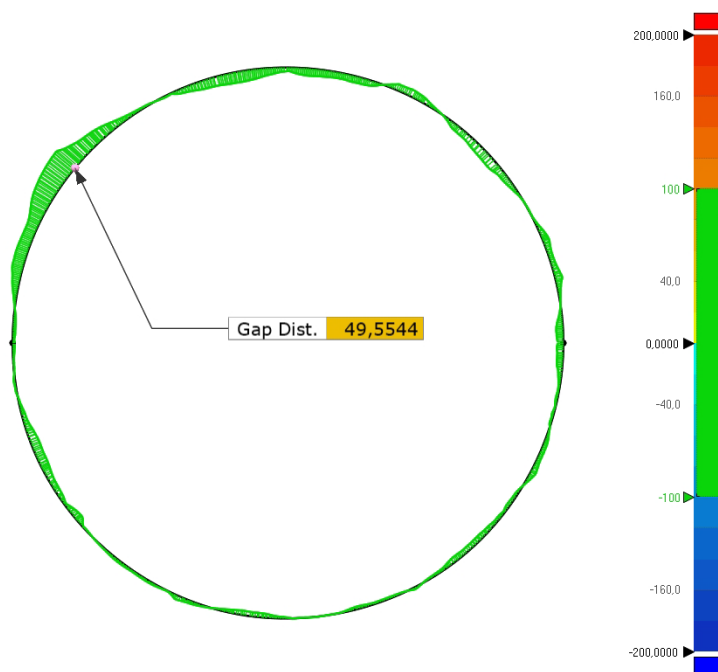
The 2D comparison of the foundation reveals one part where the gap distance is bigger than for the rest of the foundation. The largest gap distance in this area is 49,5mm. Based on the knowledge of the feature fitting of this cylindrical foundation, it is safe to say that the BIM model is a perfectly round circle. The area with the deviation is the same as discovered and showed in Figure 6.13, and caused by a noisy scan data. The noise cause some deviation in the analysis, but the reverse modelling method creates a very good approximation of the real bridge pier foundation.



(a) Analysis position



(b) Actual deviation



(c) 10x deviation

Figure 6.30: 2D analysis of the pier foundation

## 6.3 Summary of Accuracy Analysis

Overall, the accuracy analysis shows quite promising results. Some deviations are larger than the accepted tolerance level for the specific bridge part. However, the geometry is identical, and approximately 97% of the reverse BIM model lies within  $\pm 30mm$  deviation. The largest deviations can be found in the parts of the BIM model where an extension is performed, and are not representative for the feature fitting and reverse BIM modelling. A complete 3D scan of a bridge without occlusions or sources of noise would eliminate the deviations at the extended parts and would result in an even more accurate BIM model.

The average overall deviation of the reverse model of the Dade Bridge is found to be just 11,9mm, i.e. well within the set tolerance levels for concrete bridges in Norway. Moreover, the deviation analysis is negatively affected by a small number of larger deviations due to an incomplete point cloud, meaning that the overall accuracy is even better than the values which is presented in this section.

## 7 Conclusion and Prospect

In this master's thesis, the process of reverse modelling a physical structure and its immediate terrain to a digital, 3D BIM model based on point clouds gathered from terrestrial 3D laser scanners has been detailed. State-of-the-art BIM software and data collection methods were implemented, resulting in an automatic process of generating an as-built, reverse 3D BIM model of the box girder bridge "Dade Bridge" in Chuxiong, China.

By automating the reverse modelling method, the cost and time to develop as-built BIM models are significantly reduced compared to manual reverse modelling, while also providing a highly accurate reverse BIM model. In the proposed modelling method, several new technologies within the AEC-industry have been used for the creation of a reverse bridge model. A digital, 3D reconstruction of a physical bridge was generated using a combination of 3D laser scanners (Leica Scanstation P50), point cloud processing software (Geomagic Control X), BIM software (Revit) and algorithms from both numerical (Matlab) and visual (Dynamo) programming software with parametric capabilities.

The scan data collected with the TLS was initially merged in Matlab using an improved ICP algorithm based on k-d tree data structure. Subsequently, the merged point cloud of the superstructure and substructure were segmented into three structural parts; box girders, pier columns and a pier foundation. Each structural part was further segmented into smaller point clouds, representing single piers, sections of box girders and a standalone pier foundation. Advanced algorithm scripting was successfully used to extract the characteristic parameters of the aforementioned point cloud segments, and visual programming software was subsequently used to generate the reverse model based on the extracted data from the feature fitting algorithm. Lastly, the generated reverse model of the bridge was exported to a BIM software along with a reverse model of its immediate terrain, also generated from point cloud data from the aforementioned TLS.

An accuracy analysis is used to check the consistency of the BIM model. The accuracy analysis of the reverse model in comparison to its reference point cloud indicates promising results regarding the overall accuracy of the generated geometry. Approximately 97% of the complete reverse BIM model lies within a tolerance of  $\pm 30$  mm, while the average

overall deviation is only 11,9 mm. The actual accuracy level is in reality even better than these numbers indicate, as large deviations occur where the point cloud is missing which in turn increases the average deviation values. From the accuracy analysis, it is apparent that the geometrical similarities between the reverse 3D model and the scanned structure are substantial, only showing deviation outside the set tolerance level at especially vulnerable positions with quickly changing geometry as well as positions with occluded point cloud data.

The proposed method of reverse modelling shows promising results with a great possibility for adaption to different bridge structures and similar structural shapes. Automating the process of developing as-built BIM models will improve the inspection method and possibly increase the safety of the huge amount of bridges around the world. It is apparent that reverse engineering has a promising future within the AEC-industry, not only providing digital construction information for ageing structures, but also as an integral part of an improved inspection method.

Although the proposed reverse modelling method in this thesis results in a highly accurate digital representation of a physical structure, future studies should be done to improve the method and reduce the margin of error even further. Some key topics to further improve the proposed reverse modelling method are:

- Enhanced building information by integrating characteristic properties within the generated solids such as, but not limited to: material properties, inspection history and expected service life.
- Improving the proposed automated process of cutting occluded scan data.
- Develop a more sophisticated extension algorithm to improve the accuracy of the extension models.
- Improving how the algorithm copes with geometrical changes.



## References

- [1] A. Baik, A. Alitany, J. Boehm, and S. Robson, “Jeddah historical building information modelling "jhbim" ndash; object library,” *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. II-5, pp. 41–47, 2014. [Online]. Available: <https://www.isprs-ann-photogramm-remote-sens-spatial-inf-sci.net/II-5/41/2014/>
- [2] A. S. of Civil Engineers (ASCE), “2017 infrastructure report card,” Report, 2017.
- [3] R. foundation, “Gb tables- road bridge maintenance 2017,” Bridge Maintenance data GB, 2015/16, 2017. [Online]. Available: [https://www.racfoundation.org/wp-content/uploads/2017/11/RAC\\_Foundation\\_Bridge\\_Maintenance\\_GB\\_2015-16.pdf](https://www.racfoundation.org/wp-content/uploads/2017/11/RAC_Foundation_Bridge_Maintenance_GB_2015-16.pdf)
- [4] S. Nilsen, Øyvind Engan, M. G. Norman, and J. Braastad, “Vegvesenet bryter reglene på 1 av 2 broer,” 2017. [Online]. Available: <https://www.vg.no/spesial/2017/de-forsomte-broene/inspeksjoner/>
- [5] NBS, *National BIM Report 2019*.
- [6] R. Staiger, “Terrestrial laser scanning technology, systems and applications,” in *2nd FIG Regional Conference Marrakech, Morocco*, vol. 1, 2003, pp. 1–10.
- [7] “Digital twins and sensor monitoring - dnv gl,” Jan 2019. [Online]. Available: <https://www.dnvgl.com/expert-story/maritime-impact/Digital-twins-and-sensor-monitoring.html>
- [8] A. Ltd, “Brent delta – 3d laser scanning – decommissioning/asset retirement,” Jun 2017. [Online]. Available: <http://www.astd.co.uk/case-study/brent-delta-3d-laser-scanning/>
- [9] J. A. Langlo, S. Bakken, O. Karud, E. Malm, and B. Andersen, “Måling av produktivitet og prestasjoner i byggenæringen,” *Trondheim: SINTEF Byggforsk*, p. 7, 2013.
- [10] H. M. Kjer and J. Wilm, “Evaluation of surface registration algorithms for pet motion correction,” B.S. thesis, Technical University of Denmark, DTU, DK-2800 Kgs. Lyngby, Denmark, 2010.
- [11] S. Vegvesen, “Håndbok n440 bruregistrering,” 2014.
- [12] I. E. Sutherland, “Sketchpad, a man-machine graphical communication system,” Ph.D. dissertation, 1963.
- [13] *Leica ScanStation P50 data sheet*, Ds 869145 ed., Leica Geosystems.
- [14] S. vegvesens Prosesskode, “2: General specifications,” *Håndbok R762. Directorate of Public Roads and Transport Department*, 2015.
- [15] “3d compare statistics control x,” May 2020. [Online]. Available: [https://support.3dsystems.com/s/article/3D-Compare-Statistics-Control-X?language=en\\_US&name=3D-Compare-Statistics-Control-X](https://support.3dsystems.com/s/article/3D-Compare-Statistics-Control-X?language=en_US&name=3D-Compare-Statistics-Control-X)
- [16] J. Ayesha, “Infrastructure: Meaning and importance: Economic growth: Economics,”

- Mar 2018. [Online]. Available: <http://www.economicdiscussion.net/infrastructure/infrastructure-meaning-and-importance-economic-growth-economics/30223>
- [17] H. Editors, “Golden gate bridge,” Feb 2015. [Online]. Available: <https://www.history.com/topics/landmarks/golden-gate-bridge>
- [18] S. vegvesen Vegdirektoratet, “Håndbok v441 bruinspeksjon,” 2019.
- [19] Vegdirektoratet, “Gjennomgang av bruforvaltninga i statens vegvesen,” Statens Vegvesen, Report, apr 2018.
- [20] B. Riveiro, H. González-Jorge, M. Varela, and D. V. Jauregui, “Validation of terrestrial laser scanning and photogrammetry techniques for the measurement of vertical underclearance and beam geometry in structural inspection of bridges,” *Measurement*, vol. 46, no. 1, pp. 784–794, 2013.
- [21] S. Jones, D. Laquidara-Carr, A. Lorenz, B. Buckley, and S. Barnett, “The business value of bim for infrastructure 2017,” *SmartMarket Report*, 2017. [Online]. Available: <https://www2.deloitte.com/content/dam/Deloitte/us/Documents/finance/us-fas-bim-infrastructure.pdf>
- [22] *Design Manual for Roads and Bridges*, Volume 3, highway structures: Inspection & maintenance, section 1 part 4 bd 63/17 ed., Highways England et al., nov 2017.
- [23] “Inspeksjon og sikkerhet.” [Online]. Available: <https://www.vegvesen.no/fag/teknologi/bruer/Inspeksjonogsikkerhet>
- [24] S. Vegvesen, “Håndbok n400 bruprosjektering, prosjektering av bruer, ferjekaier og andre bærende konstruksjoner,” 2015.
- [25] E. Rodum, *Chapter 19 Bruer og kaier*. Norwegian Public Roads Administration, 2015, p. 277–288.
- [26] N. P. R. Administration, “Fra regioner til divisjoner,” Norwegian Public Roads Administration, Utredning om organisering av Statens vegvesen fra 1. januar 2020, 2020.
- [27] “Who we are.” [Online]. Available: <https://www.cen.eu/about/Pages/default.aspx>
- [28] “Cen/tc 442 - building information modelling (bim).” [Online]. Available: [https://standards.cen.eu/dyn/www/f?p=204:7:0:::FSP\\_ORG\\_ID:1991542&cs=16AAC0F2C377A541DCA571910561FC17F](https://standards.cen.eu/dyn/www/f?p=204:7:0:::FSP_ORG_ID:1991542&cs=16AAC0F2C377A541DCA571910561FC17F)
- [29] R. Petrie, “openbim definition.” [Online]. Available: <https://www.buildingsmart.org/about/openbim/openbim-definition/>
- [30] *Industry Foundation Classes (IFC) for data sharing in the construction and facility management industries — Part 1: Data schema*, International Organization for Standardization European Standard ISO 16 739-1, Rev. 2018, 2018.
- [31] *Building construction — Organization of information about construction works — Part 3: Framework for object-oriented information*, International Organization for Standardization ISO Standard ISO 12 006-3, Rev. 2007, 2007.
- [32] *Building information modelling — Information delivery manual — Part 1:*

- Methodology and format*, International Organization for Standardization ISO Standard ISO 29 481-1, Rev. 2010, 2010.
- [33] *Organization and digitization of information about buildings and civil engineering works, including building information modelling (BIM) Information management using building information modelling Part 1: Concepts and principles*, European Commite For Standardization European Standard EN ISO 19 650-1, Rev. 2018 (E), 2018.
- [34] *Organization and digitization of information about buildings and civil engineering works, including building information modelling (BIM) Information management using building information modelling Del 2: Prosjektfasen*, European Commite For Standardization European Standard EN ISO 19 650-2, Rev. 2018 (E), 2018.
- [35] H. M. Bernstein, S. Jones, and J. Gudgel, "The business value of bim in china," *Dodge Data and Analytics*, Bedford, MA, 2015.
- [36] X. Xiong, A. Adan, B. Akinci, and D. Huber, "Automatic creation of semantically rich 3d building models from laser scanner data," *Automation in construction*, vol. 31, pp. 325–337, 2013.
- [37] P. Tang, D. Huber, B. Akinci, R. Lipman, and A. Lytle, "Automatic reconstruction of as-built building information models from laser-scanned point clouds: A review of related techniques," *Automation in construction*, vol. 19, no. 7, pp. 829–843, 2010.
- [38] G. Vosselman, S. Dijkman *et al.*, "3d building model reconstruction from point clouds and ground plans," *International archives of photogrammetry remote sensing and spatial information sciences*, vol. 34, no. 3/W4, pp. 37–44, 2001.
- [39] I. Brilakis, M. Lourakis, R. Sacks, S. Savarese, S. Christodoulou, J. Teizer, and A. Makhmalbaf, "Toward automated generation of parametric bims based on hybrid video and laser scanning data," *Advanced Engineering Informatics*, vol. 24, no. 4, pp. 456–465, 2010.
- [40] F. Bosche, C. T. Haas, and B. Akinci, "Automated recognition of 3d cad objects in site laser scans for project 3d status visualization and performance control," *Journal of Computing in Civil Engineering*, vol. 23, no. 6, pp. 311–318, 2009.
- [41] P. Tang, E. B. Anil, B. Akinci, and D. Huber, "Efficient and effective quality assessment of as-is building information models and 3d laser-scanned data," in *Computing in Civil Engineering (2011)*, 2011, pp. 486–493.
- [42] N.-J. Shih and S.-T. Huang, "3d scan information management system for construction management," *Journal of construction engineering and management*, vol. 132, no. 2, pp. 134–142, 2006.
- [43] M. Ahmed, C. Haas, and R. Haas, "Toward low-cost 3d automatic pavement distress surveying: the close range photogrammetry approach," *Canadian Journal of Civil Engineering*, vol. 38, no. 12, pp. 1301–1313, 2011.
- [44] M. Golparvar-Fard, F. Pena-Mora, and S. Savarese, "D4ar—a 4-dimensional augmented reality model for automating construction progress monitoring data collection, processing and communication," *Journal of information technology in construction*, vol. 14, no. 13, pp. 129–153, 2009.

- [45] S. El-Omari and O. Moselhi, “Integrating 3d laser scanning and photogrammetry for progress measurement of construction work,” *Automation in construction*, vol. 18, no. 1, pp. 1–9, 2008.
- [46] F. Bosché, “Automated recognition of 3d cad model objects in laser scans and calculation of as-built dimensions for dimensional compliance control in construction,” *Advanced engineering informatics*, vol. 24, no. 1, pp. 107–118, 2010.
- [47] Y. Turkan, F. Bosche, C. T. Haas, and R. Haas, “Automated progress tracking using 4d schedule and 3d sensing technologies,” *Automation in construction*, vol. 22, pp. 414–421, 2012.
- [48] H. S. Park, H. Lee, H. Adeli, and I. Lee, “A new approach for health monitoring of structures: terrestrial laser scanning,” *Computer-Aided Civil and Infrastructure Engineering*, vol. 22, no. 1, pp. 19–30, 2007.
- [49] D. F. Laefer, L. T. Hong, and M. Fitzgerald, “Processing of terrestrial laser scanning point cloud data for computational modelling of building facades short running title: Terrestrial lidar to solid models.”
- [50] H. Guan, J. Li, Y. Yu, M. Chapman, and C. Wang, “Automated road information extraction from mobile laser scanning data,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 1, pp. 194–205, 2014.
- [51] G. Rocha, L. Mateus, J. Fernández, and V. Ferreira, “A scan-to-bim methodology applied to heritage buildings,” *Heritage*, vol. 3, no. 1, pp. 47–67, 2020.
- [52] R. Volk, J. Stengel, and F. Schultmann, “Building information modeling (bim) for existing buildings—literature review and future needs,” *Automation in construction*, vol. 38, pp. 109–127, 2014.
- [53] “Software.” [Online]. Available: <https://www.3dsystems.com/software>
- [54] D. Mejia, O. Ruiz-Salguero, J. R. Sánchez, J. Posada, A. Moreno, and C. A. Cadavid, “Hybrid geometry/topology based mesh segmentation for reverse engineering,” *Computers & Graphics*, vol. 73, pp. 47–58, 2018.
- [55] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, “Pointnet: Deep learning on point sets for 3d classification and segmentation,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660.
- [56] P. Arbeláez, B. Hariharan, C. Gu, S. Gupta, L. Bourdev, and J. Malik, “Semantic segmentation using regions and parts,” in *2012 IEEE conference on computer vision and pattern recognition*. IEEE, 2012, pp. 3378–3385.
- [57] A. Adán and D. Huber, “Reconstruction of wall surfaces under occlusion and clutter in 3d indoor environments,” *Robotics Institute, Carnegie Mellon University, Pittsburgh, PA CMU-RI-TR-10-12*, 2010.
- [58] I. Stamos, G. Yu, G. Wolberg, and S. Zokai, “3d modeling using planar segments and mesh elements,” in *Third International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT’06)*. IEEE, 2006, pp. 599–606.
- [59] B. LANDSFORENING, “Digitalt veikart for bygge-, anleggs-og eiendomsnæringen for økt bærekraft og verdiskapning,” *Obtained from www. digitaltveikart. no*, 2018.

- [60] “Statsbygg lanserer digibyg,” 2016. [Online]. Available: <https://www.statsbygg.no/Nytt-fra-Statsbygg/Nyheter/2016/Statsbygg-lanserer-Digibyg/>
- [61] “Geometrisk kontroll av bru - e16 sandvika - vøyenenga,” Oct 2018. [Online]. Available: <https://www.scansurvey.no/aktuelt/geometrisk-kontroll-av-bru-e16-sandvika-voyenenga/>
- [62] E. Espe, “Measurement of homogeneity of asphalt pavements,” 2016.
- [63] “Slik skal operaen bevarast dei neste 100 åra,” Apr 2016. [Online]. Available: <https://www.statsbygg.no/Nytt-fra-Statsbygg/Nyheter/2016/Slik-skal-Operaen-bevares-de-neste-100-ar/>
- [64] T. Oppikofer, “Overvåking av skred og andre skråningsprosesser med bakkebasert laserskanning,” 2016.
- [65] S. Engh, “Potensiale for anvendelse av lidar-data i glasiologi. interpolasjon av dtm, korreksjon av intensitetsverdier og automatisk kartlegging av bresprekker,” Master’s thesis, 2013.
- [66] “Høydedata og terrengmodeller for landområdene,” Mar 2019. [Online]. Available: <https://www.kartverket.no/data/hoydedata-og-terrengmodeller/>
- [67] K. H. Barmen, A. Aas Jakobsen *et al.*, “Muligheter og begrensninger ved bruk av droner til bruinspeksjon: erfaringer fra region øst (2015-2018),” 2019.
- [68] R. Lu, I. Brilakis, and C. R. Middleton, “Detection of structural components in point clouds of existing rc bridges,” *Computer-Aided Civil and Infrastructure Engineering*, vol. 34, no. 3, pp. 191–212, 2019.
- [69] M. Bassier, B. Van Genechten, and M. Vergauwen, “Classification of sensor independent point cloud data of building objects using random forests,” *Journal of Building Engineering*, vol. 21, pp. 468–477, 2019.
- [70] G. S. Cheok, S. Leigh, and A. Rukhin, “Calibration experiments of a laser scanner,” *NASA STI/Recon Technical Report N*, vol. 2, 2002.
- [71] D. D. Lichti, “Error modelling, calibration and analysis of an am-cw terrestrial laser scanner system,” *ISPRS journal of photogrammetry and remote sensing*, vol. 61, no. 5, pp. 307–324, 2007.
- [72] T. Voegtle, I. Schwab, and T. Landes, “Influences of different materials on the measurements of a terrestrial laser scanner (tls),” in *Proc. of the XXI Congress, The International Society for Photogrammetry and Remote Sensing, ISPRS2008*, vol. 37, 2008, pp. 1061–1066.
- [73] Y. Reshetyuk, “Self-calibration and direct georeferencing in terrestrial laser scanning,” Ph.D. dissertation, KTH, 2009.
- [74] O. W. Brown and C. H. Hugenholtz, “Quantifying the effects of terrestrial laser scanner settings and survey configuration on land surface roughness measurement,” *Geosphere*, vol. 9, no. 2, pp. 367–377, 2013.
- [75] W. Boehler, M. B. Vicent, A. Marbs *et al.*, “Investigating laser scanner accuracy,” *The*

- International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 34, no. Part 5, pp. 696–701, 2003.
- [76] S. Aung, R. Ngim, and S. Lee, “Evaluation of the laser scanner as a surface measuring tool and its accuracy compared with direct facial anthropometric measurements,” *British journal of plastic surgery*, vol. 48, no. 8, pp. 551–558, 1995.
- [77] L. Rui, “Research on applied talents training of urban and rural planning major based on 3d laser scanning technology,” *Journal of Beijing City University*, no. 2, p. 5, 2017.
- [78] K. Zhang, S.-C. Chen, D. Whitman, M.-L. Shyu, J. Yan, and C. Zhang, “A progressive morphological filter for removing nonground measurements from airborne lidar data,” *IEEE transactions on geoscience and remote sensing*, vol. 41, no. 4, pp. 872–882, 2003.
- [79] Y. Arayici, “An approach for real world data modelling with the 3d terrestrial laser scanner for built environment,” *Automation in construction*, vol. 16, no. 6, pp. 816–829, 2007.
- [80] “Leica scanstation p50 – long range 3d terrestrial laser scanner.” [Online]. Available: <https://leica-geosystems.com/products/laser-scanners/scanners/leica-scanstation-p50>
- [81] S. Li, J. Wang, Z. Liang, and L. Su, “Tree point clouds registration using an improved icp algorithm based on kd-tree,” in *2016 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*. IEEE, 2016, pp. 4545–4548.
- [82] S. Rusinkiewicz and M. Levoy, “Efficient variants of the icp algorithm,” in *Proceedings Third International Conference on 3-D Digital Imaging and Modeling*. IEEE, 2001, pp. 145–152.
- [83] J. Wilm, “Iterative closest point.” [Online]. Available: <https://www.mathworks.com/matlabcentral/fileexchange/27804-iterative-closest-point>
- [84] J. Corven, “Post-tensioned box girder design manual,” Tech. Rep., 2015.
- [85] C. V. N. Agency. (2019, jan) 15,000-ton expressway bridge rotates into position in yunnan. [Online]. Available: <https://www.youtube.com/watch?v=l9CM0uOWcoM>
- [86] I. Bucher, “Circle fit,” 2004. [Online]. Available: <https://se.mathworks.com/matlabcentral/fileexchange/5557-circle-fit>
- [87] G. D. of Techniques Road Infrastructure Department, *Bridge Inspection Manual*, Ministry of Public Works and Transportation, feb 2018.

# Appendices

<b>1</b>	<b>A: Matlab Script</b>	<b>164</b>
A1	ICP Algorithm . . . . .	164
A2	Box Girder 1 Algorithm . . . . .	180
A3	Box Girder 2-9 Algorithm . . . . .	196
A4	Box Girder 10-17 Algorithm . . . . .	213
A5	Box Girder 18 Algorithm . . . . .	229
A6	Pier Column 1, 2, 5, 6 Algorithm . . . . .	245
A7	Pier Column 3, 4 Algorithm . . . . .	258
A8	Pier Foundation Algorithm . . . . .	269
<b>2</b>	<b>B: Dynamo Workspaces</b>	<b>275</b>
B1	Reverse Modelling of Box Girders (#1 to #18) . . . . .	275
B2	Reverse Modelling of Pier Columns (#1 to #6) . . . . .	277
B3	Reverse Modelling of Pier Foundation . . . . .	279
<b>3</b>	<b>C: Geomagic Control X Report</b>	<b>281</b>
C1	Geomagic Control X Deviation Analysis Report . . . . .	281

# **1 A: Matlab Script**

## **A1 ICP Algorithm**



## Contents

---

- [ICP](#)
- [ICP function](#)

## ICP

---

```
clc;
clear all
close all

A=load('A.txt');
B=load('B.txt');
A=A(:,1:3);
B=B(:,1:3);

figure(1)
scatter3(A(:,1),A(:,2),A(:,3),'.y')
axis equal

figure(2)
scatter3(B(:,1),B(:,2),B(:,3),'.b')
axis equal

figure(3)
scatter3(A(:,1),A(:,2),A(:,3),'.y')
hold on
scatter3(B(:,1),B(:,2),B(:,3),'.b')
axis equal

D=A';
M=B';
n=10;
[Ricp, Ticp, ER, t] = icp(M, D, n, 'Matching', 'kdtree');
n=size(D,2);
Dicp = Ricp * D + repmat(Ticp, 1, n);
Dicp=Dicp';
fid=fopen('C:\Users\Eier\Desktop\Matlab master\happy ICP\K.txt','w');
for i=1:size(Dicp,1)
    fprintf(fid, '%.4f %.4f %.4f\r\n',Dicp(i,1),Dicp(i,2),Dicp(i,3));
end
```

## ICP function

```
% Jakob Wilm (2020). Iterative Closest Point (https://www.mathworks.com/matlabcentral/fileexchange/27804-iterative-closest-point),  
% MATLAB Central File Exchange. Retrieved June 10, 2020.
```

```
function [TR, TT, ER, t] = icp(q,p,varargin)  
% Perform the Iterative Closest Point algorithm on three dimensional point  
% clouds.  
%  
% [TR, TT] = icp(q,p) returns the rotation matrix TR and translation  
% vector TT that minimizes the distances from (TR * p + TT) to q.  
% p is a 3xm matrix and q is a 3xn matrix.  
%  
% [TR, TT] = icp(q,p,k) forces the algorithm to make k iterations  
% exactly. The default is 10 iterations.  
%  
% [TR, TT, ER] = icp(q,p,k) also returns the RMS of errors for k  
% iterations in a (k+1)x1 vector. ER(0) is the initial error.  
%  
% [TR, TT, ER, t] = icp(q,p,k) also returns the calculation times per  
% iteration in a (k+1)x1 vector. t(0) is the time consumed for preprocessing.  
%  
% Additional settings may be provided in a parameter list:  
%  
% Boundary  
%     {} | 1x? vector  
%     If EdgeRejection is set, a vector can be provided that indexes into  
%     q and specifies which points of q are on the boundary.  
%  
% EdgeRejection  
%     {false} | true  
%     If EdgeRejection is true, point matches to edge vertices of q are  
%     ignored. Requires that boundary points of q are specified using  
%     Boundary or that a triangulation matrix for q is provided.  
%  
% Extrapolation  
%     {false} | true  
%     If Extrapolation is true, the iteration direction will be evaluated  
%     and extrapolated if possible using the method outlined by  
%     Besl and McKay 1992.
```

```
%  
% Matching  
%     {bruteForce} | Delaunay | kDtree  
%     Specifies how point matching should be done.  
%     bruteForce is usually the slowest and kDtree is the fastest.  
%     Note that the kDtree option is depends on the Statistics Toolbox  
%     v. 7.3 or higher.  
%  
% Minimize  
%     {point} | plane | lmaPoint  
%     Defines whether point to point or point to plane minimization  
%     should be performed. point is based on the SVD approach and is  
%     usually the fastest. plane will often yield higher accuracy. It  
%     uses linearized angles and requires surface normals for all points  
%     in q. Calculation of surface normals requires substantial pre  
%     processing.  
%     The option lmaPoint does point to point minimization using the non  
%     linear least squares Levenberg Marquardt algorithm. Results are  
%     generally the same as in points, but computation time may differ.  
%  
% Normals  
%     {} | n x 3 matrix  
%     A matrix of normals for the n points in q might be provided.  
%     Normals of q are used for point to plane minimization.  
%     Else normals will be found through a PCA of the 4 nearest  
%     neighbors.  
%  
% ReturnAll  
%     {false} | true  
%     Determines whether R and T should be returned for all iterations  
%     or only for the last one. If this option is set to true, R will be  
%     a 3x3x(k+1) matrix and T will be a 3x1x(k+1) matrix.  
%  
% Triangulation  
%     {} | ? x 3 matrix  
%     A triangulation matrix for the points in q can be provided,  
%     enabling EdgeRejection. The elements should index into q, defining  
%     point triples that act together as triangles.  
%  
% Verbose  
%     {false} | true  
%     Enables extrapolation output in the Command Window.  
%  
% Weight
```

```

%      {@(match)ones(1,m)} | Function handle
%      For point or plane minimization, a function handle to a weighting
%      function can be provided. The weighting function will be called
%      with one argument, a 1xm vector that specifies point pairs by
%      indexing into q. The weighting function should return a 1xm vector
%      of weights for every point pair.
%
% WorstRejection
%      {0} | scalar in ]0; 1[
%      Reject a given percentage of the worst point pairs, based on their
%      Euclidean distance.
%
% Martin Kjer and Jakob Wilm, Technical University of Denmark, 2012

% Use the inputParser class to validate input arguments.
inp = inputParser;

inp.addRequired('q', @(x)isreal(x) && size(x,1) == 3);
inp.addRequired('p', @(x)isreal(x) && size(x,1) == 3);

inp.addOptional('iter', 10, @(x)x > 0 && x < 10^5);

inp.addParamValue('Boundary', [], @(x)size(x,1) == 1);

inp.addParamValue('EdgeRejection', false, @(x)islogical(x));

inp.addParamValue('Extrapolation', false, @(x)islogical(x));

validMatching = {'bruteForce', 'Delaunay', 'kdtree'};
inp.addParamValue('Matching', 'bruteForce', @(x)any(strcmpi(x,validMatching)));

validMinimize = {'point', 'plane', 'lmapoint'};
inp.addParamValue('Minimize', 'point', @(x)any(strcmpi(x,validMinimize)));

inp.addParamValue('Normals', [], @(x)isreal(x) && size(x,1) == 3);

inp.addParamValue('NormalsData', [], @(x)isreal(x) && size(x,1) == 3);

inp.addParamValue('ReturnAll', false, @(x)islogical(x));

inp.addParamValue('Triangulation', [], @(x)isreal(x) && size(x,2) == 3);

inp.addParamValue('Verbose', false, @(x)islogical(x));

```

```

inp.addParamValue('Weight', @(x)ones(1,length(x)), @(x)isa(x, 'function_handle'));

inp.addParamValue('WorstRejection', 0, @(x)isscalar(x) && x > 0 && x < 1);

inp.parse(q,p,varargin{:});
arg = inp.Results;
clear('inp');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Actual implementation

% Allocate vector for RMS of errors in every iteration.
t = zeros(arg.iter+1,1);

% Start timer
tic;

Np = size(p,2);

% Transformed data point cloud
pt = p;

% Allocate vector for RMS of errors in every iteration.
ER = zeros(arg.iter+1,1);

% Initialize temporary transform vector and matrix.
T = zeros(3,1);
R = eye(3,3);

% Initialize total transform vector(s) and rotation matrix(es).
TT = zeros(3,1, arg.iter+1);
TR = repmat(eye(3,3), [1,1, arg.iter+1]);

% If Minimize == 'plane', normals are needed
if (strcmp(arg.Minimize, 'plane') && isempty(arg.Normals))
    arg.Normals = lsqnormest(q,4);
end

% If Matching == 'Delaunay', a triangulation is needed
if strcmp(arg.Matching, 'Delaunay')
    DT = DelaunayTri(transpose(q));
end

% If Matching == 'kDtree', a kD tree should be built (req. Stat. TB >= 7.3)

```

```

if strcmp(arg.Matching, 'kDtree')
    kdOBJ = KDTreeSearcher(transpose(q));
end

% If edge vertices should be rejected, find edge vertices
if arg.EdgeRejection
    if isempty(arg.Boundary)
        bdr = find_bound(q, arg.Triangulation);
    else
        bdr = arg.Boundary;
    end
end

if arg.Extrapolation
    % Initialize total transform vector (quaternion ; translation vec.)
    qq = [ones(1,arg.iter+1);zeros(6,arg.iter+1)];
    % Allocate vector for direction change and change angle.
    dq = zeros(7,arg.iter+1);
    theta = zeros(1,arg.iter+1);
end

t(1) = toc;

% Go into main iteration loop
for k=1:arg.iter

    % Do matching
    switch arg.Matching
        case 'bruteForce'
            [match mindist] = match_bruteForce(q,pt);
        case 'Delaunay'
            [match mindist] = match_Delaunay(q,pt,DT);
        case 'kDtree'
            [match mindist] = match_kDtree(q,pt,kdOBJ);
    end

    % If matches to edge vertices should be rejected
    if arg.EdgeRejection
        p_idx = not(ismember(match, bdr));
        q_idx = match(p_idx);
        mindist = mindist(p_idx);
    else
        p_idx = true(1, Np);
        q_idx = match;
    end
end

```

```

end

% If worst matches should be rejected
if arg.WorstRejection
    edge = round((1-arg.WorstRejection)*sum(p_idx));
    pairs = find(p_idx);
    [~, idx] = sort(mindist);
    p_idx(pairs(idx(edge:end))) = false;
    q_idx = match(p_idx);
    mindist = mindist(p_idx);
end

if k == 1
    ER(k) = sqrt(sum(mindist.^2)/length(mindist));
end

switch arg.Minimize
    case 'point'
        % Determine weight vector
        weights = arg.Weight(match);
        [R,T] = eq_point(q(:,q_idx),pt(:,p_idx), weights(p_idx));
    case 'plane'
        weights = arg.Weight(match);
        [R,T] = eq_plane(q(:,q_idx),pt(:,p_idx),argNormals(:,q_idx),weights(p_idx));
    case 'lmaPoint'
        [R,T] = eq_lmaPoint(q(:,q_idx),pt(:,p_idx));
end

% Add to the total transformation
TR(:, :, k+1) = R*TR(:, :, k);
TT(:, :, k+1) = R*TT(:, :, k)+T;

% Apply last transformation
pt = TR(:, :, k+1) * p + repmat(TT(:, :, k+1), 1, Np);

% Root mean of objective function
ER(k+1) = rms_error(q(:,q_idx), pt(:,p_idx));

% If Extrapolation, we might be able to move quicker
if arg.Extrapolation
    qq(:,k+1) = [rmat2quat(TR(:, :, k+1));TT(:, :, k+1)];
    dq(:,k+1) = qq(:,k+1) - qq(:,k);
    theta(k+1) = (180/pi)*acos(dot(dq(:,k),dq(:,k+1))/(norm(dq(:,k))*norm(dq(:,k+1))));
    if arg.Verbose

```

```

disp(['Direction change ' num2str(theta(k+1)) ' degree in iteration ' num2str(k)]);
end
if k>2 && theta(k+1) < 10 && theta(k) < 10
    d = [ER(k+1), ER(k), ER(k-1)];
    v = [0, -norm(dq(:,k+1)), -norm(dq(:,k))-norm(dq(:,k+1))];
    vmax = 25 * norm(dq(:,k+1));
    dv = extrapolate(v,d,vmax);
    if dv ~= 0
        q_mark = qq(:,k+1) + dv * dq(:,k+1)/norm(dq(:,k+1));
        q_mark(1:4) = q_mark(1:4)/norm(q_mark(1:4));
        qq(:,k+1) = q_mark;
        TR(:, :, k+1) = quat2rmat(qq(1:4, k+1));
        TT(:, :, k+1) = qq(5:7, k+1);
        % Reapply total transformation
        pt = TR(:, :, k+1) * p + repmat(TT(:, :, k+1), 1, Np);
        % Recalculate root mean of objective function
        % Note this is costly and only for fun!
        switch arg.Matching
            case 'bruteForce'
                [~, mindist] = match_bruteForce(q,pt);
            case 'Delaunay'
                [~, mindist] = match_Delaunay(q,pt,DT);
            case 'kDtree'
                [~, mindist] = match_kDtree(q,pt,kdOBJ);
        end
        ER(k+1) = sqrt(sum(mindist.^2)/length(mindist));
    end
end
end
t(k+1) = toc;
end

if not(arg.ReturnAll)
    TR = TR(:, :, end);
    TT = TT(:, :, end);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [match mindist] = match_bruteForce(q, p)
    m = size(p,2);
    n = size(q,2);
    match = zeros(1,m);
    mindist = zeros(1,m);

```



```

for ki=1:m
    d=zeros(1,n);
    for ti=1:3
        d=d+(q(ti,:)-p(ti,ki)).^2;
    end
    [mindist(ki),match(ki)]=min(d);
end

mindist = sqrt(mindist);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [match mindist] = match_Delaunay(q, p, DT)
    match = transpose(nearestNeighbor(DT, transpose(p)));
    mindist = sqrt(sum((p-q(:,match)).^2,1));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [match mindist] = match_kDtree(~, p, kdOBJ)
    [match mindist] = knnsearch(kdOBJ,transpose(p));
    match = transpose(match);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [R,T] = eq_point(q,p,weights)

m = size(p,2);
n = size(q,2);

% normalize weights
weights = weights ./ sum(weights);

% find data centroid and deviations from centroid
q_bar = q * transpose(weights);
q_mark = q - repmat(q_bar, 1, n);
% Apply weights
q_mark = q_mark .* repmat(weights, 3, 1);

% find data centroid and deviations from centroid
p_bar = p * transpose(weights);
p_mark = p - repmat(p_bar, 1, m);
% Apply weights
p_mark = p_mark .* repmat(weights, 3, 1);

```

```

N = p_mark*transpose(q_mark); % taking points of q in matched order

[U,~,V] = svd(N); % singular value decomposition

R = V*diag([1 1 det(U*V')])*transpose(U);

T = q_bar - R*p_bar;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [R,T] = eq_plane(q,p,n,weights)

n = n .* repmat(weights,3,1);

c = cross(p,n);

cn = vertcat(c,n);

C = cn*transpose(cn);

b = - [sum(sum((p-q).*repmat(cn(1,:),3,1).*n));
       sum(sum((p-q).*repmat(cn(2,:),3,1).*n));
       sum(sum((p-q).*repmat(cn(3,:),3,1).*n));
       sum(sum((p-q).*repmat(cn(4,:),3,1).*n));
       sum(sum((p-q).*repmat(cn(5,:),3,1).*n));
       sum(sum((p-q).*repmat(cn(6,:),3,1).*n))];

X = C\b;

cx = cos(X(1)); cy = cos(X(2)); cz = cos(X(3));
sx = sin(X(1)); sy = sin(X(2)); sz = sin(X(3));

R = [cy*cz  cz*sx*sy-cx*sz  cx*cz*sy+sx*sz;
     cy*sz  cx*cz+sx*sy*sz  cx*sy*sz-cz*sx;
     -sy  cy*sx  cx*cy];

T = X(4:6);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [R,T] = eq_lmaPoint(q,p)

Rx = @(a)[1      0      0;
          0      cos(a)  -sin(a)];

```

```

    0    sin(a)  cos(a)];

Ry = @(b)[cos(b)    0    sin(b);
          0         1    0;
          -sin(b)   0    cos(b)];

Rz = @(g)[cos(g)   -sin(g) 0;
          sin(g)   cos(g)  0;
          0        0      1];

Rot = @(x)Rx(x(1))*Ry(x(2))*Rz(x(3));

myfun = @(x,xdata)Rot(x(1:3))*xdata+repmat(x(4:6),1,length(xdata));

options = optimset('Algorithm', 'levenberg-marquardt');
x = lsqcurvefit(myfun, zeros(6,1), p, q, [], [], options);

R = Rot(x(1:3));
T = x(4:6);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Extrapolation in quaternion space. Details are found in:
%
% Besl, P., & McKay, N. (1992). A method for registration of 3-D shapes.
% IEEE Transactions on pattern analysis and machine intelligence, 239?256.

function [dv] = extrapolate(v,d,vmax)

p1 = polyfit(v,d,1); % linear fit
p2 = polyfit(v,d,2); % parabolic fit
v1 = -p1(2)/p1(1); % linear zero crossing
v2 = -p2(2)/(2*p2(1)); % polynomial top point

if issorted([0 v2 v1 vmax]) || issorted([0 v2 vmax v1])
    disp('Parabolic update!');
    dv = v2;
elseif issorted([0 v1 v2 vmax]) || issorted([0 v1 vmax v2])...
    || (v2 < 0 && issorted([0 v1 vmax]))
    disp('Line based update!');
    dv = v1;
elseif v1 > vmax && v2 > vmax

```

```

disp('Maximum update!');
dv = vmax;
else
disp('No extrapolation!');
dv = 0;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Determine the RMS error between two point equally sized point clouds with
% point correspondance.
% ER = rms_error(p1,p2) where p1 and p2 are 3xn matrices.

function ER = rms_error(p1,p2)
dsq = sum(power(p1 - p2, 2),1);
ER = sqrt(mean(dsq));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Converts (orthogonal) rotation matrices R to (unit) quaternion
% representations
%
% Input: A 3x3xn matrix of rotation matrices
% Output: A 4xn matrix of n corresponding quaternions
%
% http://en.wikipedia.org/wiki/Rotation\_matrix#Quaternion

function quaternion = rmat2quat(R)

Qxx = R(1,1,:);
Qxy = R(1,2,:);
Qxz = R(1,3,:);
Qyx = R(2,1,:);
Qyy = R(2,2,:);
Qyz = R(2,3,:);
Qzx = R(3,1,:);
Qzy = R(3,2,:);
Qzz = R(3,3,:);

w = 0.5 * sqrt(1+Qxx+Qyy+Qzz);
x = 0.5 * sign(Qzy-Qyz) .* sqrt(1+Qxx-Qyy-Qzz);
y = 0.5 * sign(Qxz-Qzx) .* sqrt(1-Qxx+Qyy-Qzz);
z = 0.5 * sign(Qyx-Qxy) .* sqrt(1-Qxx-Qyy+Qzz);

```

```

quaternion = reshape([w;x;y;z],4,[]);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Converts (unit) quaternion representations to (orthogonal) rotation matrices R
%
% Input: A 4xn matrix of n quaternions
% Output: A 3x3xn matrix of corresponding rotation matrices
%
% http://en.wikipedia.org/wiki/Quaternions\_and\_spatial\_rotation#From\_a\_quaternion\_to\_an\_orthogonal\_matrix

function R = quat2rmat(quaternion)
q0(1,1,:) = quaternion(1,:);
qx(1,1,:) = quaternion(2,:);
qy(1,1,:) = quaternion(3,:);
qz(1,1,:) = quaternion(4,:);

R = [q0.^2+qx.^2-qy.^2-qz.^2 2*qx.*qy-2*q0.*qz 2*qx.*qz+2*q0.*qy;
     2*qx.*qy+2*q0.*qz q0.^2-qx.^2+qy.^2-qz.^2 2*qy.*qz-2*q0.*qx;
     2*qx.*qz-2*q0.*qy 2*qy.*qz+2*q0.*qx q0.^2-qx.^2-qy.^2+qz.^2];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Least squares normal estimation from point clouds using PCA
%
% H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle.
% Surface reconstruction from unorganized points.
% In Proceedings of ACM Siggraph, pages 71:78, 1992.
%
% p should be a matrix containing the horizontally concatenated column
% vectors with points. k is a scalar indicating how many neighbors the
% normal estimation is based upon.
%
% Note that for large point sets, the function performs significantly
% faster if Statistics Toolbox >= v. 7.3 is installed.
%
% Jakob Wilm 2010

function n = lsqnormest(p, k)
m = size(p,2);
n = zeros(3,m);

v = ver('stats');
```

```

if str2double(v.Version) >= 7.5
    neighbors = transpose(knnsearch(transpose(p), transpose(p), 'k', k+1));
else
    neighbors = k_nearest_neighbors(p, p, k+1);
end

for i = 1:m
    x = p(:,neighbors(2:end, i));
    p_bar = 1/k * sum(x,2);

    P = (x - repmat(p_bar,1,k)) * transpose(x - repmat(p_bar,1,k)); %spd matrix P
    %P = 2*cov(x);

    [V,D] = eig(P);

    [~, idx] = min(diag(D)); % choses the smallest eigenvalue

    n(:,i) = V(:,idx); % returns the corresponding eigenvector
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Program to find the k - nearest neighbors (kNN) within a set of points.
% Distance metric used: Euclidean distance
%
% Note that this function makes repetitive use of min(), which seems to be
% more efficient than sort() for k < 30.

function [neighborIds neighborDistances] = k_nearest_neighbors(dataMatrix, queryMatrix, k)

numDataPoints = size(dataMatrix,2);
numQueryPoints = size(queryMatrix,2);

neighborIds = zeros(k,numQueryPoints);
neighborDistances = zeros(k,numQueryPoints);

D = size(dataMatrix, 1); %dimensionality of points

for i=1:numQueryPoints
    d=zeros(1,numDataPoints);
    for t=1:D % this is to avoid slow repmat()
        d=d+(dataMatrix(t,:)-queryMatrix(t,i)).^2;
    end
    for j=1:k

```

```

    [s,t] = min(d);
    neighborIds(j,i)=t;
    neighborDistances(j,i)=sqrt(s);
    d(t) = NaN; % remove found number from d
end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Boundary point determination. Given a set of 3D points and a
% corresponding triangle representation, returns those point indices that
% define the border/edge of the surface.

function bound = find_bound(pts, poly)

%Correcting polygon indices and converting datatype
poly = double(poly);
pts = double(pts);

%Calculating freeboundary points:
TR = TriRep(poly, pts(1,:)', pts(2,:)', pts(3,:)');
FF = freeBoundary(TR);

%Output
bound = FF(:,1);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

## **A2    Box Girder 1 Algorithm**



## Contents

---

- [Box girder 1](#)
- [Rotate](#)
- [Deleting bad part](#)
- [Slice](#)
- [Vertical cut to find outer parts](#)
- [Horizontal cut](#)
- [Vertical cut](#)
- [Intersection points](#)
- [Rotate intersection point](#)
- [Write Excel file](#)

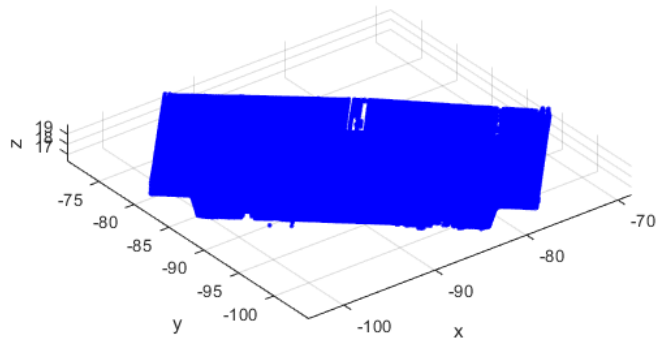
## Box girder 1

---

```
clear all
close all
clc

A=dlmread('Box_1.txt');    %%Reading the txt-file of the point cloud
a=A(:,[1,2,3]);           %%Matrix of the geometrical data
x=a(:,1);
y=a(:,2);
z=a(:,3);

figure(1)
scatter3(x,y,z,'.b')
hold on
xlabel('x')
ylabel('y')
zlabel('z')
grid on
axis('equal')
```



## Rotate

Find angle to rotate

