Development of Synthetic Point Cloud Data and Deep Learning Algorithms for Automatic Segmentation and Defect Detection in Masonry Bridges



Yixiong Jing

Wolfson College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Hilary 2025

Abstract

Masonry arch bridges constitute a significant portion of the European bridge stock, with many bridges being over 100 years old. These ageing bridges require periodic site inspection to identify potential structural degradations such as cracks, water leakage, and material degradation to ensure their safety and functionality. However, defect inspections on ageing masonry bridges are often challenging due to their varied geometries and the absence of detailed construction records. Traditional manual defect inspection methods, which typically involve visual assessments conducted by trained inspectors to identify visible defects (e.g., cracks, water leakage, etc.), are time-consuming, subjective, and prone to errors. This research proposes a novel framework that leverages point clouds and deep learning to automate defect inspections of masonry bridges.

To enable structural analysis and defect inspections, it is necessary to segment individual components from masonry bridge point clouds. Therefore, a 3D deep learning neural network called BridgeNet is developed to perform automatic segmentation, substituting the previously labour-intensive manual process. BridgeNet is trained on a synthetic dataset (e.g., generated to address the scarcity of the labelled real-world data), whilst being tested on real masonry bridge point clouds. BridgeNet achieves state-of-the-art (SOTA) performance, demonstrating the effectiveness of synthetic data for training and the efficiency of its architecture.

Building on BridgeNet, a Transformer-based architecture called BridgeNetv2 is introduced. BridgeNetv2 incorporates a Local Geo-Transformer to leverage the attention mechanism on local regions and a Global Geo-Transformer to capture global relationships within the point cloud. BridgeNetv2, which is also trained on the synthetic dataset, surpasses BridgeNet in real masonry bridge point clouds in all metrics. Moreover, BridgeNetv2 inherits the lightweight and memory-efficient

features of BridgeNet, making it ideal for large-scale civil engineering applications.

Subsequently, the research explores automating defect inspections in cracks from the point cloud of individual components by leveraging geometric distortions. *PatchCore*, an image-based anomaly detection method, is adapted for 3D point clouds. Initially, Fast Point Feature Histograms (FPFH) are employed to extract geometric features. *PatchCore* is then applied to synthetic point clouds with labelled cracks, which are generated using 3D finite element modelling and graphical modelling. Unlike traditional image-based methods that only detect surface cracks, the proposed method successfully identifies both surface and internal cracks in arches. However, it is noticed that this method encounters difficulties in detecting small curvature changes and in-plane geometric distortions.

To address these limitations, this research introduces a novel 3D multimodal feature, 3DMulti-FPFHI, that fuses geometric and intensity features to enhance defect inspections by improving accuracy and enabling the detection of non-geometric defect types (e.g., water leakage). The new feature is integrated into the *PatchCore* anomaly detection framework. Apart from testing on synthetic point clouds, this method is validated on point clouds from real masonry bridge arches and a full-scale experimental tunnel model to demonstrate the generalizability of the proposed method. Results show that incorporating intensity information significantly enhances detection performance compared to using FPFH alone, particularly in cases where FPFH fails. Additionally, the 3DMulti-FPFHI effectively captures water leakage in masonry bridge arches. The proposed framework offers a scalable and adaptable solution for automating ageing civil infrastructure defect inspections.

This thesis is submitted to the Department of Engineering Science, The University of Oxford, in fulfilment of the requirements for the degree of Doctor of Philosophy. This thesis is entirely my own work, and except where otherwise stated, describes my own research.

Yixiong Jing, Jan 2025.

Acknowledgement

I would like to express my deepest gratitude to my supervisors, Professor Sinan Acikgoz and Professor Brian Sheil, for their invaluable mentorship and patient guidance throughout my research. Their expertise, dedication, and encouragement have been instrumental in shaping this work and inspiring me to strive for excellence.

I am also sincerely thankful to my colleague Jiaxin Zhong, whose insightful suggestions were crucial in helping me address some of the intricate challenges in my work. My heartfelt appreciation also extends to my colleague Lin Wei, who generously provided the data essential for this research.

I owe a special debt of gratitude to my parents, Zemin Jing and Yuzhong Liu, for their unwavering financial support and belief in me throughout my studies. To my grandmother, Xinlan Liu, your constant encouragement and emotional support have been a source of immense strength and comfort. I am equally grateful to my girlfriend, Furong Jin, for her unwavering love and understanding, which have been a steady and inspiring presence during this journey.

I would also like to acknowledge my friends, both at Oxford and beyond, for their companionship and encouragement. While it is impossible to name everyone here, I am especially grateful to Kathy Wen, Markella Zormpa, Hardik Parwana, Luis Angel, Hanyu Hu, Ziyao Zhang, Xinyu Liu, Yuhang He, Tao Li, Yimin Hu, and Kun Ni. Your friendship and support have helped me navigate through some of the hardest times.

Finally, I extend my heartfelt thanks to anyone who has contributed to this journey, knowingly or unknowingly. This work would not have been possible without all of you.

Contents

1 Introduction **9**

1.1 Background 9

1.2 Research objectives 16

1.3 Outline of the thesis and contributions 17

2 Literature review **19**

2.1 Segmentation of large-scale point cloud of infrastructures with 3D DL 19

2.1.1 Point cloud representation 19

2.1.2 Semantic segmentation 21

2.1.3 Instance segmentation 33

2.1.4 3D point cloud synthetic dataset 35

2.1.5 3D point cloud segmentation in Infrastructure and their challenges 38

2.2 Defect inspection in infrastructure: classification, localization, and quantification 41

2.2.1 Supervised DL methods for defect detection in infrastructure 42

2.2.2 Overview of unsupervised learning for anomaly detection . . 46

2.2.3 Anomaly detection for defect inspection on infrastructure . . 56

2.3 Summary 57

3 Segmentation of masonry arch bridge point clouds 60

3.1 Introduction . . . 63

3.2 BridgeNet . . . 66

3.2.1 Residual Feature Encoder . . . 68

3.2.2 Global Feature Extraction . . . 70

3.3 Dataset . . . 70

3.3.1 Real dataset . . . 71

3.3.2 Synthetic dataset . . . 73

3.4 Training results . . . 79

3.4.1 Implementation details . . . 79

3.4.2 Evaluation metrics . . . 80

3.4.3 Semantic segmentation results . . . 81

3.5 Post-processing . . . 84

3.5.1 Instance segmentation . . . 84

3.5.2 Geometric feature extraction . . . 84

3.6 Conclusion . . . 86

4 Transformer-based segmentation of masonry arch bridge point clouds 89

4.1 Introduction . . . 92

4.2 Methodology . . . 96

4.2.1 Local Geo-Transformer . . . 97

4.2.2 Multi-head Local Geo-Attention . . . 99

4.2.3 Local Geo-Embedding . . . 100

4.2.4 Global Geo-Transformer . . . 102

4.3 Experiments and Results . . . 102

4.3.1 Implemention details . . . 102

4.3.2 Semantic segmentation results . . . 103

4.3.3 Ablation tests . . . 104
4.3.4 Efficiency analysis . . . 106
4.4 Concluding remarks . . . 107

5 Anomaly detection of cracks in synthetic bridge point cloud 110
5.1 Introduction . . . 113
5.2 Generation of synthetic data . . . 117
5.2.1 Simulation of the masonry arch model . . . 117
5.2.2 Synthetic labelled point cloud generation . . . 120
5.3 Methodology for anomaly detection . . . 121
5.3.1 FPFH computation and feature aggregation . . . 121
5.3.2 Coreset subsampling . . . 123
5.3.3 Anomaly detection . . . 124
5.4 Results . . . 125
5.4.1 Implementation details and accuracy metrics . . . 125
5.4.2 Anomaly detection on synthetic masonry arch data . . . 126
5.4.3 Anomaly detection on synthetic masonry arch bridge data . 134
5.5 Conclusions . . . 135