```
[maxX, indexOfMaxX] = max(x);
yAtMaxX = y(indexOfMaxX);
[minY, indexOfMinY] = min(y);
xAtMinY = x(indexOfMinY);

u=[maxX-xAtMinY, yAtMaxX- minY];
v=[1,0];

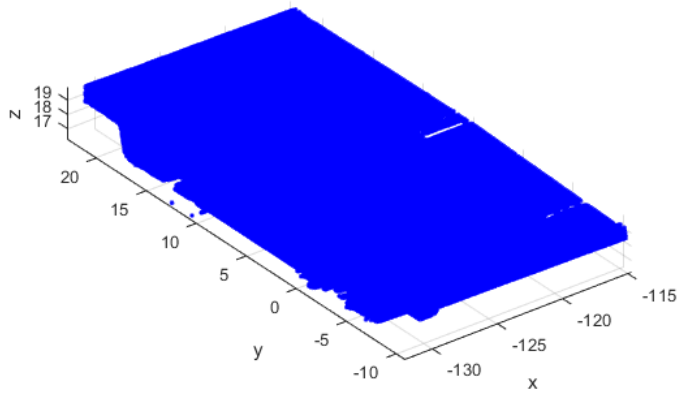
slope=u(:,2)/u(:,1);
theta=atan(slope);           %%Angle

Rz=[cos(theta)  -sin(theta)  0;  %%Rotation matrix
    sin(theta)  cos(theta)  0;
    0  0  1];

%%Rotate the whole pier
rotM=a*Rz;

%%Rotated matrix
x1=rotM(:,1);
y1=rotM(:,2);
z1=rotM(:,3);

figure(2)
scatter3(x1,y1,z1,'.b')
hold on
xlabel('x')
ylabel('y')
zlabel('z')
grid on
axis('equal')
```



### Deleting bad part

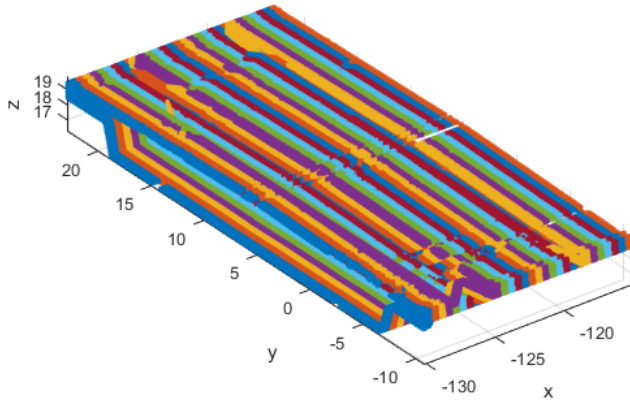
```
rowsToDelete = rotM(:,1) < min(rotM(:,1))+2;
rotM(rowsToDelete,:) = [];
```

### Slice

```
%%Dividing the rotated Box girder into main segments
xRngs=[min(rotM(:,1)):0.5: max(rotM(:,1))];
for ind = 1:numel(xRngs)-1

    rows = rotM(:,1)>xRngs(ind) & rotM(:,1)<xRngs(ind+1); %check to see which rows of b are between
    %xRng(ind) and xRng(ind+1)
    segM(ind).SmallerM = rotM(rows,:); %segM is the segmented matrix of the rotated pier

    figure(3)
    scatter3(segM(ind).SmallerM(:,1),segM(ind).SmallerM(:,2),segM(ind).SmallerM(:,3),'.') %Scatter the rotated pier
    hold on
    xlabel('x')
    ylabel('y')
    zlabel('z')
    grid on
    axis('equal')
end
```



#### Vertical cut to find outer parts

```

%%Splitting the box Vertically, finding the two outer sides
for ind = 1:numel(xRngs)-1
MaxY_Box(ind)=max(segM(ind).SmallerM(:,2));
MinY_Box(ind)=min(segM(ind).SmallerM(:,2));

e=0.01;
kVer(ind) = floor((MaxY_Box(ind)-MinY_Box(ind))/e);
for i = 1:kVer(ind)
mVer(ind).SmallerVer(i,1) = MinY_Box(ind)+e*(i-1);
A2Ver(ind).SmallerA2Ver= segM(ind).SmallerM((segM(ind).SmallerM(:,2)>=mVer(ind).SmallerVer(i,1) & segM(ind).SmallerM(:,2)<=mVer(ind).SmallerVer(i,1)+e),:);
ggVer_Box(ind).SmallerggVer_Box{i,1}=A2Ver(ind).SmallerA2Ver;          %%Counting the points in each microsegment
if isempty(A2Ver(ind).SmallerA2Ver)==1
continue
end
end
ggVer_Box(ind).SmallerggVer_Box=cell2mat(ggVer_Box(ind).SmallerggVer_Box);
ggVer_mid3_side1(ind).SmallerggVer_mid3_side1=ggVer_Box(ind).SmallerggVer_Box((ggVer_Box(ind).SmallerggVer_Box(:,2)<=MinY_Box(ind)+0.2),:);
ggVer_mid3_side2(ind).SmallerggVer_mid3_side2=ggVer_Box(ind).SmallerggVer_Box((ggVer_Box(ind).SmallerggVer_Box(:,2)>=MaxY_Box(ind)-0.2),:);
end

%%Removing unregular part
for ind = 1:numel(xRngs)-1
%side 1
MaxZ_Box_side1(ind)= max(ggVer_mid3_side1(ind).SmallerggVer_mid3_side1(:,3));
MinZ_Box_side1(ind)= min(ggVer_mid3_side1(ind).SmallerggVer_mid3_side1(:,3));

mid3_side1(ind).smallermid3_side1= ggVer_mid3_side1(ind).SmallerggVer_mid3_side1((ggVer_mid3_side1(ind).SmallerggVer_mid3_side1(:,3)<=MaxZ_Box_side1(ind)-0.1),:);
mid3_side1(ind).smallermid3_side1= ggVer_mid3_side1(ind).SmallerggVer_mid3_side1((ggVer_mid3_side1(ind).SmallerggVer_mid3_side1(:,3)>=MinZ_Box_side1(ind)+0.2),:);

%side 2
MaxZ_Box_side2(ind)= max(ggVer_mid3_side2(ind).SmallerggVer_mid3_side2(:,3));
MinZ_Box_side2(ind)= min(ggVer_mid3_side2(ind).SmallerggVer_mid3_side2(:,3));

mid3_side2(ind).smallermid3_side2= ggVer_mid3_side2(ind).SmallerggVer_mid3_side2((ggVer_mid3_side2(ind).SmallerggVer_mid3_side2(:,3)<=MaxZ_Box_side2(ind)-0.1),:);

```

```

mid3_side2(ind).smallermid3_side2= ggVer_mid3_side2(ind).SmallerggVer_mid3_side2((ggVer_mid3_side2(ind).SmallerggVer_mid3_side2(:,3))>=MinZ_Box_side2(ind)+0.2),:);

end

%%Excluding the outer part from the point cloud
for ind = 1:numel(xRngs)-1
[ia, ~] = ismember(segM(ind).SmallerM, ggVer_mid3_side1(ind).SmallerggVer_mid3_side1, 'rows');
segM(ind).SmallerM(ia, :) = [];
[ia, ib] = ismember(segM(ind).SmallerM, ggVer_mid3_side2(ind).SmallerggVer_mid3_side2, 'rows');
segM(ind).SmallerM(ia, :) = [];
segM(ind).SmallerM = segM(ind).SmallerM(segM(ind).SmallerM(:,2) >= min(ggVer_mid3_side1(ind).SmallerggVer_mid3_side1(:,2)), :), :);
segM(ind).SmallerM = segM(ind).SmallerM(segM(ind).SmallerM(:,2) <= max(ggVer_mid3_side2(ind).SmallerggVer_mid3_side2(:,2)), :), :);
end

%%Figure of Boxgirder without the outer part
for ind = 1:numel(xRngs)-1
figure(4)
scatter3(segM(ind).SmallerM(:,1),segM(ind).SmallerM(:,2),segM(ind).SmallerM(:,3),'.b')
hold on
scatter3(mid3_side1(ind).smallermid3_side1(:,1),mid3_side1(ind).smallermid3_side1(:,2),mid3_side1(ind).smallermid3_side1(:,3),'.r')
hold on
scatter3(mid3_side2(ind).smallermid3_side2(:,1),mid3_side2(ind).smallermid3_side2(:,2),mid3_side2(ind).smallermid3_side2(:,3),'.g')
axis equal
end

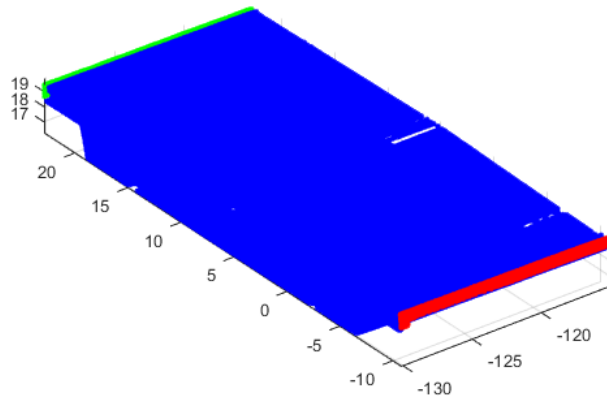
%%Find mean, max and min values
for ind = 1:numel(xRngs)-1
maximumZ(ind)= max(segM(ind).SmallerM(:,3));
minimumZ(ind)=min(segM(ind).SmallerM(:,3));

maximumY(ind)= max(segM(ind).SmallerM(:,2));
minimumY(ind)=min(segM(ind).SmallerM(:,2));

columnMean(ind).SmallercolumnMean = mean(segM(ind).SmallerM,1);

meanx(ind).Smallermeanx=columnMean(ind).SmallercolumnMean(1,1);
meany(ind).SmallermeanY=columnMean(ind).SmallercolumnMean(1,2);
meanZ(ind).SmallermeanZ=columnMean(ind).SmallercolumnMean(1,3);
end

```



### Horizontal cut

```

%%Dividing the segments into microsegments in the horizontal direction,
%%finding changes in amount of points
for ind = 1:numel(xRngs)-1

e1=0.05;
kHor(ind) = floor((maximumZ(ind)-minimumZ(ind))/e1);           %How many times its divided
for i = 1:kHor(ind)
mHor(ind).SmallermHor(i,1) = minimumZ(ind)+e1*(i-1);           %For which values its divided for each segment
A2Hor(ind).SmallerA2Hor= segM(ind).SmallerM((segM(ind).SmallerM(:,3)>=mHor(ind).SmallermHor(i,1) & segM(ind).SmallerM(:,3)<=mHor(ind).SmallermHor(i,1)+e1,:)); %logical indexing
ggHor(ind).SmallerggHor{i,1}=A2Hor(ind).SmallerA2Hor;           %Counting the points in each microsegment
if isempty(A2Hor(ind).SmallerA2Hor)=1
continue
end
end
for i=1:kHor(ind)-1
ChangeHor(ind).smallerChangeHor{i,:}=size(ggHor(ind).SmallerggHor{i,1})/size(ggHor(ind).SmallerggHor{i+1,1});

ChangeHor(ind).smallerChangeHor{i,:}=ChangeHor(ind).smallerChangeHor{i,1}(:,1);
ChangeHor1(ind).smallerChangeHor1=cell2mat(ChangeHor(ind).smallerChangeHor);
end
end

%%Finding the positions to cut
for ind = 1:numel(xRngs)-1

[m_1(ind).smallerm_1,idx_1(ind).smalleridx_1] = max(ChangeHor1(ind).smallerChangeHor1(1:floor(kHor(ind)/2),:));
ChangeHor1(ind).smallerChangeHor1(idx_1(ind).smalleridx_1) = NaN ;
idx_2(ind).smalleridx_2= find(diff(ChangeHor1(ind).smallerChangeHor1,2)==0,1); %The removed outer part always makes this position to be several repeting 1 %prøve å endre

if idx_1(ind).smalleridx_1 > idx_2(ind).smalleridx_2
[m_3(ind).smallerm_3,idx_3(ind).smalleridx_3] = min(ChangeHor1(ind).smallerChangeHor1(idx_2(ind).smalleridx_2:idx_1(ind).smalleridx_1) );
else
[m_3(ind).smallerm_3,idx_3(ind).smalleridx_3] = min(ChangeHor1(ind).smallerChangeHor1(idx_1(ind).smalleridx_1:idx_2(ind).smalleridx_2) );
end

```

```

m_4(ind).smallerm_4 = min(ChangeHor1(ind).smallerChangeHor1(idx_2(ind).smalleridx_2:end)) ;
idx_4(ind).smalleridx_4 = find(ChangeHor1(ind).smallerChangeHor1==m_4(ind).smallerm_4);

CutPosition(ind).smallerCutPosition=sort([idx_1(ind).smalleridx_1;idx_2(ind).smalleridx_2;idx_3(ind).smalleridx_3;idx_4(ind).smalleridx_4]);

end

%%Removing the ends from both sides of the mid 1,cos horizontal cut
for ind = 1:numel(xRngs)-1

up(ind).smallerup=ggHor(ind).SmallerggHor(1:CutPosition(ind).smallerCutPosition(1,:),:);
up1(ind).smallerup1=cell2mat(up(ind).smallerup);

mid1(ind).smallermid1=ggHor(ind).SmallerggHor(CutPosition(ind).smallerCutPosition(1,:)+1:CutPosition(ind).smallerCutPosition(2,:),:);
mid1(ind).smallermid1=mid1(ind).smallermid1(5:end-4);
mid11(ind).smallermid11=cell2mat(mid1(ind).smallermid1);

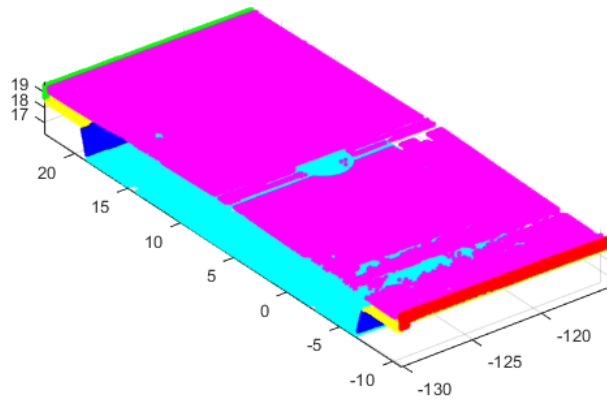
mid2(ind).smallermid2=ggHor(ind).SmallerggHor(CutPosition(ind).smallerCutPosition(2,:)+1:CutPosition(ind).smallerCutPosition(3,:),:);
mid22(ind).smallermid22=cell2mat(mid2(ind).smallermid2);

%%mid3 is already splitted as the outer parts

down(ind).smallerdown=ggHor(ind).SmallerggHor(CutPosition(ind).smallerCutPosition(4,:)+1:end,:);
down1(ind).smallerdown1=cell2mat(down(ind).smallerdown);

figure(5)
scatter3(up1(ind).smallerup1(:,1),up1(ind).smallerup1(:,2),up1(ind).smallerup1(:,3),'.c')
hold on
scatter3(mid11(ind).smallermid11(:,1),mid11(ind).smallermid11(:,2),mid11(ind).smallermid11(:,3),'.b')
hold on
scatter3(mid22(ind).smallermid22(:,1),mid22(ind).smallermid22(:,2),mid22(ind).smallermid22(:,3),'.y')
hold on
scatter3(down1(ind).smallerdown1(:,1),down1(ind).smallerdown1(:,2),down1(ind).smallerdown1(:,3),'.m')
hold on
scatter3(mid3_side1(ind).smallermid3_side1(:,1),mid3_side1(ind).smallermid3_side1(:,2),mid3_side1(ind).smallermid3_side1(:,3),'.r')
hold on
scatter3(mid3_side2(ind).smallermid3_side2(:,1),mid3_side2(ind).smallermid3_side2(:,2),mid3_side2(ind).smallermid3_side2(:,3),'.g')
axis equal
end

```



### Vertical cut

```

%%splitting the sides
for ind = 1:numel(xRngs)-1
%%Up is has only one side

Mid1_side1(ind).smallerMid1_side1=find(mid11(ind).smallermid11(:,2)< (meanY(ind).SmallermeanY));
Mid1_side2(ind).smallerMid1_side2=find(mid11(ind).smallermid11(:,2)> (meanY(ind).SmallermeanY));
Mid1_side1(ind).smallerMid1_side1=mid11(ind).smallermid11(Mid1_side1(ind).smallerMid1_side1,:);
Mid1_side2(ind).smallerMid1_side2=mid11(ind).smallermid11(Mid1_side2(ind).smallerMid1_side2,:);

Mid2_side1(ind).smallerMid2_side1=find(mid22(ind).smallermid22(:,2)< (meanY(ind).SmallermeanY));
Mid2_side2(ind).smallerMid2_side2=find(mid22(ind).smallermid22(:,2)> (meanY(ind).SmallermeanY));
Mid2_side1(ind).smallerMid2_side1=mid22(ind).smallermid22(Mid2_side1(ind).smallerMid2_side1,:);
Mid2_side2(ind).smallerMid2_side2=mid22(ind).smallermid22(Mid2_side2(ind).smallerMid2_side2,:);

Down_side1(ind).smallerDown_side1=find(down1(ind).smallerdown1(:,2)< (meanY(ind).SmallermeanY));
Down_side2(ind).smallerDown_side2=find(down1(ind).smallerdown1(:,2)> (meanY(ind).SmallermeanY));
Down_side1(ind).smallerDown_side1=down1(ind).smallerdown1(Down_side1(ind).smallerDown_side1,:);
Down_side2(ind).smallerDown_side2=down1(ind).smallerdown1(Down_side2(ind).smallerDown_side2,:);
end

%%splitting Mid2_side1 Vertically, removing the ends from both sides
for ind = 1:numel(xRngs)-1
MaxY_Mid2_side1(ind)=max(Mid2_side1(ind).smallerMid2_side1(:,2));
MinY_Mid2_side1(ind)=min(Mid2_side1(ind).smallerMid2_side1(:,2));
e=0.1;
kVer(ind) = floor((MaxY_Mid2_side1(ind)-MinY_Mid2_side1(ind))/e);
for i = 1:kVer(ind)
mVer(ind).SmallermVer(i,1) = MinY_Mid2_side1(ind)+e*(i-1);
A2Ver(ind).SmallerA2Ver= Mid2_side1(ind).smallerMid2_side1((Mid2_side1(ind).smallerMid2_side1(:,2))>=mVer(ind).SmallermVer(i,1) & Mid2_side1(ind).smallerMid2_side1(:,2)<=mVer(ind).SmallermVer(i,1)+e),:); %logical indexing
ggVer_mid2_side1(ind).SmallerggVer_mid2_side1{i,1}=A2Ver(ind).SmallerA2Ver; %Counting the points in each microsegment
if isempty(A2Ver(ind).SmallerA2Ver)=1
continue
end
end

```



```

ggVer_mid2_side1(ind).SmallerggVer_mid2_side1=ggVer_mid2_side1(ind).SmallerggVer_mid2_side1(2:end-2);
ggVer_mid2_side1(ind).SmallerggVer_mid2_side1=cell2mat(ggVer_mid2_side1(ind).SmallerggVer_mid2_side1);
end
%%splitting Mid2_side2 Vertically, removing the ends from both sides
for ind = 1:numel(xRngs)-1
MaxY_Mid2_side2(ind)=max(Mid2_side2(ind).smallerMid2_side2(:,2));
MinY_Mid2_side2(ind)=min(Mid2_side2(ind).smallerMid2_side2(:,2));
e=0.1;
kVer(ind) = floor((MaxY_Mid2_side2(ind)-MinY_Mid2_side2(ind))/e);
for i = 1:kVer(ind)
mVer(ind).SmallerVer(i,1) = MinY_Mid2_side2(ind)+e*(i-1);
A2Ver(ind).SmallerA2Ver= Mid2_side2(ind).smallerMid2_side2((Mid2_side2(ind).smallerMid2_side2(:,2)>=mVer(ind).SmallerVer(i,1) & Mid2_side2(ind).smallerMid2_side2(:,2)<=mVer(ind).SmallerVer(i,1)+e),:); %logical indexing
ggVer_mid2_side2(ind).SmallerggVer_mid2_side2{i,1}=A2Ver(ind).SmallerA2Ver; %Counting the points in each microsegment
if isempty(A2Ver(ind).SmallerA2Ver)==1
continue
end
end
ggVer_mid2_side2(ind).SmallerggVer_mid2_side2=ggVer_mid2_side2(ind).SmallerggVer_mid2_side2(2:end-2);
ggVer_mid2_side2(ind).SmallerggVer_mid2_side2=cell2mat(ggVer_mid2_side2(ind).SmallerggVer_mid2_side2);
end

%%splitting Up vertically, removing the ends from both sides
for ind = 1:numel(xRngs)-1
MaxY_Up(ind)=max(up1(ind).smallerup1(:,2));
MinY_Up(ind)=min(up1(ind).smallerup1(:,2));
e=0.1;
kVer(ind) = floor((MaxY_Up(ind)-MinY_Up(ind))/e);
for i = 1:kVer(ind)
mVer(ind).SmallerVer(i,1) = MinY_Up(ind)+e*(i-1);
A2Ver(ind).SmallerA2Ver= up1(ind).smallerup1((up1(ind).smallerup1(:,2)>=mVer(ind).SmallerVer(i,1) & up1(ind).smallerup1(:,2)<=mVer(ind).SmallerVer(i,1)+e),:); %logical indexing
ggVer_Up(ind).SmallerggVer_Up{i,1}=A2Ver(ind).SmallerA2Ver; %Counting the points in each microsegment
if isempty(A2Ver(ind).SmallerA2Ver)==1
continue
end
end
ggVer_Up(ind).SmallerggVer_Up=ggVer_Up(ind).SmallerggVer_Up(10:end-10);
ggVer_Up(ind).SmallerggVer_Up=cell2mat(ggVer_Up(ind).SmallerggVer_Up);
end

%%splitting Down_side1 vertically, removing the ends from both sides
for ind = 1:numel(xRngs)-1
MaxY_Down_side1(ind)=max(Down_side1(ind).smallerDown_side1(:,2));
MinY_Down_side1(ind)=min(Down_side1(ind).smallerDown_side1(:,2));
e=0.1;
kVer(ind) = floor((MaxY_Down_side1(ind)-MinY_Down_side1(ind))/e);
for i = 1:kVer(ind)
mVer(ind).SmallerVer(i,1) = MinY_Down_side1(ind)+e*(i-1);
A2Ver(ind).SmallerA2Ver= Down_side1(ind).smallerDown_side1((Down_side1(ind).smallerDown_side1(:,2)>=mVer(ind).SmallerVer(i,1) & Down_side1(ind).smallerDown_side1(:,2)<=mVer(ind).SmallerVer(i,1)+e),:); %logical indexing
ggVer_Down_side1(ind).SmallerggVer_Down_side1{i,1}=A2Ver(ind).SmallerA2Ver; %Counting the points in each microsegment
if isempty(A2Ver(ind).SmallerA2Ver)==1
continue
end
end
ggVer_Down_side1(ind).SmallerggVer_Down_side1=ggVer_Down_side1(ind).SmallerggVer_Down_side1(20:end-20);
ggVer_Down_side1(ind).SmallerggVer_Down_side1=cell2mat(ggVer_Down_side1(ind).SmallerggVer_Down_side1);
end

%%splitting Down_side2 vertically, removing the ends from both sides
for ind = 1:numel(xRngs)-1
MaxY_Down_side2(ind)=max(Down_side2(ind).smallerDown_side2(:,2));
MinY_Down_side2(ind)=min(Down_side2(ind).smallerDown_side2(:,2));
e=0.1;
kVer(ind) = floor((MaxY_Down_side2(ind)-MinY_Down_side2(ind))/e);
for i = 1:kVer(ind)

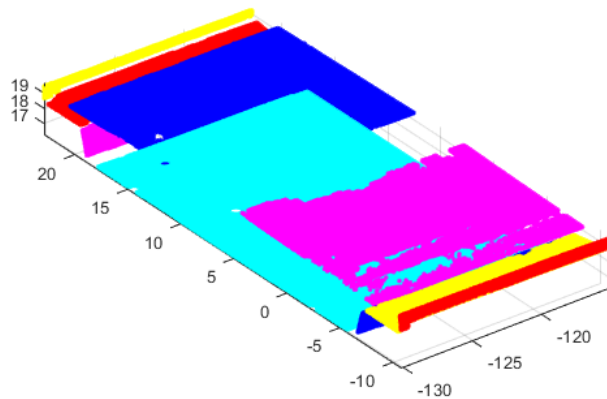
```

```

mVer(ind).SmallermVer(i,1) = MinY_Down_side2(ind)+e*(i-1);
A2Ver(ind).SmallerA2Ver= Down_side2(ind).smallerDown_side2((Down_side2(ind).smallerDown_side2(:,2))>=mVer(ind).SmallermVer(i,1) & Down_side2(ind).smallerDown_side2(:,2)<=mVer(ind).SmallermVer(i,1)+e),:); %logical indexing
ggVer_Down_side2(ind).SmallerggVer_Down_side2{i,1}=A2Ver(ind).SmallerA2Ver; %Counting the points in each microsegment
if isempty(A2Ver(ind).SmallerA2Ver)==1
continue
end
end
ggVer_Down_side2(ind).SmallerggVer_Down_side2=ggVer_Down_side2(ind).SmallerggVer_Down_side2(20:end-20);
ggVer_Down_side2(ind).SmallerggVer_Down_side2=cell2mat(ggVer_Down_side2(ind).SmallerggVer_Down_side2);
end

%%Figure of the divided box girder
for ind = 1:numel(xRngs)-1
figure(6)
scatter3(ggVer_Up(ind).SmallerggVer_Up(:,1),ggVer_Up(ind).SmallerggVer_Up(:,2),ggVer_Up(ind).SmallerggVer_Up(:,3),'.c')
hold on
scatter3(Mid1_side1(ind).smallerMid1_side1(:,1),Mid1_side1(ind).smallerMid1_side1(:,2),Mid1_side1(ind).smallerMid1_side1(:,3),'.b')
hold on
scatter3(Mid1_side2(ind).smallerMid1_side2(:,1),Mid1_side2(ind).smallerMid1_side2(:,2),Mid1_side2(ind).smallerMid1_side2(:,3),'.m')
hold on
scatter3(ggVer_mid2_side1(ind).SmallerggVer_mid2_side1(:,1),ggVer_mid2_side1(ind).SmallerggVer_mid2_side1(:,2),ggVer_mid2_side1(ind).SmallerggVer_mid2_side1(:,3),'.y')
hold on
scatter3(ggVer_mid2_side2(ind).SmallerggVer_mid2_side2(:,1),ggVer_mid2_side2(ind).SmallerggVer_mid2_side2(:,2),ggVer_mid2_side2(ind).SmallerggVer_mid2_side2(:,3),'.r')
hold on
scatter3(mid3_side1(ind).smallermid3_side1(:,1),mid3_side1(ind).smallermid3_side1(:,2),mid3_side1(ind).smallermid3_side1(:,3),'.r')
hold on
scatter3(mid3_side2(ind).smallermid3_side2(:,1),mid3_side2(ind).smallermid3_side2(:,2),mid3_side2(ind).smallermid3_side2(:,3),'.y')
hold on
scatter3( ggVer_Down_side1(ind).SmallerggVer_Down_side1(:,1), ggVer_Down_side1(ind).SmallerggVer_Down_side1(:,2), ggVer_Down_side1(ind).SmallerggVer_Down_side1(:,3),'.m')
hold on
scatter3(ggVer_Down_side2(ind).SmallerggVer_Down_side2(:,1),ggVer_Down_side2(ind).SmallerggVer_Down_side2(:,2),ggVer_Down_side2(ind).SmallerggVer_Down_side2(:,3),'.b')
axis equal
end

```



### Intersection points

```

%%Assigning one X-value to all the rows
for ind = 1:numel(xRngs)-1

    %%Flatten Up
    flatup1(ind).smallerflatup1=up1(ind).smallerup1;
    if isempty(up1(ind).smallerup1)==1
        continue
    end
    flatup1(ind).smallerflatup1(:,1)=repelem(flatup1(ind).smallerflatup1(1),size(flatup1(ind).smallerflatup1,1),1);
    pup1(ind).smallerpup1 = polyfit(flatup1(ind).smallerflatup1(:,2),flatup1(ind).smallerflatup1(:,3),1);
    fup1(ind).smallerfup1 = polyval(pup1(ind).smallerpup1,flatup1(ind).smallerflatup1(:,2));

%%Flatten Mid1_side1
    flatMid1_side1(ind).smallerflatMid1_side1=Mid1_side1(ind).smallerMid1_side1;
    if isempty(Mid1_side1(ind).smallerMid1_side1)==1
        continue
    end
    flatMid1_side1(ind).smallerflatMid1_side1(:,1)=repelem(flatMid1_side1(ind).smallerflatMid1_side1(1),size(flatMid1_side1(ind).smallerflatMid1_side1,1),1);
    pMid1_side1(ind).smallerpMid1_side1 = polyfit(flatMid1_side1(ind).smallerflatMid1_side1(:,2),flatMid1_side1(ind).smallerflatMid1_side1(:,3),1);
    fMid1_side1(ind).smallerfMid1_side1 = polyval(pMid1_side1(ind).smallerpMid1_side1,flatMid1_side1(ind).smallerflatMid1_side1(:,2));

%%Flatten Mid1_side2
    flatMid1_side2(ind).smallerflatMid1_side2=Mid1_side2(ind).smallerMid1_side2;
    if isempty(Mid1_side2(ind).smallerMid1_side2)==1
        continue
    end
    flatMid1_side2(ind).smallerflatMid1_side2(:,1)=repelem(flatMid1_side2(ind).smallerflatMid1_side2(1),size(flatMid1_side2(ind).smallerflatMid1_side2,1),1);
    pMid1_side2(ind).smallerpMid1_side2 = polyfit(flatMid1_side2(ind).smallerflatMid1_side2(:,2),flatMid1_side2(ind).smallerflatMid1_side2(:,3),1);
    fMid1_side2(ind).smallerfMid1_side2 = polyval(pMid1_side2(ind).smallerpMid1_side2,flatMid1_side2(ind).smallerflatMid1_side2(:,2));

%%Flatten Mid2_side1
    flatMid2_side1(ind).smallerflatMid2_side1=Mid2_side1(ind).smallerMid2_side1;
    if isempty(Mid2_side1(ind).smallerMid2_side1)==1
        continue
    end
    flatMid2_side1(ind).smallerflatMid2_side1(:,1)=repelem(flatMid2_side1(ind).smallerflatMid2_side1(1),size(flatMid2_side1(ind).smallerflatMid2_side1,1),1);
    pMid2_side1(ind).smallerpMid2_side1 = polyfit(flatMid2_side1(ind).smallerflatMid2_side1(:,2),flatMid2_side1(ind).smallerflatMid2_side1(:,3),1);
    fMid2_side1(ind).smallerfMid2_side1 = polyval(pMid2_side1(ind).smallerpMid2_side1,flatMid2_side1(ind).smallerflatMid2_side1(:,2));

%%Flatten Mid2_side2
    flatMid2_side2(ind).smallerflatMid2_side2=Mid2_side2(ind).smallerMid2_side2;
    if isempty(Mid2_side2(ind).smallerMid2_side2)==1
        continue
    end
    flatMid2_side2(ind).smallerflatMid2_side2(:,1)=repelem(flatMid2_side2(ind).smallerflatMid2_side2(1),size(flatMid2_side2(ind).smallerflatMid2_side2,1),1);
    pMid2_side2(ind).smallerpMid2_side2 = polyfit(flatMid2_side2(ind).smallerflatMid2_side2(:,2),flatMid2_side2(ind).smallerflatMid2_side2(:,3),1);
    fMid2_side2(ind).smallerfMid2_side2 = polyval(pMid2_side2(ind).smallerpMid2_side2,flatMid2_side2(ind).smallerflatMid2_side2(:,2));

%%Flatten Mid3_side1
    flatMid3_side1(ind).smallerflatMid3_side1=mid3_side1(ind).smallerMid3_side1;
    if isempty(mid3_side1(ind).smallerMid3_side1)==1
        continue
    end
    flatMid3_side1(ind).smallerflatMid3_side1(:,1)=repelem(flatMid3_side1(ind).smallerflatMid3_side1(1),size(flatMid3_side1(ind).smallerflatMid3_side1,1),1);
    pMid3_side1(ind).smallerpMid3_side1 = polyfit(flatMid3_side1(ind).smallerflatMid3_side1(:,2),flatMid3_side1(ind).smallerflatMid3_side1(:,3),1);
    fMid3_side1(ind).smallerfMid3_side1 = polyval(pMid3_side1(ind).smallerpMid3_side1,flatMid3_side1(ind).smallerflatMid3_side1(:,2));

%%Flatten Mid3_side2

```

```

flatMid3_side2(ind).smallerflatMid3_side2=mid3_side2(ind).smallermid3_side2;
if isempty(mid3_side2(ind).smallermid3_side2)==1
    continue
end
flatMid3_side2(ind).smallerflatMid3_side2(:,1)=repelem(flatMid3_side2(ind).smallerflatMid3_side2(1),size(flatMid3_side2(ind).smallerflatMid3_side2,1),1);
pMid3_side2(ind).smallerpMid3_side2 = polyfit(flatMid3_side2(ind).smallerflatMid3_side2(:,2),flatMid3_side2(ind).smallerflatMid3_side2(:,3),1);
fMid3_side2(ind).smallerfMid3_side2 = polyval(pMid3_side2(ind).smallerpMid3_side2,flatMid3_side2(ind).smallerflatMid3_side2(:,2));

%%Flatten Down_side1
flatDown_side1(ind).smallerflatDown_side1=Down_side1(ind).smallerDown_side1;
if isempty(Down_side1(ind).smallerDown_side1)==1
    continue
end
flatDown_side1(ind).smallerflatDown_side1(:,1)=repelem(flatDown_side1(ind).smallerflatDown_side1(1),size(flatDown_side1(ind).smallerflatDown_side1,1),1);
pflatDown_side1(ind).smallerpflatDown_side1 = polyfit(flatDown_side1(ind).smallerflatDown_side1(:,2),flatDown_side1(ind).smallerflatDown_side1(:,3),1);
fflatDown_side1(ind).smallerfflatDown_side1 = polyval(pflatDown_side1(ind).smallerpflatDown_side1,flatDown_side1(ind).smallerflatDown_side1(:,2));

%%Flatten Down_side2
flatDown_side2(ind).smallerflatDown_side2=Down_side2(ind).smallerDown_side2;
if isempty(Down_side2(ind).smallerDown_side2)==1
    continue
end
flatDown_side2(ind).smallerflatDown_side2(:,1)=repelem(flatDown_side2(ind).smallerflatDown_side2(1),size(flatDown_side2(ind).smallerflatDown_side2,1),1);
pflatDown_side2(ind).smallerpflatDown_side2 = polyfit(flatDown_side2(ind).smallerflatDown_side2(:,2),flatDown_side2(ind).smallerflatDown_side2(:,3),1);
fflatDown_side2(ind).smallerfflatDown_side2 = polyval(pflatDown_side2(ind).smallerpflatDown_side2,flatDown_side2(ind).smallerflatDown_side2(:,2));

end

%%find intersection
for ind = 1:numel(xRngs)-1

Y1_intersect(ind).smallerY1_intersect=fzero @(x) polyval(pup1(ind).smallerpup1-pMid1_side2(ind).smallerpMid1_side2,x),3);
Z1_intersect(ind).smallerZ1_intersect=polyval(pup1(ind).smallerpup1,Y1_intersect(ind).smallerY1_intersect);

Y2_intersect(ind).smallerY2_intersect=fzero @(x) polyval(pMid1_side2(ind).smallerpMid1_side2-pMid2_side2(ind).smallerpMid2_side2,x),3);
Z2_intersect(ind).smallerZ2_intersect=polyval(pMid1_side2(ind).smallerpMid1_side2,Y2_intersect(ind).smallerY2_intersect);

Y3_intersect(ind).smallerY3_intersect=fzero @(x) polyval(pMid2_side2(ind).smallerpMid2_side2-pMid3_side2(ind).smallerpMid3_side2,x),3);
Z3_intersect(ind).smallerZ3_intersect=polyval(pMid2_side2(ind).smallerpMid2_side2,Y3_intersect(ind).smallerY3_intersect);

Y4_intersect(ind).smallerY4_intersect=fzero @(x) polyval(pMid3_side2(ind).smallerpMid3_side2-pflatDown_side2(ind).smallerpflatDown_side2,x),3);
Z4_intersect(ind).smallerZ4_intersect=polyval(pMid3_side2(ind).smallerpMid3_side2,Y4_intersect(ind).smallerY4_intersect);

Y5_intersect(ind).smallerY5_intersect=fzero @(x) polyval(pflatDown_side2(ind).smallerpflatDown_side2-pflatDown_side1(ind).smallerpflatDown_side1,x),3);
Z5_intersect(ind).smallerZ5_intersect=polyval(pflatDown_side2(ind).smallerpflatDown_side2,Y5_intersect(ind).smallerY5_intersect);

Y6_intersect(ind).smallerY6_intersect=fzero @(x) polyval(pflatDown_side1(ind).smallerpflatDown_side1-pMid3_side1(ind).smallerpMid3_side1,x),3);
Z6_intersect(ind).smallerZ6_intersect=polyval(pflatDown_side1(ind).smallerpflatDown_side1,Y6_intersect(ind).smallerY6_intersect);

Y7_intersect(ind).smallerY7_intersect=fzero @(x) polyval(pMid3_side1(ind).smallerpMid3_side1-pMid2_side1(ind).smallerpMid2_side1,x),3);
Z7_intersect(ind).smallerZ7_intersect=polyval(pMid3_side1(ind).smallerpMid3_side1,Y7_intersect(ind).smallerY7_intersect);

Y8_intersect(ind).smallerY8_intersect=fzero @(x) polyval(pMid2_side1(ind).smallerpMid2_side1-pMid1_side1(ind).smallerpMid1_side1,x),3);
Z8_intersect(ind).smallerZ8_intersect=polyval(pMid2_side1(ind).smallerpMid2_side1,Y8_intersect(ind).smallerY8_intersect);

Y9_intersect(ind).smallerY9_intersect=fzero @(x) polyval(pMid1_side1(ind).smallerpMid1_side1-pup1(ind).smallerpup1,x),3);
Z9_intersect(ind).smallerZ9_intersect=polyval(pMid1_side1(ind).smallerpMid1_side1,Y9_intersect(ind).smallerY9_intersect);

%%New matrix of points with same flattened x-value from right
intersect_xyz1(ind).smallerintersect_xyz1=[flatup1(ind).smallerflatup1(1,1),Y1_intersect(ind).smallerY1_intersect,Z1_intersect(ind).smallerZ1_intersect];
intersect_xyz2(ind).smallerintersect_xyz2=[flatup1(ind).smallerflatup1(1,1),Y2_intersect(ind).smallerY2_intersect,Z2_intersect(ind).smallerZ2_intersect];

```

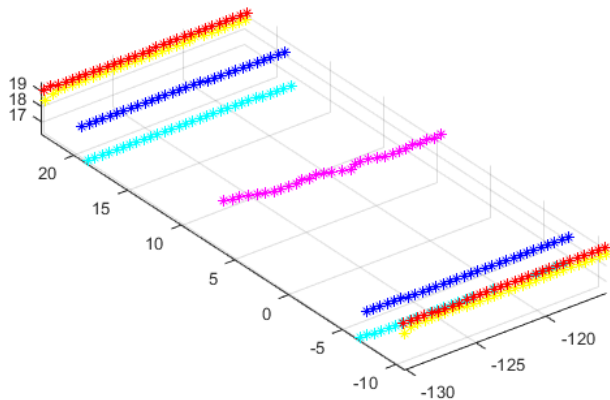
```

intersect_xyz3(ind).smallerintersect_xyz3=[flatup1(ind).smallerflatup1(1,1),Y3_intersect(ind).smallerY3_intersect,Z3_intersect(ind).smallerZ3_intersect];
intersect_xyz4(ind).smallerintersect_xyz4=[flatup1(ind).smallerflatup1(1,1),Y4_intersect(ind).smallerY4_intersect,Z4_intersect(ind).smallerZ4_intersect];
intersect_xyz5(ind).smallerintersect_xyz5=[flatup1(ind).smallerflatup1(1,1),Y5_intersect(ind).smallerY5_intersect,Z5_intersect(ind).smallerZ5_intersect];
intersect_xyz6(ind).smallerintersect_xyz6=[flatup1(ind).smallerflatup1(1,1),Y6_intersect(ind).smallerY6_intersect,Z6_intersect(ind).smallerZ6_intersect];
intersect_xyz7(ind).smallerintersect_xyz7=[flatup1(ind).smallerflatup1(1,1),Y7_intersect(ind).smallerY7_intersect,Z7_intersect(ind).smallerZ7_intersect];
intersect_xyz8(ind).smallerintersect_xyz8=[flatup1(ind).smallerflatup1(1,1),Y8_intersect(ind).smallerY8_intersect,Z8_intersect(ind).smallerZ8_intersect];
intersect_xyz9(ind).smallerintersect_xyz9=[flatup1(ind).smallerflatup1(1,1),Y9_intersect(ind).smallerY9_intersect,Z9_intersect(ind).smallerZ9_intersect];
end
%%Figure of the unrotated intersection
for ind = 1:numel(xRngs)-1

figure(7)
scatter3(intersect_xyz1(ind).smallerintersect_xyz1(:,1),intersect_xyz1(ind).smallerintersect_xyz1(:,2),intersect_xyz1(ind).smallerintersect_xyz1(:,3), '*c')
hold on
scatter3(intersect_xyz2(ind).smallerintersect_xyz2(:,1),intersect_xyz2(ind).smallerintersect_xyz2(:,2),intersect_xyz2(ind).smallerintersect_xyz2(:,3), '*b')
hold on
scatter3(intersect_xyz3(ind).smallerintersect_xyz3(:,1),intersect_xyz3(ind).smallerintersect_xyz3(:,2),intersect_xyz3(ind).smallerintersect_xyz3(:,3), '*y')
hold on
scatter3(intersect_xyz4(ind).smallerintersect_xyz4(:,1),intersect_xyz4(ind).smallerintersect_xyz4(:,2),intersect_xyz4(ind).smallerintersect_xyz4(:,3), '*r')
hold on
scatter3(intersect_xyz5(ind).smallerintersect_xyz5(:,1),intersect_xyz5(ind).smallerintersect_xyz5(:,2),intersect_xyz5(ind).smallerintersect_xyz5(:,3), '*m')
hold on
scatter3(intersect_xyz6(ind).smallerintersect_xyz6(:,1),intersect_xyz6(ind).smallerintersect_xyz6(:,2),intersect_xyz6(ind).smallerintersect_xyz6(:,3), '*r')
hold on
scatter3(intersect_xyz7(ind).smallerintersect_xyz7(:,1),intersect_xyz7(ind).smallerintersect_xyz7(:,2),intersect_xyz7(ind).smallerintersect_xyz7(:,3), '*y')
hold on
scatter3(intersect_xyz8(ind).smallerintersect_xyz8(:,1),intersect_xyz8(ind).smallerintersect_xyz8(:,2),intersect_xyz8(ind).smallerintersect_xyz8(:,3), '*b')
hold on
scatter3(intersect_xyz9(ind).smallerintersect_xyz9(:,1),intersect_xyz9(ind).smallerintersect_xyz9(:,2),intersect_xyz9(ind).smallerintersect_xyz9(:,3), '*c')
axis equal

end

```



**Rotate intersection point**

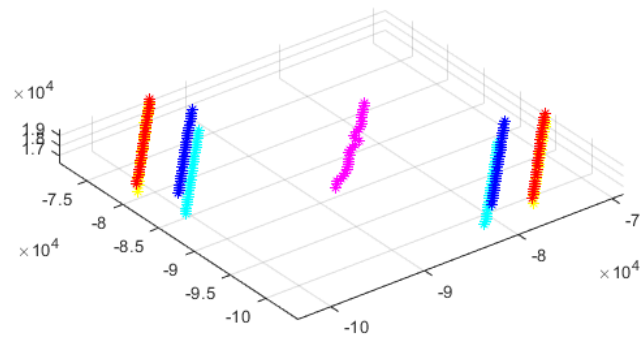
```

%%Rotate pier back to original orientation and convert from m to mm
for ind = 1:numel(xRngs)-1
rotintersect_xyz1(ind).smallerrotintersect_xyz1=intersect_xyz1(ind).smallerintersect_xyz1*inv(Rz)*10^3;
rotintersect_xyz2(ind).smallerrotintersect_xyz2=intersect_xyz2(ind).smallerintersect_xyz2*inv(Rz)*10^3;
rotintersect_xyz3(ind).smallerrotintersect_xyz3=intersect_xyz3(ind).smallerintersect_xyz3*inv(Rz)*10^3;
rotintersect_xyz4(ind).smallerrotintersect_xyz4=intersect_xyz4(ind).smallerintersect_xyz4*inv(Rz)*10^3;
rotintersect_xyz5(ind).smallerrotintersect_xyz5=intersect_xyz5(ind).smallerintersect_xyz5*inv(Rz)*10^3;
rotintersect_xyz6(ind).smallerrotintersect_xyz6=intersect_xyz6(ind).smallerintersect_xyz6*inv(Rz)*10^3;
rotintersect_xyz7(ind).smallerrotintersect_xyz7=intersect_xyz7(ind).smallerintersect_xyz7*inv(Rz)*10^3;
rotintersect_xyz8(ind).smallerrotintersect_xyz8=intersect_xyz8(ind).smallerintersect_xyz8*inv(Rz)*10^3;
rotintersect_xyz9(ind).smallerrotintersect_xyz9=intersect_xyz9(ind).smallerintersect_xyz9*inv(Rz)*10^3;
end

%%Figure of the intersection pionts, rotated back to original positions
for ind = 1:numel(xRngs)-1

figure(8)
scatter3(rotintersect_xyz1(ind).smallerrotintersect_xyz1(:,1),rotintersect_xyz1(ind).smallerrotintersect_xyz1(:,2),rotintersect_xyz1(ind).smallerrotintersect_xyz1(:,3), '*c')
hold on
scatter3(rotintersect_xyz2(ind).smallerrotintersect_xyz2(:,1),rotintersect_xyz2(ind).smallerrotintersect_xyz2(:,2),rotintersect_xyz2(ind).smallerrotintersect_xyz2(:,3), '*b')
hold on
scatter3(rotintersect_xyz3(ind).smallerrotintersect_xyz3(:,1),rotintersect_xyz3(ind).smallerrotintersect_xyz3(:,2),rotintersect_xyz3(ind).smallerrotintersect_xyz3(:,3), '*y')
hold on
scatter3(rotintersect_xyz4(ind).smallerrotintersect_xyz4(:,1),rotintersect_xyz4(ind).smallerrotintersect_xyz4(:,2),rotintersect_xyz4(ind).smallerrotintersect_xyz4(:,3), '*r')
hold on
scatter3(rotintersect_xyz5(ind).smallerrotintersect_xyz5(:,1),rotintersect_xyz5(ind).smallerrotintersect_xyz5(:,2),rotintersect_xyz5(ind).smallerrotintersect_xyz5(:,3), '*m')
hold on
scatter3(rotintersect_xyz6(ind).smallerrotintersect_xyz6(:,1),rotintersect_xyz6(ind).smallerrotintersect_xyz6(:,2),rotintersect_xyz6(ind).smallerrotintersect_xyz6(:,3), '*r')
hold on
scatter3(rotintersect_xyz7(ind).smallerrotintersect_xyz7(:,1),rotintersect_xyz7(ind).smallerrotintersect_xyz7(:,2),rotintersect_xyz7(ind).smallerrotintersect_xyz7(:,3), '*y')
hold on
scatter3(rotintersect_xyz8(ind).smallerrotintersect_xyz8(:,1),rotintersect_xyz8(ind).smallerrotintersect_xyz8(:,2),rotintersect_xyz8(ind).smallerrotintersect_xyz8(:,3), '*b')
hold on
scatter3(rotintersect_xyz9(ind).smallerrotintersect_xyz9(:,1),rotintersect_xyz9(ind).smallerrotintersect_xyz9(:,2),rotintersect_xyz9(ind).smallerrotintersect_xyz9(:,3), '*c')
axis equal
end

```



#### Write Excel file