6 A 3D multimodal feature for infrastructure anomaly detection 139
6.1 Introduction . . . 142
6.2 Methodology . . . 146
6.2.1 Geometry and intensity feature extraction and aggregation . 147
6.2.2 2D feature extraction and aggregation for CPMF . . . 148
6.2.3 Anomaly detection method . . . 149
6.3 Datasets . . . 150
6.3.1 Synthetic masonry arch dataset . . . 151
6.3.2 Masonry arch dataset . . . 154
6.3.3 Concrete segmental tunnel dataset . . . 156

6.4 Experiments and results . . . 158
6.4.1 Implementation details and metrics . . . 158
6.4.2 Results for the synthetic arch . . . 159
6.4.3 Results for the real arch . . . 168
6.4.4 Results for the experimental tunnel model . . . 172

7 Conclusions, future Work, and dataset contributions 185
7.1 Conclusions . . . 185
7.2 Future works . . . 189
7.2.1 Towards generalized segmentation frameworks . . . 189
7.2.2 Enhancements on anomaly detections on infrastructures . . . 190
7.3 Data contribution . . . 191
7.3.1 BridgePoint-Seg dataset . . . 191
7.3.2 Crack3D-Defect dataset . . . 193

References 195

Chapter 1

Introduction

1.1 Background

Masonry bridges form a critical component of the UK's transportation infrastructure, with many dating back to the 18th and 19th centuries. As reported by Page [1996], approximately 40,000 masonry bridges remain in service across the British transportation network, representing 40% of the total bridge stock. It is crucial to maintain their structural integrity as masonry bridges play a crucial role in supporting economic activity and public transportation.

Masonry bridges are vulnerable to various structural defects that compromise their integrity and functionality. These defects can be broadly categorized into four main types: surface degradation, moisture penetration, distortions, and cracking [Keshmiry et al. 2024; Brackenbury et al. 2019]. Surface degradation, including spalling, mortar loss, and other delamination of brickwork, often arises from inadequate load capacities and environmental factors such as freeze-thaw cycles, high humidity, and flooding [Solla et al. 2013]. Moisture penetration can be also attributed to environmental factors, causing water leakage, vegetation growth, and material decay [Solla et al. 2013]. Distortions are represented as geometric irregularities in structural components (e.g., arches, spandrel walls, and parapets) which are typically caused by differential settlements and material degradation over time [Brackenbury et al. 2019].

Cracking is the most common defect type in masonry bridges [Keshmiry et al. 2024]. These cracks are often introduced by settlement and ground deformation

[Latifi et al. 2023; Alessandri et al. 2015]. These cracks are observed in varied patterns on surfaces of structural components, such as longitudinal cracks near arch barrels or wing walls, transverse cracks in quarter-span regions, and diagonal cracks across arches and piers [Brackenbury et al. 2019]. Cracks can also propagate internally, causing damage that is not visible on the surface while significantly weakening the structure [Keshmiry et al. 2024].

Masonry bridges have been subjected to increasing traffic loading (e.g., estimated 36% of increments in the total number of long-distance road and rail trips per person [Consultation 2011] from 2008 to 2043) and climate change (e.g., extremely hot weather, rain&flood, etc [Tran et al. 2014]), which accelerates the development of structural defects. According to a report by the RAC Foundation [RAC Foundation 2021], a substantial proportion of masonry bridges were classified as substandard, meaning they could not support the heaviest vehicles currently permitted on UK roads. It was estimated that repairing all substandard bridges to perfect condition would require a budget of £1.16 billion. However, current budget limitations suggested that only a small fraction of the necessary work would be completed within the next five years [RAC Foundation 2021]. This indicates the urgent need for efficient and proactive maintenance strategies to mitigate potential risks and ensure the structural integrity of the masonry bridges.