```
writetable(struct2table(rotintersect_xyz1), 'Intersect_box_2.xlsx', 'sheet', 1);  
writetable(struct2table(rotintersect_xyz2), 'Intersect_box_2.xlsx', 'sheet', 2);  
writetable(struct2table(rotintersect_xyz3), 'Intersect_box_2.xlsx', 'sheet', 3);  
writetable(struct2table(rotintersect_xyz4), 'Intersect_box_2.xlsx', 'sheet', 4);  
writetable(struct2table(rotintersect_xyz5), 'Intersect_box_2.xlsx', 'sheet', 5);  
writetable(struct2table(rotintersect_xyz6), 'Intersect_box_2.xlsx', 'sheet', 6);  
writetable(struct2table(rotintersect_xyz7), 'Intersect_box_2.xlsx', 'sheet', 7);  
writetable(struct2table(rotintersect_xyz8), 'Intersect_box_2.xlsx', 'sheet', 8);  
writetable(struct2table(rotintersect_xyz9), 'Intersect_box_2.xlsx', 'sheet', 9);  
winopen 'Intersect_box_2.xlsx'
```

Published with MATLAB® R2019b

## **A3    Box Girder 2-9 Algorithm**



## Contents

---

- [Box girder 2-9](#)
- [Rotate](#)
- [Slice](#)
- [Vertical cut to find outer parts](#)
- [Horizontal cut](#)
- [Vertical cut](#)
- [Intersection points](#)
- [Rotate intersection point](#)
- [Write Excel file](#)

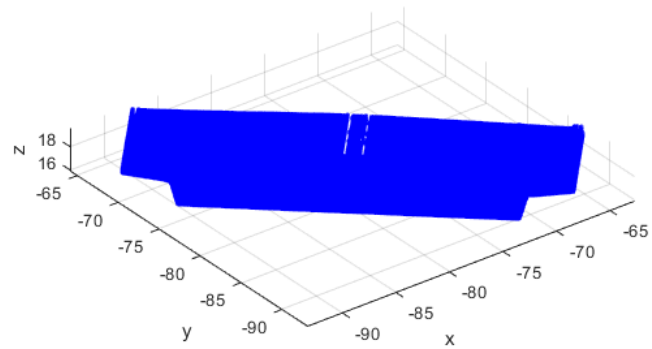
## Box girder 2-9

---

```
clear all
close all
clc

A=dlmread('Box_2.txt');    %%Reading the txt-file of the point cloud
a=A(:,[1,2,3]);           %%Matrix of the geometrical data
x=a(:,1);
y=a(:,2);
z=a(:,3);

figure(1)
scatter3(x,y,z,'.b')
hold on
xlabel('x')
ylabel('y')
zlabel('z')
grid on
axis('equal')
```



## Rotate

Find angle to rotate

```
[maxX, indexOfMaxX] = max(x);
yAtMaxX = y(indexOfMaxX);
[minY, indexOfMinY] = min(y);
xAtMinY = x(indexOfMinY);

u=[maxX-xAtMinY, yAtMaxX- minY];
v=[1,0];

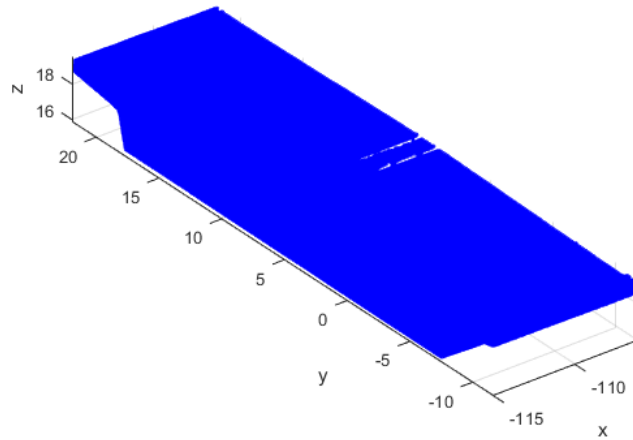
slope=u(:,2)/u(:,1);
theta=atan(slope);           %%Angle

Rz=[cos(theta)  -sin(theta)  0;   %%Rotation matrix
    sin(theta)  cos(theta)  0;
    0  0  1];

%%Rotate the whole pier
rotM=a*Rz;

%%Rotated matrix
x1=rotM(:,1);
y1=rotM(:,2);
z1=rotM(:,3);

figure(2)
scatter3(x1,y1,z1,'.b')
hold on
xlabel('x')
ylabel('y')
zlabel('z')
grid on
axis('equal')
```



### Slice

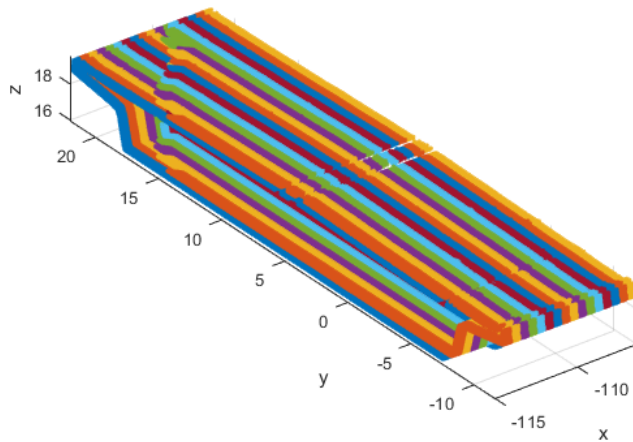
```

%%Dividing the rotated Box girder into main segments
xRngs=[min(rotM(:,1)):0.5: max(rotM(:,1))];
for ind = 1: numel(xRngs)-1

    rows = rotM(:,1)>xRngs(ind) & rotM(:,1)<xRngs(ind+1); %check to see which rows of b are between
    %xRng(ind) and xRng(ind+1)
    segM(ind).SmallerM = rotM(rows,:); %segM is the segmented matrix of the rotated pier

    figure(3)
    scatter3(segM(ind).SmallerM(:,1),segM(ind).SmallerM(:,2),segM(ind).SmallerM(:,3),'.') %Scatter the rotated pier
    hold on
    xlabel('x')
    ylabel('y')
    zlabel('z')
    grid on
    axis('equal')
end

```



#### Vertical cut to find outer parts

```

%%Splitting the box Vertically, finding the two outer sides
for ind = 1:numel(xRngs)-1
    MaxY_Box(ind)=max(segM(ind).SmallerM(:,2));
    MinY_Box(ind)=min(segM(ind).SmallerM(:,2));

    e=0.01;
    kVer(ind) = floor((MaxY_Box(ind)-MinY_Box(ind))/e);
    for i = 1:kVer(ind)
        mVer(ind).SmallerVer(i,1) = MinY_Box(ind)+e*(i-1);
        A2Ver(ind).SmallerA2Ver= segM(ind).SmallerM((segM(ind).SmallerM(:,2)>=mVer(ind).SmallerVer(i,1) & segM(ind).SmallerM(:,2)<=mVer(ind).SmallerVer(i,1)+e),:);
        ggVer_Box(ind).SmallerggVer_Box{i,1}=A2Ver(ind).SmallerA2Ver;           %%Counting the points in each microsegment
        if isempty(A2Ver(ind).SmallerA2Ver)==1
            continue
        end
    end
    ggVer_Box(ind).SmallerggVer_Box=cell2mat(ggVer_Box(ind).SmallerggVer_Box);
    ggVer_mid3_side1(ind).SmallerggVer_mid3_side1=ggVer_Box(ind).SmallerggVer_Box((ggVer_Box(ind).SmallerggVer_Box(:,2)<=MinY_Box(ind)+0.2),:);
    ggVer_mid3_side2(ind).SmallerggVer_mid3_side2=ggVer_Box(ind).SmallerggVer_Box((ggVer_Box(ind).SmallerggVer_Box(:,2)>=MaxY_Box(ind)-0.2),:);
end

%%Removing unregular part
for ind = 1:numel(xRngs)-1
    %side 1
    MaxZ_Box_side1(ind)= max(ggVer_mid3_side1(ind).SmallerggVer_mid3_side1(:,3));
    MinZ_Box_side1(ind)= min(ggVer_mid3_side1(ind).SmallerggVer_mid3_side1(:,3));

    mid3_side1(ind).smallermid3_side1= ggVer_mid3_side1(ind).SmallerggVer_mid3_side1((ggVer_mid3_side1(ind).SmallerggVer_mid3_side1(:,3)<=MaxZ_Box_side1(ind)-0.1),:);
    mid3_side1(ind).smallermid3_side1= ggVer_mid3_side1(ind).SmallerggVer_mid3_side1((ggVer_mid3_side1(ind).SmallerggVer_mid3_side1(:,3)>=MinZ_Box_side1(ind)+0.2),:);

    %side 2
    MaxZ_Box_side2(ind)= max(ggVer_mid3_side2(ind).SmallerggVer_mid3_side2(:,3));
    MinZ_Box_side2(ind)= min(ggVer_mid3_side2(ind).SmallerggVer_mid3_side2(:,3));

    mid3_side2(ind).smallermid3_side2= ggVer_mid3_side2(ind).SmallerggVer_mid3_side2((ggVer_mid3_side2(ind).SmallerggVer_mid3_side2(:,3)<=MaxZ_Box_side2(ind)-0.1),:);

```

```

mid3_side2(ind).smallermid3_side2= ggVer_mid3_side2(ind).SmallerggVer_mid3_side2((ggVer_mid3_side2(ind).SmallerggVer_mid3_side2(:,3))>=MinZ_Box_side2(ind)+0.2),:);

end

%%Excluding the outer part from the point cloud
for ind = 1:numel(xRngs)-1
[ia, ~] = ismember(segM(ind).SmallerM, ggVer_mid3_side1(ind).SmallerggVer_mid3_side1, 'rows');
segM(ind).SmallerM(ia, :) = [];
[ia, ib] = ismember(segM(ind).SmallerM, ggVer_mid3_side2(ind).SmallerggVer_mid3_side2, 'rows');
segM(ind).SmallerM(ia, :) = [];
segM(ind).SmallerM = segM(ind).SmallerM(segM(ind).SmallerM(:,2) >= min(ggVer_mid3_side1(ind).SmallerggVer_mid3_side1(:,2)), :), :);
segM(ind).SmallerM = segM(ind).SmallerM(segM(ind).SmallerM(:,2) <= max(ggVer_mid3_side2(ind).SmallerggVer_mid3_side2(:,2)), :), :);
end

%%Figure of Boxgirder without the outer part
for ind = 1:numel(xRngs)-1
figure(4)
scatter3(segM(ind).SmallerM(:,1),segM(ind).SmallerM(:,2),segM(ind).SmallerM(:,3),'.b')
hold on
scatter3(mid3_side1(ind).smallermid3_side1(:,1),mid3_side1(ind).smallermid3_side1(:,2),mid3_side1(ind).smallermid3_side1(:,3),'.r')
hold on
scatter3(mid3_side2(ind).smallermid3_side2(:,1),mid3_side2(ind).smallermid3_side2(:,2),mid3_side2(ind).smallermid3_side2(:,3),'.g')
axis equal
end

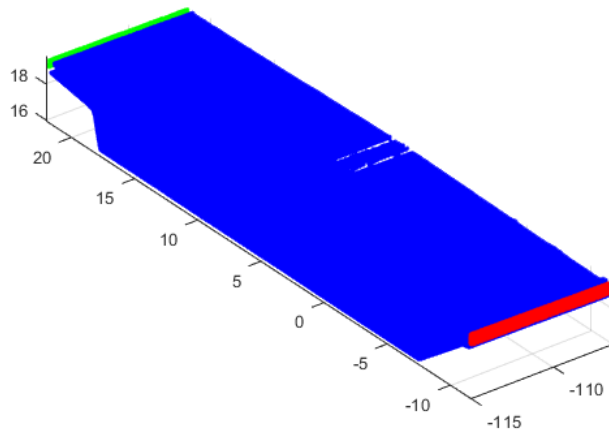
%%Find mean, max and min values
for ind = 1:numel(xRngs)-1
maximumZ(ind)= max(segM(ind).SmallerM(:,3));
minimumZ(ind)=min(segM(ind).SmallerM(:,3));

maximumY(ind)= max(segM(ind).SmallerM(:,2));
minimumY(ind)=min(segM(ind).SmallerM(:,2));

columnMean(ind).SmallercolumnMean = mean(segM(ind).SmallerM,1);

meanx(ind).Smallermeanx=columnMean(ind).SmallercolumnMean(1,1);
meanY(ind).SmallermeanY=columnMean(ind).SmallercolumnMean(1,2);
meanZ(ind).SmallermeanZ=columnMean(ind).SmallercolumnMean(1,3);
end

```



### Horizontal cut

```

%%Dividing the segments into microsegments in the horizontal direction,
%%finding changes in amount of points
for ind = 1:numel(xRngs)-1

e1=0.05;
kHor(ind) = floor((maximumZ(ind)-minimumZ(ind))/e1);           %How many times its divided
for i = 1:kHor(ind)
    mHor(ind).SmallermHor(i,1) = minimumZ(ind)+e1*(i-1);       %For which values its divided for each segment
    A2Hor(ind).SmallerA2Hor= segM(ind).SmallerM((segM(ind).SmallerM(:,3)>=mHor(ind).SmallermHor(i,1) & segM(ind).SmallerM(:,3)<=mHor(ind).SmallermHor(i,1)+e1),:); %logical indexing
    ggHor(ind).SmallerggHor{i,1}=A2Hor(ind).SmallerA2Hor;       %Counting the points in each microsegment
    if isempty(A2Hor(ind).SmallerA2Hor)=1
        continue
    end
end
for i=1:kHor(ind)-1
    ChangeHor(ind).smallerChangeHor{i,:}=size(ggHor(ind).SmallerggHor{i,1})/size(ggHor(ind).SmallerggHor{i+1,1});

ChangeHor(ind).smallerChangeHor{i,:}=ChangeHor(ind).smallerChangeHor{i,1}(:,1);
ChangeHor1(ind).smallerChangeHor1=cell2mat(ChangeHor(ind).smallerChangeHor);
end
end

%%Finding the positions to cut
for ind = 1:numel(xRngs)-1

[m_1(ind).smallerm_1,idx_1(ind).smalleridx_1] = max(ChangeHor1(ind).smallerChangeHor1(1:floor(kHor(ind)/2),:));
ChangeHor1(ind).smallerChangeHor1(idx_1(ind).smalleridx_1) = NaN ;
idx_2(ind).smalleridx_2= find(diff(ChangeHor1(ind).smallerChangeHor1,2)==0,1); %The removed outer part always makes this position to be several repeting 1 %prøve å endre

if idx_1(ind).smalleridx_1 > idx_2(ind).smalleridx_2
    [m_3(ind).smallerm_3,idx_3(ind).smalleridx_3] = min(ChangeHor1(ind).smallerChangeHor1(idx_2(ind).smalleridx_2:idx_1(ind).smalleridx_1) );
else
    [m_3(ind).smallerm_3,idx_3(ind).smalleridx_3] = min(ChangeHor1(ind).smallerChangeHor1(idx_1(ind).smalleridx_1:idx_2(ind).smalleridx_2) );
end

```

```

m_4(ind).smallerm_4 = min(ChangeHor1(ind).smallerChangeHor1(idx_2(ind).smalleridx_2:end)) ;
idx_4(ind).smalleridx_4 = find(ChangeHor1(ind).smallerChangeHor1==m_4(ind).smallerm_4);

CutPosition(ind).smallerCutPosition=sort([idx_1(ind).smalleridx_1;idx_2(ind).smalleridx_2;idx_3(ind).smalleridx_3;idx_4(ind).smalleridx_4]);

end

%%Removing the ends from both sides of the mid 1,cos horizontal cut
for ind = 1:numel(xRngs)-1

up(ind).smallerup=ggHor(ind).SmallerggHor(1:CutPosition(ind).smallerCutPosition(1,:),:);
up1(ind).smallerup1=cell2mat(up(ind).smallerup);

mid1(ind).smallermid1=ggHor(ind).SmallerggHor(CutPosition(ind).smallerCutPosition(1,:)+1:CutPosition(ind).smallerCutPosition(2,:),:);
mid1(ind).smallermid1=mid1(ind).smallermid1(5:end-4);
mid11(ind).smallermid11=cell2mat(mid1(ind).smallermid1);

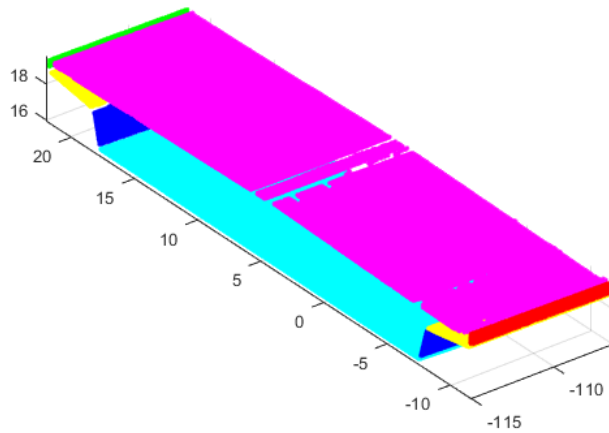
mid2(ind).smallermid2=ggHor(ind).SmallerggHor(CutPosition(ind).smallerCutPosition(2,:)+1:CutPosition(ind).smallerCutPosition(3,:),:);
mid22(ind).smallermid22=cell2mat(mid2(ind).smallermid2);

%%mid3 is already splitted as the outer parts

down(ind).smallerdown=ggHor(ind).SmallerggHor(CutPosition(ind).smallerCutPosition(4,:)+1:end,:);
down1(ind).smallerdown1=cell2mat(down(ind).smallerdown);

figure(5)
scatter3(up1(ind).smallerup1(:,1),up1(ind).smallerup1(:,2),up1(ind).smallerup1(:,3),'.c')
hold on
scatter3(mid11(ind).smallermid11(:,1),mid11(ind).smallermid11(:,2),mid11(ind).smallermid11(:,3),'.b')
hold on
scatter3(mid22(ind).smallermid22(:,1),mid22(ind).smallermid22(:,2),mid22(ind).smallermid22(:,3),'.y')
hold on
scatter3(down1(ind).smallerdown1(:,1),down1(ind).smallerdown1(:,2),down1(ind).smallerdown1(:,3),'.m')
hold on
scatter3(mid3_side1(ind).smallermid3_side1(:,1),mid3_side1(ind).smallermid3_side1(:,2),mid3_side1(ind).smallermid3_side1(:,3),'.r')
hold on
scatter3(mid3_side2(ind).smallermid3_side2(:,1),mid3_side2(ind).smallermid3_side2(:,2),mid3_side2(ind).smallermid3_side2(:,3),'.g')
axis equal
end

```



### Vertical cut

```

%%splitting the sides
for ind = 1:numel(xRngs)-1
    %%Up is has only one side

    Mid1_side1(ind).smallerMid1_side1=find(mid11(ind).smallermid11(:,2)< (meanY(ind).SmallermeanY));
    Mid1_side2(ind).smallerMid1_side2=find(mid11(ind).smallermid11(:,2)> (meanY(ind).SmallermeanY));
    Mid1_side1(ind).smallerMid1_side1=mid11(ind).smallermid11(Mid1_side1(ind).smallerMid1_side1,:);
    Mid1_side2(ind).smallerMid1_side2=mid11(ind).smallermid11(Mid1_side2(ind).smallerMid1_side2,:);

    Mid2_side1(ind).smallerMid2_side1=find(mid22(ind).smallermid22(:,2)< (meanY(ind).SmallermeanY));
    Mid2_side2(ind).smallerMid2_side2=find(mid22(ind).smallermid22(:,2)> (meanY(ind).SmallermeanY));
    Mid2_side1(ind).smallerMid2_side1=mid22(ind).smallermid22(Mid2_side1(ind).smallerMid2_side1,:);
    Mid2_side2(ind).smallerMid2_side2=mid22(ind).smallermid22(Mid2_side2(ind).smallerMid2_side2,:);

    Down_side1(ind).smallerDown_side1=find(down1(ind).smallerdown1(:,2)< (meanY(ind).SmallermeanY));
    Down_side2(ind).smallerDown_side2=find(down1(ind).smallerdown1(:,2)> (meanY(ind).SmallermeanY));
    Down_side1(ind).smallerDown_side1=down1(ind).smallerdown1(Down_side1(ind).smallerDown_side1,:);
    Down_side2(ind).smallerDown_side2=down1(ind).smallerdown1(Down_side2(ind).smallerDown_side2,:);
end

%%splitting Mid2_side1 Vertically, removing the ends from both sides
for ind = 1:numel(xRngs)-1
    MaxY_Mid2_side1(ind)=max(Mid2_side1(ind).smallerMid2_side1(:,2));
    MinY_Mid2_side1(ind)=min(Mid2_side1(ind).smallerMid2_side1(:,2));
    e=0.1;
    kVer(ind) = floor((MaxY_Mid2_side1(ind)-MinY_Mid2_side1(ind))/e);
    for i = 1:kVer(ind)
        mVer(ind).SmallermVer(i,1) = MinY_Mid2_side1(ind)+e*(i-1);
        A2Ver(ind).SmallerA2Ver = Mid2_side1(ind).smallerMid2_side1((Mid2_side1(ind).smallerMid2_side1(:,2))>=mVer(ind).SmallermVer(i,1) & Mid2_side1(ind).smallerMid2_side1(:,2)<=mVer(ind).SmallermVer(i,1)+e),:); %logical indexing
        ggVer_mid2_side1(ind).SmallerggVer_mid2_side1{i,1}=A2Ver(ind).SmallerA2Ver; %Counting the points in each microsegment
        if isempty(A2Ver(ind).SmallerA2Ver)=1
            continue
        end
    end
end

```



```

ggVer_mid2_side1(ind).SmallerggVer_mid2_side1=ggVer_mid2_side1(ind).SmallerggVer_mid2_side1(2:end-2);
ggVer_mid2_side1(ind).SmallerggVer_mid2_side1=cell2mat(ggVer_mid2_side1(ind).SmallerggVer_mid2_side1);
end
%%splitting Mid2_side2 Vertically, removing the ends from both sides
for ind = 1:numel(xRngs)-1
MaxY_Mid2_side2(ind)=max(Mid2_side2(ind).smallerMid2_side2(:,2));
MinY_Mid2_side2(ind)=min(Mid2_side2(ind).smallerMid2_side2(:,2));
e=0.1;
kVer(ind) = floor((MaxY_Mid2_side2(ind)-MinY_Mid2_side2(ind))/e);
for i = 1:kVer(ind)
mVer(ind).SmallerVer(i,1) = MinY_Mid2_side2(ind)+e*(i-1);
A2Ver(ind).SmallerA2Ver= Mid2_side2(ind).smallerMid2_side2((Mid2_side2(ind).smallerMid2_side2(:,2)>=mVer(ind).SmallerVer(i,1) & Mid2_side2(ind).smallerMid2_side2(:,2)<=mVer(ind).SmallerVer(i,1)+e),:); %logical indexing
ggVer_mid2_side2(ind).SmallerggVer_mid2_side2{i,1}=A2Ver(ind).SmallerA2Ver; %Counting the points in each microsegment
if isempty(A2Ver(ind).SmallerA2Ver)==1
continue
end
end
ggVer_mid2_side2(ind).SmallerggVer_mid2_side2=ggVer_mid2_side2(ind).SmallerggVer_mid2_side2(2:end-2);
ggVer_mid2_side2(ind).SmallerggVer_mid2_side2=cell2mat(ggVer_mid2_side2(ind).SmallerggVer_mid2_side2);
end

%%splitting Up vertically, removing the ends from both sides
for ind = 1:numel(xRngs)-1
MaxY_Up(ind)=max(up1(ind).smallerup1(:,2));
MinY_Up(ind)=min(up1(ind).smallerup1(:,2));
e=0.1;
kVer(ind) = floor((MaxY_Up(ind)-MinY_Up(ind))/e);
for i = 1:kVer(ind)
mVer(ind).SmallerVer(i,1) = MinY_Up(ind)+e*(i-1);
A2Ver(ind).SmallerA2Ver= up1(ind).smallerup1((up1(ind).smallerup1(:,2)>=mVer(ind).SmallerVer(i,1) & up1(ind).smallerup1(:,2)<=mVer(ind).SmallerVer(i,1)+e),:); %logical indexing
ggVer_Up(ind).SmallerggVer_Up{i,1}=A2Ver(ind).SmallerA2Ver; %Counting the points in each microsegment
if isempty(A2Ver(ind).SmallerA2Ver)==1
continue
end
end
ggVer_Up(ind).SmallerggVer_Up=ggVer_Up(ind).SmallerggVer_Up(10:end-10);
ggVer_Up(ind).SmallerggVer_Up=cell2mat(ggVer_Up(ind).SmallerggVer_Up);
end

%%splitting Down_side1 vertically, removing the ends from both sides
for ind = 1:numel(xRngs)-1
MaxY_Down_side1(ind)=max(Down_side1(ind).smallerDown_side1(:,2));
MinY_Down_side1(ind)=min(Down_side1(ind).smallerDown_side1(:,2));
e=0.1;
kVer(ind) = floor((MaxY_Down_side1(ind)-MinY_Down_side1(ind))/e);
for i = 1:kVer(ind)
mVer(ind).SmallerVer(i,1) = MinY_Down_side1(ind)+e*(i-1);
A2Ver(ind).SmallerA2Ver= Down_side1(ind).smallerDown_side1((Down_side1(ind).smallerDown_side1(:,2)>=mVer(ind).SmallerVer(i,1) & Down_side1(ind).smallerDown_side1(:,2)<=mVer(ind).SmallerVer(i,1)+e),:); %logical indexing
ggVer_Down_side1(ind).SmallerggVer_Down_side1{i,1}=A2Ver(ind).SmallerA2Ver; %Counting the points in each microsegment
if isempty(A2Ver(ind).SmallerA2Ver)==1
continue
end
end
ggVer_Down_side1(ind).SmallerggVer_Down_side1=ggVer_Down_side1(ind).SmallerggVer_Down_side1(20:end-20);
ggVer_Down_side1(ind).SmallerggVer_Down_side1=cell2mat(ggVer_Down_side1(ind).SmallerggVer_Down_side1);
end

%%splitting Down_side2 vertically, removing the ends from both sides
for ind = 1:numel(xRngs)-1
MaxY_Down_side2(ind)=max(Down_side2(ind).smallerDown_side2(:,2));
MinY_Down_side2(ind)=min(Down_side2(ind).smallerDown_side2(:,2));
e=0.1;
kVer(ind) = floor((MaxY_Down_side2(ind)-MinY_Down_side2(ind))/e);
for i = 1:kVer(ind)

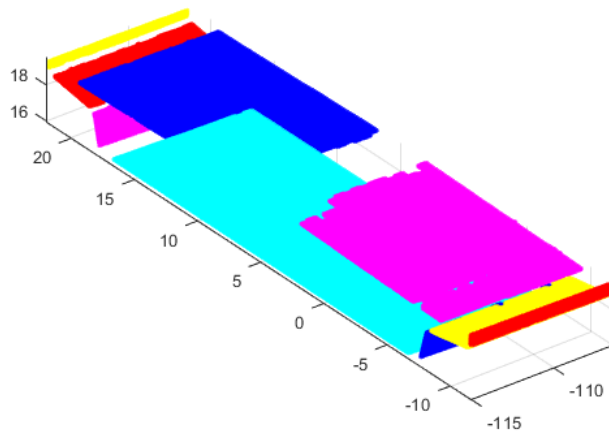
```

```

mVer(ind).SmallermVer(i,1) = MinY_Down_side2(ind)+e*(i-1);
A2Ver(ind).SmallerA2Ver= Down_side2(ind).smallerDown_side2((Down_side2(ind).smallerDown_side2(:,2))>=mVer(ind).SmallermVer(i,1) & Down_side2(ind).smallerDown_side2(:,2)<=mVer(ind).SmallermVer(i,1)+e),:); %logical indexing
ggVer_Down_side2(ind).SmallerggVer_Down_side2{i,1}=A2Ver(ind).SmallerA2Ver; %Counting the points in each microsegment
if isempty(A2Ver(ind).SmallerA2Ver)==1
continue
end
end
ggVer_Down_side2(ind).SmallerggVer_Down_side2=ggVer_Down_side2(ind).SmallerggVer_Down_side2(20:end-20);
ggVer_Down_side2(ind).SmallerggVer_Down_side2=cell2mat(ggVer_Down_side2(ind).SmallerggVer_Down_side2);
end

%%Figure of the divided box girder
for ind = 1:numel(xRngs)-1
figure(6)
scatter3(ggVer_Up(ind).SmallerggVer_Up(:,1),ggVer_Up(ind).SmallerggVer_Up(:,2),ggVer_Up(ind).SmallerggVer_Up(:,3),'c')
hold on
scatter3(Mid1_side1(ind).smallerMid1_side1(:,1),Mid1_side1(ind).smallerMid1_side1(:,2),Mid1_side1(ind).smallerMid1_side1(:,3),'b')
hold on
scatter3(Mid1_side2(ind).smallerMid1_side2(:,1),Mid1_side2(ind).smallerMid1_side2(:,2),Mid1_side2(ind).smallerMid1_side2(:,3),'m')
hold on
scatter3(ggVer_mid2_side1(ind).SmallerggVer_mid2_side1(:,1),ggVer_mid2_side1(ind).SmallerggVer_mid2_side1(:,2),ggVer_mid2_side1(ind).SmallerggVer_mid2_side1(:,3),'y')
hold on
scatter3(ggVer_mid2_side2(ind).SmallerggVer_mid2_side2(:,1),ggVer_mid2_side2(ind).SmallerggVer_mid2_side2(:,2),ggVer_mid2_side2(ind).SmallerggVer_mid2_side2(:,3),'r')
hold on
scatter3(mid3_side1(ind).smallermid3_side1(:,1),mid3_side1(ind).smallermid3_side1(:,2),mid3_side1(ind).smallermid3_side1(:,3),'r')
hold on
scatter3(mid3_side2(ind).smallermid3_side2(:,1),mid3_side2(ind).smallermid3_side2(:,2),mid3_side2(ind).smallermid3_side2(:,3),'y')
hold on
scatter3( ggVer_Down_side1(ind).SmallerggVer_Down_side1(:,1), ggVer_Down_side1(ind).SmallerggVer_Down_side1(:,2), ggVer_Down_side1(ind).SmallerggVer_Down_side1(:,3),'m')
hold on
scatter3(ggVer_Down_side2(ind).SmallerggVer_Down_side2(:,1),ggVer_Down_side2(ind).SmallerggVer_Down_side2(:,2),ggVer_Down_side2(ind).SmallerggVer_Down_side2(:,3),'b')
axis equal
end

```



### Intersection points

```

%%Assigning one X-value to all the rows
for ind = 1:numel(xRngs)-1

    %%Flatten Up
    flatup1(ind).smallerflatup1=up1(ind).smallerup1;
    if isempty(up1(ind).smallerup1)==1
        continue
    end
    flatup1(ind).smallerflatup1(:,1)=repelem(flatup1(ind).smallerflatup1(1),size(flatup1(ind).smallerflatup1,1),1);
    pup1(ind).smallerpup1 = polyfit(flatup1(ind).smallerflatup1(:,2),flatup1(ind).smallerflatup1(:,3),1);
    fup1(ind).smallerfup1 = polyval(pup1(ind).smallerpup1,flatup1(ind).smallerflatup1(:,2));

%%Flatten Mid1_side1
    flatMid1_side1(ind).smallerflatMid1_side1=Mid1_side1(ind).smallerMid1_side1;
    if isempty(Mid1_side1(ind).smallerMid1_side1)==1
        continue
    end
    flatMid1_side1(ind).smallerflatMid1_side1(:,1)=repelem(flatMid1_side1(ind).smallerflatMid1_side1(1),size(flatMid1_side1(ind).smallerflatMid1_side1,1),1);
    pMid1_side1(ind).smallerpMid1_side1 = polyfit(flatMid1_side1(ind).smallerflatMid1_side1(:,2),flatMid1_side1(ind).smallerflatMid1_side1(:,3),1);
    fMid1_side1(ind).smallerfMid1_side1 = polyval(pMid1_side1(ind).smallerpMid1_side1,flatMid1_side1(ind).smallerflatMid1_side1(:,2));

%%Flatten Mid1_side2
    flatMid1_side2(ind).smallerflatMid1_side2=Mid1_side2(ind).smallerMid1_side2;
    if isempty(Mid1_side2(ind).smallerMid1_side2)==1
        continue
    end
    flatMid1_side2(ind).smallerflatMid1_side2(:,1)=repelem(flatMid1_side2(ind).smallerflatMid1_side2(1),size(flatMid1_side2(ind).smallerflatMid1_side2,1),1);
    pMid1_side2(ind).smallerpMid1_side2 = polyfit(flatMid1_side2(ind).smallerflatMid1_side2(:,2),flatMid1_side2(ind).smallerflatMid1_side2(:,3),1);
    fMid1_side2(ind).smallerfMid1_side2 = polyval(pMid1_side2(ind).smallerpMid1_side2,flatMid1_side2(ind).smallerflatMid1_side2(:,2));

%%Flatten Mid2_side1
    flatMid2_side1(ind).smallerflatMid2_side1=Mid2_side1(ind).smallerMid2_side1;
    if isempty(Mid2_side1(ind).smallerMid2_side1)==1
        continue
    end
    flatMid2_side1(ind).smallerflatMid2_side1(:,1)=repelem(flatMid2_side1(ind).smallerflatMid2_side1(1),size(flatMid2_side1(ind).smallerflatMid2_side1,1),1);
    pMid2_side1(ind).smallerpMid2_side1 = polyfit(flatMid2_side1(ind).smallerflatMid2_side1(:,2),flatMid2_side1(ind).smallerflatMid2_side1(:,3),1);
    fMid2_side1(ind).smallerfMid2_side1 = polyval(pMid2_side1(ind).smallerpMid2_side1,flatMid2_side1(ind).smallerflatMid2_side1(:,2));

%%Flatten Mid2_side2
    flatMid2_side2(ind).smallerflatMid2_side2=Mid2_side2(ind).smallerMid2_side2;
    if isempty(Mid2_side2(ind).smallerMid2_side2)==1
        continue
    end
    flatMid2_side2(ind).smallerflatMid2_side2(:,1)=repelem(flatMid2_side2(ind).smallerflatMid2_side2(1),size(flatMid2_side2(ind).smallerflatMid2_side2,1),1);
    pMid2_side2(ind).smallerpMid2_side2 = polyfit(flatMid2_side2(ind).smallerflatMid2_side2(:,2),flatMid2_side2(ind).smallerflatMid2_side2(:,3),1);
    fMid2_side2(ind).smallerfMid2_side2 = polyval(pMid2_side2(ind).smallerpMid2_side2,flatMid2_side2(ind).smallerflatMid2_side2(:,2));

%%Flatten Mid3_side1
    flatMid3_side1(ind).smallerflatMid3_side1=mid3_side1(ind).smallerMid3_side1;
    if isempty(mid3_side1(ind).smallerMid3_side1)==1
        continue
    end
    flatMid3_side1(ind).smallerflatMid3_side1(:,1)=repelem(flatMid3_side1(ind).smallerflatMid3_side1(1),size(flatMid3_side1(ind).smallerflatMid3_side1,1),1);
    pMid3_side1(ind).smallerpMid3_side1 = polyfit(flatMid3_side1(ind).smallerflatMid3_side1(:,2),flatMid3_side1(ind).smallerflatMid3_side1(:,3),1);
    fMid3_side1(ind).smallerfMid3_side1 = polyval(pMid3_side1(ind).smallerpMid3_side1,flatMid3_side1(ind).smallerflatMid3_side1(:,2));

%%Flatten Mid3_side2

```

```

flatMid3_side2(ind).smallerflatMid3_side2=mid3_side2(ind).smallermid3_side2;
if isempty(mid3_side2(ind).smallermid3_side2)==1
    continue
end
flatMid3_side2(ind).smallerflatMid3_side2(:,1)=repelem(flatMid3_side2(ind).smallerflatMid3_side2(1),size(flatMid3_side2(ind).smallerflatMid3_side2,1),1);
pMid3_side2(ind).smallerpMid3_side2 = polyfit(flatMid3_side2(ind).smallerflatMid3_side2(:,2),flatMid3_side2(ind).smallerflatMid3_side2(:,3),1);
fMid3_side2(ind).smallerfMid3_side2 = polyval(pMid3_side2(ind).smallerpMid3_side2,flatMid3_side2(ind).smallerflatMid3_side2(:,2));

%%Flatten Down_side1
flatDown_side1(ind).smallerflatDown_side1=Down_side1(ind).smallerDown_side1;
if isempty(Down_side1(ind).smallerDown_side1)==1
    continue
end
flatDown_side1(ind).smallerflatDown_side1(:,1)=repelem(flatDown_side1(ind).smallerflatDown_side1(1),size(flatDown_side1(ind).smallerflatDown_side1,1),1);
pflatDown_side1(ind).smallerpflatDown_side1 = polyfit(flatDown_side1(ind).smallerflatDown_side1(:,2),flatDown_side1(ind).smallerflatDown_side1(:,3),1);
fflatDown_side1(ind).smallerfflatDown_side1 = polyval(pflatDown_side1(ind).smallerpflatDown_side1,flatDown_side1(ind).smallerflatDown_side1(:,2));

%%Flatten Down_side2
flatDown_side2(ind).smallerflatDown_side2=Down_side2(ind).smallerDown_side2;
if isempty(Down_side2(ind).smallerDown_side2)==1
    continue
end
flatDown_side2(ind).smallerflatDown_side2(:,1)=repelem(flatDown_side2(ind).smallerflatDown_side2(1),size(flatDown_side2(ind).smallerflatDown_side2,1),1);
pflatDown_side2(ind).smallerpflatDown_side2 = polyfit(flatDown_side2(ind).smallerflatDown_side2(:,2),flatDown_side2(ind).smallerflatDown_side2(:,3),1);
fflatDown_side2(ind).smallerfflatDown_side2 = polyval(pflatDown_side2(ind).smallerpflatDown_side2,flatDown_side2(ind).smallerflatDown_side2(:,2));

end

%%find intersection
for ind = 1:numel(xRngs)-1

Y1_intersect(ind).smallerY1_intersect=fzero @(x) polyval(pup1(ind).smallerpup1-pMid1_side2(ind).smallerpMid1_side2,x),3);
Z1_intersect(ind).smallerZ1_intersect=polyval(pup1(ind).smallerpup1,Y1_intersect(ind).smallerY1_intersect);

Y2_intersect(ind).smallerY2_intersect=fzero @(x) polyval(pMid1_side2(ind).smallerpMid1_side2-pMid2_side2(ind).smallerpMid2_side2,x),3);
Z2_intersect(ind).smallerZ2_intersect=polyval(pMid1_side2(ind).smallerpMid1_side2,Y2_intersect(ind).smallerY2_intersect);

Y3_intersect(ind).smallerY3_intersect=fzero @(x) polyval(pMid2_side2(ind).smallerpMid2_side2-pMid3_side2(ind).smallerpMid3_side2,x),3);
Z3_intersect(ind).smallerZ3_intersect=polyval(pMid2_side2(ind).smallerpMid2_side2,Y3_intersect(ind).smallerY3_intersect);

Y4_intersect(ind).smallerY4_intersect=fzero @(x) polyval(pMid3_side2(ind).smallerpMid3_side2-pflatDown_side2(ind).smallerpflatDown_side2,x),3);
Z4_intersect(ind).smallerZ4_intersect=polyval(pMid3_side2(ind).smallerpMid3_side2,Y4_intersect(ind).smallerY4_intersect);

Y5_intersect(ind).smallerY5_intersect=fzero @(x) polyval(pflatDown_side2(ind).smallerpflatDown_side2-pflatDown_side1(ind).smallerpflatDown_side1,x),3);
Z5_intersect(ind).smallerZ5_intersect=polyval(pflatDown_side2(ind).smallerpflatDown_side2,Y5_intersect(ind).smallerY5_intersect);

Y6_intersect(ind).smallerY6_intersect=fzero @(x) polyval(pflatDown_side1(ind).smallerpflatDown_side1-pMid3_side1(ind).smallerpMid3_side1,x),3);
Z6_intersect(ind).smallerZ6_intersect=polyval(pflatDown_side1(ind).smallerpflatDown_side1,Y6_intersect(ind).smallerY6_intersect);

Y7_intersect(ind).smallerY7_intersect=fzero @(x) polyval(pMid3_side1(ind).smallerpMid3_side1-pMid2_side1(ind).smallerpMid2_side1,x),3);
Z7_intersect(ind).smallerZ7_intersect=polyval(pMid3_side1(ind).smallerpMid3_side1,Y7_intersect(ind).smallerY7_intersect);

Y8_intersect(ind).smallerY8_intersect=fzero @(x) polyval(pMid2_side1(ind).smallerpMid2_side1-pMid1_side1(ind).smallerpMid1_side1,x),3);
Z8_intersect(ind).smallerZ8_intersect=polyval(pMid2_side1(ind).smallerpMid2_side1,Y8_intersect(ind).smallerY8_intersect);

Y9_intersect(ind).smallerY9_intersect=fzero @(x) polyval(pMid1_side1(ind).smallerpMid1_side1-pup1(ind).smallerpup1,x),3);
Z9_intersect(ind).smallerZ9_intersect=polyval(pMid1_side1(ind).smallerpMid1_side1,Y9_intersect(ind).smallerY9_intersect);

%%New matrix of points with same flattened x-value from right
intersect_xyz1(ind).smallerintersect_xyz1=[flatup1(ind).smallerflatup1(1,1),Y1_intersect(ind).smallerY1_intersect,Z1_intersect(ind).smallerZ1_intersect];
intersect_xyz2(ind).smallerintersect_xyz2=[flatup1(ind).smallerflatup1(1,1),Y2_intersect(ind).smallerY2_intersect,Z2_intersect(ind).smallerZ2_intersect];

```

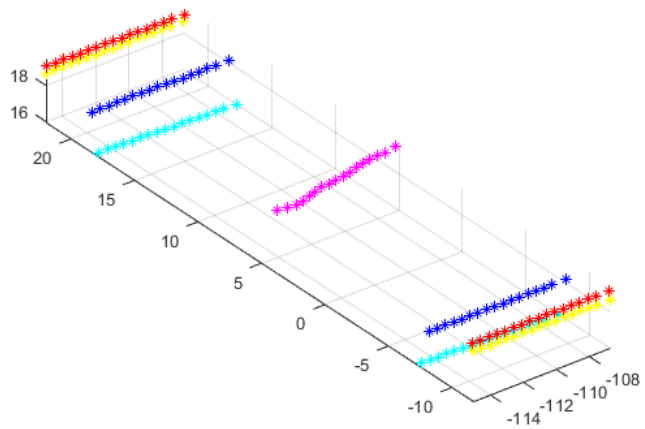
```

intersect_xyz3(ind).smallerintersect_xyz3=[flatup1(ind).smallerflatup1(1,1),Y3_intersect(ind).smallerY3_intersect,Z3_intersect(ind).smallerZ3_intersect];
intersect_xyz4(ind).smallerintersect_xyz4=[flatup1(ind).smallerflatup1(1,1),Y4_intersect(ind).smallerY4_intersect,Z4_intersect(ind).smallerZ4_intersect];
intersect_xyz5(ind).smallerintersect_xyz5=[flatup1(ind).smallerflatup1(1,1),Y5_intersect(ind).smallerY5_intersect,Z5_intersect(ind).smallerZ5_intersect];
intersect_xyz6(ind).smallerintersect_xyz6=[flatup1(ind).smallerflatup1(1,1),Y6_intersect(ind).smallerY6_intersect,Z6_intersect(ind).smallerZ6_intersect];
intersect_xyz7(ind).smallerintersect_xyz7=[flatup1(ind).smallerflatup1(1,1),Y7_intersect(ind).smallerY7_intersect,Z7_intersect(ind).smallerZ7_intersect];
intersect_xyz8(ind).smallerintersect_xyz8=[flatup1(ind).smallerflatup1(1,1),Y8_intersect(ind).smallerY8_intersect,Z8_intersect(ind).smallerZ8_intersect];
intersect_xyz9(ind).smallerintersect_xyz9=[flatup1(ind).smallerflatup1(1,1),Y9_intersect(ind).smallerY9_intersect,Z9_intersect(ind).smallerZ9_intersect];
end
%%Figure of the unrotated intersection
for ind = 1:numel(xRngs)-1

figure(7)
scatter3(intersect_xyz1(ind).smallerintersect_xyz1(:,1),intersect_xyz1(ind).smallerintersect_xyz1(:,2),intersect_xyz1(ind).smallerintersect_xyz1(:,3), '*c')
hold on
scatter3(intersect_xyz2(ind).smallerintersect_xyz2(:,1),intersect_xyz2(ind).smallerintersect_xyz2(:,2),intersect_xyz2(ind).smallerintersect_xyz2(:,3), '*b')
hold on
scatter3(intersect_xyz3(ind).smallerintersect_xyz3(:,1),intersect_xyz3(ind).smallerintersect_xyz3(:,2),intersect_xyz3(ind).smallerintersect_xyz3(:,3), '*y')
hold on
scatter3(intersect_xyz4(ind).smallerintersect_xyz4(:,1),intersect_xyz4(ind).smallerintersect_xyz4(:,2),intersect_xyz4(ind).smallerintersect_xyz4(:,3), '*r')
hold on
scatter3(intersect_xyz5(ind).smallerintersect_xyz5(:,1),intersect_xyz5(ind).smallerintersect_xyz5(:,2),intersect_xyz5(ind).smallerintersect_xyz5(:,3), '*m')
hold on
scatter3(intersect_xyz6(ind).smallerintersect_xyz6(:,1),intersect_xyz6(ind).smallerintersect_xyz6(:,2),intersect_xyz6(ind).smallerintersect_xyz6(:,3), '*r')
hold on
scatter3(intersect_xyz7(ind).smallerintersect_xyz7(:,1),intersect_xyz7(ind).smallerintersect_xyz7(:,2),intersect_xyz7(ind).smallerintersect_xyz7(:,3), '*y')
hold on
scatter3(intersect_xyz8(ind).smallerintersect_xyz8(:,1),intersect_xyz8(ind).smallerintersect_xyz8(:,2),intersect_xyz8(ind).smallerintersect_xyz8(:,3), '*b')
hold on
scatter3(intersect_xyz9(ind).smallerintersect_xyz9(:,1),intersect_xyz9(ind).smallerintersect_xyz9(:,2),intersect_xyz9(ind).smallerintersect_xyz9(:,3), '*c')
axis equal

end

```



**Rotate intersection point**

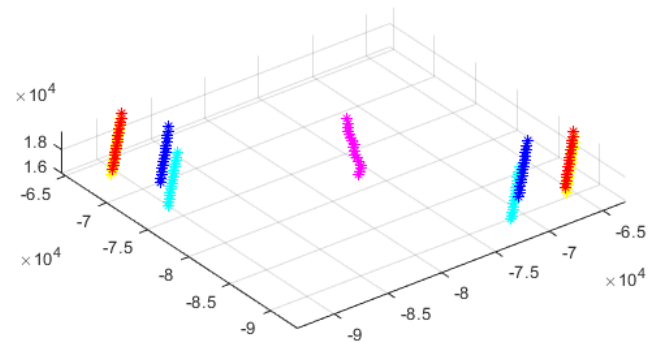
```