Effective and efficient maintenance of the vast stock of masonry bridges depends on frequent defect inspection to provide comprehensive evaluations of their structural condition. Traditional defect inspection methods predominantly rely on visual assessments of the scale and severity of defects, which are performed manually by inspectors. However, defect inspection relying on human inspectors is often subjective, time-consuming, and labour-intensive [Moore et al. 2001; Brent M. Phares et al. 2004; Huthwohl et al. 2016]. Moreover, manual inspections can struggle to access hard-to-reach areas, leaving critical components inadequately evaluated. These limitations not only compromise the comprehension of defect inspection but also lead to the risk of fatal work-related accidents during manual assessments [Fernandes 2013].

To overcome these challenges, automated inspection methods leveraging advances in deep learning (DL) have emerged as promising alternatives [Cha et al. 2018; Katsigiannis et al. 2023; Loverdos and Sarhosis 2022a]. DL, which is fully data-

driven, can provide consistent and automatic defect inspection, reducing reliance on human inspectors while improving efficiency and reliability. Recent applications of image-based DL models have demonstrated high accuracy in detecting cracks in masonry bridges [Daniel Brackenbury 2022a]. Moreover, these methods can be seamlessly integrated with Unmanned Aerial Vehicles (UAVs), allowing automated image collection with comprehensive coverage of infrastructure. Such integration improves both the efficiency and safety of inspections, as UAVs can substitute the inspectors to access hard-to-reach areas [Mandirola et al. 2022; I.-H. Kim et al. 2018].

While 2D imaging has proven effective for defect inspection in masonry bridges, it is inherently limited in capturing geometric information. In contrast, laser scanning provides an efficient, non-contact method for documenting the in-situ 3D geometry of large-scale civil engineering infrastructure using point clouds [Luhmann et al. 2020a]. Point clouds enable detailed assessments of the geometry of masonry bridges, enabling the extraction of critical parameters such as arch rise and the evaluation of deformations. This geometric information, which is challenging to measure accurately in the field, is essential for numerical analysis [Helmerich et al. 2012; Orbán and Gutermann 2009] and structural monitoring [Acikgoz et al. 2017b; C. Ye et al. 2018].

Segmentation is a crucial pre-processing step for effectively exploiting geometric information from point clouds, enabling efficient extraction of properties for defect inspection. Previous studies have focused on segmenting masonry units and mortar regions to measure mortar recess and pinning in historic buildings [Valero et al. 2018; Valero et al. 2019]. Despite these approaches effectively capturing defects related to surface degradation, they omitted the high-level geometric information necessary for evaluating structural integrity comprehensively. For masonry bridges, segmentation of individual structural components, such as arches, spandrel walls, and parapets, is essential for extracting geometric parameters like arch rise and span length. These parameters are critical for identifying structural defects, such as distortions and cracks, which potentially introduce geometric distortions to masonry bridges.

Traditional segmentation methods [Riveiro et al. 2016; R. Lu et al. 2019a] relied on handcrafted features and heuristic pipelines, often built upon presumptions about

the geometry of infrastructures. However, these approaches are case-specific and require extensive domain knowledge, limiting their scalability and generalizability.

Recent advancements in DL [Qi et al. 2017a; Qi et al. 2017b] have overcome these limitations by offering unified frameworks for segmentation tasks across various infrastructure types. These models were applied to infrastructure point clouds [H. Kim et al. 2020; Liangliang Zhao and Weidong Tao 2020; Haznedar et al. 2023; W. Lin et al. 2024b], achieving robust and accurate segmentation results while demonstrating significant generalizability.

However, two impediments need to be addressed before applying DL to masonry bridge point clouds:

(1) DL relies on well-labelled datasets with substantial data volume to achieve accurate segmentation results. However, collecting and labelling masonry bridge point clouds is a non-trivial process, which can be time-consuming and labour-intensive.

(2) Masonry bridge point clouds often contain millions of points, resulting in high computational costs when adapting DL methods. Existing DL architectures, commonly used for other infrastructure types, are not optimized for handling such large-scale datasets.