%%Rotate pier back to original orientation and convert from m to mm
for ind = 1:numel(xRngs)-1
rotintersect_xyz1(ind).smallerrotintersect_xyz1=intersect_xyz1(ind).smallerintersect_xyz1*inv(Rz)*10^3;
rotintersect_xyz2(ind).smallerrotintersect_xyz2=intersect_xyz2(ind).smallerintersect_xyz2*inv(Rz)*10^3;
rotintersect_xyz3(ind).smallerrotintersect_xyz3=intersect_xyz3(ind).smallerintersect_xyz3*inv(Rz)*10^3;
rotintersect_xyz4(ind).smallerrotintersect_xyz4=intersect_xyz4(ind).smallerintersect_xyz4*inv(Rz)*10^3;
rotintersect_xyz5(ind).smallerrotintersect_xyz5=intersect_xyz5(ind).smallerintersect_xyz5*inv(Rz)*10^3;
rotintersect_xyz6(ind).smallerrotintersect_xyz6=intersect_xyz6(ind).smallerintersect_xyz6*inv(Rz)*10^3;
rotintersect_xyz7(ind).smallerrotintersect_xyz7=intersect_xyz7(ind).smallerintersect_xyz7*inv(Rz)*10^3;
rotintersect_xyz8(ind).smallerrotintersect_xyz8=intersect_xyz8(ind).smallerintersect_xyz8*inv(Rz)*10^3;
rotintersect_xyz9(ind).smallerrotintersect_xyz9=intersect_xyz9(ind).smallerintersect_xyz9*inv(Rz)*10^3;
end

%%Figure of the intersection points, rotated back to original positions
for ind = 1:numel(xRngs)-1

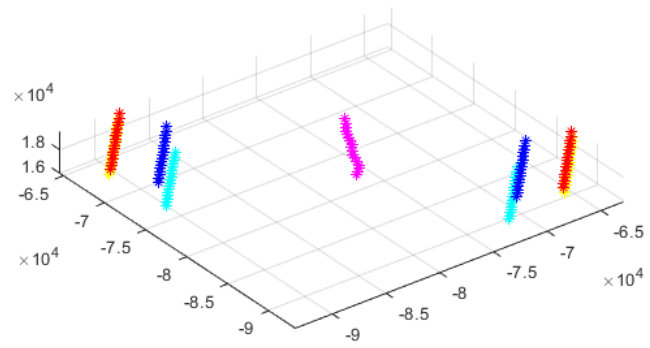
figure(8)
scatter3(rotintersect_xyz1(ind).smallerrotintersect_xyz1(:,1),rotintersect_xyz1(ind).smallerrotintersect_xyz1(:,2),rotintersect_xyz1(ind).smallerrotintersect_xyz1(:,3), 'c')
hold on
scatter3(rotintersect_xyz2(ind).smallerrotintersect_xyz2(:,1),rotintersect_xyz2(ind).smallerrotintersect_xyz2(:,2),rotintersect_xyz2(ind).smallerrotintersect_xyz2(:,3), 'b')
hold on
scatter3(rotintersect_xyz3(ind).smallerrotintersect_xyz3(:,1),rotintersect_xyz3(ind).smallerrotintersect_xyz3(:,2),rotintersect_xyz3(ind).smallerrotintersect_xyz3(:,3), 'y')
hold on
scatter3(rotintersect_xyz4(ind).smallerrotintersect_xyz4(:,1),rotintersect_xyz4(ind).smallerrotintersect_xyz4(:,2),rotintersect_xyz4(ind).smallerrotintersect_xyz4(:,3), 'r')
hold on
scatter3(rotintersect_xyz5(ind).smallerrotintersect_xyz5(:,1),rotintersect_xyz5(ind).smallerrotintersect_xyz5(:,2),rotintersect_xyz5(ind).smallerrotintersect_xyz5(:,3), 'm')
hold on
scatter3(rotintersect_xyz6(ind).smallerrotintersect_xyz6(:,1),rotintersect_xyz6(ind).smallerrotintersect_xyz6(:,2),rotintersect_xyz6(ind).smallerrotintersect_xyz6(:,3), 'r')
hold on
scatter3(rotintersect_xyz7(ind).smallerrotintersect_xyz7(:,1),rotintersect_xyz7(ind).smallerrotintersect_xyz7(:,2),rotintersect_xyz7(ind).smallerrotintersect_xyz7(:,3), 'y')
hold on
scatter3(rotintersect_xyz8(ind).smallerrotintersect_xyz8(:,1),rotintersect_xyz8(ind).smallerrotintersect_xyz8(:,2),rotintersect_xyz8(ind).smallerrotintersect_xyz8(:,3), 'b')
hold on
scatter3(rotintersect_xyz9(ind).smallerrotintersect_xyz9(:,1),rotintersect_xyz9(ind).smallerrotintersect_xyz9(:,2),rotintersect_xyz9(ind).smallerrotintersect_xyz9(:,3), 'c')
axis equal
end

```



#### Write Excel file

```
writetable(struct2table(rotintersect_xyz1), 'Intersect_box_2.xlsx', 'sheet', 1);  
writetable(struct2table(rotintersect_xyz2), 'Intersect_box_2.xlsx', 'sheet', 2);  
writetable(struct2table(rotintersect_xyz3), 'Intersect_box_2.xlsx', 'sheet', 3);  
writetable(struct2table(rotintersect_xyz4), 'Intersect_box_2.xlsx', 'sheet', 4);  
writetable(struct2table(rotintersect_xyz5), 'Intersect_box_2.xlsx', 'sheet', 5);  
writetable(struct2table(rotintersect_xyz6), 'Intersect_box_2.xlsx', 'sheet', 6);  
writetable(struct2table(rotintersect_xyz7), 'Intersect_box_2.xlsx', 'sheet', 7);  
writetable(struct2table(rotintersect_xyz8), 'Intersect_box_2.xlsx', 'sheet', 8);  
writetable(struct2table(rotintersect_xyz9), 'Intersect_box_2.xlsx', 'sheet', 9);  
winopen 'Intersect_box_2.xlsx'
```



---

Published with MATLAB® R2019b



**A4    Box Girder 10-17  
Algorithm**

## Contents

---

- [Box girder 10-17](#)
- [Rotate](#)
- [Slice](#)
- [Vertical cut to find outer parts](#)
- [Horizontal cut](#)
- [Vertical cut](#)
- [Intersection points](#)
- [Rotate intersection point](#)
- [write an Excel file](#)

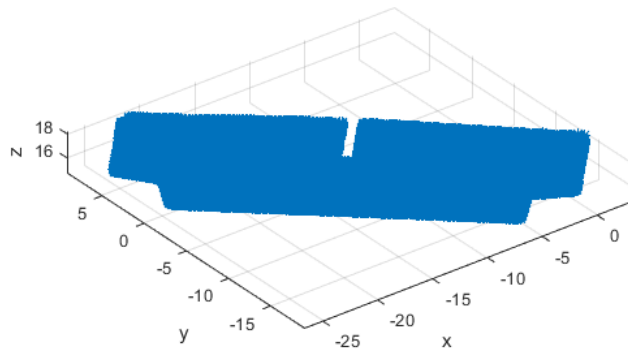
## Box girder 10-17

---

```
clear all
close all
clc

A=dlmread('Box_17.txt');    %%Reading the txt-file of the point cloud
a=A(:,[1,2,3]);           %%Matrix of the geometrical data
x=a(:,1);
y=a(:,2);
z=a(:,3);

figure(1)
scatter3(x,y,z,'*')
hold on
xlabel('x')
ylabel('y')
zlabel('z')
grid on
axis('equal')
```



## Rotate

```

%find angle to rotate

[maxX, indexOfMaxX] = max(x);
yAtMaxX = y(indexOfMaxX);
[minY, indexOfMinY] = min(y);
xAtMinY = x(indexOfMinY);

u=[maxX-xAtMinY, yAtMaxX- minY];
v=[1,0];

slope=u(:,2)/u(:,1);
theta=atan(slope);           %%Angle

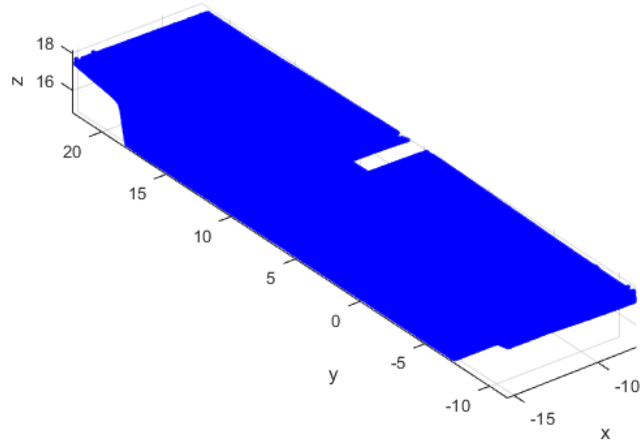
R=[cos(theta)  -sin(theta)  0;  %%Rotation matrix
   sin(theta)  cos(theta)  0;
   0  0  1];

%%Rotate the whole pier
rotM=a*R;                    %%rotated matrix

%%Rotated matrix
x1=rotM(:,1);
y1=rotM(:,2);
z1=rotM(:,3);

figure(2)
scatter3(x1,y1,z1,'.b')
hold on
xlabel('x')
ylabel('y')
zlabel('z')
grid on
axis('equal')

```



### Slice

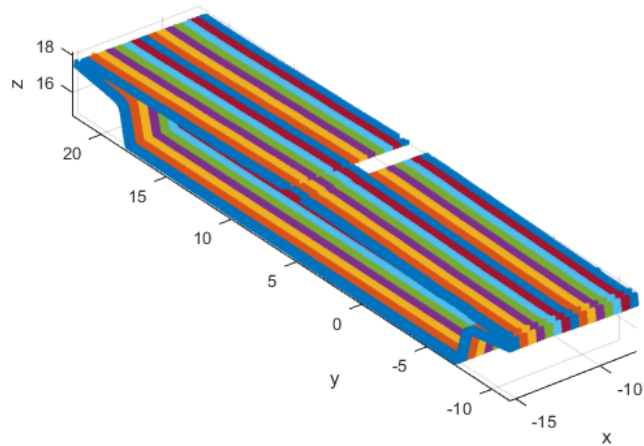
```

%%Dividing the rotated Box girder into main segments
xRngs=[min(rotM(:,1)):0.5: max(rotM(:,1))];
for ind = 1: numel(xRngs)-1

    rows = rotM(:,1)>xRngs(ind) & rotM(:,1)<xRngs(ind+1); %check to see which rows of b are between
    %xRng(ind) and xRng(ind+1)
    segM(ind).SmallerM = rotM(rows,:); %segM is the segmented matrix of the rotated pier

    figure(3)
    scatter3(segM(ind).SmallerM(:,1),segM(ind).SmallerM(:,2),segM(ind).SmallerM(:,3),'.') %Scatter the rotated pier
    hold on
    xlabel('x')
    ylabel('y')
    zlabel('z')
    grid on
    axis('equal')
end

```



#### Vertical cut to find outer parts

```

%%splitting the box vertically, finding the two outer sides
for ind = 1:numel(xRngs)-1
MaxY_Box(ind)=max(segM(ind).SmallerM(:,2));
MinY_Box(ind)=min(segM(ind).SmallerM(:,2));
e=0.01;
kVer(ind) = floor((MaxY_Box(ind)-MinY_Box(ind))/e);
for i = 1:kVer(ind)
mVer(ind).SmallermVer(i,1) = MinY_Box(ind)+e*(i-1);
A2Ver(ind).SmallerA2Ver= segM(ind).SmallerM((segM(ind).SmallerM(:,2)>=mVer(ind).SmallermVer(i,1) & segM(ind).SmallerM(:,2)<=mVer(ind).SmallermVer(i,1)+e),:); %logical indexing
ggVer_Box(ind).SmallerggVer_Box{i,1}=A2Ver(ind).SmallerA2Ver; %Counting the points in each microsegment
if isempty(A2Ver(ind).SmallerA2Ver)=1
continue
end
end
ggVer_Box(ind).SmallerggVer_Box=cell2mat(ggVer_Box(ind).SmallerggVer_Box);
ggVer_mid3_side1(ind).SmallerggVer_mid3_side1=ggVer_Box(ind).SmallerggVer_Box((ggVer_Box(ind).SmallerggVer_Box(:,2)<=MinY_Box(ind)+0.2),:);
ggVer_mid3_side2(ind).SmallerggVer_mid3_side2=ggVer_Box(ind).SmallerggVer_Box((ggVer_Box(ind).SmallerggVer_Box(:,2)>=MaxY_Box(ind)-0.2),:);
end

%%removing unregular part
for ind = 1:numel(xRngs)-1
%side 1
MaxZ_Box_side1(ind)= max(ggVer_mid3_side1(ind).SmallerggVer_mid3_side1(:,3));
MinZ_Box_side1(ind)= min(ggVer_mid3_side1(ind).SmallerggVer_mid3_side1(:,3));

mid3_side1(ind).smallermid3_side1= ggVer_mid3_side1(ind).SmallerggVer_mid3_side1((ggVer_mid3_side1(ind).SmallerggVer_mid3_side1(:,3)<=MaxZ_Box_side1(ind)-0.1),:);
mid3_side1(ind).smallermid3_side1= ggVer_mid3_side1(ind).SmallerggVer_mid3_side1((ggVer_mid3_side1(ind).SmallerggVer_mid3_side1(:,3)>=MinZ_Box_side1(ind)+0.2),:);

%%side 2, missing point cloud
end

%%Excluding the outer part from the point cloud
for ind = 1:numel(xRngs)-1

```

```

[ia, ~] = ismember(segM(ind).SmallerM, ggVer_mid3_side1(ind).SmallerggVer_mid3_side1, 'rows');
segM(ind).SmallerM(ia, :) = [];
[ia, ib] = ismember(segM(ind).SmallerM, ggVer_mid3_side2(ind).SmallerggVer_mid3_side2, 'rows');
segM(ind).SmallerM(ia, :) = [];
segM(ind).SmallerM = segM(ind).SmallerM(segM(ind).SmallerM(:,2) >= min(ggVer_mid3_side1(ind).SmallerggVer_mid3_side1(:,2)), :));
segM(ind).SmallerM = segM(ind).SmallerM(segM(ind).SmallerM(:,2) <= max(ggVer_mid3_side2(ind).SmallerggVer_mid3_side2(:,2)), :));

end

%%Figure of Boxgirder without the outer part
for ind = 1:numel(xRngs)-1
figure(4)
scatter3(segM(ind).SmallerM(:,1),segM(ind).SmallerM(:,2),segM(ind).SmallerM(:,3),'.r')
hold on
% scatter3(ggVer_mid3_side1(9).SmallerggVer_mid3_side1(:,1),ggVer_mid3_side1(9).SmallerggVer_mid3_side1(:,2),ggVer_mid3_side1(9).SmallerggVer_mid3_side1(:,3),'.b')
% hold on
% scatter3(ggVer_mid3_side2(9).SmallerggVer_mid3_side2(:,1),ggVer_mid3_side2(9).SmallerggVer_mid3_side2(:,2),ggVer_mid3_side2(9).SmallerggVer_mid3_side2(:,3),'.g')
axis equal
end

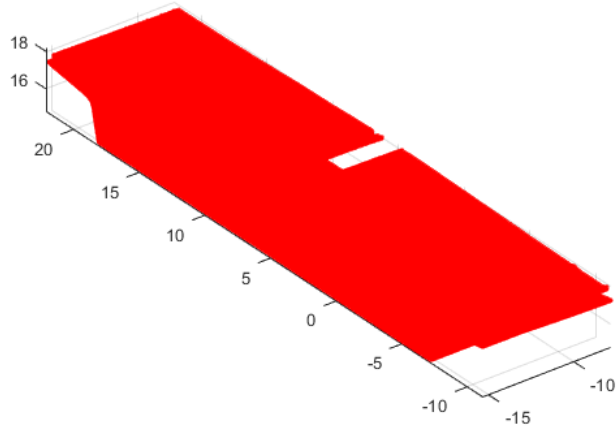
%%Find mean, max and min values
for ind = 1:numel(xRngs)-1
maximumZ(ind)= max(segM(ind).SmallerM(:,3));
minimumZ(ind)=min(segM(ind).SmallerM(:,3));

maximumY(ind)= max(segM(ind).SmallerM(:,2));
minimumY(ind)=min(segM(ind).SmallerM(:,2));

columnMean(ind).SmallercolumnMean = mean(segM(ind).SmallerM,1);           %%Find the mean x-values of each segment

meanx(ind).Smallermeanx=columnMean(ind).SmallercolumnMean(1,1);
meanY(ind).SmallermeanY=columnMean(ind).SmallercolumnMean(1,2);
meanZ(ind).SmallermeanZ=columnMean(ind).SmallercolumnMean(1,3);
end

```



### Horizontal cut

```

%%Dividing the segments into microsegments in the horizontal direction,
%%finding changes in amount of points
for ind = 1:numel(xRngs)-1

e1=0.05;
kHor(ind) = floor((maximumZ(ind)-minimumZ(ind))/e1);           %How many times its divided
for i = 1:kHor(ind)
    mHor(ind).SmallermHor(i,1) = minimumZ(ind)+e1*(i-1);        %For which values its divided for each segment
    A2Hor(ind).SmallerA2Hor= segM(ind).SmallerM((segM(ind).SmallerM(:,3))>=mHor(ind).SmallermHor(i,1) & segM(ind).SmallerM(:,3)<=mHor(ind).SmallermHor(i,1)+e1,:); %logical indexing
    ggHor(ind).SmallerggHor{i,1}=A2Hor(ind).SmallerA2Hor;        %Counting the points in each microsegment
    if isempty(A2Hor(ind).SmallerA2Hor)=1
        continue
    end
end
for i=1:kHor(ind)-1
ChangeHor(ind).smallerChangeHor{i,:}=size(ggHor(ind).SmallerggHor{i,1})/size(ggHor(ind).SmallerggHor{i+1,1});

ChangeHor(ind).smallerChangeHor{i,:}=ChangeHor(ind).smallerChangeHor{i,1}(:,1);
ChangeHor1(ind).smallerChangeHor1=cell2mat(ChangeHor(ind).smallerChangeHor);
end
end
%%finding the positions to cut
for ind = 1:numel(xRngs)-1

[m_1(ind).smallerm_1,idx_1(ind).smalleridx_1] = max(ChangeHor1(ind).smallerChangeHor1(1:floor(kHor(ind)/2),:));
ChangeHor1(ind).smallerChangeHor1(idx_1(ind).smalleridx_1) = NaN ;
idx_2(ind).smalleridx_2= find(diff(ChangeHor1(ind).smallerChangeHor1,2)==0,1); %%The removed outer part always makes this position to be several repeting 1

if idx_1(ind).smalleridx_1 > idx_2(ind).smalleridx_2
    [m_3(ind).smallerm_3,idx_3(ind).smalleridx_3] = min(ChangeHor1(ind).smallerChangeHor1(idx_2(ind).smalleridx_2:idx_1(ind).smalleridx_1)) ;
else
    [m_3(ind).smallerm_3,idx_3(ind).smalleridx_3] = min(ChangeHor1(ind).smallerChangeHor1(idx_1(ind).smalleridx_1:idx_2(ind).smalleridx_2)) ;
end
m_4(ind).smallerm_4 = min(ChangeHor1(ind).smallerChangeHor1(idx_2(ind).smalleridx_2:end)) ;

```

```

idx_4(ind).smalleridx_4 = find(ChangeHor1(ind).smallerChangeHor1==m_4(ind).smallerm_4);

CutPosition(ind).smallerCutPosition=sort([idx_1(ind).smalleridx_1;idx_2(ind).smalleridx_2;idx_3(ind).smalleridx_3;idx_4(ind).smalleridx_4]);

end
%%splitting the parts
%%Removing the ends from both sides of the mid 1,cos horizontal cut
for ind = 1:numel(xRngs)-1

up(ind).smallerup=ggHor(ind).SmallerggHor(1:CutPosition(ind).smallerCutPosition(1,:),:);
up1(ind).smallerup1=cell2mat(up(ind).smallerup);

mid1(ind).smallermid1=ggHor(ind).SmallerggHor(CutPosition(ind).smallerCutPosition(1,:)+1:CutPosition(ind).smallerCutPosition(2,:),:);
mid1(ind).smallermid1=mid1(ind).smallermid1(5:end-4);
mid11(ind).smallermid11=cell2mat(mid1(ind).smallermid1);

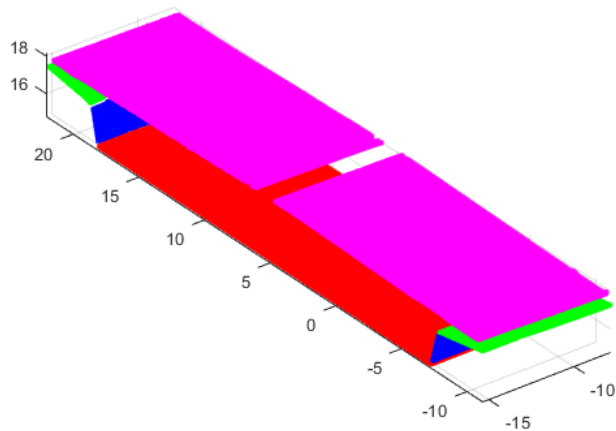
mid2(ind).smallermid2=ggHor(ind).SmallerggHor(CutPosition(ind).smallerCutPosition(2,:)+1:CutPosition(ind).smallerCutPosition(3,:),:);
mid22(ind).smallermid22=cell2mat(mid2(ind).smallermid2);

%%mid3 is already splitted as the outer parts

down(ind).smallerdown=ggHor(ind).SmallerggHor(CutPosition(ind).smallerCutPosition(4,:)+1:end,:);
down1(ind).smallerdown1=cell2mat(down(ind).smallerdown);

figure(5)
scatter3(up1(ind).smallerup1(:,1),up1(ind).smallerup1(:,2),up1(ind).smallerup1(:,3),'.r')
hold on
scatter3(mid11(ind).smallermid11(:,1),mid11(ind).smallermid11(:,2),mid11(ind).smallermid11(:,3),'.b')
hold on
scatter3(mid22(ind).smallermid22(:,1),mid22(ind).smallermid22(:,2),mid22(ind).smallermid22(:,3),'.g')
hold on
scatter3(down1(ind).smallerdown1(:,1),down1(ind).smallerdown1(:,2),down1(ind).smallerdown1(:,3),'.m')
axis equal
end

```





## Vertical cut

```

%%Splitting the sides
for ind = 1:numel(xRngs)-1
    %%Up is has only one side

    Mid1_side1(ind).smallerMid1_side1=find(mid11(ind).smallermid11(:,2)< (meanY(ind).SmallermeanY));
    Mid1_side2(ind).smallerMid1_side2=find(mid11(ind).smallermid11(:,2)> (meanY(ind).SmallermeanY));
    Mid1_side1(ind).smallerMid1_side1=mid11(ind).smallermid11(Mid1_side1(ind).smallerMid1_side1,:);
    Mid1_side2(ind).smallerMid1_side2=mid11(ind).smallermid11(Mid1_side2(ind).smallerMid1_side2,:);

    Mid2_side1(ind).smallerMid2_side1=find(mid22(ind).smallermid22(:,2)< (meanY(ind).SmallermeanY));
    Mid2_side2(ind).smallerMid2_side2=find(mid22(ind).smallermid22(:,2)> (meanY(ind).SmallermeanY));
    Mid2_side1(ind).smallerMid2_side1=mid22(ind).smallermid22(Mid2_side1(ind).smallerMid2_side1,:);
    Mid2_side2(ind).smallerMid2_side2=mid22(ind).smallermid22(Mid2_side2(ind).smallerMid2_side2,:);

    Down_side1(ind).smallerDown_side1=find(down1(ind).smallerdown1(:,2)< (meanY(ind).SmallermeanY));
    Down_side2(ind).smallerDown_side2=find(down1(ind).smallerdown1(:,2)> (meanY(ind).SmallermeanY));
    Down_side1(ind).smallerDown_side1=down1(ind).smallerdown1(Down_side1(ind).smallerDown_side1,:);
    Down_side2(ind).smallerDown_side2=down1(ind).smallerdown1(Down_side2(ind).smallerDown_side2,:);

end

%%splitting Mid2_side1 Vertically, removing the ends from both sides
for ind = 1:numel(xRngs)-1
    MaxY_Mid2_side1(ind)=max(Mid2_side1(ind).smallerMid2_side1(:,2));
    MinY_Mid2_side1(ind)=min(Mid2_side1(ind).smallerMid2_side1(:,2));
    e=0.1;
    kVer(ind) = floor((MaxY_Mid2_side1(ind)-MinY_Mid2_side1(ind))/e);
    for i = 1:kVer(ind)
        mVer(ind).SmallermVer(i,1) = MinY_Mid2_side1(ind)+e*(i-1);
        A2Ver(ind).SmallerA2Ver= Mid2_side1(ind).smallerMid2_side1(Mid2_side1(ind).smallerMid2_side1(:,2)>=mVer(ind).SmallermVer(i,1) & Mid2_side1(ind).smallerMid2_side1(:,2)<=mVer(ind).SmallermVer(i,1)+e,:); %logical indexing
        ggVer_mid2_side1(ind).SmallerggVer_mid2_side1{i,1}=A2Ver(ind).SmallerA2Ver; %Counting the points in each microsegment
        if isempty(A2Ver(ind).SmallerA2Ver)==1
            continue
        end
    end
    ggVer_mid2_side1(ind).SmallerggVer_mid2_side1=ggVer_mid2_side1(ind).SmallerggVer_mid2_side1(2:end-2);
    ggVer_mid2_side1(ind).SmallerggVer_mid2_side1=cell2mat(ggVer_mid2_side1(ind).SmallerggVer_mid2_side1);
end

%%splitting Mid2_side2 Vertically, removing the ends from both sides
for ind = 1:numel(xRngs)-1
    MaxY_Mid2_side2(ind)=max(Mid2_side2(ind).smallerMid2_side2(:,2));
    MinY_Mid2_side2(ind)=min(Mid2_side2(ind).smallerMid2_side2(:,2));
    e=0.1;
    kVer(ind) = floor((MaxY_Mid2_side2(ind)-MinY_Mid2_side2(ind))/e);
    for i = 1:kVer(ind)
        mVer(ind).SmallermVer(i,1) = MinY_Mid2_side2(ind)+e*(i-1);
        A2Ver(ind).SmallerA2Ver= Mid2_side2(ind).smallerMid2_side2(Mid2_side2(ind).smallerMid2_side2(:,2)>=mVer(ind).SmallermVer(i,1) & Mid2_side2(ind).smallerMid2_side2(:,2)<=mVer(ind).SmallermVer(i,1)+e,:); %logical indexing
        ggVer_mid2_side2(ind).SmallerggVer_mid2_side2{i,1}=A2Ver(ind).SmallerA2Ver; %Counting the points in each microsegment
        if isempty(A2Ver(ind).SmallerA2Ver)==1
            continue
        end
    end
    ggVer_mid2_side2(ind).SmallerggVer_mid2_side2=ggVer_mid2_side2(ind).SmallerggVer_mid2_side2(2:end-2);
    ggVer_mid2_side2(ind).SmallerggVer_mid2_side2=cell2mat(ggVer_mid2_side2(ind).SmallerggVer_mid2_side2);
end

%%splitting Up vertically, removing the ends from both sides
for ind = 1:numel(xRngs)-1
    MaxY_Up(ind)=max(up1(ind).smallerup1(:,2));
    MinY_Up(ind)=min(up1(ind).smallerup1(:,2));
    e=0.1;
    kVer(ind) = floor((MaxY_Up(ind)-MinY_Up(ind))/e);

```

```

for i = 1:kVer(ind)
    mVer(ind).SmallerVer(i,1) = MinY_Up(ind)+e*(i-1);
    A2Ver(ind).SmallerA2Ver= up1(ind).smallerup1((up1(ind).smallerup1(:,2)>=mVer(ind).SmallerVer(i,1) & up1(ind).smallerup1(:,2)<=mVer(ind).SmallerVer(i,1)+e),:); %Logical indexing
    ggVer_Up(ind).SmallerggVer_Up{i,1}=A2Ver(ind).SmallerA2Ver; %Counting the points in each microsegment
    if isempty(A2Ver(ind).SmallerA2Ver)==1
        continue
    end
end
ggVer_Up(ind).SmallerggVer_Up=ggVer_Up(ind).SmallerggVer_Up(10:end-10);
ggVer_Up(ind).SmallerggVer_Up=cell2mat(ggVer_Up(ind).SmallerggVer_Up);
end

%%splitting Down_side1 vertically, removing the ends from both sides
for ind = 1:numel(xRngs)-1
    MaxY_Down_side1(ind)=max(Down_side1(ind).smallerDown_side1(:,2));
    MinY_Down_side1(ind)=min(Down_side1(ind).smallerDown_side1(:,2));
    e=0.1;
    kVer(ind) = floor((MaxY_Down_side1(ind)-MinY_Down_side1(ind))/e);
    for i = 1:kVer(ind)
        mVer(ind).SmallerVer(i,1) = MinY_Down_side1(ind)+e*(i-1);
        A2Ver(ind).SmallerA2Ver= Down_side1(ind).smallerDown_side1((Down_side1(ind).smallerDown_side1(:,2)>=mVer(ind).SmallerVer(i,1) & Down_side1(ind).smallerDown_side1(:,2)<=mVer(ind).SmallerVer(i,1)+e),:); %Logical indexing
        ggVer_Down_side1(ind).SmallerggVer_Down_side1{i,1}=A2Ver(ind).SmallerA2Ver; %Counting the points in each microsegment
        if isempty(A2Ver(ind).SmallerA2Ver)==1
            continue
        end
    end
    ggVer_Down_side1(ind).SmallerggVer_Down_side1=ggVer_Down_side1(ind).SmallerggVer_Down_side1(20:end-20);
    ggVer_Down_side1(ind).SmallerggVer_Down_side1=cell2mat(ggVer_Down_side1(ind).SmallerggVer_Down_side1);
end

%%splitting Down_side2 vertically, removing the ends from both sides
for ind = 1:numel(xRngs)-1
    MaxY_Down_side2(ind)=max(Down_side2(ind).smallerDown_side2(:,2));
    MinY_Down_side2(ind)=min(Down_side2(ind).smallerDown_side2(:,2));
    e=0.1;
    kVer(ind) = floor((MaxY_Down_side2(ind)-MinY_Down_side2(ind))/e);
    for i = 1:kVer(ind)
        mVer(ind).SmallerVer(i,1) = MinY_Down_side2(ind)+e*(i-1);
        A2Ver(ind).SmallerA2Ver= Down_side2(ind).smallerDown_side2((Down_side2(ind).smallerDown_side2(:,2)>=mVer(ind).SmallerVer(i,1) & Down_side2(ind).smallerDown_side2(:,2)<=mVer(ind).SmallerVer(i,1)+e),:); %Logical indexing
        ggVer_Down_side2(ind).SmallerggVer_Down_side2{i,1}=A2Ver(ind).SmallerA2Ver; %Counting the points in each microsegment
        if isempty(A2Ver(ind).SmallerA2Ver)==1
            continue
        end
    end
    ggVer_Down_side2(ind).SmallerggVer_Down_side2=ggVer_Down_side2(ind).SmallerggVer_Down_side2(20:end-20);
    ggVer_Down_side2(ind).SmallerggVer_Down_side2=cell2mat(ggVer_Down_side2(ind).SmallerggVer_Down_side2);
end

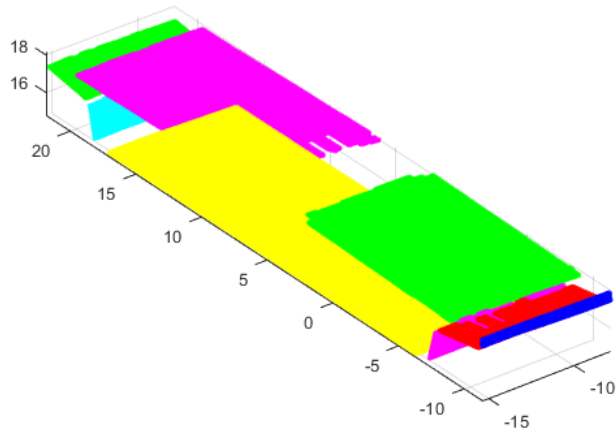
%%Figure of the divided box girder
for ind = 1:numel(xRngs)-1
    figure(6)
    scatter3(ggVer_Up(ind).SmallerggVer_Up(:,1),ggVer_Up(ind).SmallerggVer_Up(:,2),ggVer_Up(ind).SmallerggVer_Up(:,3),'.y')
    hold on
    scatter3(Mid1_side1(ind).smallerMid1_side1(:,1),Mid1_side1(ind).smallerMid1_side1(:,2),Mid1_side1(ind).smallerMid1_side1(:,3),'.m')
    hold on
    scatter3(Mid1_side2(ind).smallerMid1_side2(:,1),Mid1_side2(ind).smallerMid1_side2(:,2),Mid1_side2(ind).smallerMid1_side2(:,3),'.c')
    hold on
    scatter3(ggVer_mid2_side1(ind).SmallerggVer_mid2_side1(:,1),ggVer_mid2_side1(ind).SmallerggVer_mid2_side1(:,2),ggVer_mid2_side1(ind).SmallerggVer_mid2_side1(:,3),'.n')
    hold on
    scatter3(ggVer_mid2_side2(ind).SmallerggVer_mid2_side2(:,1),ggVer_mid2_side2(ind).SmallerggVer_mid2_side2(:,2),ggVer_mid2_side2(ind).SmallerggVer_mid2_side2(:,3),'.g')
    hold on
    scatter3(mid3_side1(ind).smallermid3_side1(:,1),mid3_side1(ind).smallermid3_side1(:,2),mid3_side1(ind).smallermid3_side1(:,3),'.b')
    hold on
    % scatter3(mid3_side2(ind).smallermid3_side2(:,1),mid3_side2(ind).smallermid3_side2(:,2),mid3_side2(ind).smallermid3_side2(:,3),'.k')

```

```

% hold on
scatter3( ggVer_Down_side1(ind).SmallerggVer_Down_side1(:,1), ggVer_Down_side1(ind).SmallerggVer_Down_side1(:,2), ggVer_Down_side1(ind).SmallerggVer_Down_side1(:,3), '.g')
hold on
scatter3(ggVer_Down_side2(ind).SmallerggVer_Down_side2(:,1),ggVer_Down_side2(ind).SmallerggVer_Down_side2(:,2),ggVer_Down_side2(ind).SmallerggVer_Down_side2(:,3), '.m')
axis equal
end

```



### Intersection points

```

%%Assigning one X-value to all the rows
for ind = 1:numel(xRngs)-1

    %%Flatten Up
    flatup1(ind).smallerflatup1=up1(ind).smallerup1;
    if isempty(up1(ind).smallerup1)==1
        continue
    end
    flatup1(ind).smallerflatup1(:,1)=repelem(flatup1(ind).smallerflatup1(1),size(flatup1(ind).smallerflatup1,1),1);
    pup1(ind).smallerpup1 = polyfit(flatup1(ind).smallerflatup1(:,2),flatup1(ind).smallerflatup1(:,3),1);
    fup1(ind).smallerfup1 = polyval(pup1(ind).smallerpup1,flatup1(ind).smallerflatup1(:,2));

    %%Flatten Mid1_side1
    flatMid1_side1(ind).smallerflatMid1_side1=Mid1_side1(ind).smallerMid1_side1;
    if isempty(Mid1_side1(ind).smallerMid1_side1)==1
        continue
    end
    flatMid1_side1(ind).smallerflatMid1_side1(:,1)=repelem(flatMid1_side1(ind).smallerflatMid1_side1(1),size(flatMid1_side1(ind).smallerflatMid1_side1,1),1);
    pMid1_side1(ind).smallerpMid1_side1 = polyfit(flatMid1_side1(ind).smallerflatMid1_side1(:,2),flatMid1_side1(ind).smallerflatMid1_side1(:,3),1);
    fMid1_side1(ind).smallerfMid1_side1 = polyval(pMid1_side1(ind).smallerpMid1_side1,flatMid1_side1(ind).smallerflatMid1_side1(:,2));

    %%Flatten Mid1_side2
    flatMid1_side2(ind).smallerflatMid1_side2=Mid1_side2(ind).smallerMid1_side2;
    if isempty(Mid1_side2(ind).smallerMid1_side2)==1

```

```

    continue
end
flatMid1_side2(ind).smallerflatMid1_side2(:,1)=repelem(flatMid1_side2(ind).smallerflatMid1_side2(1),size(flatMid1_side2(ind).smallerflatMid1_side2,1),1);
pMid1_side2(ind).smallerpMid1_side2 = polyfit(flatMid1_side2(ind).smallerflatMid1_side2(:,2),flatMid1_side2(ind).smallerflatMid1_side2(:,3),1);
fMid1_side2(ind).smallerfMid1_side2 = polyval(pMid1_side2(ind).smallerpMid1_side2,flatMid1_side2(ind).smallerflatMid1_side2(:,2));

%%Flatten Mid2_side1
flatMid2_side1(ind).smallerflatMid2_side1=Mid2_side1(ind).smallerMid2_side1;
if isempty(Mid2_side1(ind).smallerMid2_side1)==1
    continue
end
flatMid2_side1(ind).smallerflatMid2_side1(:,1)=repelem(flatMid2_side1(ind).smallerflatMid2_side1(1),size(flatMid2_side1(ind).smallerflatMid2_side1,1),1);
pMid2_side1(ind).smallerpMid2_side1 = polyfit(flatMid2_side1(ind).smallerflatMid2_side1(:,2),flatMid2_side1(ind).smallerflatMid2_side1(:,3),1);
fMid2_side1(ind).smallerfMid2_side1 = polyval(pMid2_side1(ind).smallerpMid2_side1,flatMid2_side1(ind).smallerflatMid2_side1(:,2));

%%Flatten Mid2_side2
flatMid2_side2(ind).smallerflatMid2_side2=Mid2_side2(ind).smallerMid2_side2;
if isempty(Mid2_side2(ind).smallerMid2_side2)==1
    continue
end
flatMid2_side2(ind).smallerflatMid2_side2(:,1)=repelem(flatMid2_side2(ind).smallerflatMid2_side2(1),size(flatMid2_side2(ind).smallerflatMid2_side2,1),1);
pMid2_side2(ind).smallerpMid2_side2 = polyfit(flatMid2_side2(ind).smallerflatMid2_side2(:,2),flatMid2_side2(ind).smallerflatMid2_side2(:,3),1);
fMid2_side2(ind).smallerfMid2_side2 = polyval(pMid2_side2(ind).smallerpMid2_side2,flatMid2_side2(ind).smallerflatMid2_side2(:,2));

%%Flatten Mid3_side1
flatMid3_side1(ind).smallerflatMid3_side1=mid3_side1(ind).smallermid3_side1;
if isempty(mid3_side1(ind).smallermid3_side1)==1
    continue
end
flatMid3_side1(ind).smallerflatMid3_side1(:,1)=repelem(flatMid3_side1(ind).smallerflatMid3_side1(1),size(flatMid3_side1(ind).smallerflatMid3_side1,1),1);
pMid3_side1(ind).smallerpMid3_side1 = polyfit(flatMid3_side1(ind).smallerflatMid3_side1(:,2),flatMid3_side1(ind).smallerflatMid3_side1(:,3),1);
fMid3_side1(ind).smallerfMid3_side1 = polyval(pMid3_side1(ind).smallerpMid3_side1,flatMid3_side1(ind).smallerflatMid3_side1(:,2));

%%Flatten Down_side1
flatDown_side1(ind).smallerflatDown_side1=Down_side1(ind).smallerDown_side1;
if isempty(Down_side1(ind).smallerDown_side1)==1
    continue
end
flatDown_side1(ind).smallerflatDown_side1(:,1)=repelem(flatDown_side1(ind).smallerflatDown_side1(1),size(flatDown_side1(ind).smallerflatDown_side1,1),1);
pflatDown_side1(ind).smallerpflatDown_side1 = polyfit(flatDown_side1(ind).smallerflatDown_side1(:,2),flatDown_side1(ind).smallerflatDown_side1(:,3),1);
fflatDown_side1(ind).smallerfflatDown_side1 = polyval(pflatDown_side1(ind).smallerpflatDown_side1,flatDown_side1(ind).smallerflatDown_side1(:,2));

%%Flatten Down_side2
flatDown_side2(ind).smallerflatDown_side2=Down_side2(ind).smallerDown_side2;
if isempty(Down_side2(ind).smallerDown_side2)==1
    continue
end
flatDown_side2(ind).smallerflatDown_side2(:,1)=repelem(flatDown_side2(ind).smallerflatDown_side2(1),size(flatDown_side2(ind).smallerflatDown_side2,1),1);
pflatDown_side2(ind).smallerpflatDown_side2 = polyfit(flatDown_side2(ind).smallerflatDown_side2(:,2),flatDown_side2(ind).smallerflatDown_side2(:,3),1);
fflatDown_side2(ind).smallerfflatDown_side2 = polyval(pflatDown_side2(ind).smallerpflatDown_side2,flatDown_side2(ind).smallerflatDown_side2(:,2));

end

%%find intersection
for ind = 1:numel(xRngs)-1

Y1_intersect(ind).smallerY1_intersect=fzero(@(x) polyval(pup1(ind).smallerpup1-pMid1_side2(ind).smallerpMid1_side2,x),3);
Z1_intersect(ind).smallerZ1_intersect=polyval(pup1(ind).smallerpup1,Y1_intersect(ind).smallerY1_intersect);

```

```

Y2_intersect(ind).smallerY2_intersect=fzero(@(x) polyval(pMid1_side2(ind).smallerpMid1_side2-pMid2_side2(ind).smallerpMid2_side2,x),3);
Z2_intersect(ind).smallerZ2_intersect=polyval(pMid1_side2(ind).smallerpMid1_side2,Y2_intersect(ind).smallerY2_intersect);

%Intersection point 3 becomes the endpoint_3, because of the missing part

%Intersection point 4 is found in the next for-loop

Y5_intersect(ind).smallerY5_intersect=fzero(@(x) polyval(pflatDown_side2(ind).smallerpflatDown_side2-pflatDown_side1(ind).smallerpflatDown_side1,x),3);
Z5_intersect(ind).smallerZ5_intersect=polyval(pflatDown_side2(ind).smallerpflatDown_side2,Y5_intersect(ind).smallerY5_intersect);

Y6_intersect(ind).smallerY6_intersect=fzero(@(x) polyval(pflatDown_side1(ind).smallerpflatDown_side1-pMid3_side1(ind).smallerpMid3_side1,x),3);
Z6_intersect(ind).smallerZ6_intersect=polyval(pflatDown_side1(ind).smallerpflatDown_side1,Y6_intersect(ind).smallerY6_intersect);

Y7_intersect(ind).smallerY7_intersect=fzero(@(x) polyval(pMid3_side1(ind).smallerpMid3_side1-pMid2_side1(ind).smallerpMid2_side1,x),3);
Z7_intersect(ind).smallerZ7_intersect=polyval(pMid3_side1(ind).smallerpMid3_side1,Y7_intersect(ind).smallerY7_intersect);

Y8_intersect(ind).smallerY8_intersect=fzero(@(x) polyval(pMid2_side1(ind).smallerpMid2_side1-pMid1_side1(ind).smallerpMid1_side1,x),3);
Z8_intersect(ind).smallerZ8_intersect=polyval(pMid2_side1(ind).smallerpMid2_side1,Y8_intersect(ind).smallerY8_intersect);

Y9_intersect(ind).smallerY9_intersect=fzero(@(x) polyval(pMid1_side1(ind).smallerpMid1_side1-pup1(ind).smallerpup1,x),3);
Z9_intersect(ind).smallerZ9_intersect=polyval(pMid1_side1(ind).smallerpMid1_side1,Y9_intersect(ind).smallerY9_intersect);

%%New matrix of points with same flattened x-value from right
intersect_xyz1(ind).smallerintersect_xyz1=[flatup1(ind).smallerflatup1(1,1),Y1_intersect(ind).smallerY1_intersect,Z1_intersect(ind).smallerZ1_intersect];
intersect_xyz2(ind).smallerintersect_xyz2=[flatup1(ind).smallerflatup1(1,1),Y2_intersect(ind).smallerY2_intersect,Z2_intersect(ind).smallerZ2_intersect];
% intersect_xyz3(ind).smallerintersect_xyz3=[flatup1(ind).smallerflatup1(1,1),Y3_intersect(ind).smallerY3_intersect,Z3_intersect(ind).smallerZ3_intersect];
% intersect_xyz4(ind).smallerintersect_xyz4=[flatup1(ind).smallerflatup1(1,1),Y4_intersect(ind).smallerY4_intersect,Z4_intersect(ind).smallerZ4_intersect];
intersect_xyz5(ind).smallerintersect_xyz5=[flatup1(ind).smallerflatup1(1,1),Y5_intersect(ind).smallerY5_intersect,Z5_intersect(ind).smallerZ5_intersect];
intersect_xyz6(ind).smallerintersect_xyz6=[flatup1(ind).smallerflatup1(1,1),Y6_intersect(ind).smallerY6_intersect,Z6_intersect(ind).smallerZ6_intersect];
intersect_xyz7(ind).smallerintersect_xyz7=[flatup1(ind).smallerflatup1(1,1),Y7_intersect(ind).smallerY7_intersect,Z7_intersect(ind).smallerZ7_intersect];
intersect_xyz8(ind).smallerintersect_xyz8=[flatup1(ind).smallerflatup1(1,1),Y8_intersect(ind).smallerY8_intersect,Z8_intersect(ind).smallerZ8_intersect];
intersect_xyz9(ind).smallerintersect_xyz9=[flatup1(ind).smallerflatup1(1,1),Y9_intersect(ind).smallerY9_intersect,Z9_intersect(ind).smallerZ9_intersect];
end

%Find endpoint where the missing points are, mid3_side2
for ind = 1:numel(xRngs)-1
[~,idx]=max(Mid2_side2(ind).smallerMid2_side2(:,2));
endpoint_3(ind).smallerendpoint_3=Mid2_side2(ind).smallerMid2_side2(idx,:);

[~,idx]=max(Down_side2(ind).smallerDown_side2(:,2));
endpoint_4(ind).smallerendpoint_4=Down_side2(ind).smallerDown_side2(idx,:);
end

%Find intersection of missing part, intersection point 4
for ind = 1:numel(xRngs)-1
[~,idx]=min(Down_side2(ind).smallerDown_side2(:,2));
startpoint_4(ind).smallerstartpoint_4=Down_side2(ind).smallerDown_side2(idx,:);

Y4_intersect(ind).smallerY4_intersect=endpoint_3(ind).smallerendpoint_3(:,2);
coefficients(ind).smallercoefficients=polyfit([startpoint_4(ind).smallerstartpoint_4(:,2), endpoint_4(ind).smallerendpoint_4(:,2)], [startpoint_4(ind).smallerstartpoint_4(:,3), endpoint_4(ind).smallerendpoint_4(:,3)], 1);
Z4_intersect(ind).smallerZ4_intersect=coefficients(ind).smallercoefficients(:,1)*Y4_intersect(ind).smallerY4_intersect+coefficients(ind).smallercoefficients(:,2);

intersect_xyz3(ind).smallerintersect_xyz3=[flatup1(ind).smallerflatup1(1,1),endpoint_3(ind).smallerendpoint_3(:,2),endpoint_3(ind).smallerendpoint_3(:,3)];
intersect_xyz4(ind).smallerintersect_xyz4=[flatup1(ind).smallerflatup1(1,1),Y4_intersect(ind).smallerY4_intersect,Z4_intersect(ind).smallerZ4_intersect];

end
%%Figure of the unrotated intersection
for ind = 1:numel(xRngs)-1

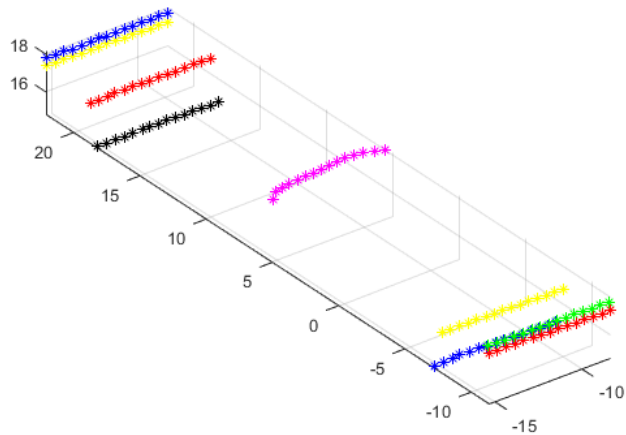
figure(7)
scatter3(intersect_xyz1(ind).smallerintersect_xyz1(:,1),intersect_xyz1(ind).smallerintersect_xyz1(:,2),intersect_xyz1(ind).smallerintersect_xyz1(:,3), '*k')
hold on

```