To address the impediment (1), researchers have increasingly turned to synthetic point cloud generation. By utilizing graphical simulators such as Blender, point clouds can be synthesized to augment or even substitute the real-world data for training DL models [Narazaki et al. 2021; Morbidoni et al. 2020; W. Lin et al. 2024c]. Synthetic point cloud datasets have demonstrated success in applications such as reinforced concrete bridges [Narazaki et al. 2021] and tunnel segmentation [W. Lin et al. 2024c], achieving accurate segmentation results by leveraging synthetic datasets.

There has been limited research on developing DL architectures for large-scale infrastructure point cloud segmentation to address the impediment (2). In contrast, computationally efficient architectures have been well-explored in the autonomous driving domain, which is designed to accelerate point cloud segmentation on the inference stage [Hu et al. 2020; Choy et al. 2019a; Ran Cheng et al. 2021; Kai Liu et al. 2020] to achieve real-time performance. The pioneering work, RandLA-Net

[Hu et al. 2020], employs random sampling for efficient downsampling and incorporates lightweight feature extraction blocks, enabling real-time segmentation performance. Despite these advancements, such computationally efficient architectures remain underexplored for masonry bridge point cloud segmentation, leaving a significant gap in the adaptions of technologies.

The segmentation on masonry bridge point clouds enables defect inspection by utilizing geometric information of individual structural components. Vision-based defect inspection has predominantly relied on 2D images, which leverage well-established DL convolutional neural networks (CNNs) to automate the classification and quantification of defects such as cracks and spalling [Benyun Zhao et al. 2024; Munawar et al. 2022; Rao et al. 2021]. Despite these methods having achieved high accuracy, defect detections on 2D images are often influenced by environmental factors like lighting conditions and shadows. Moreover, while 2D images are effective for identifying superficial defects, they cannot provide measurements of defect dimensions in 3D, such as crack depth or the extent of material loss. Additionally, internal defects, which are invisible on the surface, cannot be captured using 2D images, whilst being critical for comprehensive structural evaluations.

To overcome the limitations of 2D image-based approaches, recent studies have focused on detecting structural defects associated with geometric distortions using point clouds. Several methods identify geometric deviations by comparing structural components to predefined primitive geometries, such as planes or cylinders, to detect anomalies like deformations or displacements [Pesci et al. 2011; W. Liu et al. 2011; C. Ye et al. 2018; Dong et al. 2023]. These distortions are typically characterized by abrupt changes in elevation, normal vectors, or curvature. Additionally, Mohammadi et al. [2019] proposed detecting damage by analyzing the statistical distribution of geometric features to identify anomalous data without relying on predefined reference geometries. By leveraging point clouds, these approaches provide robust and accurate results in detecting defects by exploiting geometric information without reliance on colour information, which can be influenced by environmental factors [Mohammadi et al. 2019; Dong et al. 2023]. However, their reliance on case-specific workflows and assumptions on primitive geometries limits their scalability and adaptability across different infrastructure

types.

Recent advancements in unsupervised learning for anomaly detection in 3D point clouds present a promising framework for more generalizable solutions [Chu et al. 2023; Yue Wang et al. 2023; J. Liu et al. 2024; Bergmann and Sattlegger 2023; Pang et al. 2022]. Unlike traditional methods that depend on predefined geometric references or heuristic workflows, unsupervised learning approaches aim to generalize anomaly detection by training DL models to capture the statistical distribution of normal data. By measuring deviations between test data and the learned distribution of normal data, these models provide robust predictions on anomalous data. For masonry bridges, anomalous data can be regarded as structural defects for adaptions of anomaly detection.

However, unsupervised learning methods are significantly limited in adapting to defect inspection on the infrastructure point clouds. Firstly, defects related to geometric distortions, such as cracks, are often small compared to the size of masonry bridge point clouds. Capturing such defects requires dense point clouds to avoid losing localized geometric distortion during downsampling. Despite processing dense point clouds with DL methods is computationally infeasible due to the massive memory consumption required, making DL methods impractical for real-world applications.