```

scatter3(intersect_xyz2(ind).smallerintersect_xyz2(:,1),intersect_xyz2(ind).smallerintersect_xyz2(:,2),intersect_xyz2(ind).smallerintersect_xyz2(:,3), '*r')
hold on
scatter3(intersect_xyz3(ind).smallerintersect_xyz3(:,1),intersect_xyz3(ind).smallerintersect_xyz3(:,2),intersect_xyz3(ind).smallerintersect_xyz3(:,3), '*y')
hold on
scatter3(intersect_xyz4(ind).smallerintersect_xyz4(:,1),intersect_xyz4(ind).smallerintersect_xyz4(:,2),intersect_xyz4(ind).smallerintersect_xyz4(:,3), '*b')
hold on
scatter3(intersect_xyz5(ind).smallerintersect_xyz5(:,1),intersect_xyz5(ind).smallerintersect_xyz5(:,2),intersect_xyz5(ind).smallerintersect_xyz5(:,3), '*m')
hold on
scatter3(intersect_xyz6(ind).smallerintersect_xyz6(:,1),intersect_xyz6(ind).smallerintersect_xyz6(:,2),intersect_xyz6(ind).smallerintersect_xyz6(:,3), '*g')
hold on
scatter3(intersect_xyz7(ind).smallerintersect_xyz7(:,1),intersect_xyz7(ind).smallerintersect_xyz7(:,2),intersect_xyz7(ind).smallerintersect_xyz7(:,3), '*r')
hold on
scatter3(intersect_xyz8(ind).smallerintersect_xyz8(:,1),intersect_xyz8(ind).smallerintersect_xyz8(:,2),intersect_xyz8(ind).smallerintersect_xyz8(:,3), '*y')
hold on
scatter3(intersect_xyz9(ind).smallerintersect_xyz9(:,1),intersect_xyz9(ind).smallerintersect_xyz9(:,2),intersect_xyz9(ind).smallerintersect_xyz9(:,3), '*b')
axis equal
end

```



#### Rotate intersection point

```

%%Rotate pier back to original orientation and convert from m to mm
for ind = 1:numel(xRngs)-1
rotintersect_xyz1(ind).smallerrotintersect_xyz1=intersect_xyz1(ind).smallerintersect_xyz1*inv(R)*10^3;
rotintersect_xyz2(ind).smallerrotintersect_xyz2=intersect_xyz2(ind).smallerintersect_xyz2*inv(R)*10^3;
rotintersect_xyz3(ind).smallerrotintersect_xyz3=intersect_xyz3(ind).smallerintersect_xyz3*inv(R)*10^3;
rotintersect_xyz4(ind).smallerrotintersect_xyz4=intersect_xyz4(ind).smallerintersect_xyz4*inv(R)*10^3;
rotintersect_xyz5(ind).smallerrotintersect_xyz5=intersect_xyz5(ind).smallerintersect_xyz5*inv(R)*10^3;
rotintersect_xyz6(ind).smallerrotintersect_xyz6=intersect_xyz6(ind).smallerintersect_xyz6*inv(R)*10^3;
rotintersect_xyz7(ind).smallerrotintersect_xyz7=intersect_xyz7(ind).smallerintersect_xyz7*inv(R)*10^3;
rotintersect_xyz8(ind).smallerrotintersect_xyz8=intersect_xyz8(ind).smallerintersect_xyz8*inv(R)*10^3;
rotintersect_xyz9(ind).smallerrotintersect_xyz9=intersect_xyz9(ind).smallerintersect_xyz9*inv(R)*10^3;
end

```

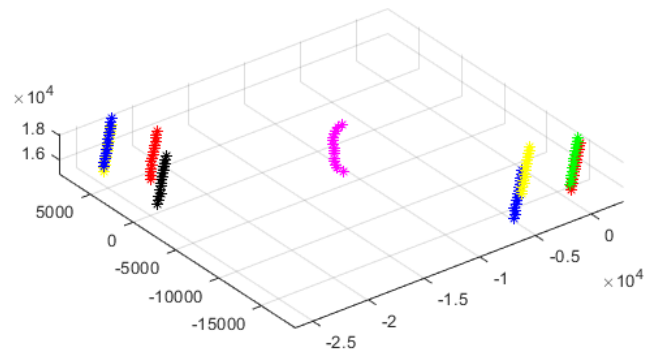
```

%%Figure of the intersection points, rotated back to original positions
for ind = 1:numel(xRngs)-1

figure(8)
scatter3(rotintersect_xyz1(ind).smallerrotintersect_xyz1(:,1),rotintersect_xyz1(ind).smallerrotintersect_xyz1(:,2),rotintersect_xyz1(ind).smallerrotintersect_xyz1(:,3), '*k')
hold on
scatter3(rotintersect_xyz2(ind).smallerrotintersect_xyz2(:,1),rotintersect_xyz2(ind).smallerrotintersect_xyz2(:,2),rotintersect_xyz2(ind).smallerrotintersect_xyz2(:,3), '*r')
hold on
scatter3(rotintersect_xyz3(ind).smallerrotintersect_xyz3(:,1),rotintersect_xyz3(ind).smallerrotintersect_xyz3(:,2),rotintersect_xyz3(ind).smallerrotintersect_xyz3(:,3), '*y')
hold on
scatter3(rotintersect_xyz4(ind).smallerrotintersect_xyz4(:,1),rotintersect_xyz4(ind).smallerrotintersect_xyz4(:,2),rotintersect_xyz4(ind).smallerrotintersect_xyz4(:,3), '*b')
hold on
scatter3(rotintersect_xyz5(ind).smallerrotintersect_xyz5(:,1),rotintersect_xyz5(ind).smallerrotintersect_xyz5(:,2),rotintersect_xyz5(ind).smallerrotintersect_xyz5(:,3), '*m')
hold on
scatter3(rotintersect_xyz6(ind).smallerrotintersect_xyz6(:,1),rotintersect_xyz6(ind).smallerrotintersect_xyz6(:,2),rotintersect_xyz6(ind).smallerrotintersect_xyz6(:,3), '*g')
hold on
scatter3(rotintersect_xyz7(ind).smallerrotintersect_xyz7(:,1),rotintersect_xyz7(ind).smallerrotintersect_xyz7(:,2),rotintersect_xyz7(ind).smallerrotintersect_xyz7(:,3), '*r')
hold on
scatter3(rotintersect_xyz8(ind).smallerrotintersect_xyz8(:,1),rotintersect_xyz8(ind).smallerrotintersect_xyz8(:,2),rotintersect_xyz8(ind).smallerrotintersect_xyz8(:,3), '*y')
hold on
scatter3(rotintersect_xyz9(ind).smallerrotintersect_xyz9(:,1),rotintersect_xyz9(ind).smallerrotintersect_xyz9(:,2),rotintersect_xyz9(ind).smallerrotintersect_xyz9(:,3), '*b')
axis equal

end

```



#### write an Excel file

```

writetable(struct2table(rotintersect_xyz1), 'Intersect_box_17.xlsx', 'sheet', 1);
writetable(struct2table(rotintersect_xyz2), 'Intersect_box_17.xlsx', 'sheet', 2);
writetable(struct2table(rotintersect_xyz3), 'Intersect_box_17.xlsx', 'sheet', 3);
writetable(struct2table(rotintersect_xyz4), 'Intersect_box_17.xlsx', 'sheet', 4);
writetable(struct2table(rotintersect_xyz5), 'Intersect_box_17.xlsx', 'sheet', 5);
writetable(struct2table(rotintersect_xyz6), 'Intersect_box_17.xlsx', 'sheet', 6);
writetable(struct2table(rotintersect_xyz7), 'Intersect_box_17.xlsx', 'sheet', 7);

```

```
writetable(struct2table(rotintersect_xyz8), 'Intersect_box_17.xlsx', 'sheet', 8);  
writetable(struct2table(rotintersect_xyz9), 'Intersect_box_17.xlsx', 'sheet', 9);  
winopen 'Intersect_box_17.xlsx'
```

Warning: Added specified worksheet.  
Warning: Added specified worksheet.  
Warning: Added specified worksheet.  
Warning: Added specified worksheet.  
Warning: Added specified worksheet.  
Warning: Added specified worksheet.  
Warning: Added specified worksheet.  
Warning: Added specified worksheet.

---

Published with MATLAB® R2019b



## **A5    Box Girder 18 Algorithm**

## Contents

---

- [Box girder 18](#)
- [Rotate](#)
- [Deleting bad part](#)
- [Slice](#)
- [Vertical cut to find outer parts](#)
- [Horizontal cut](#)
- [Vertical cut](#)
- [Intersection points](#)
- [Rotate intersection point](#)
- [write an Excel file](#)

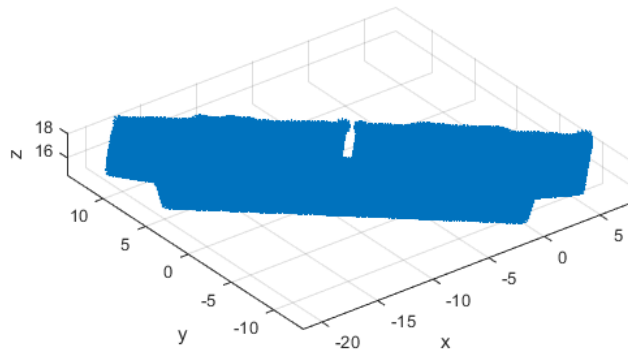
## Box girder 18

---

```
clear all
close all
clc

A=dlmread('Box_18.txt');    %%Reading the txt-file of the point cloud
a=A(:,[1,2,3]);           %%Matrix of the geometrical data
x=a(:,1);
y=a(:,2);
z=a(:,3);

figure(1)
scatter3(x,y,z,'*')
hold on
xlabel('x')
ylabel('y')
zlabel('z')
grid on
axis('equal')
```



### Rotate

```

%find angle to rotate

[maxX, indexOfMaxX] = max(x);
yAtMaxX = y(indexOfMaxX);
[minY, indexOfMinY] = min(y);
xAtMinY = x(indexOfMinY);

u=[maxX-xAtMinY, yAtMaxX- minY];
v=[1,0];

slope=u(:,2)/u(:,1);
theta=atan(slope);           %%Angle

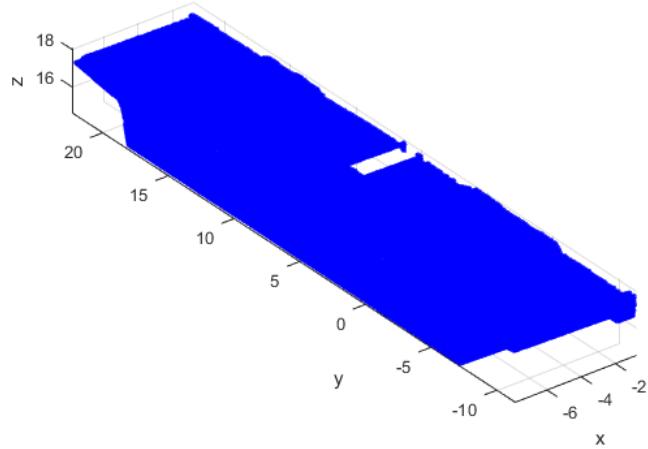
R=[cos(theta)  -sin(theta)  0;  %%Rotation matrix
   sin(theta)  cos(theta)  0;
   0  0  1];

%%Rotate the whole pier
rotM=a*R;                    %%rotated matrix

%%Rotated matrix
x1=rotM(:,1);
y1=rotM(:,2);
z1=rotM(:,3);

figure(2)
scatter3(x1,y1,z1,'.b')
hold on
xlabel('x')
ylabel('y')
zlabel('z')
grid on
axis('equal')

```



### Deleting bad part

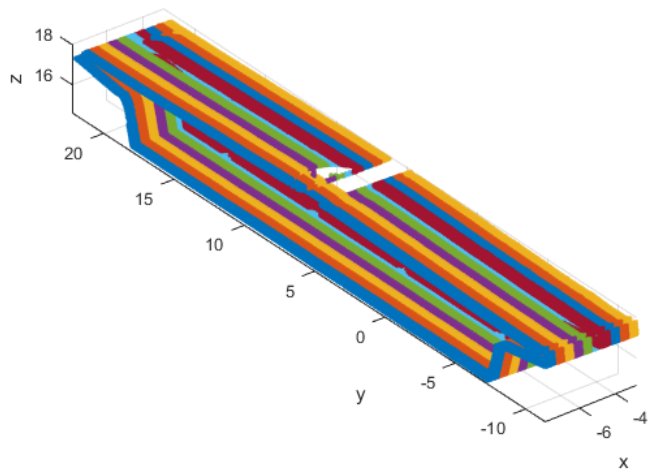
```
rowsToDelete = rotM(:,1) > max(rotM(:,1))-2;
rotM(rowsToDelete,:) = [];
```

### Slice

```
%%Dividing the rotated Box girder into main segments
xRngs=[min(rotM(:,1)):0.5: max(rotM(:,1))];
for ind = 1:numel(xRngs)-1

    rows = rotM(:,1)>=xRngs(ind) & rotM(:,1)<xRngs(ind+1); %check to see which rows of b are between
    %xRng(ind) and xRng(ind+1)
    segM(ind).SmallerM = rotM(rows,:); %segM is the segmented matrix of the rotated pier

    figure(3)
    scatter3(segM(ind).SmallerM(:,1),segM(ind).SmallerM(:,2),segM(ind).SmallerM(:,3),'.') %Scatter the rotated pier
    hold on
    xlabel('x')
    ylabel('y')
    zlabel('z')
    grid on
    axis('equal')
end
```



#### Vertical cut to find outer parts

```

%%splitting the box vertically, finding the two outer sides
for ind = 1:numel(xRngs)-1
MaxY_Box(ind)=max(segM(ind).SmallerM(:,2));
MinY_Box(ind)=min(segM(ind).SmallerM(:,2));
e=0.01;
kVer(ind) = floor((MaxY_Box(ind)-MinY_Box(ind))/e);
for i = 1:kVer(ind)
mVer(ind).SmallerMVer(i,1) = MinY_Box(ind)+e*(i-1);
A2Ver(ind).SmallerA2Ver= segM(ind).SmallerM((segM(ind).SmallerM(:,2)>=mVer(ind).SmallerMVer(i,1) & segM(ind).SmallerM(:,2)<=mVer(ind).SmallerMVer(i,1)+e),:); %logical indexing
ggVer_Box(ind).SmallerggVer_Box{i,1}=A2Ver(ind).SmallerA2Ver; %Counting the points in each microsegment
if isempty(A2Ver(ind).SmallerA2Ver)=1
continue
end
end
ggVer_Box(ind).SmallerggVer_Box=cell2mat(ggVer_Box(ind).SmallerggVer_Box);
ggVer_mid3_side1(ind).SmallerggVer_mid3_side1=ggVer_Box(ind).SmallerggVer_Box((ggVer_Box(ind).SmallerggVer_Box(:,2)<=MinY_Box(ind)+0.2),:);
ggVer_mid3_side2(ind).SmallerggVer_mid3_side2=ggVer_Box(ind).SmallerggVer_Box((ggVer_Box(ind).SmallerggVer_Box(:,2)>=MaxY_Box(ind)-0.2),:);
end

%%removing unregular part
for ind = 1:numel(xRngs)-1
%side 1
MaxZ_Box_side1(ind)= max(ggVer_mid3_side1(ind).SmallerggVer_mid3_side1(:,3));
MinZ_Box_side1(ind)= min(ggVer_mid3_side1(ind).SmallerggVer_mid3_side1(:,3));

mid3_side1(ind).smallermid3_side1= ggVer_mid3_side1(ind).SmallerggVer_mid3_side1((ggVer_mid3_side1(ind).SmallerggVer_mid3_side1(:,3)<=MaxZ_Box_side1(ind)-0.1),:);
mid3_side1(ind).smallermid3_side1= ggVer_mid3_side1(ind).SmallerggVer_mid3_side1((ggVer_mid3_side1(ind).SmallerggVer_mid3_side1(:,3)>=MinZ_Box_side1(ind)+0.2),:);

%%side 2, missing point cloud
end

%%Excluding the outer part from the point cloud
for ind = 1:numel(xRngs)-1

```

```

[ia, ~] = ismember(segM(ind).SmallerM, ggVer_mid3_side1(ind).SmallerggVer_mid3_side1, 'rows');
segM(ind).SmallerM(ia, :) = [];
[ia, ib] = ismember(segM(ind).SmallerM, ggVer_mid3_side2(ind).SmallerggVer_mid3_side2, 'rows');
segM(ind).SmallerM(ia, :) = [];
segM(ind).SmallerM = segM(ind).SmallerM(segM(ind).SmallerM(:,2) >= min(ggVer_mid3_side1(ind).SmallerggVer_mid3_side1(:,2)), :));
segM(ind).SmallerM = segM(ind).SmallerM(segM(ind).SmallerM(:,2) <= max(ggVer_mid3_side2(ind).SmallerggVer_mid3_side2(:,2)), :));

end

%%Figure of Boxgirder without the outer part
for ind = 1:numel(xRngs)-1
figure(4)
scatter3(segM(ind).SmallerM(:,1),segM(ind).SmallerM(:,2),segM(ind).SmallerM(:,3),'.r')
hold on
% scatter3(ggVer_mid3_side1(9).SmallerggVer_mid3_side1(:,1),ggVer_mid3_side1(9).SmallerggVer_mid3_side1(:,2),ggVer_mid3_side1(9).SmallerggVer_mid3_side1(:,3),'.b')
% hold on
% scatter3(ggVer_mid3_side2(9).SmallerggVer_mid3_side2(:,1),ggVer_mid3_side2(9).SmallerggVer_mid3_side2(:,2),ggVer_mid3_side2(9).SmallerggVer_mid3_side2(:,3),'.g')
axis equal
end

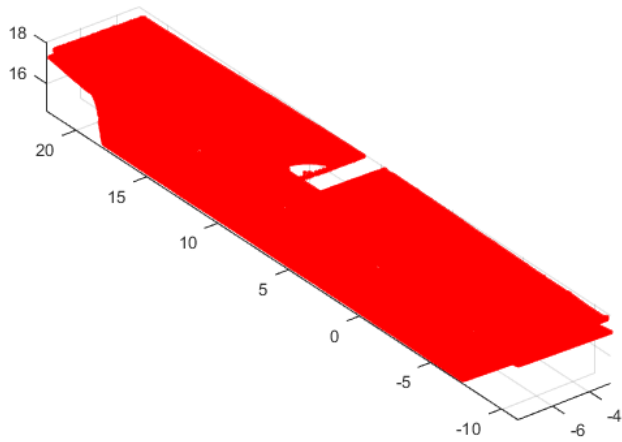
%%Find mean, max and min values
for ind = 1:numel(xRngs)-1
maximumZ(ind)= max(segM(ind).SmallerM(:,3));
minimumZ(ind)=min(segM(ind).SmallerM(:,3));

maximumY(ind)= max(segM(ind).SmallerM(:,2));
minimumY(ind)=min(segM(ind).SmallerM(:,2));

columnMean(ind).SmallercolumnMean = mean(segM(ind).SmallerM,1);           %%Find the mean x-values of each segment

meanx(ind).Smallermeanx=columnMean(ind).SmallercolumnMean(1,1);
meanY(ind).SmallermeanY=columnMean(ind).SmallercolumnMean(1,2);
meanZ(ind).SmallermeanZ=columnMean(ind).SmallercolumnMean(1,3);
end

```



### Horizontal cut

```

%%Dividing the segments into microsegments in the horizontal direction,
%%finding changes in amount of points
for ind = 1:numel(xRngs)-1

e1=0.05;
kHor(ind) = floor((maximumZ(ind)-minimumZ(ind))/e1);           %How many times its divided
for i = 1:kHor(ind)
    mHor(ind).SmallermHor(i,1) = minimumZ(ind)+e1*(i-1);       %For which values its divided for each segment
    A2Hor(ind).SmallerA2Hor= segM(ind).SmallerM((segM(ind).SmallerM(:,3)>=mHor(ind).SmallermHor(i,1) & segM(ind).SmallerM(:,3)<=mHor(ind).SmallermHor(i,1)+e1),:); %logical indexing
    ggHor(ind).SmallerggHor{i,1}=A2Hor(ind).SmallerA2Hor;       %%Counting the points in each microsegment
    if isempty(A2Hor(ind).SmallerA2Hor)=1
        continue
    end
end
for i=1:kHor(ind)-1
ChangeHor(ind).smallerChangeHor{i,:}=size(ggHor(ind).SmallerggHor{i,1})/size(ggHor(ind).SmallerggHor{i+1,1});

ChangeHor(ind).smallerChangeHor{i,:}=ChangeHor(ind).smallerChangeHor{i,1}(:,1);
ChangeHor1(ind).smallerChangeHor1=cell2mat(ChangeHor(ind).smallerChangeHor);
end
end
%%finding the positions to cut
for ind = 1:numel(xRngs)-1

[m_1(ind).smallerm_1,idx_1(ind).smalleridx_1] = max(ChangeHor1(ind).smallerChangeHor1(1:floor(kHor(ind)/2),:));
ChangeHor1(ind).smallerChangeHor1(idx_1(ind).smalleridx_1) = NaN ;
idx_2(ind).smalleridx_2= find(diff(ChangeHor1(ind).smallerChangeHor1,2)==0,1); %%The removed outer part always makes this position to be several repeting 1

if idx_1(ind).smalleridx_1 > idx_2(ind).smalleridx_2
    [m_3(ind).smallerm_3,idx_3(ind).smalleridx_3] = min(ChangeHor1(ind).smallerChangeHor1(idx_2(ind).smalleridx_2:idx_1(ind).smalleridx_1)) ;
else
    [m_3(ind).smallerm_3,idx_3(ind).smalleridx_3] = min(ChangeHor1(ind).smallerChangeHor1(idx_1(ind).smalleridx_1:idx_2(ind).smalleridx_2)) ;
end
m_4(ind).smallerm_4 = min(ChangeHor1(ind).smallerChangeHor1(idx_2(ind).smalleridx_2:end)) ;

```

```

idx_4(ind).smalleridx_4 = find(ChangeHor1(ind).smallerChangeHor1==m_4(ind).smallerm_4);

CutPosition(ind).smallerCutPosition=sort([idx_1(ind).smalleridx_1;idx_2(ind).smalleridx_2;idx_3(ind).smalleridx_3;idx_4(ind).smalleridx_4]);

end
%%splitting the parts
%%Removing the ends from both sides of the mid 1,cos horizontal cut
for ind = 1:numel(xRngs)-1

up(ind).smallerup=ggHor(ind).SmallerggHor(1:CutPosition(ind).smallerCutPosition(1,:),:);
up1(ind).smallerup1=cell2mat(up(ind).smallerup);

mid1(ind).smallermid1=ggHor(ind).SmallerggHor(CutPosition(ind).smallerCutPosition(1,:)+1:CutPosition(ind).smallerCutPosition(2,:),:);
mid1(ind).smallermid1=mid1(ind).smallermid1(5:end-4);
mid11(ind).smallermid11=cell2mat(mid1(ind).smallermid1);

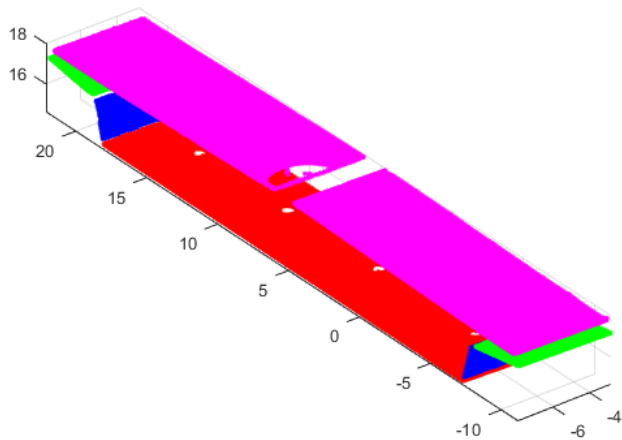
mid2(ind).smallermid2=ggHor(ind).SmallerggHor(CutPosition(ind).smallerCutPosition(2,:)+1:CutPosition(ind).smallerCutPosition(3,:),:);
mid22(ind).smallermid22=cell2mat(mid2(ind).smallermid2);

%%mid3 is already splitted as the outer parts

down(ind).smallerdown=ggHor(ind).SmallerggHor(CutPosition(ind).smallerCutPosition(4,:)+1:end,:);
down1(ind).smallerdown1=cell2mat(down(ind).smallerdown);

figure(5)
scatter3(up1(ind).smallerup1(:,1),up1(ind).smallerup1(:,2),up1(ind).smallerup1(:,3),'.r')
hold on
scatter3(mid11(ind).smallermid11(:,1),mid11(ind).smallermid11(:,2),mid11(ind).smallermid11(:,3),'.b')
hold on
scatter3(mid22(ind).smallermid22(:,1),mid22(ind).smallermid22(:,2),mid22(ind).smallermid22(:,3),'.g')
hold on
scatter3(down1(ind).smallerdown1(:,1),down1(ind).smallerdown1(:,2),down1(ind).smallerdown1(:,3),'.m')
axis equal
end

```





## Vertical cut

```

%%Splitting the sides
for ind = 1:numel(xRngs)-1
    %%Up is has only one side

    Mid1_side1(ind).smallerMid1_side1=find(mid11(ind).smallermid11(:,2)< (meanY(ind).SmallermeanY));
    Mid1_side2(ind).smallerMid1_side2=find(mid11(ind).smallermid11(:,2)> (meanY(ind).SmallermeanY));
    Mid1_side1(ind).smallerMid1_side1=mid11(ind).smallermid11(Mid1_side1(ind).smallerMid1_side1,:);
    Mid1_side2(ind).smallerMid1_side2=mid11(ind).smallermid11(Mid1_side2(ind).smallerMid1_side2,:);

    Mid2_side1(ind).smallerMid2_side1=find(mid22(ind).smallermid22(:,2)< (meanY(ind).SmallermeanY));
    Mid2_side2(ind).smallerMid2_side2=find(mid22(ind).smallermid22(:,2)> (meanY(ind).SmallermeanY));
    Mid2_side1(ind).smallerMid2_side1=mid22(ind).smallermid22(Mid2_side1(ind).smallerMid2_side1,:);
    Mid2_side2(ind).smallerMid2_side2=mid22(ind).smallermid22(Mid2_side2(ind).smallerMid2_side2,:);

    Down_side1(ind).smallerDown_side1=find(down1(ind).smallerdown1(:,2)< (meanY(ind).SmallermeanY));
    Down_side2(ind).smallerDown_side2=find(down1(ind).smallerdown1(:,2)> (meanY(ind).SmallermeanY));
    Down_side1(ind).smallerDown_side1=down1(ind).smallerdown1(Down_side1(ind).smallerDown_side1,:);
    Down_side2(ind).smallerDown_side2=down1(ind).smallerdown1(Down_side2(ind).smallerDown_side2,:);

end

%%splitting Mid2_side1 Vertically, removing the ends from both sides
for ind = 1:numel(xRngs)-1
    MaxY_Mid2_side1(ind)=max(Mid2_side1(ind).smallerMid2_side1(:,2));
    MinY_Mid2_side1(ind)=min(Mid2_side1(ind).smallerMid2_side1(:,2));
    e=0.1;
    kVer(ind) = floor((MaxY_Mid2_side1(ind)-MinY_Mid2_side1(ind))/e);
    for i = 1:kVer(ind)
        mVer(ind).SmallerVer(i,1) = MinY_Mid2_side1(ind)+e*(i-1);
        A2Ver(ind).SmallerA2Ver= Mid2_side1(ind).smallerMid2_side1(Mid2_side1(ind).smallerMid2_side1(:,2)>=mVer(ind).SmallerVer(i,1) & Mid2_side1(ind).smallerMid2_side1(:,2)<=mVer(ind).SmallerVer(i,1)+e,:); %logical indexing
        ggVer_mid2_side1(ind).SmallerggVer_mid2_side1{i,1}=A2Ver(ind).SmallerA2Ver; %Counting the points in each microsegment
        if isempty(A2Ver(ind).SmallerA2Ver)==1
            continue
        end
    end
    ggVer_mid2_side1(ind).SmallerggVer_mid2_side1=ggVer_mid2_side1(ind).SmallerggVer_mid2_side1(2:end-2);
    ggVer_mid2_side1(ind).SmallerggVer_mid2_side1=cell2mat(ggVer_mid2_side1(ind).SmallerggVer_mid2_side1);
end

%%splitting Mid2_side2 Vertically, removing the ends from both sides
for ind = 1:numel(xRngs)-1
    MaxY_Mid2_side2(ind)=max(Mid2_side2(ind).smallerMid2_side2(:,2));
    MinY_Mid2_side2(ind)=min(Mid2_side2(ind).smallerMid2_side2(:,2));
    e=0.1;
    kVer(ind) = floor((MaxY_Mid2_side2(ind)-MinY_Mid2_side2(ind))/e);
    for i = 1:kVer(ind)
        mVer(ind).SmallerVer(i,1) = MinY_Mid2_side2(ind)+e*(i-1);
        A2Ver(ind).SmallerA2Ver= Mid2_side2(ind).smallerMid2_side2(Mid2_side2(ind).smallerMid2_side2(:,2)>=mVer(ind).SmallerVer(i,1) & Mid2_side2(ind).smallerMid2_side2(:,2)<=mVer(ind).SmallerVer(i,1)+e,:); %logical indexing
        ggVer_mid2_side2(ind).SmallerggVer_mid2_side2{i,1}=A2Ver(ind).SmallerA2Ver; %Counting the points in each microsegment
        if isempty(A2Ver(ind).SmallerA2Ver)==1
            continue
        end
    end
    ggVer_mid2_side2(ind).SmallerggVer_mid2_side2=ggVer_mid2_side2(ind).SmallerggVer_mid2_side2(2:end-2);
    ggVer_mid2_side2(ind).SmallerggVer_mid2_side2=cell2mat(ggVer_mid2_side2(ind).SmallerggVer_mid2_side2);
end

%%splitting Up vertically, removing the ends from both sides
for ind = 1:numel(xRngs)-1
    MaxY_Up(ind)=max(up1(ind).smallerup1(:,2));
    MinY_Up(ind)=min(up1(ind).smallerup1(:,2));
    e=0.1;
    kVer(ind) = floor((MaxY_Up(ind)-MinY_Up(ind))/e);

```

```

for i = 1:kVer(ind)
    mVer(ind).SmallerVer(i,1) = MinY_Up(ind)+e*(i-1);
    A2Ver(ind).SmallerA2Ver= up1(ind).smallerup1((up1(ind).smallerup1(:,2)>=mVer(ind).SmallerVer(i,1) & up1(ind).smallerup1(:,2)<=mVer(ind).SmallerVer(i,1)+e),:); %Logical indexing
    ggVer_Up(ind).SmallerggVer_Up{i,1}=A2Ver(ind).SmallerA2Ver; %Counting the points in each microsegment
    if isempty(A2Ver(ind).SmallerA2Ver)==1
        continue
    end
end
ggVer_Up(ind).SmallerggVer_Up=ggVer_Up(ind).SmallerggVer_Up(10:end-10);
ggVer_Up(ind).SmallerggVer_Up=cell2mat(ggVer_Up(ind).SmallerggVer_Up);
end

%%splitting Down_side1 vertically, removing the ends from both sides
for ind = 1:numel(xRngs)-1
    MaxY_Down_side1(ind)=max(Down_side1(ind).smallerDown_side1(:,2));
    MinY_Down_side1(ind)=min(Down_side1(ind).smallerDown_side1(:,2));
    e=0.1;
    kVer(ind) = floor((MaxY_Down_side1(ind)-MinY_Down_side1(ind))/e);
    for i = 1:kVer(ind)
        mVer(ind).SmallerVer(i,1) = MinY_Down_side1(ind)+e*(i-1);
        A2Ver(ind).SmallerA2Ver= Down_side1(ind).smallerDown_side1((Down_side1(ind).smallerDown_side1(:,2)>=mVer(ind).SmallerVer(i,1) & Down_side1(ind).smallerDown_side1(:,2)<=mVer(ind).SmallerVer(i,1)+e),:); %Logical indexing
        ggVer_Down_side1(ind).SmallerggVer_Down_side1{i,1}=A2Ver(ind).SmallerA2Ver; %Counting the points in each microsegment
        if isempty(A2Ver(ind).SmallerA2Ver)==1
            continue
        end
    end
    ggVer_Down_side1(ind).SmallerggVer_Down_side1=ggVer_Down_side1(ind).SmallerggVer_Down_side1(20:end-20);
    ggVer_Down_side1(ind).SmallerggVer_Down_side1=cell2mat(ggVer_Down_side1(ind).SmallerggVer_Down_side1);
end

%%splitting Down_side2 vertically, removing the ends from both sides
for ind = 1:numel(xRngs)-1
    MaxY_Down_side2(ind)=max(Down_side2(ind).smallerDown_side2(:,2));
    MinY_Down_side2(ind)=min(Down_side2(ind).smallerDown_side2(:,2));
    e=0.1;
    kVer(ind) = floor((MaxY_Down_side2(ind)-MinY_Down_side2(ind))/e);
    for i = 1:kVer(ind)
        mVer(ind).SmallerVer(i,1) = MinY_Down_side2(ind)+e*(i-1);
        A2Ver(ind).SmallerA2Ver= Down_side2(ind).smallerDown_side2((Down_side2(ind).smallerDown_side2(:,2)>=mVer(ind).SmallerVer(i,1) & Down_side2(ind).smallerDown_side2(:,2)<=mVer(ind).SmallerVer(i,1)+e),:); %Logical indexing
        ggVer_Down_side2(ind).SmallerggVer_Down_side2{i,1}=A2Ver(ind).SmallerA2Ver; %Counting the points in each microsegment
        if isempty(A2Ver(ind).SmallerA2Ver)==1
            continue
        end
    end
    ggVer_Down_side2(ind).SmallerggVer_Down_side2=ggVer_Down_side2(ind).SmallerggVer_Down_side2(20:end-20);
    ggVer_Down_side2(ind).SmallerggVer_Down_side2=cell2mat(ggVer_Down_side2(ind).SmallerggVer_Down_side2);
end

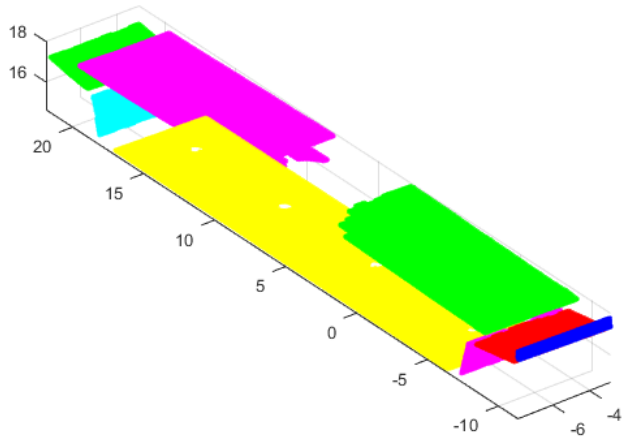
%%Figure of the divided box girder
for ind = 1:numel(xRngs)-1
    figure(6)
    scatter3(ggVer_Up(ind).SmallerggVer_Up(:,1),ggVer_Up(ind).SmallerggVer_Up(:,2),ggVer_Up(ind).SmallerggVer_Up(:,3),'.y')
    hold on
    scatter3(Mid1_side1(ind).smallerMid1_side1(:,1),Mid1_side1(ind).smallerMid1_side1(:,2),Mid1_side1(ind).smallerMid1_side1(:,3),'.m')
    hold on
    scatter3(Mid1_side2(ind).smallerMid1_side2(:,1),Mid1_side2(ind).smallerMid1_side2(:,2),Mid1_side2(ind).smallerMid1_side2(:,3),'.c')
    hold on
    scatter3(ggVer_mid2_side1(ind).SmallerggVer_mid2_side1(:,1),ggVer_mid2_side1(ind).SmallerggVer_mid2_side1(:,2),ggVer_mid2_side1(ind).SmallerggVer_mid2_side1(:,3),'.n')
    hold on
    scatter3(ggVer_mid2_side2(ind).SmallerggVer_mid2_side2(:,1),ggVer_mid2_side2(ind).SmallerggVer_mid2_side2(:,2),ggVer_mid2_side2(ind).SmallerggVer_mid2_side2(:,3),'.g')
    hold on
    scatter3(mid3_side1(ind).smallermid3_side1(:,1),mid3_side1(ind).smallermid3_side1(:,2),mid3_side1(ind).smallermid3_side1(:,3),'.b')
    hold on
    % scatter3(mid3_side2(ind).smallermid3_side2(:,1),mid3_side2(ind).smallermid3_side2(:,2),mid3_side2(ind).smallermid3_side2(:,3),'.k')

```

```

% hold on
scatter3( ggVer_Down_side1(ind).SmallerggVer_Down_side1(:,1), ggVer_Down_side1(ind).SmallerggVer_Down_side1(:,2), ggVer_Down_side1(ind).SmallerggVer_Down_side1(:,3), '.g')
hold on
scatter3(ggVer_Down_side2(ind).SmallerggVer_Down_side2(:,1),ggVer_Down_side2(ind).SmallerggVer_Down_side2(:,2),ggVer_Down_side2(ind).SmallerggVer_Down_side2(:,3), '.m')
axis equal
end

```



### Intersection points

```

%%Assigning one X-value to all the rows
for ind = 1:numel(xRngs)-1

    %%Flatten Up
    flatup1(ind).smallerflatup1=up1(ind).smallerup1;
    if isempty(up1(ind).smallerup1)==1
        continue
    end
    flatup1(ind).smallerflatup1(:,1)=repelem(flatup1(ind).smallerflatup1(1),size(flatup1(ind).smallerflatup1,1),1);
    pup1(ind).smallerpup1 = polyfit(flatup1(ind).smallerflatup1(:,2),flatup1(ind).smallerflatup1(:,3),1);
    fup1(ind).smallerfup1 = polyval(pup1(ind).smallerpup1,flatup1(ind).smallerflatup1(:,2));

    %%Flatten Mid1_side1
    flatMid1_side1(ind).smallerflatMid1_side1=Mid1_side1(ind).smallerMid1_side1;
    if isempty(Mid1_side1(ind).smallerMid1_side1)==1
        continue
    end
    flatMid1_side1(ind).smallerflatMid1_side1(:,1)=repelem(flatMid1_side1(ind).smallerflatMid1_side1(1),size(flatMid1_side1(ind).smallerflatMid1_side1,1),1);
    pMid1_side1(ind).smallerpMid1_side1 = polyfit(flatMid1_side1(ind).smallerflatMid1_side1(:,2),flatMid1_side1(ind).smallerflatMid1_side1(:,3),1);
    fMid1_side1(ind).smallerfMid1_side1 = polyval(pMid1_side1(ind).smallerpMid1_side1,flatMid1_side1(ind).smallerflatMid1_side1(:,2));

    %%Flatten Mid1_side2
    flatMid1_side2(ind).smallerflatMid1_side2=Mid1_side2(ind).smallerMid1_side2;
    if isempty(Mid1_side2(ind).smallerMid1_side2)==1

```

```

    continue
end
flatMid1_side2(ind).smallerflatMid1_side2(:,1)=repelem(flatMid1_side2(ind).smallerflatMid1_side2(1),size(flatMid1_side2(ind).smallerflatMid1_side2,1),1);
pMid1_side2(ind).smallerpMid1_side2 = polyfit(flatMid1_side2(ind).smallerflatMid1_side2(:,2),flatMid1_side2(ind).smallerflatMid1_side2(:,3),1);
fMid1_side2(ind).smallerfMid1_side2 = polyval(pMid1_side2(ind).smallerpMid1_side2,flatMid1_side2(ind).smallerflatMid1_side2(:,2));

%%Flatten Mid2_side1
flatMid2_side1(ind).smallerflatMid2_side1=Mid2_side1(ind).smallerMid2_side1;
if isempty(Mid2_side1(ind).smallerMid2_side1)==1
    continue
end
flatMid2_side1(ind).smallerflatMid2_side1(:,1)=repelem(flatMid2_side1(ind).smallerflatMid2_side1(1),size(flatMid2_side1(ind).smallerflatMid2_side1,1),1);
pMid2_side1(ind).smallerpMid2_side1 = polyfit(flatMid2_side1(ind).smallerflatMid2_side1(:,2),flatMid2_side1(ind).smallerflatMid2_side1(:,3),1);
fMid2_side1(ind).smallerfMid2_side1 = polyval(pMid2_side1(ind).smallerpMid2_side1,flatMid2_side1(ind).smallerflatMid2_side1(:,2));

%%Flatten Mid2_side2
flatMid2_side2(ind).smallerflatMid2_side2=Mid2_side2(ind).smallerMid2_side2;
if isempty(Mid2_side2(ind).smallerMid2_side2)==1
    continue
end
flatMid2_side2(ind).smallerflatMid2_side2(:,1)=repelem(flatMid2_side2(ind).smallerflatMid2_side2(1),size(flatMid2_side2(ind).smallerflatMid2_side2,1),1);
pMid2_side2(ind).smallerpMid2_side2 = polyfit(flatMid2_side2(ind).smallerflatMid2_side2(:,2),flatMid2_side2(ind).smallerflatMid2_side2(:,3),1);
fMid2_side2(ind).smallerfMid2_side2 = polyval(pMid2_side2(ind).smallerpMid2_side2,flatMid2_side2(ind).smallerflatMid2_side2(:,2));

%%Flatten Mid3_side1
flatMid3_side1(ind).smallerflatMid3_side1=mid3_side1(ind).smallerMid3_side1;
if isempty(mid3_side1(ind).smallerMid3_side1)==1
    continue
end
flatMid3_side1(ind).smallerflatMid3_side1(:,1)=repelem(flatMid3_side1(ind).smallerflatMid3_side1(1),size(flatMid3_side1(ind).smallerflatMid3_side1,1),1);
pMid3_side1(ind).smallerpMid3_side1 = polyfit(flatMid3_side1(ind).smallerflatMid3_side1(:,2),flatMid3_side1(ind).smallerflatMid3_side1(:,3),1);
fMid3_side1(ind).smallerfMid3_side1 = polyval(pMid3_side1(ind).smallerpMid3_side1,flatMid3_side1(ind).smallerflatMid3_side1(:,2));

%%Flatten Down_side1
flatDown_side1(ind).smallerflatDown_side1=Down_side1(ind).smallerDown_side1;
if isempty(Down_side1(ind).smallerDown_side1)==1
    continue
end
flatDown_side1(ind).smallerflatDown_side1(:,1)=repelem(flatDown_side1(ind).smallerflatDown_side1(1),size(flatDown_side1(ind).smallerflatDown_side1,1),1);
pflatDown_side1(ind).smallerpflatDown_side1 = polyfit(flatDown_side1(ind).smallerflatDown_side1(:,2),flatDown_side1(ind).smallerflatDown_side1(:,3),1);
fflatDown_side1(ind).smallerfflatDown_side1 = polyval(pflatDown_side1(ind).smallerpflatDown_side1,flatDown_side1(ind).smallerflatDown_side1(:,2));

%%Flatten Down_side2
flatDown_side2(ind).smallerflatDown_side2=Down_side2(ind).smallerDown_side2;
if isempty(Down_side2(ind).smallerDown_side2)==1
    continue
end
flatDown_side2(ind).smallerflatDown_side2(:,1)=repelem(flatDown_side2(ind).smallerflatDown_side2(1),size(flatDown_side2(ind).smallerflatDown_side2,1),1);
pflatDown_side2(ind).smallerpflatDown_side2 = polyfit(flatDown_side2(ind).smallerflatDown_side2(:,2),flatDown_side2(ind).smallerflatDown_side2(:,3),1);
fflatDown_side2(ind).smallerfflatDown_side2 = polyval(pflatDown_side2(ind).smallerpflatDown_side2,flatDown_side2(ind).smallerflatDown_side2(:,2));

end

%%find intersection
for ind = 1:numel(xRngs)-1

Y1_intersect(ind).smallerY1_intersect=fzero(@(x) polyval(pup1(ind).smallerpup1-pMid1_side2(ind).smallerpMid1_side2,x),3);
Z1_intersect(ind).smallerZ1_intersect=polyval(pup1(ind).smallerpup1,Y1_intersect(ind).smallerY1_intersect);

```

```

Y2_intersect(ind).smallerY2_intersect=fzero(@(x) polyval(pMid1_side2(ind).smallerpMid1_side2-pMid2_side2(ind).smallerpMid2_side2,x),3);
Z2_intersect(ind).smallerZ2_intersect=polyval(pMid1_side2(ind).smallerpMid1_side2,Y2_intersect(ind).smallerY2_intersect);

%Intersection point 3 becomes the endpoint_3, because of the missing part

%Intersection point 4 is found in the next for-loop

Y5_intersect(ind).smallerY5_intersect=fzero(@(x) polyval(pflatDown_side2(ind).smallerpflatDown_side2-pflatDown_side1(ind).smallerpflatDown_side1,x),3);
Z5_intersect(ind).smallerZ5_intersect=polyval(pflatDown_side2(ind).smallerpflatDown_side2,Y5_intersect(ind).smallerY5_intersect);

Y6_intersect(ind).smallerY6_intersect=fzero(@(x) polyval(pflatDown_side1(ind).smallerpflatDown_side1-pMid3_side1(ind).smallerpMid3_side1,x),3);
Z6_intersect(ind).smallerZ6_intersect=polyval(pflatDown_side1(ind).smallerpflatDown_side1,Y6_intersect(ind).smallerY6_intersect);

Y7_intersect(ind).smallerY7_intersect=fzero(@(x) polyval(pMid3_side1(ind).smallerpMid3_side1-pMid2_side1(ind).smallerpMid2_side1,x),3);
Z7_intersect(ind).smallerZ7_intersect=polyval(pMid3_side1(ind).smallerpMid3_side1,Y7_intersect(ind).smallerY7_intersect);

Y8_intersect(ind).smallerY8_intersect=fzero(@(x) polyval(pMid2_side1(ind).smallerpMid2_side1-pMid1_side1(ind).smallerpMid1_side1,x),3);
Z8_intersect(ind).smallerZ8_intersect=polyval(pMid2_side1(ind).smallerpMid2_side1,Y8_intersect(ind).smallerY8_intersect);

Y9_intersect(ind).smallerY9_intersect=fzero(@(x) polyval(pMid1_side1(ind).smallerpMid1_side1-pup1(ind).smallerpup1,x),3);
Z9_intersect(ind).smallerZ9_intersect=polyval(pMid1_side1(ind).smallerpMid1_side1,Y9_intersect(ind).smallerY9_intersect);

%%New matrix of points with same flattened x-value from right
intersect_xyz1(ind).smallerintersect_xyz1=[flatup1(ind).smallerflatup1(1,1),Y1_intersect(ind).smallerY1_intersect,Z1_intersect(ind).smallerZ1_intersect];
intersect_xyz2(ind).smallerintersect_xyz2=[flatup1(ind).smallerflatup1(1,1),Y2_intersect(ind).smallerY2_intersect,Z2_intersect(ind).smallerZ2_intersect];
% intersect_xyz3(ind).smallerintersect_xyz3=[flatup1(ind).smallerflatup1(1,1),Y3_intersect(ind).smallerY3_intersect,Z3_intersect(ind).smallerZ3_intersect];
% intersect_xyz4(ind).smallerintersect_xyz4=[flatup1(ind).smallerflatup1(1,1),Y4_intersect(ind).smallerY4_intersect,Z4_intersect(ind).smallerZ4_intersect];
intersect_xyz5(ind).smallerintersect_xyz5=[flatup1(ind).smallerflatup1(1,1),Y5_intersect(ind).smallerY5_intersect,Z5_intersect(ind).smallerZ5_intersect];
intersect_xyz6(ind).smallerintersect_xyz6=[flatup1(ind).smallerflatup1(1,1),Y6_intersect(ind).smallerY6_intersect,Z6_intersect(ind).smallerZ6_intersect];
intersect_xyz7(ind).smallerintersect_xyz7=[flatup1(ind).smallerflatup1(1,1),Y7_intersect(ind).smallerY7_intersect,Z7_intersect(ind).smallerZ7_intersect];
intersect_xyz8(ind).smallerintersect_xyz8=[flatup1(ind).smallerflatup1(1,1),Y8_intersect(ind).smallerY8_intersect,Z8_intersect(ind).smallerZ8_intersect];
intersect_xyz9(ind).smallerintersect_xyz9=[flatup1(ind).smallerflatup1(1,1),Y9_intersect(ind).smallerY9_intersect,Z9_intersect(ind).smallerZ9_intersect];
end

%%Find endpoint where the missing points are, mid3_side2
for ind = 1:numel(xRngs)-1
[~,idx]=max(Mid2_side2(ind).smallerMid2_side2(:,2));
endpoint_3(ind).smallerendpoint_3=Mid2_side2(ind).smallerMid2_side2(idx,:);

[~,idx]=max(Down_side2(ind).smallerDown_side2(:,2));
endpoint_4(ind).smallerendpoint_4=Down_side2(ind).smallerDown_side2(idx,:);
end

%%Find intersection of missing part, intersection point 4
for ind = 1:numel(xRngs)-1
[~,idx]=min(Down_side2(ind).smallerDown_side2(:,2));
startpoint_4(ind).smallerstartpoint_4=Down_side2(ind).smallerDown_side2(idx,:);

Y4_intersect(ind).smallerY4_intersect=endpoint_3(ind).smallerendpoint_3(:,2);
coefficients(ind).smallercoefficients= polyfit([startpoint_4(ind).smallerstartpoint_4(:,2), endpoint_4(ind).smallerendpoint_4(:,2)], [startpoint_4(ind).smallerstartpoint_4(:,3), endpoint_4(ind).smallerendpoint_4(:,3)], 1);
Z4_intersect(ind).smallerZ4_intersect=coefficients(ind).smallercoefficients(:,1)*Y4_intersect(ind).smallerY4_intersect+coefficients(ind).smallercoefficients(:,2);

intersect_xyz3(ind).smallerintersect_xyz3=[flatup1(ind).smallerflatup1(1,1),endpoint_3(ind).smallerendpoint_3(:,2),endpoint_3(ind).smallerendpoint_3(:,3)];
intersect_xyz4(ind).smallerintersect_xyz4=[flatup1(ind).smallerflatup1(1,1),Y4_intersect(ind).smallerY4_intersect,Z4_intersect(ind).smallerZ4_intersect];

end
%%Figure of the unrotated intersection
for ind = 1:numel(xRngs)-1

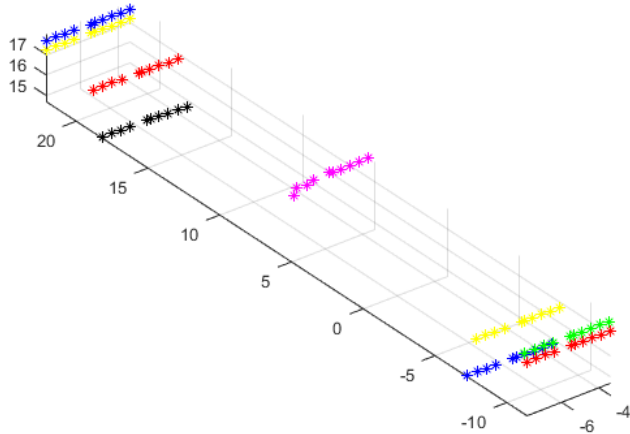
figure(7)
scatter3(intersect_xyz1(ind).smallerintersect_xyz1(:,1),intersect_xyz1(ind).smallerintersect_xyz1(:,2),intersect_xyz1(ind).smallerintersect_xyz1(:,3), '*k')
hold on

```

```

scatter3(intersect_xyz2(ind).smallerintersect_xyz2(:,1),intersect_xyz2(ind).smallerintersect_xyz2(:,2),intersect_xyz2(ind).smallerintersect_xyz2(:,3),'*r')
hold on
scatter3(intersect_xyz3(ind).smallerintersect_xyz3(:,1),intersect_xyz3(ind).smallerintersect_xyz3(:,2),intersect_xyz3(ind).smallerintersect_xyz3(:,3),'*y')
hold on
scatter3(intersect_xyz4(ind).smallerintersect_xyz4(:,1),intersect_xyz4(ind).smallerintersect_xyz4(:,2),intersect_xyz4(ind).smallerintersect_xyz4(:,3),'*b')
hold on
scatter3(intersect_xyz5(ind).smallerintersect_xyz5(:,1),intersect_xyz5(ind).smallerintersect_xyz5(:,2),intersect_xyz5(ind).smallerintersect_xyz5(:,3),'*m')
hold on
scatter3(intersect_xyz6(ind).smallerintersect_xyz6(:,1),intersect_xyz6(ind).smallerintersect_xyz6(:,2),intersect_xyz6(ind).smallerintersect_xyz6(:,3),'*g')
hold on
scatter3(intersect_xyz7(ind).smallerintersect_xyz7(:,1),intersect_xyz7(ind).smallerintersect_xyz7(:,2),intersect_xyz7(ind).smallerintersect_xyz7(:,3),'*r')
hold on
scatter3(intersect_xyz8(ind).smallerintersect_xyz8(:,1),intersect_xyz8(ind).smallerintersect_xyz8(:,2),intersect_xyz8(ind).smallerintersect_xyz8(:,3),'*y')
hold on
scatter3(intersect_xyz9(ind).smallerintersect_xyz9(:,1),intersect_xyz9(ind).smallerintersect_xyz9(:,2),intersect_xyz9(ind).smallerintersect_xyz9(:,3),'*b')
axis equal
end

```



### Rotate intersection point

```

%%Rotate pier back to original orientation and convert from m to mm
for ind = 1:numel(xRngs)-1
rotintersect_xyz1(ind).smallerrotintersect_xyz1=intersect_xyz1(ind).smallerintersect_xyz1*inv(R)*10^3;
rotintersect_xyz2(ind).smallerrotintersect_xyz2=intersect_xyz2(ind).smallerintersect_xyz2*inv(R)*10^3;
rotintersect_xyz3(ind).smallerrotintersect_xyz3=intersect_xyz3(ind).smallerintersect_xyz3*inv(R)*10^3;
rotintersect_xyz4(ind).smallerrotintersect_xyz4=intersect_xyz4(ind).smallerintersect_xyz4*inv(R)*10^3;
rotintersect_xyz5(ind).smallerrotintersect_xyz5=intersect_xyz5(ind).smallerintersect_xyz5*inv(R)*10^3;
rotintersect_xyz6(ind).smallerrotintersect_xyz6=intersect_xyz6(ind).smallerintersect_xyz6*inv(R)*10^3;
rotintersect_xyz7(ind).smallerrotintersect_xyz7=intersect_xyz7(ind).smallerintersect_xyz7*inv(R)*10^3;
rotintersect_xyz8(ind).smallerrotintersect_xyz8=intersect_xyz8(ind).smallerintersect_xyz8*inv(R)*10^3;
rotintersect_xyz9(ind).smallerrotintersect_xyz9=intersect_xyz9(ind).smallerintersect_xyz9*inv(R)*10^3;
end

```

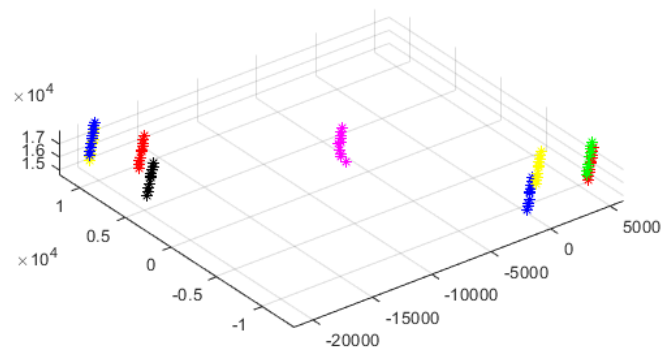
```

%%Figure of the intersection points, rotated back to original positions
for ind = 1:numel(xRngs)-1

figure(8)
scatter3(rotintersect_xyz1(ind).smallerrotintersect_xyz1(:,1),rotintersect_xyz1(ind).smallerrotintersect_xyz1(:,2),rotintersect_xyz1(ind).smallerrotintersect_xyz1(:,3), '*k')
hold on
scatter3(rotintersect_xyz2(ind).smallerrotintersect_xyz2(:,1),rotintersect_xyz2(ind).smallerrotintersect_xyz2(:,2),rotintersect_xyz2(ind).smallerrotintersect_xyz2(:,3), '*r')
hold on
scatter3(rotintersect_xyz3(ind).smallerrotintersect_xyz3(:,1),rotintersect_xyz3(ind).smallerrotintersect_xyz3(:,2),rotintersect_xyz3(ind).smallerrotintersect_xyz3(:,3), '*y')
hold on
scatter3(rotintersect_xyz4(ind).smallerrotintersect_xyz4(:,1),rotintersect_xyz4(ind).smallerrotintersect_xyz4(:,2),rotintersect_xyz4(ind).smallerrotintersect_xyz4(:,3), '*b')
hold on
scatter3(rotintersect_xyz5(ind).smallerrotintersect_xyz5(:,1),rotintersect_xyz5(ind).smallerrotintersect_xyz5(:,2),rotintersect_xyz5(ind).smallerrotintersect_xyz5(:,3), '*m')
hold on
scatter3(rotintersect_xyz6(ind).smallerrotintersect_xyz6(:,1),rotintersect_xyz6(ind).smallerrotintersect_xyz6(:,2),rotintersect_xyz6(ind).smallerrotintersect_xyz6(:,3), '*g')
hold on
scatter3(rotintersect_xyz7(ind).smallerrotintersect_xyz7(:,1),rotintersect_xyz7(ind).smallerrotintersect_xyz7(:,2),rotintersect_xyz7(ind).smallerrotintersect_xyz7(:,3), '*r')
hold on
scatter3(rotintersect_xyz8(ind).smallerrotintersect_xyz8(:,1),rotintersect_xyz8(ind).smallerrotintersect_xyz8(:,2),rotintersect_xyz8(ind).smallerrotintersect_xyz8(:,3), '*y')
hold on
scatter3(rotintersect_xyz9(ind).smallerrotintersect_xyz9(:,1),rotintersect_xyz9(ind).smallerrotintersect_xyz9(:,2),rotintersect_xyz9(ind).smallerrotintersect_xyz9(:,3), '*b')
axis equal

end

```



#### write an Excel file

```

writetable(struct2table(rotintersect_xyz1), 'Intersect_box_17.xlsx', 'sheet', 1);
writetable(struct2table(rotintersect_xyz2), 'Intersect_box_17.xlsx', 'sheet', 2);
writetable(struct2table(rotintersect_xyz3), 'Intersect_box_17.xlsx', 'sheet', 3);
writetable(struct2table(rotintersect_xyz4), 'Intersect_box_17.xlsx', 'sheet', 4);
writetable(struct2table(rotintersect_xyz5), 'Intersect_box_17.xlsx', 'sheet', 5);
writetable(struct2table(rotintersect_xyz6), 'Intersect_box_17.xlsx', 'sheet', 6);
writetable(struct2table(rotintersect_xyz7), 'Intersect_box_17.xlsx', 'sheet', 7);

```

```
writetable(struct2table(rotintersect_xyz8), 'Intersect_box_17.xlsx', 'sheet', 8);  
writetable(struct2table(rotintersect_xyz9), 'Intersect_box_17.xlsx', 'sheet', 9);  
winopen 'Intersect_box_17.xlsx'
```

---

*Published with MATLAB® R2019b*



**A6 Pier Column 1, 2, 5, 6  
Algorithm**

## Contents

---

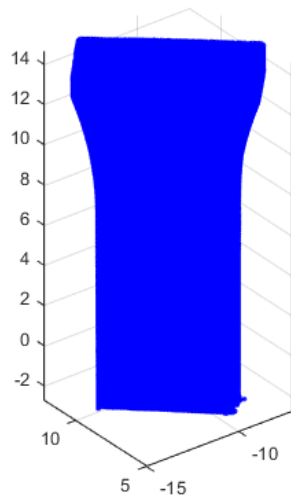
- Pier1,2,5,6
- Flip
- Cut
- Rotate
- Slice
- Parts
- Sides
- Intersection points
- Rotate and Flip back
- Write Excel file

## Pier1,2,5,6

---

```
clear all
close all
clc

A=dlmread('Pier_2_ny.txt');           %%Reading the txt-file of the point cloud
a=A(:,[1,2,3]);                       %%Matrix of the geometrical data
figure(1)
scatter3(a(:,1),a(:,2),a(:,3),'b')
axis equal
```



## Flip

---

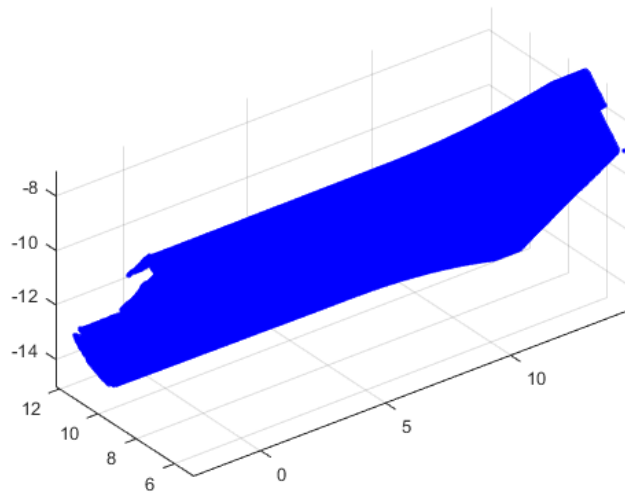
```

b=flip1r(a);           %%flipping the pier
x=b(:,1);
y=b(:,2);
z=b(:,3);
figure(2)
scatter3(b(:,1),b(:,2),b(:,3),'.b')
axis equal

% %%Dividing the pier into smaller segments to find the angle

%The ranges of the matrices are designated by being between n and n+1
xRngs1=[min(b(:,1)):0.25: max(b(:,1))];           %%Find range from min to max, divided by ...   %%Can change
for ind = 1:numel(xRngs1)-1
    rowsOriginal = b(:,1)>=xRngs1(ind) & b(:,1)<xRngs1(ind+1);           %%Check to see which rows of b are between xRng(ind) and xRng(ind+1)
    M(ind).SmallerM = b(rowsOriginal,:);           %%Segmented matrix
end

```



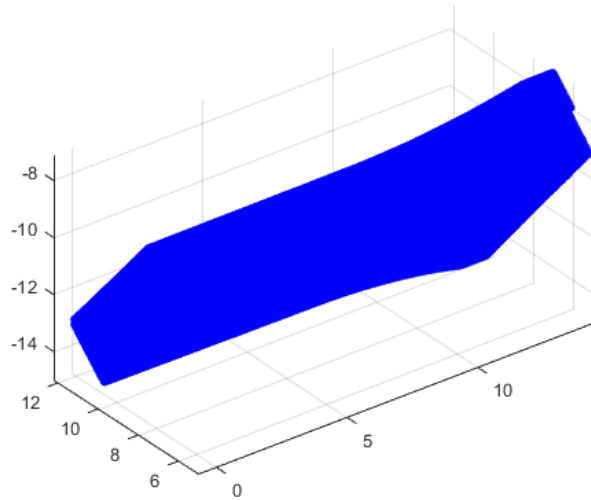
### Cut

```

rowsToDelete = b(:,1) < min(b(:,1))+2 | b(:,1) > max(b(:,1))-0.2;
b(rowsToDelete,:) = [];

figure(4)
scatter3(b(:,1),b(:,2),b(:,3),'.b')
axis equal

```



## Rotate

```

%%find angle to rotate pier

c=M(20).SmallerM;      %%matrix for the first segment
xM=c(:,1);
yM=c(:,2);
zM=c(:,3);

[maxY, indexOfMaxY] = max(yM);
zAtMaxY = zM(indexOfMaxY);
[minZ, indexOfMinZ] = min(zM);
yAtMinZ = yM(indexOfMinZ);

u=[maxY-yAtMinZ, zAtMaxY- minZ];
v=[1,0];

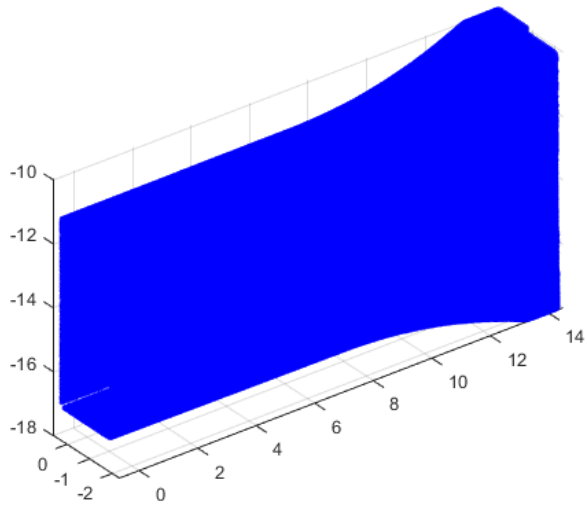
slope=u(:,2)/u(:,1);
theta=atan(slope);      %%Angle

R=[1 0 0;              %%Rotation matrix
   0 cos(theta) -sin(theta);
   0 sin(theta) cos(theta)];

%%Rotate the whole pier
rotM=b*R;               %%rotated matrix

x1=rotM(:,1);
y1=rotM(:,2);
z1=rotM(:,3);
figure(3)
scatter3(rotM(:,1),rotM(:,2),rotM(:,3),'b')
axis equal

```



### Slice

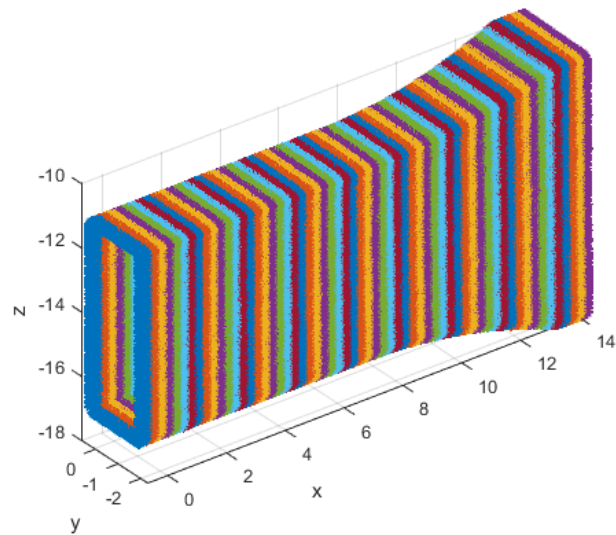
```

%%Dividing the rotated pier into main segments
xRngs=[min(b(:,1)):0.25: max(b(:,1))];
for ind = 1:numel(xRngs)-1

    rows = rotM(:,1)>=xRngs(ind) & b(:,1)<xRngs(ind+1); %check to see which rows of b are between
        %xRng(ind) and xRng(ind+1)
    segM(ind).SmallerM = rotM(rows,:); %segM is the segmented matrix of the rotated pier

    figure(5)
    scatter3(segM(ind).SmallerM(:,1),segM(ind).SmallerM(:,2),segM(ind).SmallerM(:,3),'*') %Scatter the rotated pier
    hold on
    xlabel('x')
    ylabel('y')
    zlabel('z')
    grid on
    axis('equal')
end

```



## Parts

```

%%Find center of the pier
for ind = 1:numel(xRngs)-1

maximumZ(ind)= max(segM(ind).SmallerM(:,3));
minimumZ(ind)=min(segM(ind).SmallerM(:,3));

maximumY(ind)= max(segM(ind).SmallerM(:,2));
minimumY(ind)=min(segM(ind).SmallerM(:,2));

columnMean(ind).SmallercolumnMean = mean(segM(ind).SmallerM,1);           %%Find the mean x-values of each segment

meanY(ind).SmallermeanY=columnMean(ind).SmallercolumnMean(1,2);
meanZ(ind).SmallermeanZ=columnMean(ind).SmallercolumnMean(1,3);

%%Find the appropriate lenght from center and out, before the curve starts
LZ(ind)=maximumZ(ind)-minimumZ(ind);
LY(ind)=maximumY(ind)-minimumY(ind);
lz(ind)=(LZ(ind)/2)-1;                                                    %Can change
ly(ind)=(LY(ind)/2)-0.5;                                                %Can change

%%Find all the points that lies in the wanted range from center
YY(ind).SmallerYY= find(segM(ind).SmallerM(:,2) < (meanY(ind).SmallermeanY+ly(ind)) & segM(ind).SmallerM(:,2) > (meanY(ind).SmallermeanY-ly(ind)));
ZZ(ind).SmallerZZ= find(segM(ind).SmallerM(:,3) < (meanZ(ind).SmallermeanZ+lz(ind)) & segM(ind).SmallerM(:,3) > (meanZ(ind).SmallermeanZ-lz(ind)));

YYmatrix(ind).SmallerYYmatrix=segM(ind).SmallerM(YY(ind).SmallerYY,:);
ZZmatrix(ind).SmallerZZmatrix=segM(ind).SmallerM(ZZ(ind).SmallerZZ,:);

%%Find the center of the pier
centerX(ind).SmallercenterX=mean(YYmatrix(ind).SmallerYYmatrix(:,1));
centerY(ind).SmallercenterY=mean(YYmatrix(ind).SmallerYYmatrix(:,2));
centerZ(ind).SmallercenterZ=mean(ZZmatrix(ind).SmallerZZmatrix(:,3));

```

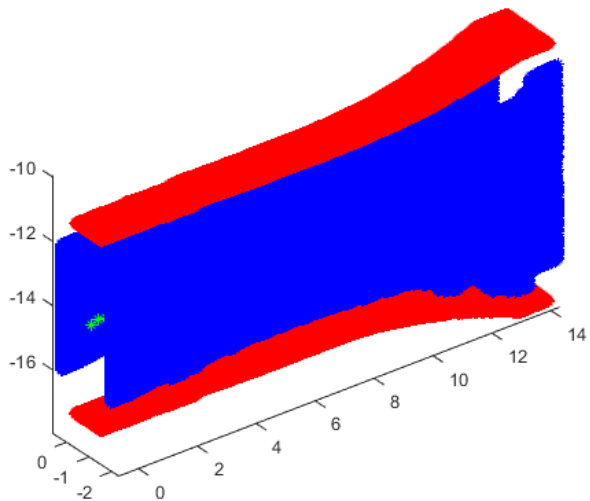
```

CENTER(ind).SmallerCENTER=[centerX(ind).SmallercenterX,centerY(ind).SmallercenterY,centerZ(ind).SmallercenterZ];

CENTER_mat = cell2mat(struct2cell(CENTER));

figure(6) %Shows the center of the pier and the range in the Y and Z direction
plot3(YMatrix(ind).SmallerYMatrix(:,1),YMatrix(ind).SmallerYMatrix(:,2),YMatrix(ind).SmallerYMatrix(:,3),'*r')
hold on
plot3(ZMatrix(ind).SmallerZMatrix(:,1),ZMatrix(ind).SmallerZMatrix(:,2),ZMatrix(ind).SmallerZMatrix(:,3),'*b')
hold on
plot3(CENTER(ind).SmallerCENTER(:,1),CENTER(ind).SmallerCENTER(:,2),CENTER(ind).SmallerCENTER(:,3),'*g')
axis equal
end

```



## Sides

```

for ind = 1:numel(xRngs)-1
%%Dividing the segments into microsegments in the horizontal direction

maximumZ(ind)= max(YMatrix(ind).SmallerYMatrix(:,3));
minimumZ(ind)=min(YMatrix(ind).SmallerYMatrix(:,3));
e=0.005;
kHor(ind) = floor((maximumZ(ind)-minimumZ(ind))/e); %How many times its divided
for i = 1:kHor(ind)
mHor(ind).SmallermHor(i,1) = minimumZ(ind)+e*(i-1); %For which values its divided for each segment
A2Hor(ind).SmallerA2Hor= YMatrix(ind).SmallerYMatrix((YMatrix(ind).SmallerYMatrix(:,3)>=mHor(ind).SmallermHor(i,1) & YMatrix(ind).SmallerYMatrix(:,3)<=mHor(ind).SmallermHor(i,1)+e),:); %logical indexing
ggHor(ind).SmallerggHor{i,1}=A2Hor(ind).SmallerA2Hor; %Counting the points in each microsegment
if isempty(A2Hor(ind).SmallerA2Hor)==1
continue
end
end
end

```

```

%%Dividing the segments into microsegments in the vertical direction

maximumY(ind)= max(ZZmatrix(ind).SmallerZZmatrix(:,2));
minimumY(ind)=min(ZZmatrix(ind).SmallerZZmatrix(:,2));

kVer(ind) = floor((maximumY(ind)-minimumY(ind))/e);
for i = 1:kVer(ind)
    mVer(ind).SmallerVer(i,1) = minimumY(ind)+e*(i-1);
    A2Ver(ind).SmallerA2Ver= ZZmatrix(ind).SmallerZZmatrix((ZZmatrix(ind).SmallerZZmatrix(:,2)>=mVer(ind).SmallerVer(i,1) & ZZmatrix(ind).SmallerZZmatrix(:,2)<=mVer(ind).SmallerVer(i,1)+e),:);
    ggVer(ind).SmallerggVer{i,1}=A2Ver(ind).SmallerA2Ver;
    if isempty(A2Ver(ind).SmallerA2Ver)==1
        continue
    end
end
end

for ind = 1:numel(xRngs)-1
    %%splitting the two sides

    Right1(ind).smallerRight1=ggVer(ind).SmallerggVer(1:find(cellfun(@isempty,ggVer(ind).SmallerggVer),1),:);
    Right(ind).smallerRight=cell2mat(Right1(ind).smallerRight1);

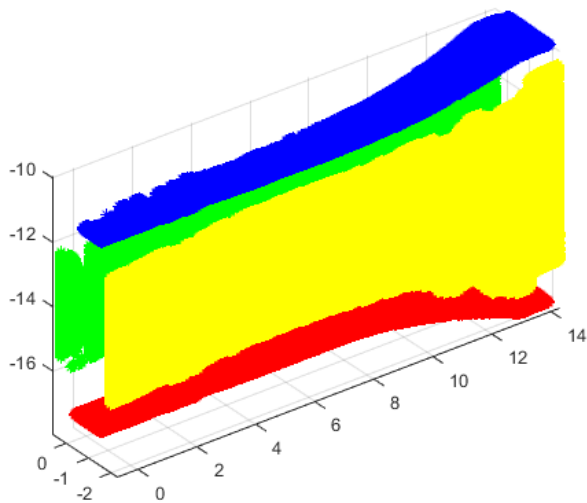
    Left1(ind).smallerLeft1=ggVer(ind).SmallerggVer(find(cellfun(@isempty,ggVer(ind).SmallerggVer),1,'last'):end,:);
    Left(ind).smallerLeft=cell2mat(Left1(ind).smallerLeft1);

    Lower1(ind).smallerLower1=ggHor(ind).SmallerggHor(1:find(cellfun(@isempty,ggHor(ind).SmallerggHor),1),:);
    Lower(ind).smallerLower=cell2mat(Lower1(ind).smallerLower1);

    Upper1(ind).smallerUpper1=ggHor(ind).SmallerggHor(find(cellfun(@isempty,ggHor(ind).SmallerggHor),1,'last'):end,:);
    Upper(ind).smallerUpper=cell2mat(Upper1(ind).smallerUpper1);
    figure(7)
    scatter3(Lower(ind).smallerLower(:,1),Lower(ind).smallerLower(:,2),Lower(ind).smallerLower(:,3),'*r')
    hold on
    scatter3(Upper(ind).smallerUpper(:,1),Upper(ind).smallerUpper(:,2),Upper(ind).smallerUpper(:,3),'*b')
    hold on
    scatter3(Left(ind).smallerLeft(:,1),Left(ind).smallerLeft(:,2),Left(ind).smallerLeft(:,3),'*g')
    hold on
    scatter3(Right(ind).smallerRight(:,1),Right(ind).smallerRight(:,2),Right(ind).smallerRight(:,3),'*y')
    axis equal
end

```





### Intersection points

```

for ind = 1:numel(xRngs)-1
    %%Assigning one X-value to all the rows
    %%Flatten the right side
    flatRight(ind).smallerflatRight=Right(ind).smallerRight;
    flatRight(ind).smallerflatRight(:,1)=repelem(flatRight(ind).smallerflatRight(1),size(flatRight(ind).smallerflatRight,1),1);
    pRight(ind).smallerpRight = polyfit(flatRight(ind).smallerflatRight(:,2),flatRight(ind).smallerflatRight(:,3),1);
    fRight(ind).smallerfRight = polyval(pRight(ind).smallerpRight,flatRight(ind).smallerflatRight(:,2));

    %%flatten the left side
    flatLeft(ind).smallerflatLeft=Left(ind).smallerLeft;
    flatLeft(ind).smallerflatLeft(:,1)=repelem(flatLeft(ind).smallerflatLeft(1),size(flatLeft(ind).smallerflatLeft,1),1);
    pLeft(ind).smallerpLeft = polyfit(flatLeft(ind).smallerflatLeft(:,2),flatLeft(ind).smallerflatLeft(:,3),1);
    fLeft(ind).smallerfLeft = polyval(pLeft(ind).smallerpLeft,flatLeft(ind).smallerflatLeft(:,2));

    %%Flatten the lower
    flatLower(ind).smallerflatLower=Lower(ind).smallerLower;
    flatLower(ind).smallerflatLower(:,1)=repelem(flatLower(ind).smallerflatLower(1),size(flatLower(ind).smallerflatLower,1),1);
    pLower(ind).smallerpLower = polyfit(flatLower(ind).smallerflatLower(:,2),flatLower(ind).smallerflatLower(:,3),1);
    fLower(ind).smallerfLower = polyval(pLower(ind).smallerpLower,flatLower(ind).smallerflatLower(:,2));

    %%Flatten the Upper
    flatUpper(ind).smallerflatUpper=Upper(ind).smallerUpper;
    % if isempty(Upper(ind).smallerUpper)==1
    % continue
    % end
    flatUpper(ind).smallerflatUpper(:,1)=repelem(flatUpper(ind).smallerflatUpper(1),size(flatUpper(ind).smallerflatUpper,1),1);
    pUpper(ind).smallerpUpper = polyfit(flatUpper(ind).smallerflatUpper(:,2),flatUpper(ind).smallerflatUpper(:,3),1);
    fUpper(ind).smallerfUpper = polyval(pUpper(ind).smallerpUpper,flatUpper(ind).smallerflatUpper(:,2));
end

for ind = 1:numel(xRngs)-1

```

```

%%find intersection
Y1_intersect(ind).smallerY1_intersect=fzero(@(x) polyval(pLeft(ind).smallerpLeft-pLower(ind).smallerpLower,x),3);
Z1_intersect(ind).smallerZ1_intersect=polyval(pLeft(ind).smallerpLeft,Y1_intersect(ind).smallerY1_intersect);

Y2_intersect(ind).smallerY2_intersect=fzero(@(x) polyval(pLeft(ind).smallerpLeft-pUpper(ind).smallerpUpper,x),3);
Z2_intersect(ind).smallerZ2_intersect=polyval(pLeft(ind).smallerpLeft,Y2_intersect(ind).smallerY2_intersect);

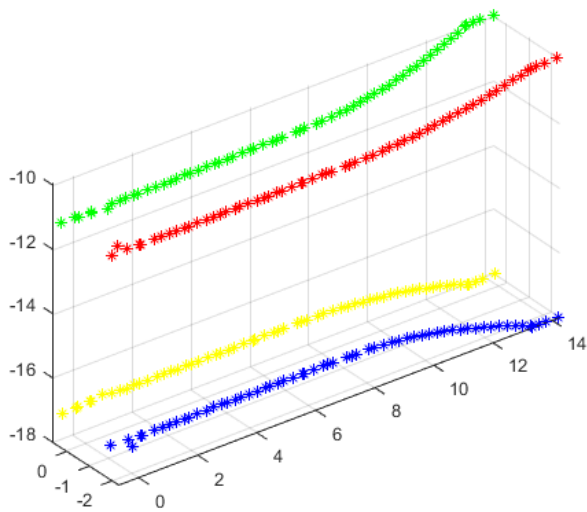
Y3_intersect(ind).smallerY3_intersect=fzero(@(x) polyval(pRight(ind).smallerpRight-pUpper(ind).smallerpUpper,x),3);
Z3_intersect(ind).smallerZ3_intersect=polyval(pRight(ind).smallerpRight,Y3_intersect(ind).smallerY3_intersect);

Y4_intersect(ind).smallerY4_intersect=fzero(@(x) polyval(pRight(ind).smallerpRight-pLower(ind).smallerpLower,x),3);
Z4_intersect(ind).smallerZ4_intersect=polyval(pRight(ind).smallerpRight,Y4_intersect(ind).smallerY4_intersect);

end

for ind = 1:numel(xRngs)-1
    %%New matrix of points with same flattened x-value from right
    intersect_xyz1(ind).smallerintersect_xyz1=[flatRight(ind).smallerflatRight(1,1),Y1_intersect(ind).smallerY1_intersect,Z1_intersect(ind).smallerZ1_intersect];
    intersect_xyz2(ind).smallerintersect_xyz2=[flatRight(ind).smallerflatRight(1,1),Y2_intersect(ind).smallerY2_intersect,Z2_intersect(ind).smallerZ2_intersect];
    intersect_xyz3(ind).smallerintersect_xyz3=[flatRight(ind).smallerflatRight(1,1),Y3_intersect(ind).smallerY3_intersect,Z3_intersect(ind).smallerZ3_intersect];
    intersect_xyz4(ind).smallerintersect_xyz4=[flatRight(ind).smallerflatRight(1,1),Y4_intersect(ind).smallerY4_intersect,Z4_intersect(ind).smallerZ4_intersect];
end
for ind = 1:numel(xRngs)-1
    figure(8)
    scatter3(intersect_xyz1(ind).smallerintersect_xyz1(:,1),intersect_xyz1(ind).smallerintersect_xyz1(:,2),intersect_xyz1(ind).smallerintersect_xyz1(:,3), '*y')
    hold on
    scatter3(intersect_xyz2(ind).smallerintersect_xyz2(:,1),intersect_xyz2(ind).smallerintersect_xyz2(:,2),intersect_xyz2(ind).smallerintersect_xyz2(:,3), '*g')
    hold on
    scatter3(intersect_xyz3(ind).smallerintersect_xyz3(:,1),intersect_xyz3(ind).smallerintersect_xyz3(:,2),intersect_xyz3(ind).smallerintersect_xyz3(:,3), '*r')
    hold on
    scatter3(intersect_xyz4(ind).smallerintersect_xyz4(:,1),intersect_xyz4(ind).smallerintersect_xyz4(:,2),intersect_xyz4(ind).smallerintersect_xyz4(:,3), '*b')
    axis equal
end

```



### Rotate and Flip back

```

for ind = 1:numel(xRngs)-1
%Rotate pier back to original orientation and convert from m to mm
rotintersect_xyz1(ind).smallerotintersect_xyz1=intersect_xyz1(ind).smallerintersect_xyz1*inv(R)*10^3;
rotintersect_xyz2(ind).smallerotintersect_xyz2=intersect_xyz2(ind).smallerintersect_xyz2*inv(R)*10^3;
rotintersect_xyz3(ind).smallerotintersect_xyz3=intersect_xyz3(ind).smallerintersect_xyz3*inv(R)*10^3;
rotintersect_xyz4(ind).smallerotintersect_xyz4=intersect_xyz4(ind).smallerintersect_xyz4*inv(R)*10^3;
%flip pier back to original angle (upwards)
rotintersect_xyz1flip(ind).smallerotintersect_xyz1flip=flipplr(rotintersect_xyz1(ind).smallerotintersect_xyz1);
rotintersect_xyz2flip(ind).smallerotintersect_xyz2flip=flipplr(rotintersect_xyz2(ind).smallerotintersect_xyz2);
rotintersect_xyz3flip(ind).smallerotintersect_xyz3flip=flipplr(rotintersect_xyz3(ind).smallerotintersect_xyz3);
rotintersect_xyz4flip(ind).smallerotintersect_xyz4flip=flipplr(rotintersect_xyz4(ind).smallerotintersect_xyz4);
end

%Shows the intersection points of the rotated pier at with and without original orientation
for ind = 1:numel(xRngs)-1
    figure(9)
    scatter3(rotintersect_xyz1(ind).smallerotintersect_xyz1(:,1),rotintersect_xyz1(ind).smallerotintersect_xyz1(:,2),rotintersect_xyz1(ind).smallerotintersect_xyz1(:,3), '*y')
    hold on
    scatter3(rotintersect_xyz2(ind).smallerotintersect_xyz2(:,1),rotintersect_xyz2(ind).smallerotintersect_xyz2(:,2),rotintersect_xyz2(ind).smallerotintersect_xyz2(:,3), '*g')
    hold on
    scatter3(rotintersect_xyz3(ind).smallerotintersect_xyz3(:,1),rotintersect_xyz3(ind).smallerotintersect_xyz3(:,2),rotintersect_xyz3(ind).smallerotintersect_xyz3(:,3), '*r')
    hold on
    scatter3(rotintersect_xyz4(ind).smallerotintersect_xyz4(:,1),rotintersect_xyz4(ind).smallerotintersect_xyz4(:,2),rotintersect_xyz4(ind).smallerotintersect_xyz4(:,3), '*b')
    axis equal

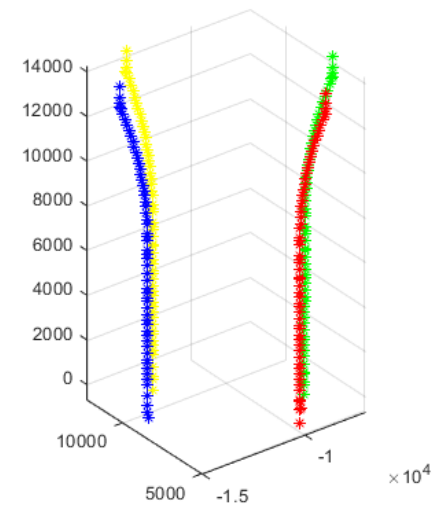
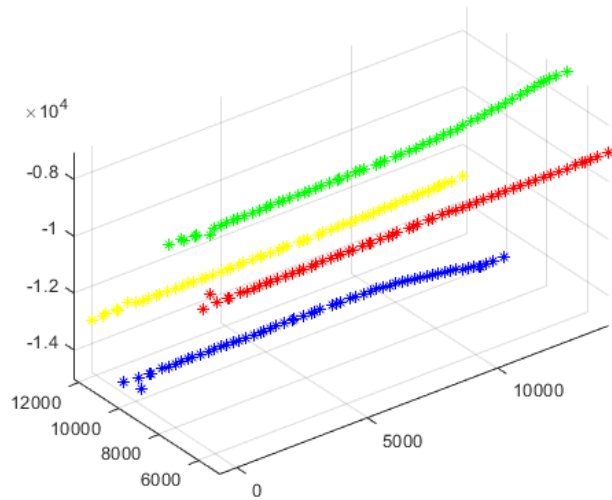
figure(10)
scatter3(rotintersect_xyz1flip(ind).smallerotintersect_xyz1flip(:,1),rotintersect_xyz1flip(ind).smallerotintersect_xyz1flip(:,2),rotintersect_xyz1flip(ind).smallerotintersect_xyz1flip(:,3), '*y')
hold on
scatter3(rotintersect_xyz2flip(ind).smallerotintersect_xyz2flip(:,1),rotintersect_xyz2flip(ind).smallerotintersect_xyz2flip(:,2),rotintersect_xyz2flip(ind).smallerotintersect_xyz2flip(:,3), '*g')
hold on

```

```

scatter3(rotintersect_xyz3flip(ind).smallerrotintersect_xyz3flip(:,1),rotintersect_xyz3flip(ind).smallerrotintersect_xyz3flip(:,2),rotintersect_xyz3flip(ind).smallerrotintersect_xyz3flip(:,3), '*r')
hold on
scatter3(rotintersect_xyz4flip(ind).smallerrotintersect_xyz4flip(:,1),rotintersect_xyz4flip(ind).smallerrotintersect_xyz4flip(:,2),rotintersect_xyz4flip(ind).smallerrotintersect_xyz4flip(:,3), '*b')
hold on
%scatter3(A(:,1),A(:,2),A(:,3), '*m')
axis equal
end

```



### Write Excel file

```
%write a Excel file of a struct
```

```

writetable(struct2table(rotintersect_xyz1flip), 'Intersect_pier2.xlsx', 'sheet', 1);
writetable(struct2table(rotintersect_xyz2flip), 'Intersect_pier2.xlsx', 'sheet', 2);
writetable(struct2table(rotintersect_xyz3flip), 'Intersect_pier2.xlsx', 'sheet', 3);
writetable(struct2table(rotintersect_xyz4flip), 'Intersect_pier2.xlsx', 'sheet', 4);
winopen 'Intersect_pier2.xlsx'

```

```
%write an excel file of the original z-values
```

```

writematrix(max(a(:,3))*10^3, 'Original_z_values_pier2.xlsx', 'sheet', 1, 'Range', 'A1');
writematrix(min(a(:,3))*10^3, 'Original_z_values_pier2.xlsx', 'sheet', 1, 'Range', 'A2');

```

```

Warning: Added specified worksheet.
Warning: Added specified worksheet.
Warning: Added specified worksheet.

```



## **A7 Pier Column 3, 4 Algorithm**

## Contents

---

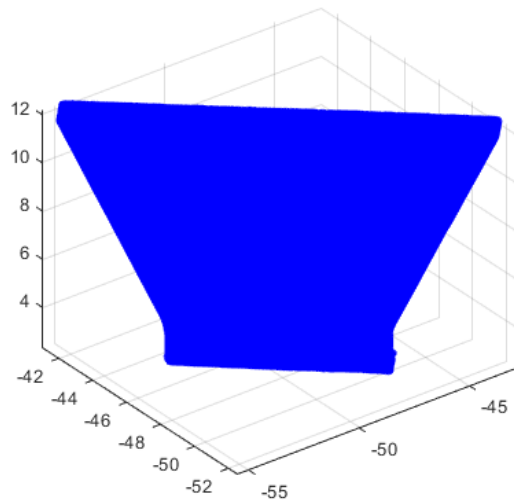
- [Pier 3,4](#)
- [Flip](#)
- [Cut](#)
- [Rotate](#)
- [Slice](#)
- [Parts](#)
- [Sides](#)
- [Intersection points](#)
- [Rotate and Flip](#)

## Pier 3,4

---

```
clear all
close all
clc

A=dlmread('Pier4.txt');           %%Reading the txt-file of the point cloud
a=A(:,[1,2,3]);                  %%Matrix of the geometrical data
figure(1)
scatter3(a(:,1),a(:,2),a(:,3),'.b')
axis equal
```



## Flip

---

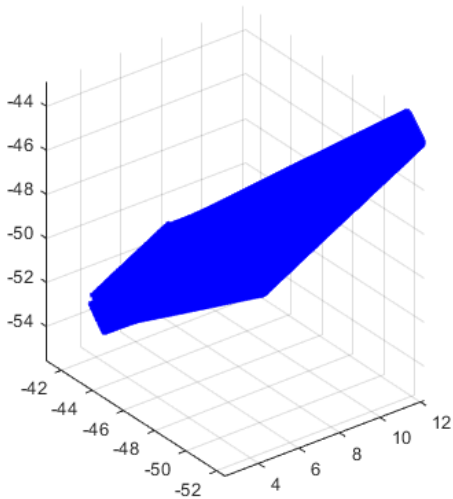
```

b=flip1r(a);           %%flipping the pier
x=b(:,1);
y=b(:,2);
z=b(:,3);
figure(2)
scatter3(b(:,1),b(:,2),b(:,3),'.b')
axis equal

%   %%Dividing the pier into smaller segments to find the angle

%The ranges of the matrices are designated by being between n and n+1
xRngs1=[min(b(:,1)):0.25: max(b(:,1))];           %%Find range from min to max, divided by ...   %%Can change
for ind = 1:numel(xRngs1)-1
    rowsOriginal = b(:,1)>=xRngs1(ind) & b(:,1)<xRngs1(ind+1);           %%Check to see which rows of b are between xRng(ind) and xRng(ind+1)
    M(ind).SmallerM = b(rowsOriginal,:);           %%Segmented matrix
end

```



### Cut

```
% No cut for pier 4
```

### Rotate

```

%%find angle to rotate pier

c=M(20).SmallerM;           %%matrix for the first segment
xM=c(:,1);
yM=c(:,2);
zM=c(:,3);

```



```

[maxY, indexOfMaxY] = max(yM);
zAtMaxY = zM(indexOfMaxY);
[minZ, indexOfMinZ] = min(zM);
yAtMinZ = yM(indexOfMinZ);

u=[maxY-yAtMinZ, zAtMaxY- minZ];
v=[1,0];

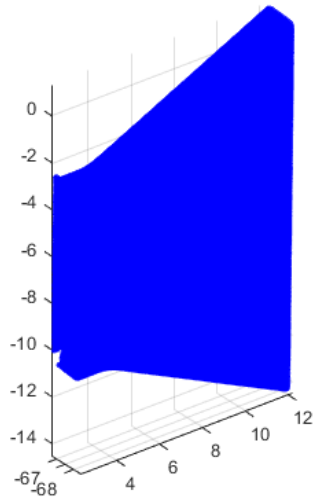
slope=u(:,2)/u(:,1);
theta=atan(slope);           %%Angle

R=[1 0 0;                    %%Rotation matrix
   0 cos(theta) -sin(theta);
   0 sin(theta)  cos(theta)];

%%Rotate the whole pier
rotM=b*R;                    %%rotated matrix

x1=rotM(:,1);
y1=rotM(:,2);
z1=rotM(:,3);
figure(3)
scatter3(rotM(:,1),rotM(:,2),rotM(:,3),'.b')
axis equal

```



### Slice

```

%%Dividing the rotated pier into main segments
xRngs=[min(b(:,1)):0.25: max(b(:,1))];
for ind = 1:numel(xRngs)-1

    rows = rotM(:,1)>=xRngs(ind) & b(:,1)<xRngs(ind+1); %check to see which rows of b are between