Secondly, there is a lack of suitable training data for defect inspection in masonry bridges. Existing synthetic point cloud generation methods primarily focus on creating idealized geometries [Narazaki et al. 2021; W. Lin et al. 2024c] and omit realistic deformations caused by factors such as initial settlements. These deformations introduce slight geometric imperfections that reflect the mechanical behaviour of masonry bridges and should be regarded as normal data rather than defects. Therefore, to validate the performance of unsupervised learning methods, deviations should be measured between normal data with geometric imperfections and anomalous data representing actual anomalies, rather than using idealized geometries as normal data.

Recently, researchers introduced the *PatchCore* framework for anomaly detection in 3D point clouds [Horwitz and Hoshen 2023; Cao et al. 2023], achieving state-of-the-art (SOTA) performance compared to DL-based methods. Instead of relying

on DL models, *PatchCore* utilizes the handcrafted Fast Point Feature Histograms (FPFH) descriptor [Rusu et al. 2009] to extract geometric features from point clouds. This method significantly reduces the data requirements typically associated with DL methods, allowing anomaly detection to be performed on minimally two pairs of point clouds. Additionally, *PatchCore* avoids the high GPU memory demands of DL models, enabling the efficient processing of dense point clouds for individual structural components.

PatchCore leverages a memory bank constructed from FPFH features of the normal point cloud. When evaluating a test point cloud, *PatchCore* pairs between FPFH features of test points and those in the memory bank by measuring the minimal Euclidean distances, where higher deviations indicate potential anomalies. Such an approach allows direct comparisons between a current inspection point cloud and a prior reference of the same masonry bridge. *PatchCore* avoids the need for training data and predefined ideal geometries, making it applicable to real-world anomaly detection in masonry bridge point clouds.

Despite *PatchCore* alleviating the data requirements, the validation of the anomaly detection algorithm still requires point clouds with labelled defects. Labelling visible defects, such as surface cracks or spalling, can be accomplished using photogrammetry or manual annotations derived from high-resolution scans. However, identifying and labelling internal defects within masonry structures requires techniques such as the ground-penetrating radar [Solla et al. 2012] or ultrasonic testing [Hassani and Dackermann 2023]. These methods require significant time, expertise, and costly equipment, making them impractical in real-world applications.

Overall, effective inspection of masonry bridges is crucial for enabling proactive maintenance to preserve structural integrity. While advancements in DL and laser scanning enable potentially automated inspections, their implementations to masonry bridges are hindered by data scarcity, high computational costs, and the absence of algorithms for defect detection in large-scale point clouds. This study systematically addresses these challenges to advance automated bridge inspection methodologies.

1.2 Research objectives

This research aims to first address critical challenges in the segmentation of large-scale masonry bridge point clouds using advanced DL techniques. To overcome the high costs and labour-intensive of collecting real data for training DL, a synthetic masonry bridge point cloud dataset will be developed. Leveraging this dataset, the research will focus on designing lightweight DL models optimized for large-scale point cloud segmentation. These architectures will emphasize computational efficiency while maintaining high segmentation accuracy, making them suitable for processing the large-scale masonry bridge point cloud dataset with complex geometries.

In addition to segmentation, this study seeks to advance automatic defect inspection (e.g., cracks and water patches) directly on masonry bridge point clouds. By integrating the geometric and intensity information in 3D point clouds, the research will leverage a multimodal feature, 3DMulti-FPFHI, to capture the characteristics of the two types of defects. The 3DMulti-FPFHI will be incorporated into an anomaly detection algorithm to enable the reliable identification of defects. This approach aims to enhance the robustness and generalizability of defect inspection, addressing critical limitations in current inspection methodologies for infrastructure point clouds.

To support the anomaly detection algorithm, a novel framework will be established for simulating masonry bridge point clouds with realistic structural deformations. This framework will integrate physical simulations of structural behaviour with advanced graphical rendering tools to produce realistic synthetic datasets, i.e., synthetic point clouds considering the mechanical behaviours of masonry bridges. These datasets will enable thorough validation of the anomaly detection algorithm and support the development of data-driven methods requiring extensive training datasets.

By integrating these objectives, this research aims to advance the automation, efficiency, and reliability of masonry bridge inspections, offering scalable solutions for structural maintenance and safety assessments.