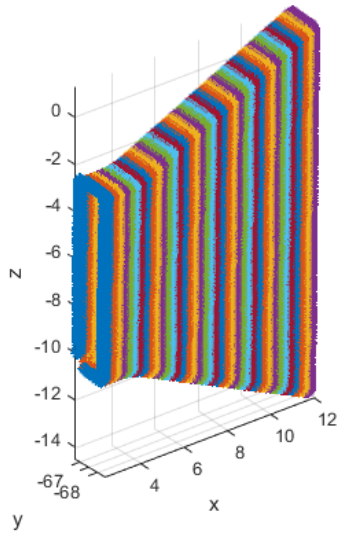
```

```

    %xRng(ind) and xRng(ind+1)
    segM(ind).SmallerM = rotM(rows,:);           %%segM is the segmented matrix of the rotated pier

    figure(5)
    scatter3(segM(ind).SmallerM(:,1),segM(ind).SmallerM(:,2),segM(ind).SmallerM(:,3),'*') %%Scatter the rotated pier
    hold on
    xlabel('x')
    ylabel('y')
    zlabel('z')
    grid on
    axis('equal')
end

```



## Parts

```

%%Find center of the pier
for ind = 1:numel(xRngs)-1

    maximumZ(ind)= max(segM(ind).SmallerM(:,3));
    minimumZ(ind)=min(segM(ind).SmallerM(:,3));

    maximumY(ind)= max(segM(ind).SmallerM(:,2));
    minimumY(ind)=min(segM(ind).SmallerM(:,2));

    columnMean(ind).SmallercolumnMean = mean(segM(ind).SmallerM,1);           %%Find the mean x-values of each segment

    meanY(ind).SmallermeanY=columnMean(ind).SmallercolumnMean(1,2);
    meanZ(ind).SmallermeanZ=columnMean(ind).SmallercolumnMean(1,3);

    %%Find the appropriate lenght from center and out, before the curve starts
    LZ(ind)=maximumZ(ind)-minimumZ(ind);
    LY(ind)=maximumY(ind)-minimumY(ind);

```

```

lz(ind)=(LZ(ind)/2)-1;
ly(ind)=(LY(ind)/2)-0.5;
%Can change
%Can change

%%Find all the points that lies in the wanted range from center
YY(ind).SmallerYY= find(segM(ind).SmallerM(:,2) < (meanY(ind).SmallermeanY+ly(ind)) & segM(ind).SmallerM(:,2) > (meanY(ind).SmallermeanY-ly(ind)));
ZZ(ind).SmallerZZ= find(segM(ind).SmallerM(:,3) < (meanZ(ind).SmallermeanZ+lz(ind)) & segM(ind).SmallerM(:,3) > (meanZ(ind).SmallermeanZ-lz(ind)));

YYmatrix(ind).SmallerYYmatrix=segM(ind).SmallerM(YY(ind).SmallerYY,:);
ZZmatrix(ind).SmallerZZmatrix=segM(ind).SmallerM(ZZ(ind).SmallerZZ,:);

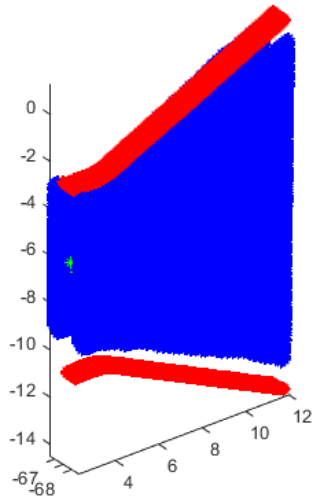
%%Find the center of the pier
centerX(ind).SmallercenterX=mean(YYmatrix(ind).SmallerYYmatrix(:,1));
centerY(ind).SmallercenterY=mean(YYmatrix(ind).SmallerYYmatrix(:,2));
centerZ(ind).SmallercenterZ=mean(ZZmatrix(ind).SmallerZZmatrix(:,3));

CENTER(ind).SmallerCENTER=[centerX(ind).SmallercenterX,centerY(ind).SmallercenterY,centerZ(ind).SmallercenterZ];

CENTER_mat = cell2mat(struct2cell(CENTER));

figure(6) %%Shows the center of the pier and the range in the Y and Z direction
plot3(YYmatrix(ind).SmallerYYmatrix(:,1),YYmatrix(ind).SmallerYYmatrix(:,2),YYmatrix(ind).SmallerYYmatrix(:,3), '*r')
hold on
plot3(ZZmatrix(ind).SmallerZZmatrix(:,1),ZZmatrix(ind).SmallerZZmatrix(:,2),ZZmatrix(ind).SmallerZZmatrix(:,3), '*b')
hold on
plot3(CENTER(ind).SmallerCENTER(:,1),CENTER(ind).SmallerCENTER(:,2),CENTER(ind).SmallerCENTER(:,3), '*g')
axis equal
end

```



## Sides

```

for ind = 1:numel(xRngs)-1
%%Dividing the segments into microsegments in the horizontal direction

```

```

maximumZ(ind)= max(YYmatrix(ind).SmallerYYmatrix(:,3));
minimumZ(ind)=min(YYmatrix(ind).SmallerYYmatrix(:,3));
e=0.005;
kHor(ind) = floor((maximumZ(ind)-minimumZ(ind))/e);           %How many times its divided
for i = 1:kHor(ind)
    mHor(ind).SmallermHor(i,1) = minimumZ(ind)+e*(i-1);       %For which values its divided for each segment
    A2Hor(ind).SmallerA2Hor= YYmatrix(ind).SmallerYYmatrix((YYmatrix(ind).SmallerYYmatrix(:,3)>=mHor(ind).SmallermHor(i,1) & YYmatrix(ind).SmallerYYmatrix(:,3)<=mHor(ind).SmallermHor(i,1)+e),:); %logical indexing
    ggHor(ind).SmallerggHor{i,1}=A2Hor(ind).SmallerA2Hor;      %Counting the points in each microsegment
    if isempty(A2Hor(ind).SmallerA2Hor)==1
        continue
    end
end

%%Dividing the segments into microsegments in the vertical direction

maximumY(ind)= max(ZZmatrix(ind).SmallerZZmatrix(:,2));
minimumY(ind)=min(ZZmatrix(ind).SmallerZZmatrix(:,2));

kVer(ind) = floor((maximumY(ind)-minimumY(ind))/e);           %can change %OBSOBS SJEKK OM A2Hor og A2Ver har verdier, hvis ikke gjør om på denne forløkken, og evt øverst i xRngs
for i = 1:kVer(ind)
    mVer(ind).SmallermVer(i,1) = minimumY(ind)+e*(i-1);
    A2Ver(ind).SmallerA2Ver= ZZmatrix(ind).SmallerZZmatrix((ZZmatrix(ind).SmallerZZmatrix(:,2)>=mVer(ind).SmallermVer(i,1) & ZZmatrix(ind).SmallerZZmatrix(:,2)<=mVer(ind).SmallermVer(i,1)+e),:); %logical indexing
    ggVer(ind).SmallerggVer{i,1}=A2Ver(ind).SmallerA2Ver;      %Counting the points in each microsegment
    if isempty(A2Ver(ind).SmallerA2Ver)==1
        continue
    end
end

for ind = 1:numel(xRngs)-1
    %%splitting the two sides

    Right1(ind).smallerRight1=ggVer(ind).SmallerggVer(1:find(cellfun(@isempty,ggVer(ind).SmallerggVer),1),:);
    Right(ind).smallerRight=cell2mat(Right1(ind).smallerRight1);

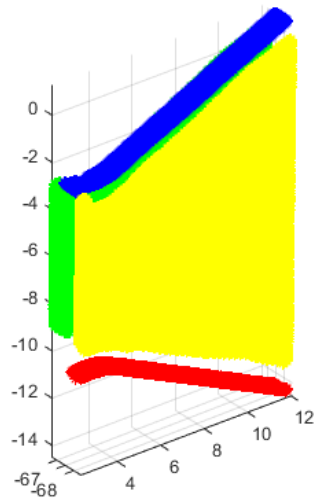
    Left1(ind).smallerLeft1=ggVer(ind).SmallerggVer(find(cellfun(@isempty,ggVer(ind).SmallerggVer),1,'last'):end,:);
    Left(ind).smallerLeft=cell2mat(Left1(ind).smallerLeft1);

    Lower1(ind).smallerLower1=ggHor(ind).SmallerggHor(1:find(cellfun(@isempty,ggHor(ind).SmallerggHor),1),:);
    Lower(ind).smallerLower=cell2mat(Lower1(ind).smallerLower1);

    Upper1(ind).smallerUpper1=ggHor(ind).SmallerggHor(find(cellfun(@isempty,ggHor(ind).SmallerggHor),1,'last'):end,:);
    Upper(ind).smallerUpper=cell2mat(Upper1(ind).smallerUpper1);

    figure(7)
    scatter3(Lower(ind).smallerLower(:,1),Lower(ind).smallerLower(:,2),Lower(ind).smallerLower(:,3),'*r')
    hold on
    scatter3(Upper(ind).smallerUpper(:,1),Upper(ind).smallerUpper(:,2),Upper(ind).smallerUpper(:,3),'*b')
    hold on
    scatter3(Left(ind).smallerLeft(:,1),Left(ind).smallerLeft(:,2),Left(ind).smallerLeft(:,3),'*g')
    hold on
    scatter3(Right(ind).smallerRight(:,1),Right(ind).smallerRight(:,2),Right(ind).smallerRight(:,3),'*y')
    axis equal
end

```



### Intersection points

```

for ind = 1:numel(xRngs)-1
    %%Assigning one X-value to all the rows
    %%Flatten the right side
    flatRight(ind).smallerflatRight=Right(ind).smallerRight;
    flatRight(ind).smallerflatRight(:,1)=repelem(flatRight(ind).smallerflatRight(1),size(flatRight(ind).smallerflatRight,1),1);
    pRight(ind).smallerpRight = polyfit(flatRight(ind).smallerflatRight(:,2),flatRight(ind).smallerflatRight(:,3),1);
    fRight(ind).smallerfRight = polyval(pRight(ind).smallerpRight,flatRight(ind).smallerflatRight(:,2));

    %%flatten the left side
    flatLeft(ind).smallerflatLeft=Left(ind).smallerLeft;
    flatLeft(ind).smallerflatLeft(:,1)=repelem(flatLeft(ind).smallerflatLeft(1),size(flatLeft(ind).smallerflatLeft,1),1);
    pLeft(ind).smallerpLeft = polyfit(flatLeft(ind).smallerflatLeft(:,2),flatLeft(ind).smallerflatLeft(:,3),1);
    fLeft(ind).smallerfLeft = polyval(pLeft(ind).smallerpLeft,flatLeft(ind).smallerflatLeft(:,2));

    %%Flatten the lower
    flatLower(ind).smallerflatLower=Lower(ind).smallerLower;
    flatLower(ind).smallerflatLower(:,1)=repelem(flatLower(ind).smallerflatLower(1),size(flatLower(ind).smallerflatLower,1),1);
    pLower(ind).smallerpLower = polyfit(flatLower(ind).smallerflatLower(:,2),flatLower(ind).smallerflatLower(:,3),1);
    fLower(ind).smallerfLower = polyval(pLower(ind).smallerpLower,flatLower(ind).smallerflatLower(:,2));

    %%Flatten the Upper
    flatUpper(ind).smallerflatUpper=Upper(ind).smallerUpper;
    % if isempty(Upper(ind).smallerUpper)==1
    % continue
    % end
    flatUpper(ind).smallerflatUpper(:,1)=repelem(flatUpper(ind).smallerflatUpper(1),size(flatUpper(ind).smallerflatUpper,1),1);
    pUpper(ind).smallerpUpper = polyfit(flatUpper(ind).smallerflatUpper(:,2),flatUpper(ind).smallerflatUpper(:,3),1);
    fUpper(ind).smallerfUpper = polyval(pUpper(ind).smallerpUpper,flatUpper(ind).smallerflatUpper(:,2));
end

for ind = 1:numel(xRngs)-1

```

```

%%find intersection
Y1_intersect(ind).smallerY1_intersect=fzero(@(x) polyval(pLeft(ind).smallerpLeft-pLower(ind).smallerpLower,x),3);
Z1_intersect(ind).smallerZ1_intersect=polyval(pLeft(ind).smallerpLeft,Y1_intersect(ind).smallerY1_intersect);

Y2_intersect(ind).smallerY2_intersect=fzero(@(x) polyval(pLeft(ind).smallerpLeft-pUpper(ind).smallerpUpper,x),3);
Z2_intersect(ind).smallerZ2_intersect=polyval(pLeft(ind).smallerpLeft,Y2_intersect(ind).smallerY2_intersect);

Y3_intersect(ind).smallerY3_intersect=fzero(@(x) polyval(pRight(ind).smallerpRight-pUpper(ind).smallerpUpper,x),3);
Z3_intersect(ind).smallerZ3_intersect=polyval(pRight(ind).smallerpRight,Y3_intersect(ind).smallerY3_intersect);

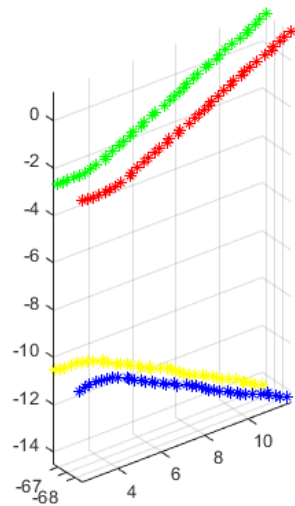
Y4_intersect(ind).smallerY4_intersect=fzero(@(x) polyval(pRight(ind).smallerpRight-pLower(ind).smallerpLower,x),3);
Z4_intersect(ind).smallerZ4_intersect=polyval(pRight(ind).smallerpRight,Y4_intersect(ind).smallerY4_intersect);

end

for ind = 1:numel(xRngs)-1
%%New matrix of points with same flattened x-value from right
intersect_xyz1(ind).smallerintersect_xyz1=[flatRight(ind).smallerflatRight(1,1),Y1_intersect(ind).smallerY1_intersect,Z1_intersect(ind).smallerZ1_intersect];
intersect_xyz2(ind).smallerintersect_xyz2=[flatRight(ind).smallerflatRight(1,1),Y2_intersect(ind).smallerY2_intersect,Z2_intersect(ind).smallerZ2_intersect];
intersect_xyz3(ind).smallerintersect_xyz3=[flatRight(ind).smallerflatRight(1,1),Y3_intersect(ind).smallerY3_intersect,Z3_intersect(ind).smallerZ3_intersect];
intersect_xyz4(ind).smallerintersect_xyz4=[flatRight(ind).smallerflatRight(1,1),Y4_intersect(ind).smallerY4_intersect,Z4_intersect(ind).smallerZ4_intersect];
end
for ind = 1:numel(xRngs)-1
    figure(8)
scatter3(intersect_xyz1(ind).smallerintersect_xyz1(:,1),intersect_xyz1(ind).smallerintersect_xyz1(:,2),intersect_xyz1(ind).smallerintersect_xyz1(:,3), '*y')
hold on
scatter3(intersect_xyz2(ind).smallerintersect_xyz2(:,1),intersect_xyz2(ind).smallerintersect_xyz2(:,2),intersect_xyz2(ind).smallerintersect_xyz2(:,3), '*g')
hold on
scatter3(intersect_xyz3(ind).smallerintersect_xyz3(:,1),intersect_xyz3(ind).smallerintersect_xyz3(:,2),intersect_xyz3(ind).smallerintersect_xyz3(:,3), '*r')
hold on
scatter3(intersect_xyz4(ind).smallerintersect_xyz4(:,1),intersect_xyz4(ind).smallerintersect_xyz4(:,2),intersect_xyz4(ind).smallerintersect_xyz4(:,3), '*b')
axis equal
end

```

Warning: Polynomial is not unique; degree >= number of data points.



### Rotate and Flip

```

for ind = 1:numel(xRngs)-1
%Rotate pier back to original orientation
rotintersect_xyz1(ind).smallerotintersect_xyz1=intersect_xyz1(ind).smallerintersect_xyz1*inv(R);
rotintersect_xyz2(ind).smallerotintersect_xyz2=intersect_xyz2(ind).smallerintersect_xyz2*inv(R);
rotintersect_xyz3(ind).smallerotintersect_xyz3=intersect_xyz3(ind).smallerintersect_xyz3*inv(R);
rotintersect_xyz4(ind).smallerotintersect_xyz4=intersect_xyz4(ind).smallerintersect_xyz4*inv(R);
%flip pier back to original angle (upwards)
rotintersect_xyz1flip(ind).smallerotintersect_xyz1flip=flipplr(rotintersect_xyz1(ind).smallerotintersect_xyz1);
rotintersect_xyz2flip(ind).smallerotintersect_xyz2flip=flipplr(rotintersect_xyz2(ind).smallerotintersect_xyz2);
rotintersect_xyz3flip(ind).smallerotintersect_xyz3flip=flipplr(rotintersect_xyz3(ind).smallerotintersect_xyz3);
rotintersect_xyz4flip(ind).smallerotintersect_xyz4flip=flipplr(rotintersect_xyz4(ind).smallerotintersect_xyz4);
end

for ind = 1:numel(xRngs)-1
    figure(9)
scatter3(rotintersect_xyz1(ind).smallerotintersect_xyz1(:,1),rotintersect_xyz1(ind).smallerotintersect_xyz1(:,2),rotintersect_xyz1(ind).smallerotintersect_xyz1(:,3), '*y')
hold on
scatter3(rotintersect_xyz2(ind).smallerotintersect_xyz2(:,1),rotintersect_xyz2(ind).smallerotintersect_xyz2(:,2),rotintersect_xyz2(ind).smallerotintersect_xyz2(:,3), '*g')
hold on
scatter3(rotintersect_xyz3(ind).smallerotintersect_xyz3(:,1),rotintersect_xyz3(ind).smallerotintersect_xyz3(:,2),rotintersect_xyz3(ind).smallerotintersect_xyz3(:,3), '*r')
hold on
scatter3(rotintersect_xyz4(ind).smallerotintersect_xyz4(:,1),rotintersect_xyz4(ind).smallerotintersect_xyz4(:,2),rotintersect_xyz4(ind).smallerotintersect_xyz4(:,3), '*b')
axis equal

figure(10)
scatter3(rotintersect_xyz1flip(ind).smallerotintersect_xyz1flip(:,1),rotintersect_xyz1flip(ind).smallerotintersect_xyz1flip(:,2),rotintersect_xyz1flip(ind).smallerotintersect_xyz1flip(:,3), '*y')
hold on
scatter3(rotintersect_xyz2flip(ind).smallerotintersect_xyz2flip(:,1),rotintersect_xyz2flip(ind).smallerotintersect_xyz2flip(:,2),rotintersect_xyz2flip(ind).smallerotintersect_xyz2flip(:,3), '*g')
hold on
scatter3(rotintersect_xyz3flip(ind).smallerotintersect_xyz3flip(:,1),rotintersect_xyz3flip(ind).smallerotintersect_xyz3flip(:,2),rotintersect_xyz3flip(ind).smallerotintersect_xyz3flip(:,3), '*r')

```

```

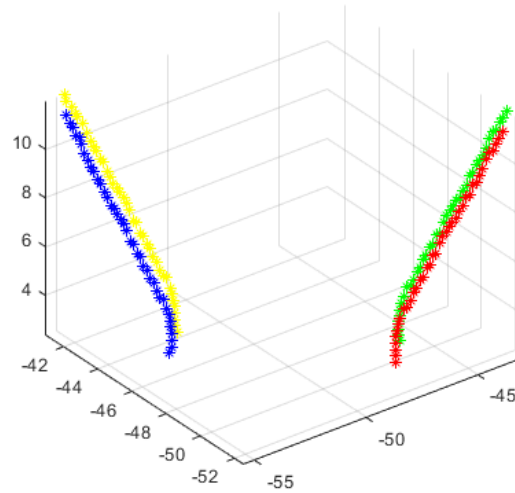
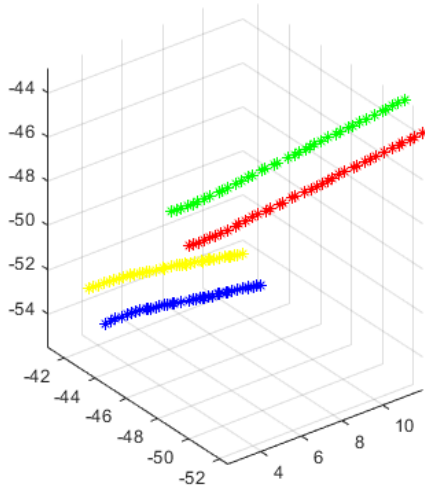
hold on
scatter3(rotintersect_xyz4flip(ind).smallerrotintersect_xyz4flip(:,1),rotintersect_xyz4flip(ind).smallerrotintersect_xyz4flip(:,2),rotintersect_xyz4flip(ind).smallerrotintersect_xyz4flip(:,3), '*b')
hold on
axis equal
end

%write a Excel file of a struct

writetable(struct2table(rotintersect_xyz1flip), 'Intersect_pier4.xlsx', 'sheet',1);
writetable(struct2table(rotintersect_xyz2flip), 'Intersect_pier4.xlsx', 'sheet',2);
writetable(struct2table(rotintersect_xyz3flip), 'Intersect_pier4.xlsx', 'sheet',3);
writetable(struct2table(rotintersect_xyz4flip), 'Intersect_pier4.xlsx', 'sheet',4);
winopen 'Intersect_pier4.xlsx'

%write an excel file of the original z-values
writematrix(a(:,3), 'Original_z_values_pier4.xlsx', 'sheet',1);

```





## **A8 Pier Foundation Algorithm**

## Contents

---

- [Pier foundation](#)
- [Slice](#)
- [Circlefit](#)
- [Write Excel file](#)
- [Circlefit Function](#)

## Pier foundation

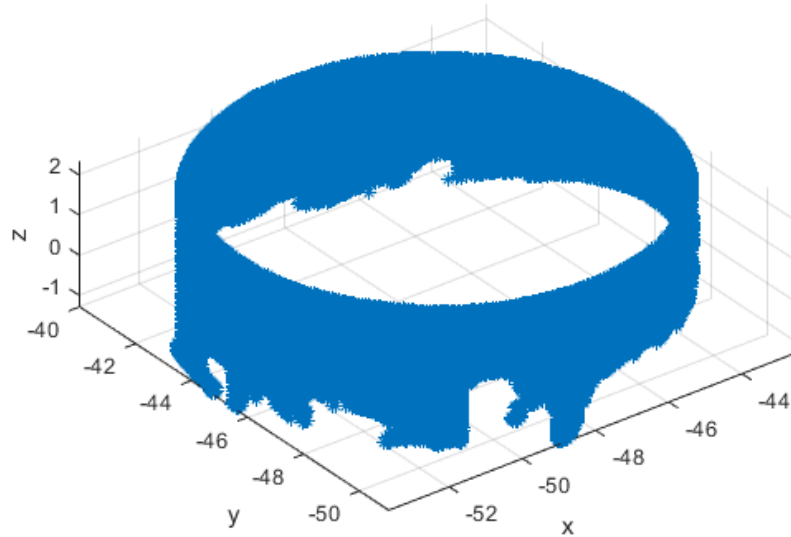
---

```
%https://se.mathworks.com/matlabcentral/fileexchange/5557-circle-fit

clear all
close all
clc

A=dlmread('Foundation.txt');      %%Reading the txt-file of the point cloud
a=A(:,[1,2,3]);                  %%Matrix of the geometrical data
x=a(:,1);
y=a(:,2);
z=a(:,3);

figure(1)
scatter3(x,y,z, '*')
hold on
xlabel('x')
ylabel('y')
zlabel('z')
grid on
axis('equal')
```



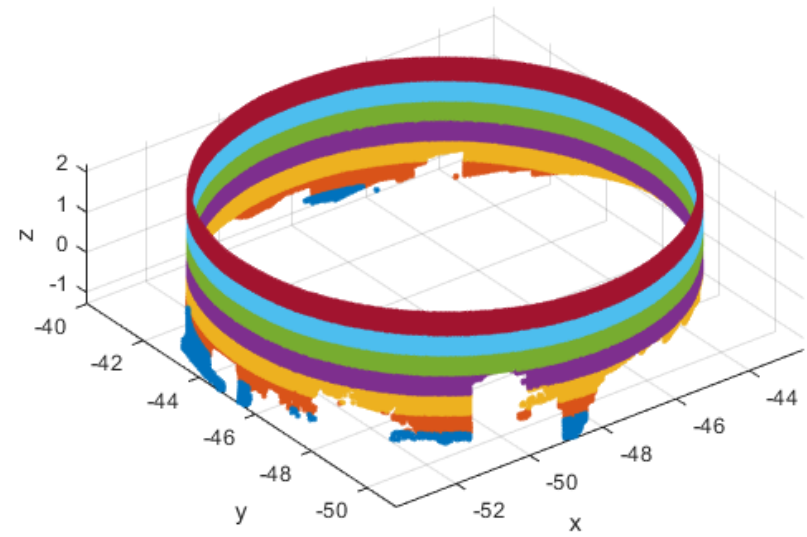
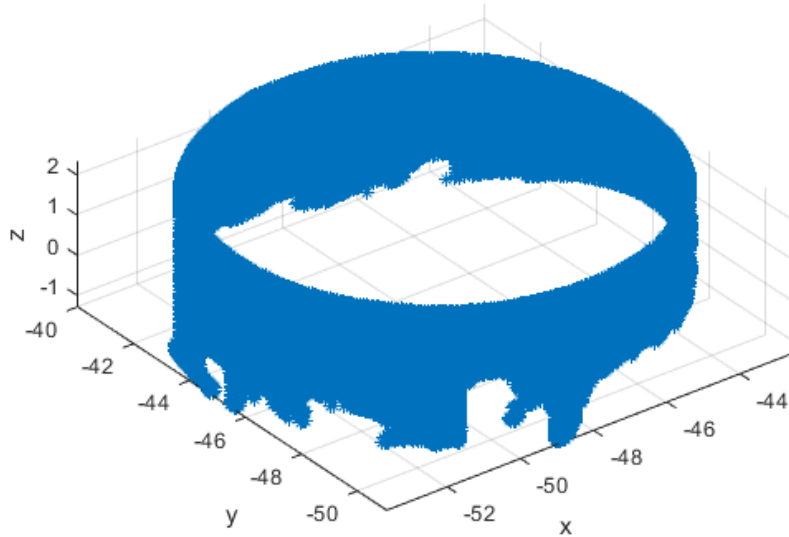
## Slice

```

%%Dividing the foundation into main segments
xRngs=[min(a(:,3)):0.5: max(a(:,3))];
for ind = 1:numel(xRngs)-1

    rows = a(:,3)>=xRngs(ind) & a(:,3)<xRngs(ind+1); %check to see which rows of b are between
    %xRng(ind) and xRng(ind+1)
    segM(ind).SmallerM = a(rows,:);
end
for ind = 1:numel(xRngs)-1
figure(2)
scatter3(segM(ind).SmallerM(:,1),segM(ind).SmallerM(:,2),segM(ind).SmallerM(:,3),'.')    %%Scatter the foundation
hold on
xlabel('x')
ylabel('y')
zlabel('z')
grid on
axis('equal')
end

```



### Circlefit

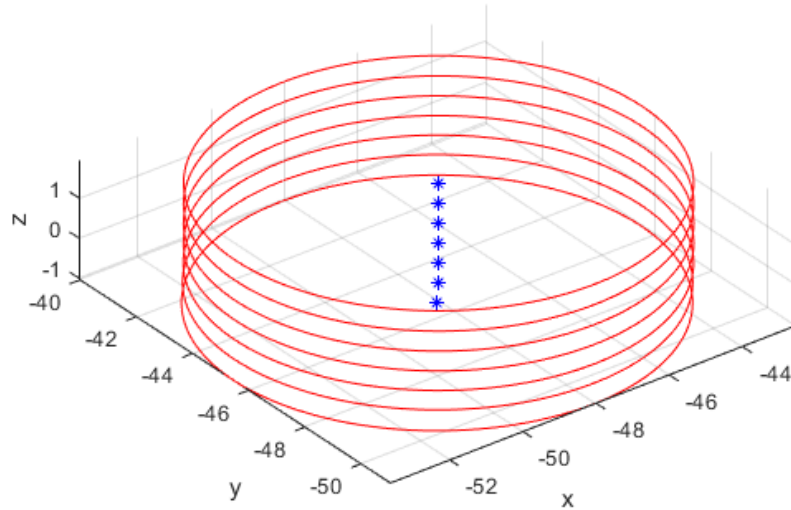
```

for ind = 1:numel(xRngs)-1
[xc(ind),yc(ind),R(ind),a]=circlefit(segM(ind).SmallerM(:,1),segM(ind).SmallerM(:,2));

Center(ind).smallerCenter=[xc(ind); yc(ind);mean(segM(ind).SmallerM(:,3))];

end
for ind = 1:numel(xRngs)-1
figure(3)
scatter3(Center(ind).smallerCenter(1),Center(ind).smallerCenter(2),Center(ind).smallerCenter(3),'*b')    %%Scatter the rotated pier
hold on
plotCircle3D([Center(ind).smallerCenter(1),Center(ind).smallerCenter(2),Center(ind).smallerCenter(3)],[0, 0, 1],R(ind))
hold on
xlabel('x')
ylabel('y')
zlabel('z')
grid on
axis('equal')
end
Max_z=max(z);
Min_z=min(z);

```



### Write Excel file

```
%transpose matrices for excel organization
xct=transpose(xc);
yct=transpose(yc);
Rt=transpose(R);

% %write an excel file of the characteristic values for the foundation
writematrix(xct(:, :)*10^3, 'Foundation_Characteristic_Values.xlsx', 'sheet', 1);
writematrix(yct(:, :)*10^3, 'Foundation_Characteristic_Values.xlsx', 'sheet', 2);
writematrix(Max_z(:, :)*10^3, 'Foundation_Characteristic_Values.xlsx', 'sheet', 3);
writematrix(Min_z(:, :)*10^3, 'Foundation_Characteristic_Values.xlsx', 'sheet', 4);
writematrix(Rt(:, :)*10^3, 'Foundation_Characteristic_Values.xlsx', 'sheet', 5);
```

%Instead of using the Z-values from segM, the dynamo-script will use Min\_z and Max\_z values divided into # amount of points as the z-values of segM are mean values and not the max and min values.

### Circlefit Function

```
function [xc,yc,R,a]=circlefit(x,y)
% CIRCLEFIT fits a circle in x,y plane
% (x-a)^2 + (y-b)^2 = R^2
% x^2+y^2+a(1)*x+a(2)*y+a(3)=0
```

```
n=length(x);
xx=x.*x;
yy=y.*y;
xy=x.*y;

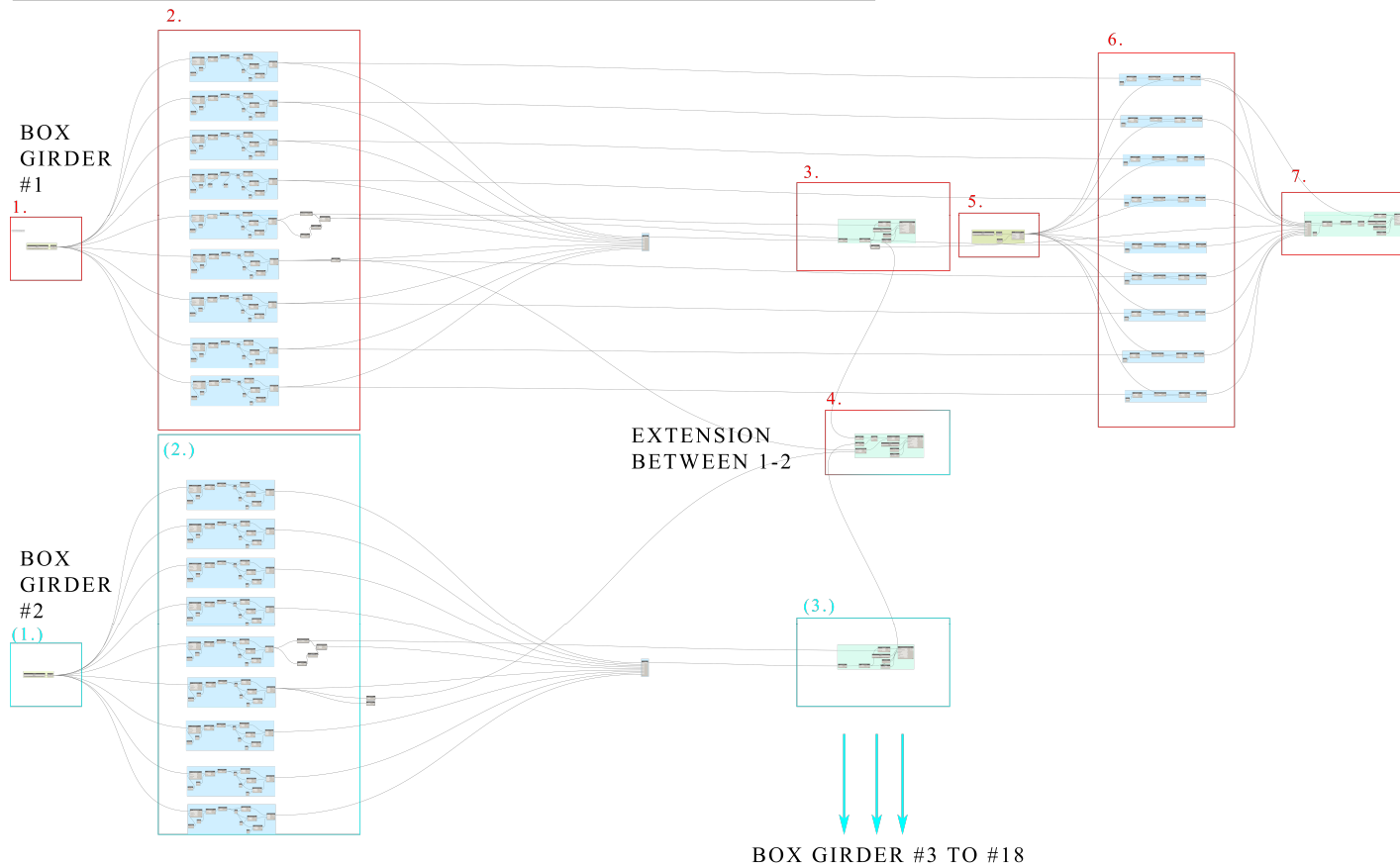
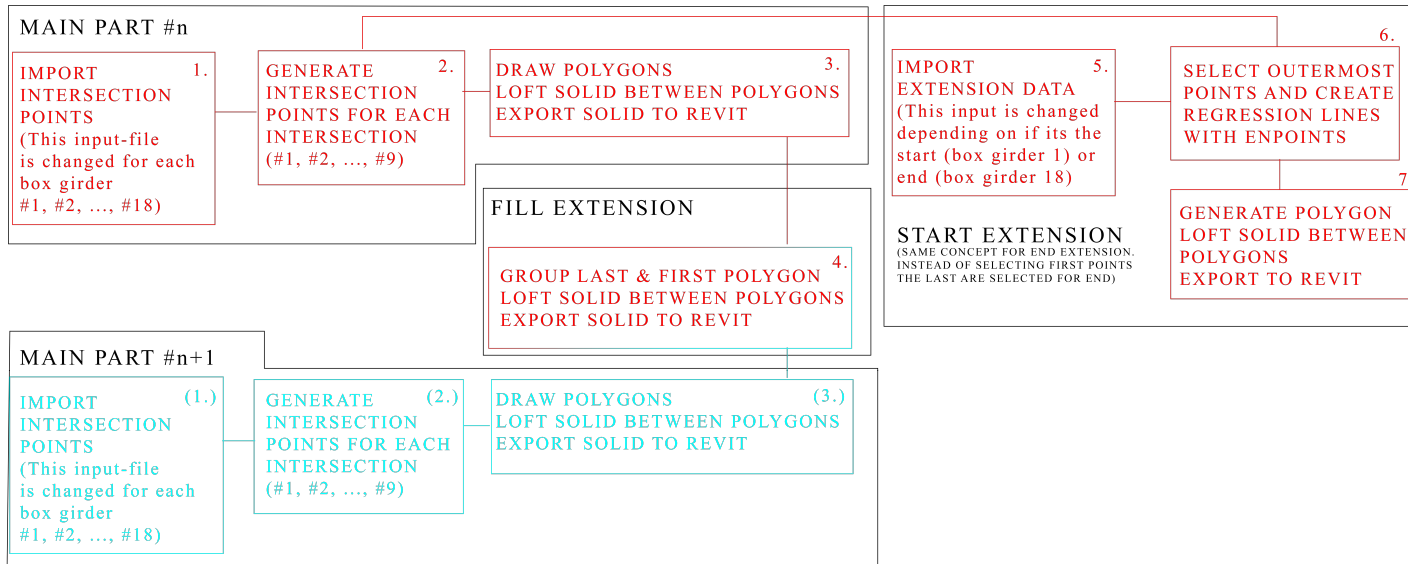
A=[sum(x) sum(y) n;sum(xy) sum(yy) sum(y);sum(xx) sum(xy) sum(x)];
B=[-sum(xx+yy);-sum(xx.*y+yy.*y);-sum(xx.*x+xy.*y)];
a=A\B;
xc = -0.5*a(1);
yc = -0.5*a(2);
R = sqrt(-(a(3)-xc^2-yc^2));
end
```

---

Published with MATLAB® R2019b

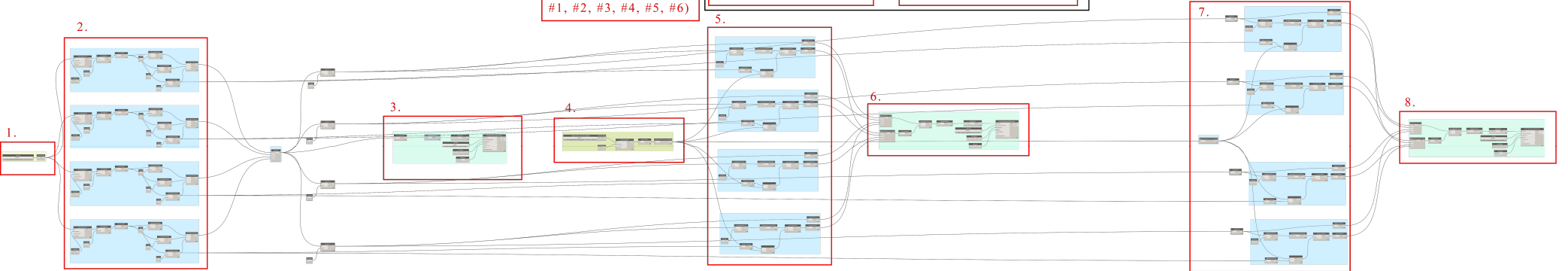
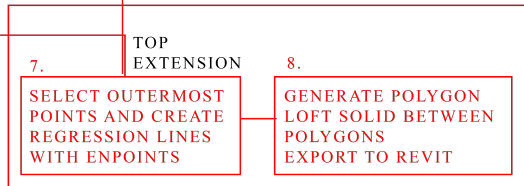
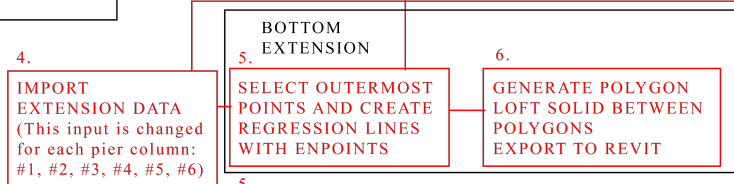
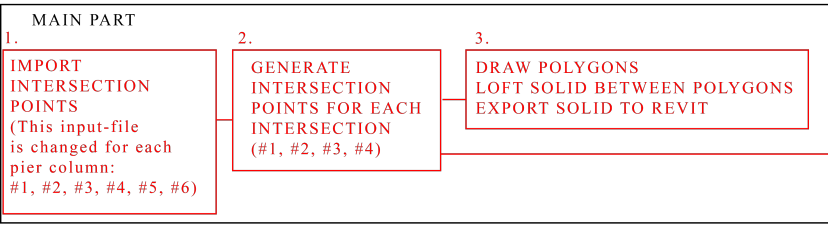
## **2 B: Dynamo Workspaces**

### **B1 Reverse Modelling of Box Girders (#1 to #18)**

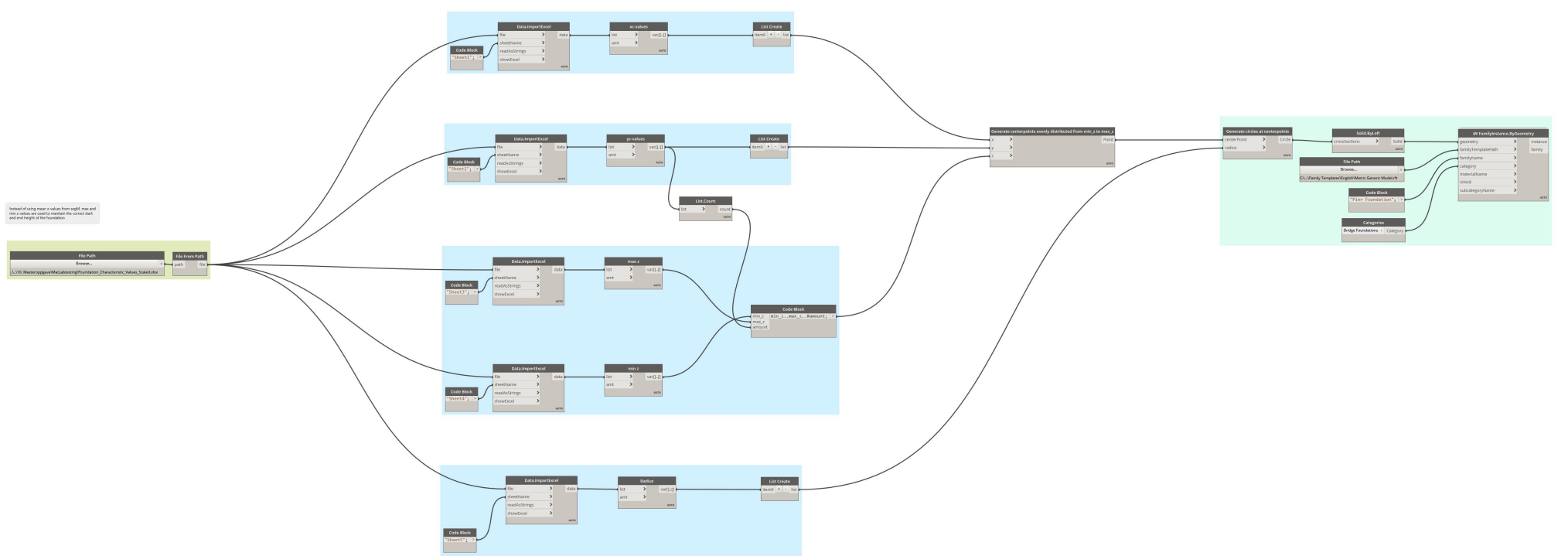
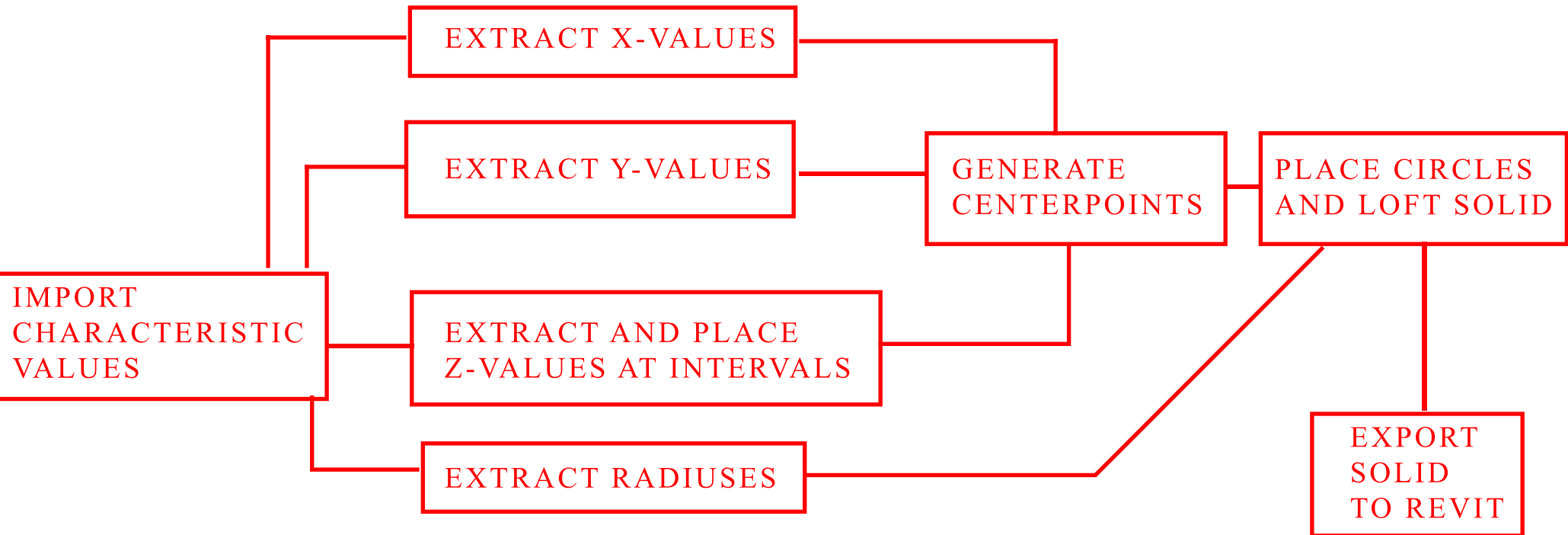




## **B2 Reverse Modelling of Pier Columns (#1 to #6)**



## **B3 Reverse Modelling of Pier Foundation**



### **3 C: Geomagic Control X Report**

#### **C1 Geomagic Control X Deviation Analysis Report**

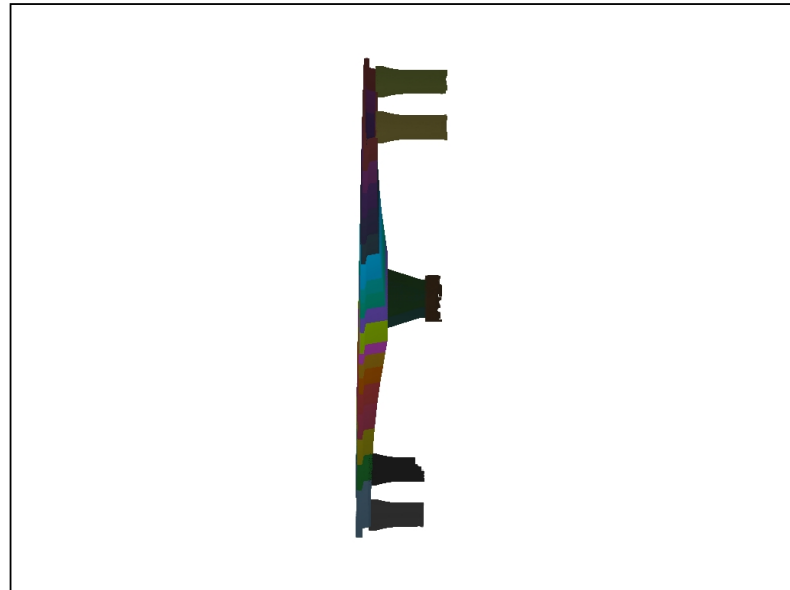
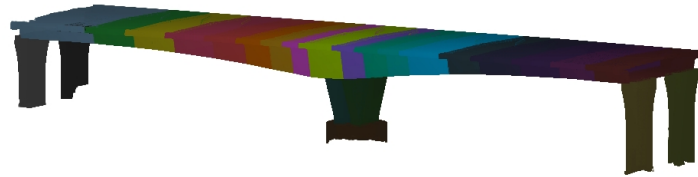


Product Name	Dade Bridge Accuracy analysis
Part Name	-
Part Number	-
Department	-
Inspector	-
Date	Jun 09, 2020
Unit	mm

**Disclaimer**

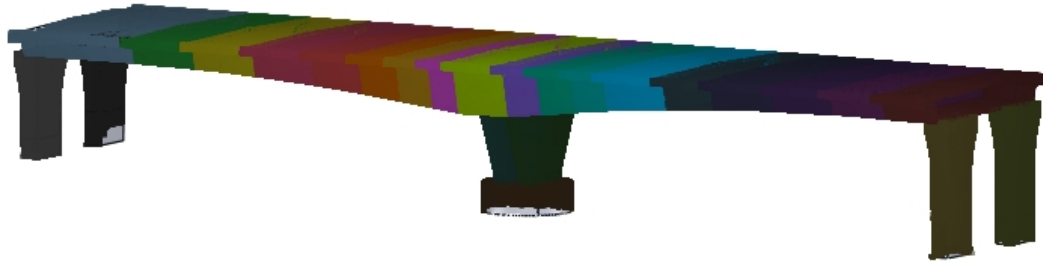
The results of this analysis and forecastings are believed to be reliable but are not to be construed as providing a warranty, including any warranty of merchantability or fitness for purpose, or representation for which 3D Systems, Inc. assumes legal responsibility. Users should undertake sufficient verification and iterative testing to determine the suitability of any information presented. Nothing herein is to be taken as permission, inducement or recommendation by 3D Systems, Inc. to practice any patented invention without a license or to in any way infringe upon the intellectual property rights of any other party.

# Measured Data



Product Name	Dade Bridge Accuracy	Department	-	Date	Jun 09, 2020
Part Name	-	Inspector	-	Unit	mm

# Alignment

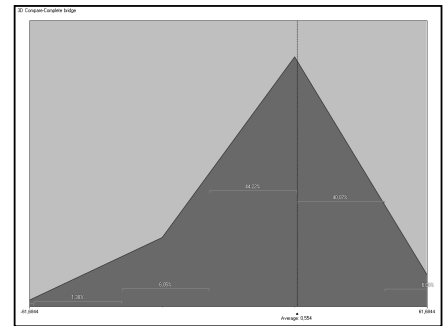
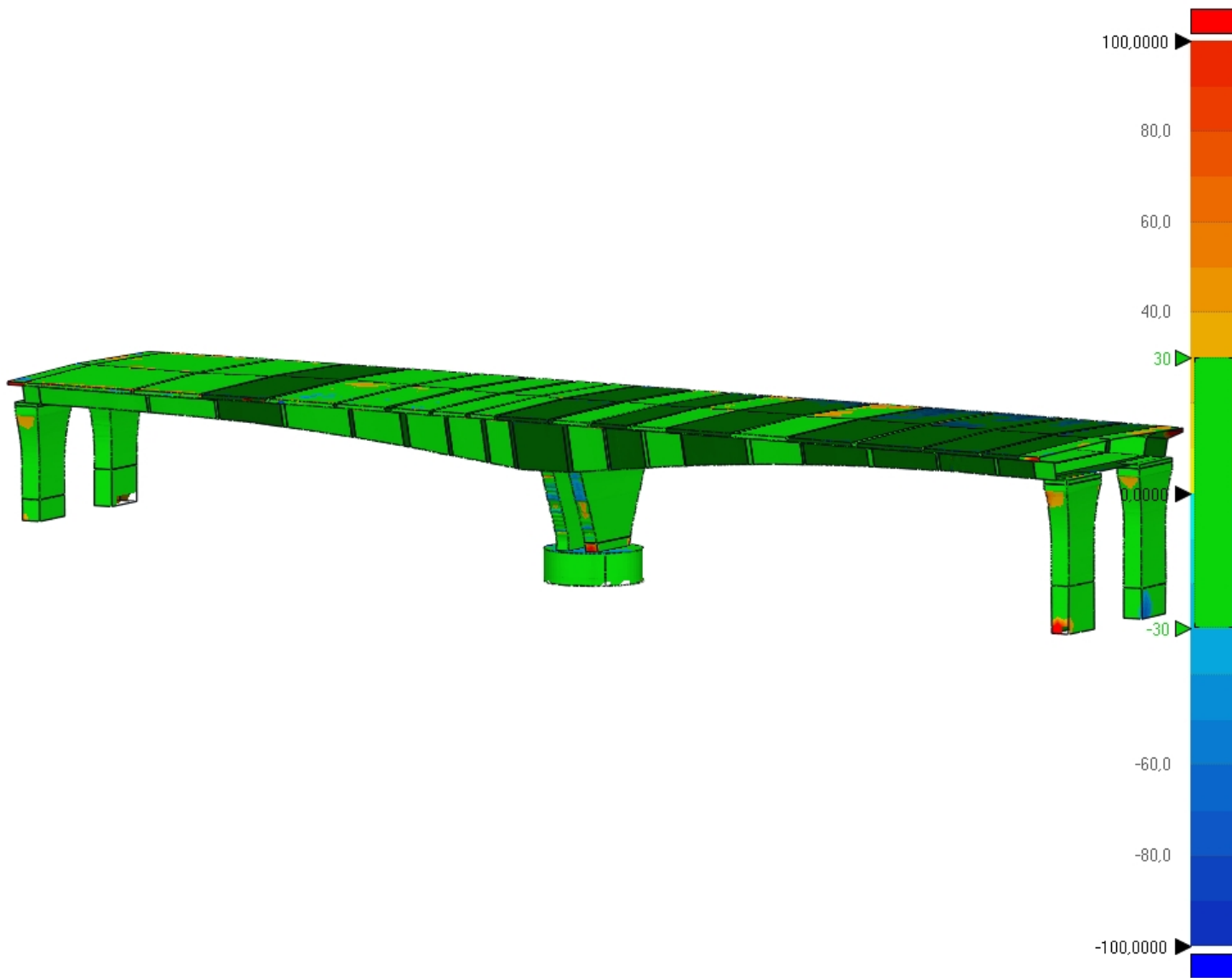


Min.	-29,7358
Max.	29,7356
Avg.	-0,0287
RMS	11,1079
Std. Dev.	11,1078
Var.	123,3836
+Avg.	8,5235
-Avg.	-8,6459

Product Name	Dade Bridge Accuracy	Department	-	Date	Jun 09, 2020
Part Name	-	Inspector	-	Unit	mm



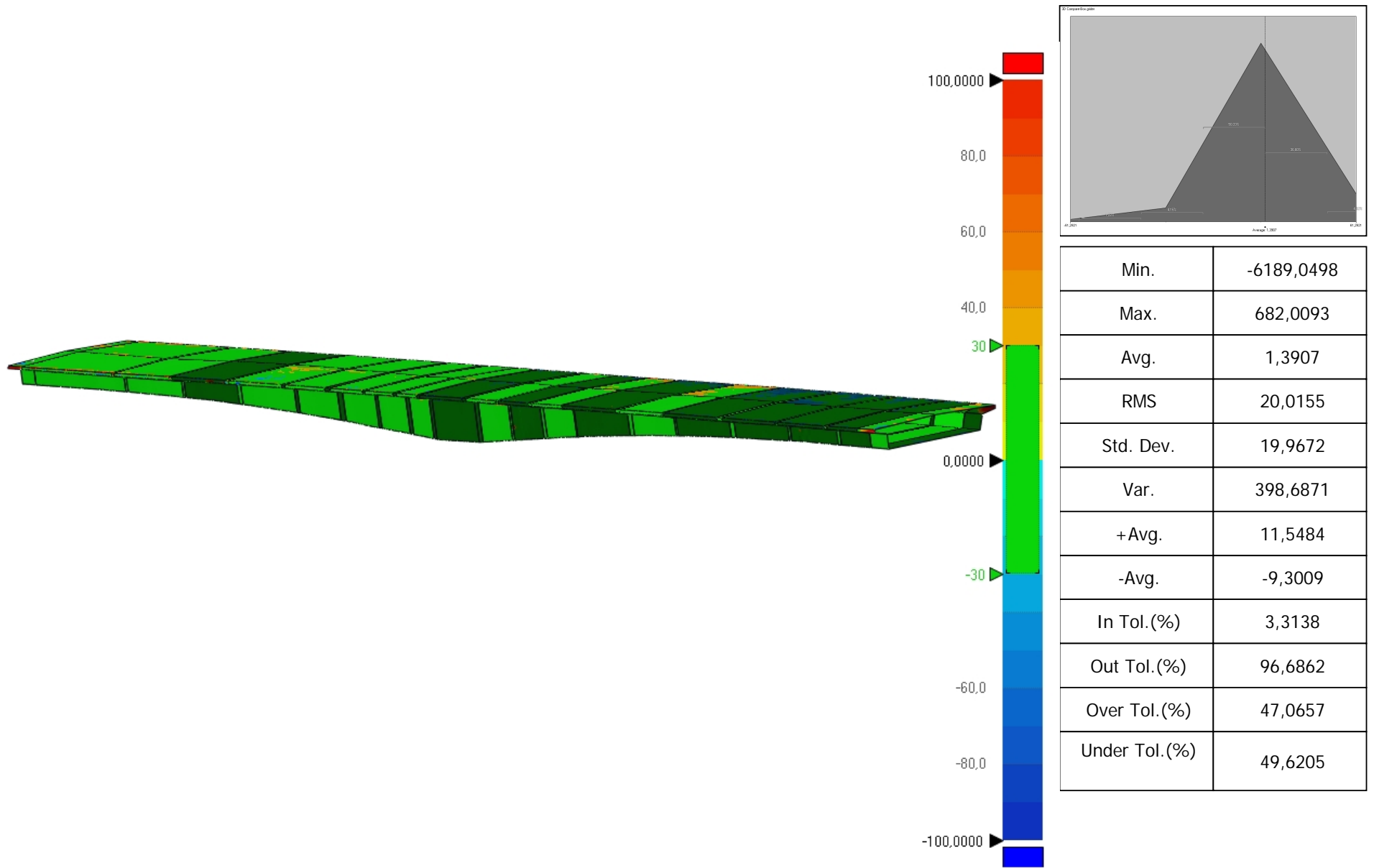
# 3D Compare-Complete bridge



Min.	-6177,3472
Max.	687,6968
Avg.	0,554
RMS	20,3843
Std. Dev.	20,3768
Var.	415,2142
+Avg.	12,3378
-Avg.	-11,4101
In Tol.(%)	2,5849
Out Tol.(%)	97,4151
Over Tol.(%)	48,332
Under Tol.(%)	49,0831

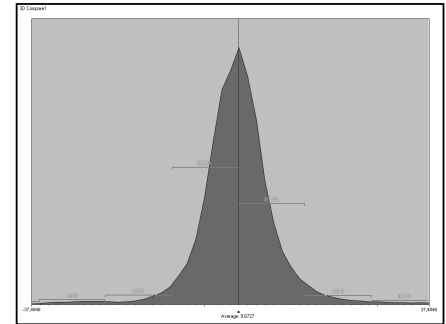
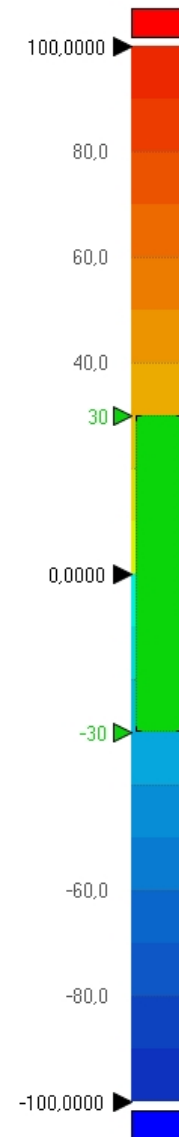
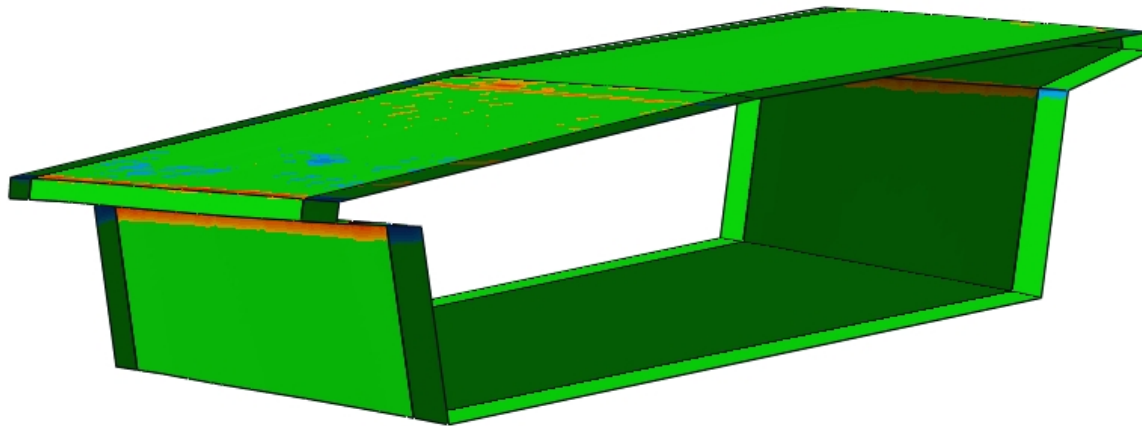
Product Name	Dade Bridge Accuracy analysis	Department	-	Date	Jun 09, 2020
Part Name	-	Inspector	-	Unit	mm

# 3D Compare-Box grider



Product Name	Dade Bridge Accuracy analysis	Department	-	Date	Jun 09, 2020
Part Name	-	Inspector	-	Unit	mm

# 3D Compare-Box5



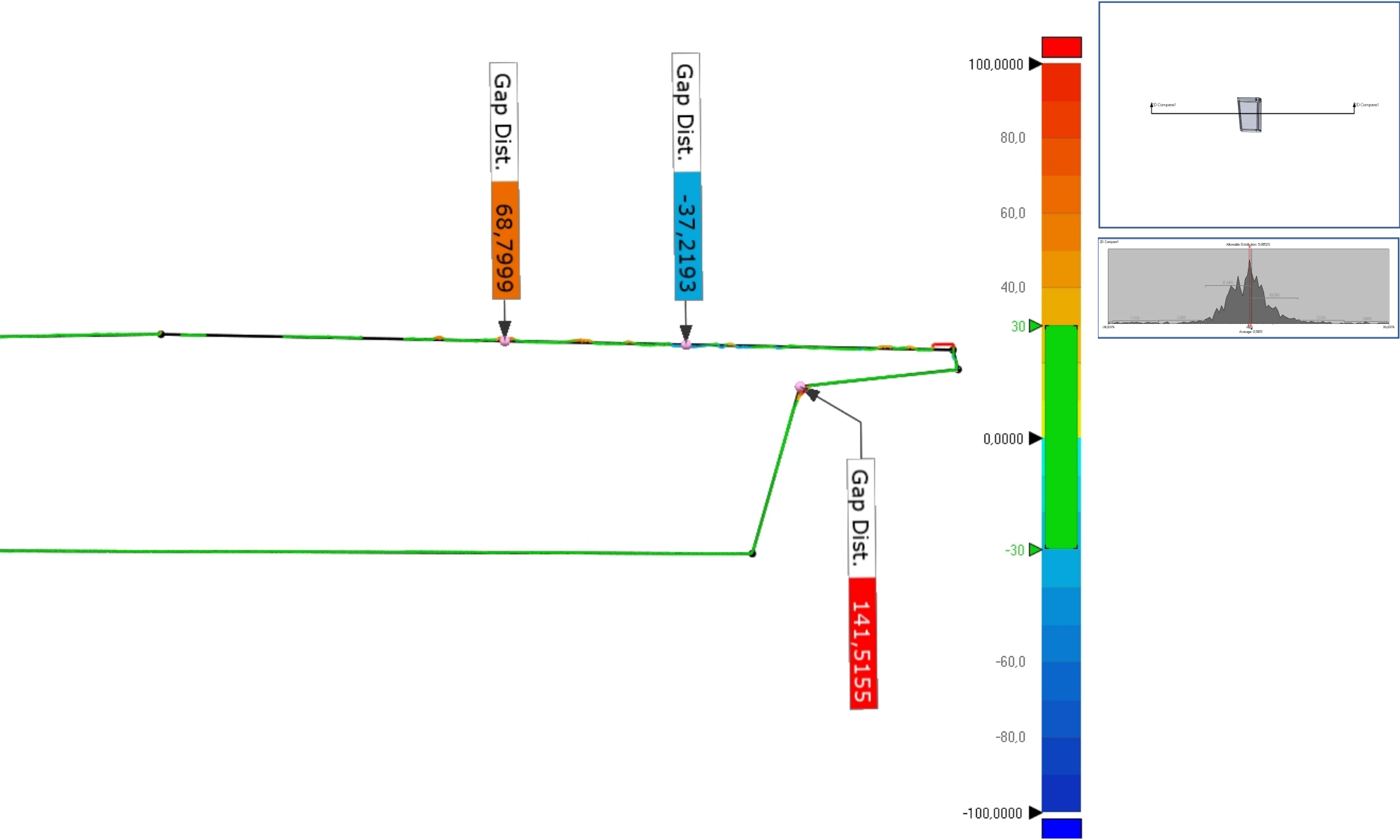
Min.	-260,3105
Max.	144,9484
Avg.	0,6727
RMS	12,2891
Std. Dev.	12,2706
Var.	150,5686
+Avg.	6,9769
-Avg.	-5,3051
In Tol.(%)	4,7721
Out Tol.(%)	95,2279
Over Tol.(%)	48,9375
Under Tol.(%)	46,2904

Product Name	Dade Bridge Accuracy analysis
Part Name	-

Department	-
Inspector	-

Date	Jun 09, 2020
Unit	mm

# 2D Compare-Box5



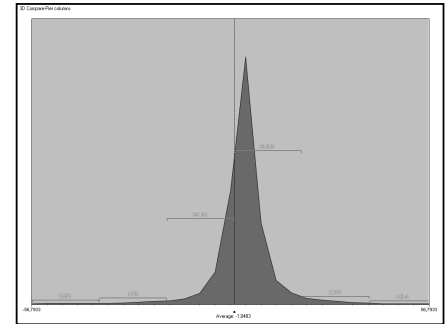
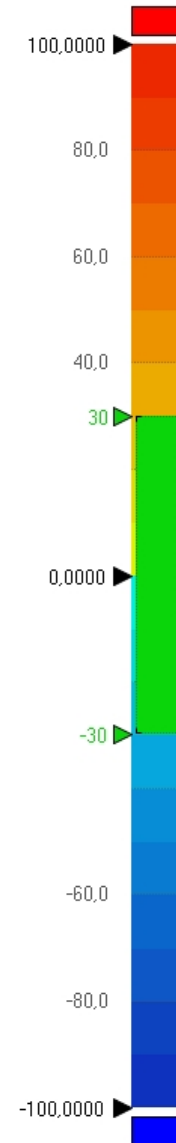
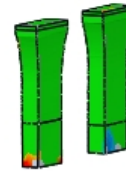
Product Name	Dade Bridge Accuracy analysis	Department	-	Date	Jun 09, 2020
Part Name	-	Inspector	-	Unit	mm

Min.	-46,2809
Max.	123,6717
Avg.	0,5651
RMS	13,103
Std. Dev.	13,0908
Var.	171,3697
+Avg.	7,9529
-Avg.	-5,7335
In Tol.(%)	5,8052
Out Tol.(%)	94,1948
Over Tol.(%)	43,3989
Under Tol.(%)	50,7959

Name	Reference Pos.		Measured Pos.		Gap Dist.	Tolerance
	X	Y	X	Y		
2D Compare1: 1	745,2543	-9999,9994	814,0192	-9997,8018	68,7999	±30
2D Compare1: 2	-332,5494	-3493,6664	-443,5502	-3405,8855	141,5155	±30
2D Compare1: 3	617,4514	-6000	580,2511	-6001,1885	-37,2193	±30
Min.	-332,5494	-9999,9994	-443,5502	-9997,8018	-37,2193	
Max.	745,2543	-3493,6664	814,0192	-3405,8855	141,5155	

Product Name	Dade Bridge Accuracy	Department	-	Date	Jun 09, 2020
Part Name	-	Inspector	-	Unit	mm

# 3D Compare-Pier columns



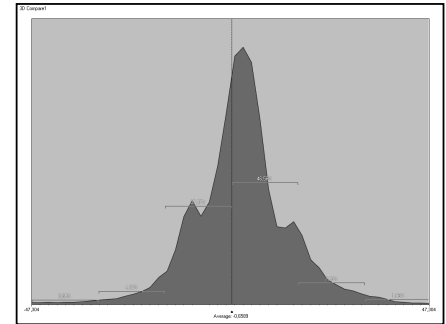
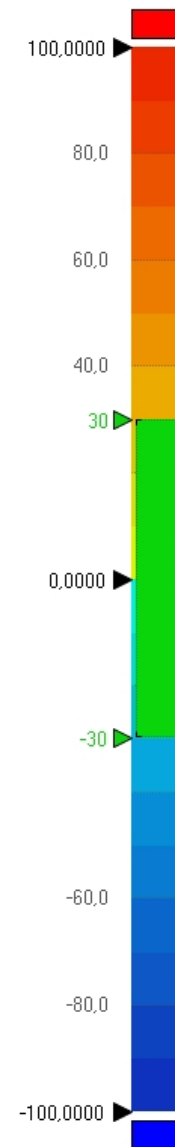
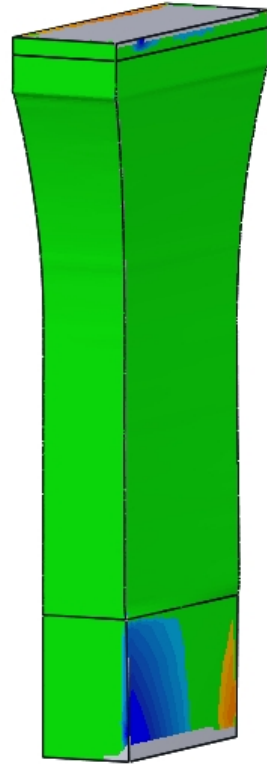
Min.	-186,2981
Max.	865,0736
Avg.	-1,0483
RMS	18,6112
Std. Dev.	18,5816
Var.	345,2774
+Avg.	6,5787
-Avg.	-8,6276
In Tol.(%)	7,5779
Out Tol.(%)	92,4221
Over Tol.(%)	46,3234
Under Tol.(%)	46,0987

Product Name	Dade Bridge Accuracy analysis
Part Name	-

Department	-
Inspector	-

Date	Jun 09, 2020
Unit	mm

# 3D Compare-Pier2



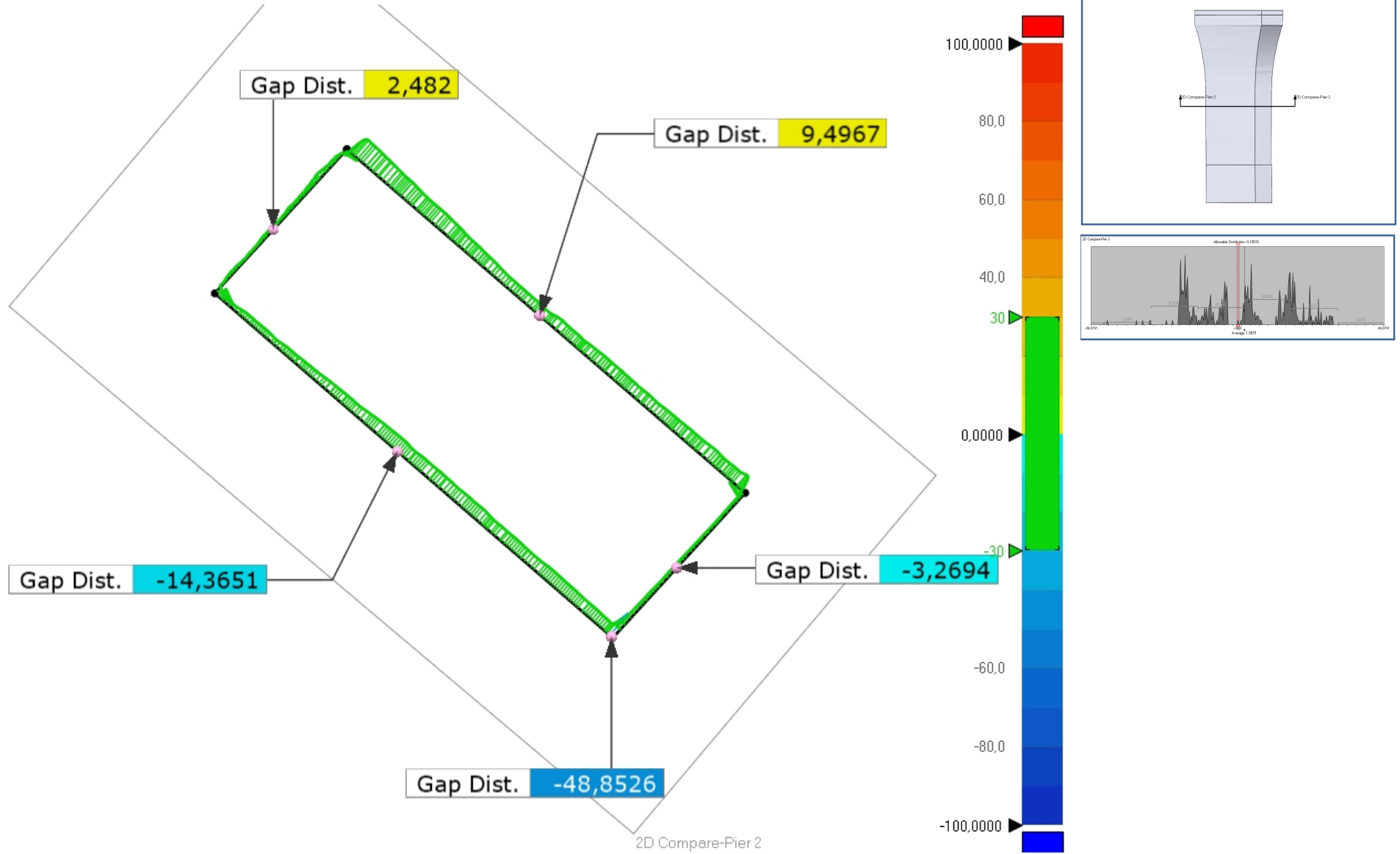
Min.	-170,5959
Max.	325,6458
Avg.	-0,6589
RMS	15,5623
Std. Dev.	15,5483
Var.	241,7512
+Avg.	8,4657
-Avg.	-10,5767
In Tol.(%)	3,1366
Out Tol.(%)	96,8634
Over Tol.(%)	46,3501
Under Tol.(%)	50,5133

Product Name	Dade Bridge Accuracy analysis
Part Name	-

Department	-
Inspector	-

Date	Jun 09, 2020
Unit	mm

# 2D Compare-Pier 2



Product Name	Dade Bridge Accuracy analysis	Department	-	Date	Jun 09, 2020
Part Name	-	Inspector	-	Unit	mm

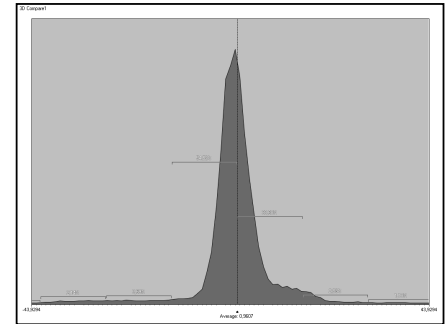
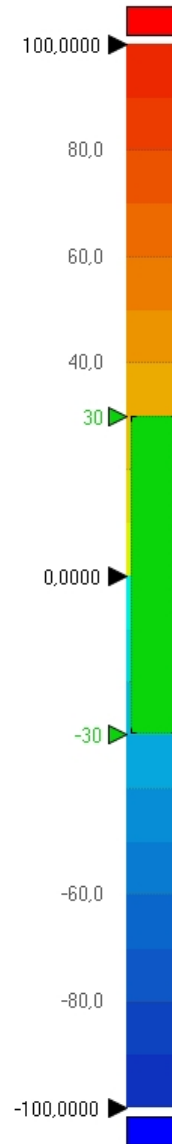
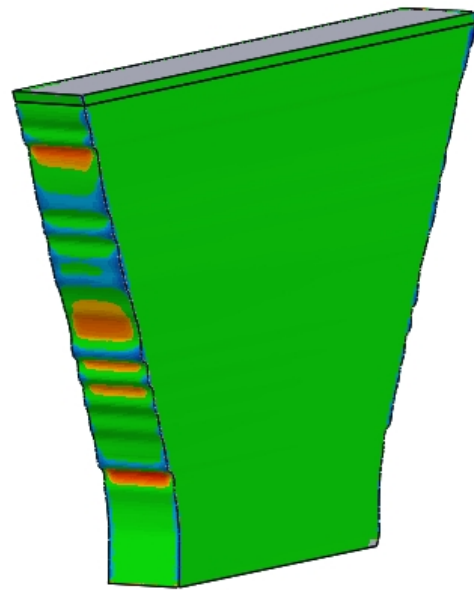


Min.	-32,0471
Max.	23,5459
Avg.	1,5875
RMS	11,605
Std. Dev.	11,4959
Var.	132,1551
+Avg.	10,6179
-Avg.	-9,5724
In Tol.(%)	0,1953
Out Tol.(%)	99,8047
Over Tol.(%)	55,0781
Under Tol.(%)	44,7266

Name	Reference Pos.		Measured Pos.		Gap Dist.	Tolerance
	X	Y	X	Y		
2D Compare-Pier 2: 1	-9593,1214	5827,6311	-9577,4922	5873,916	-48,8526	±30
2D Compare-Pier 2: 2	-11998,2906	7908,0103	-11988,8936	7918,875	-14,3651	±30
2D Compare-Pier 2: 3	-13392,1823	10398,6553	-13394,0107	10400,334	2,482	±30
2D Compare-Pier 2: 4	-10398,838	9432,4886	-10392,6377	9439,6816	9,4967	±30
2D Compare-Pier 2: 5	-8860,4252	6600,0005	-8862,8369	6602,2075	-3,2694	±30
Min.	-13392,1823	5827,6311	-13394,0107	5873,9160	-48,8526	
Max.	-8860,4252	10398,6553	-8862,8369	10400,3340	9,4967	

Product Name	Dade Bridge Accuracy	Department	-	Date	Jun 09, 2020
Part Name	-	Inspector	-	Unit	mm

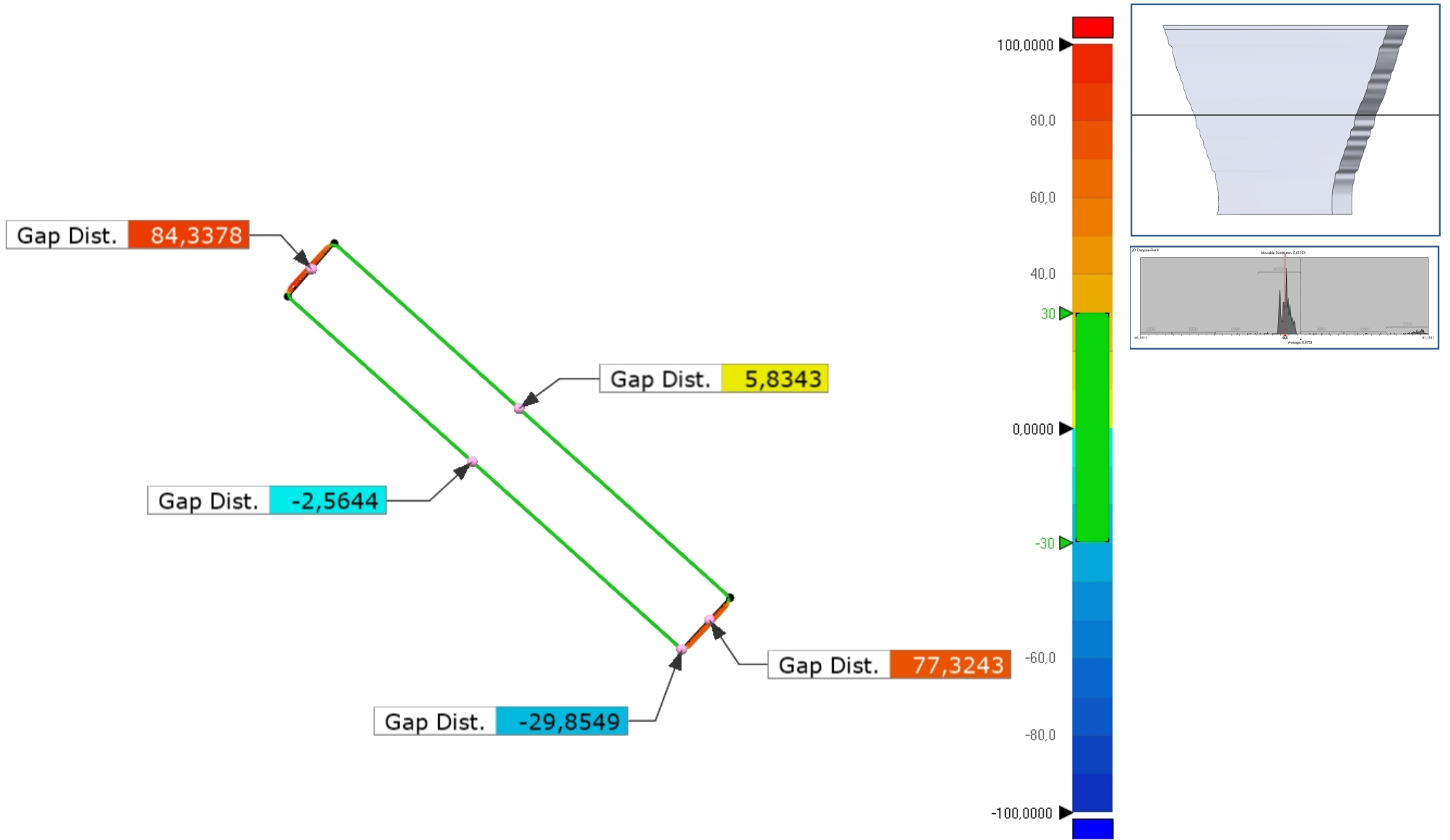
# 3D Compare-Pier4



Min.	-124,9442
Max.	138,2508
Avg.	0,9607
RMS	14,3551
Std. Dev.	14,3229
Var.	205,1451
+ Avg.	7,8885
- Avg.	-5,9988
In Tol.(%)	6,5985
Out Tol.(%)	93,4015
Over Tol.(%)	46,5356
Under Tol.(%)	46,8659

Date	Jun 09, 2020
Unit	mm

# 2D Compare-Pier 4



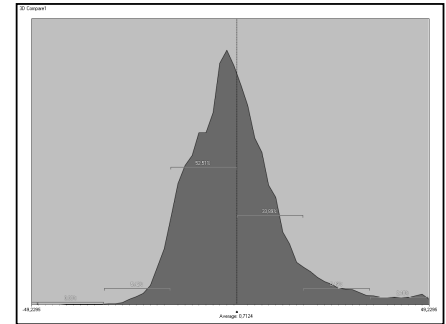
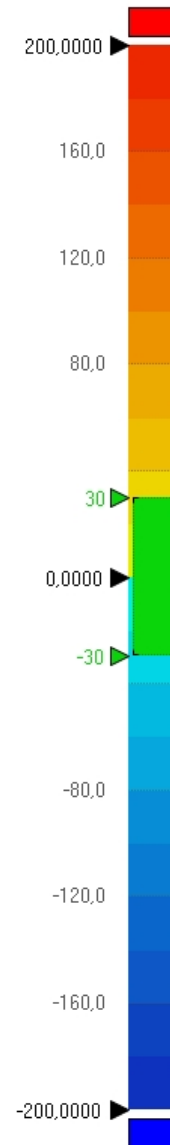
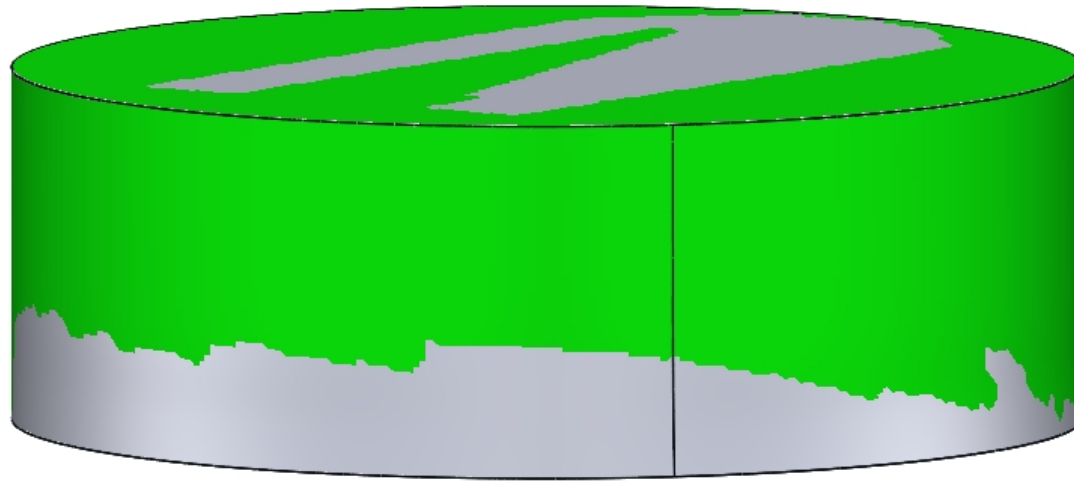
Product Name	Dade Bridge Accuracy analysis	Department	-	Date	Jun 09, 2020
Part Name	-	Inspector	-	Unit	mm

Min.	-26,9352
Max.	91,4519
Avg.	8,8759
RMS	25,6748
Std. Dev.	24,0917
Var.	580,4116
+Avg.	15,0298
-Avg.	-2,4222
In Tol.(%)	6,6116
Out Tol.(%)	93,3884
Over Tol.(%)	61,9835
Under Tol.(%)	31,405

Name	Reference Pos.		Measured Pos.		Gap Dist.	Tolerance
	X	Y	X	Y		
2D Compare-Pier 4: 1	-49000,003	-45980,567	-48996,1133	-45976,2187	5,8343	±30
2D Compare-Pier 4: 2	-50000,0031	-47115,4806	-49998,293	-47113,5703	-2,5644	±30
2D Compare-Pier 4: 3	-44930,8001	-50500,0031	-44874,2305	-50552,7187	77,3243	±30
2D Compare-Pier 4: 4	-53432,2122	-43000,0026	-53495,707	-42944,4961	84,3378	±30
2D Compare-Pier 4: 5	-45520	-51129,3861	-45525,2227	-51099,9922	-29,8549	±30
Min.	-53432,2122	-51129,3861	-53495,7070	-51099,9922	-29,8549	
Max.	-44930,8001	-43000,0026	-44874,2305	-42944,4961	84,3378	

Product Name	Dade Bridge Accuracy	Department	-	Date	Jun 09, 2020
Part Name	-	Inspector	-	Unit	mm

# 3D Compare-Foundation



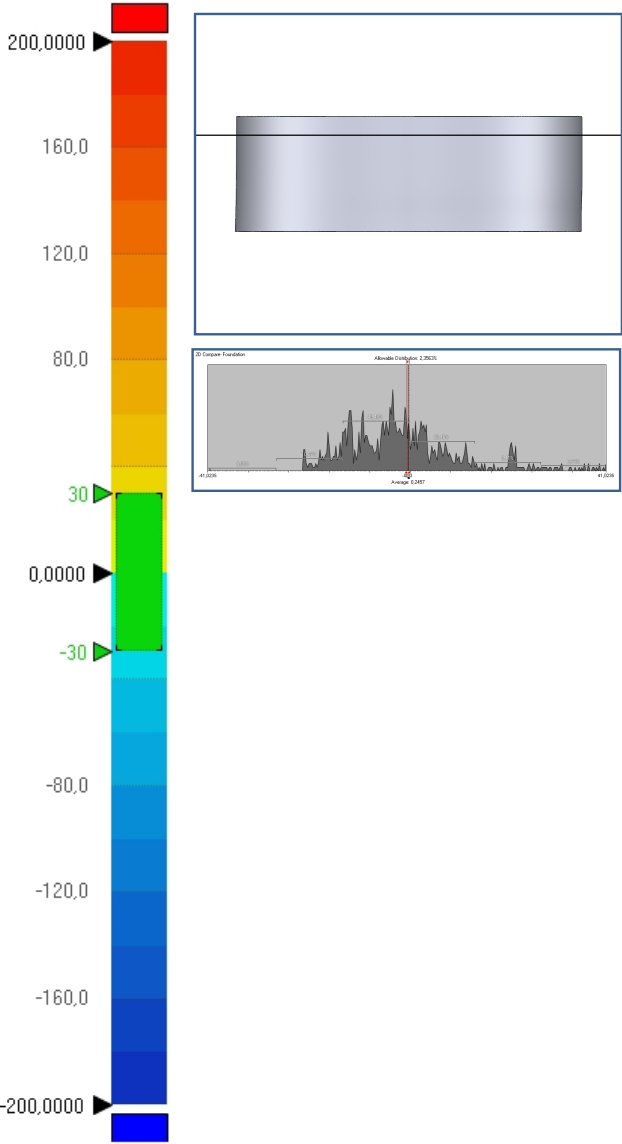
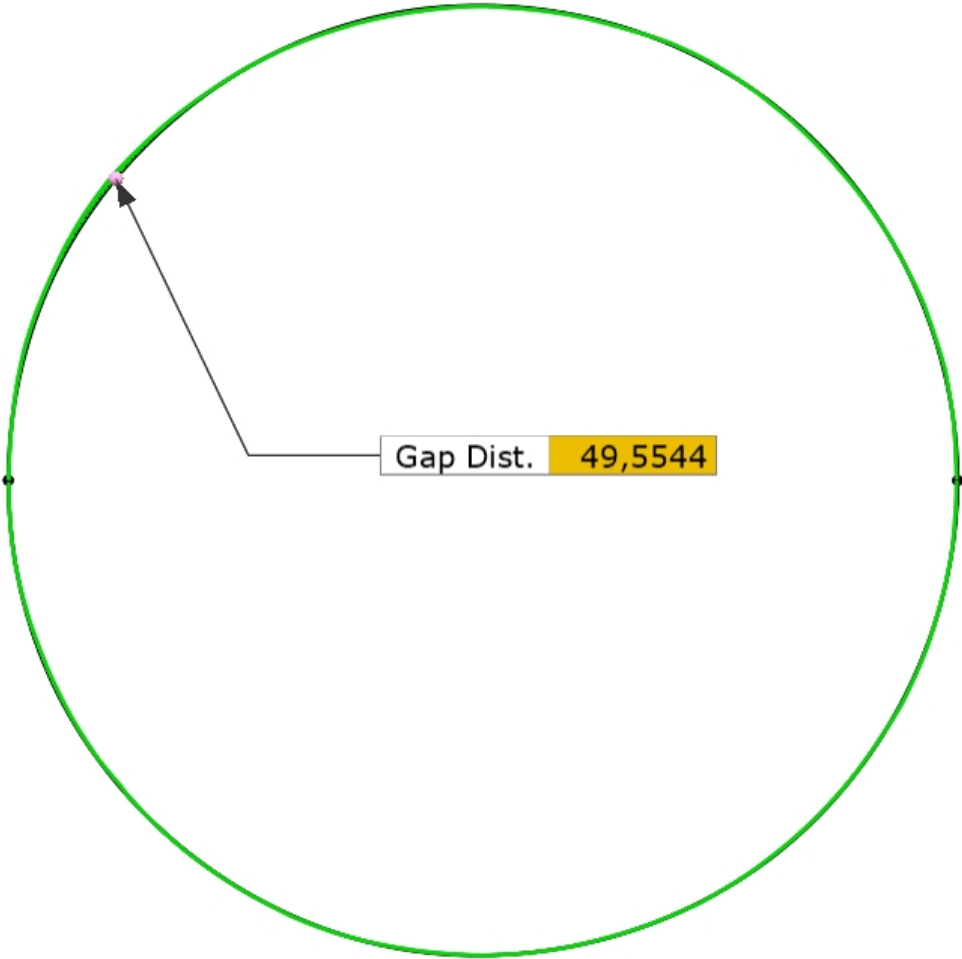
Min.	-60,0436
Max.	370,3655
Avg.	0,7124
RMS	16,188
Std. Dev.	16,1724
Var.	261,5452
+Avg.	11,1199
-Avg.	-7,7949
In Tol.(%)	2,5506
Out Tol.(%)	97,4494
Over Tol.(%)	53,7427
Under Tol.(%)	43,7067

Product Name	Dade Bridge Accuracy analysis
Part Name	-

Department	-
Inspector	-

Date	Jun 09, 2020
Unit	mm

# 2D Compare- Foundation



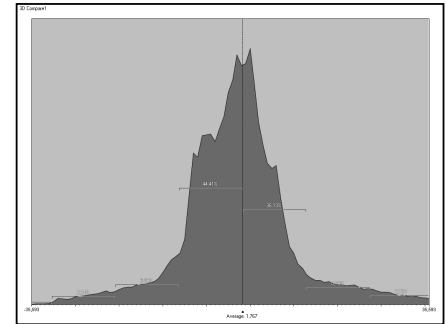
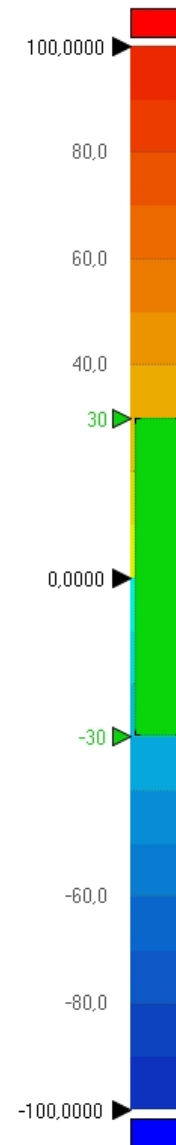
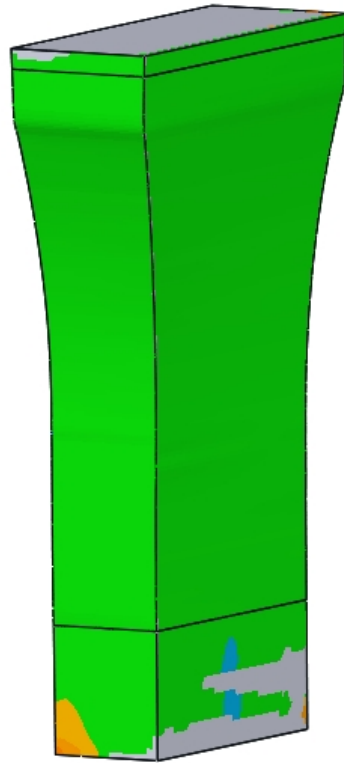
Product Name	Dade Bridge Accuracy analysis	Department	-	Date	Jun 09, 2020
Part Name	-	Inspector	-	Unit	mm

Min.	-21,3331
Max.	49,5665
Avg.	0,2457
RMS	13,5948
Std. Dev.	13,5926
Var.	184,7592
+Avg.	12,198
-Avg.	-8,0598
In Tol.(%)	2,3563
Out Tol.(%)	97,6437
Over Tol.(%)	40,1508
Under Tol.(%)	57,4929

Name	Reference Pos.		Measured Pos.		Gap Dist.	Tolerance
	X	Y	X	Y		
2D Compare- Foundation: 1	-52317,1469	-42000	-52355,2734	-41968,3437	49,5544	±100
Min.	-52317,1469	-42000,0000	-52355,2734	-41968,3437	49,5544	
Max.	-52317,1469	-42000,0000	-52355,2734	-41968,3437	49,5544	

Product Name	Dade Bridge Accuracy	Department	-	Date	Jun 09, 2020
Part Name	-	Inspector	-	Unit	mm

# 3D Compare-pier5



Min.	-81,4232
Max.	116,4729
Avg.	1,767
RMS	11,413
Std. Dev.	11,2753
Var.	127,1334
+Avg.	8,3505
-Avg.	-6,7622
In Tol.(%)	3,1748
Out Tol.(%)	96,8252
Over Tol.(%)	41,9569
Under Tol.(%)	54,8683

Product Name	[Product Name]
Part Name	[Part Name]

Department	[Department]
Inspector	[Inspector]

Date	Jun 10, 2020
Unit	mm