1.3 Outline of the thesis and contributions

In Chapter 2, we begin by exploring the SOTA 3D point cloud segmentation technologies, focusing on DL methods and their applications in infrastructure. Building upon this foundation, we review DL techniques (e.g., supervised and unsupervised learning methods) for detecting structural defects using image and point cloud data.

Chapter 3 introduces BridgeNet, a novel 3D deep learning neural network for semantic segmentation of large-scale masonry bridge point clouds. To address data scarcity, this chapter introduces a framework for synthesizing masonry bridge point clouds. BridgeNet achieves SOTA performance on real-world datasets, outperforming existing methods and enabling accurate segmentation of structural components such as arches and piers. This framework enables the extraction of critical geometric parameters essential for defect inspection, facilitating proactive maintenance.

Chapter 4 proposed the BridgeNetv2, which extends BridgeNet by incorporating Transformer-based modules to enhance feature extraction and segmentation performance, particularly for imbalanced datasets. By leveraging Local and Global Geo-Transformers, BridgeNetv2 achieves SOTA accuracy while inheriting the memory efficiency of BridgeNet, making it applicable to diverse infrastructures.

Chapter 5 introduces a novel anomaly detection framework for identifying cracks in synthetic masonry bridge point clouds using geometric distortions. This framework adapts the 2D *PatchCore* algorithm for 3D applications and integrates multi-scale FPFH to improve crack detection accuracy. Synthetic point clouds generated through finite element simulations and graphical modelling enable validation under diverse conditions, addressing challenges such as noise, initial damage, and rough textures. This approach demonstrates robust detection of both surface and internal cracks, providing a potentially generalized method for inspecting structural defects in complex infrastructure systems.

Chapter 6 introduces a novel 3D multimodal feature, 3DMulti-FPFHI, combining geometry and intensity modalities for anomaly detection in point clouds, surpassing the limitations of previous geometry-based methods. The proposed feature enhances defect detection accuracy by leveraging synthetic point clouds for statistical and parametric studies, which compensate for data scarcity in real-world

applications. The framework is demonstrated on real-world data from London Bridge Station, where the 3DMulti-FPFHI successfully captures a longitudinal crack that geometric feature (e.g., FPFH) alone fails to detect. The 3DMulti-FPFHI also enables the identification of non-geometric defects (e.g., water ingress) which introduces intensity anomalies. To evaluate generalizability, the framework is tested on a full-scale experiment tunnel dataset under increasing loading conditions, which effectively captures both intrados and extrados cracks under intensity noise. These results demonstrate the framework's robustness and adaptability across diverse infrastructure types, offering a practical solution for anomaly detection in point clouds.

Chapter 7 consolidates the key findings and contributions of this research and proposes avenues for future work.

Chapter 2

Literature review

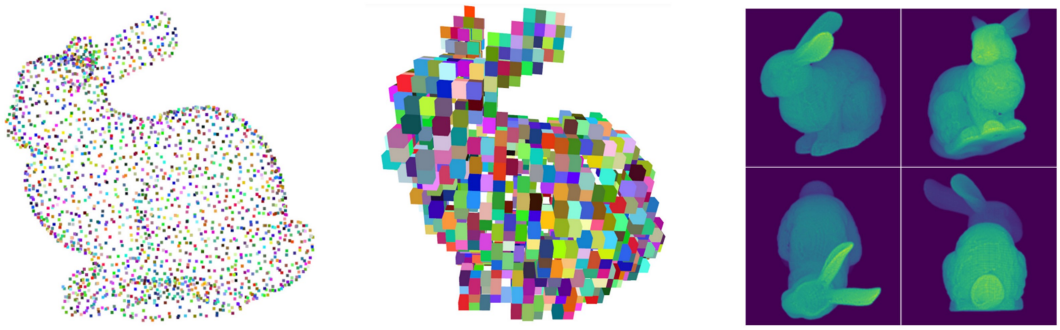
2.1 Segmentation of large-scale point cloud of infrastructures with 3D DL

2.1.1 Point cloud representation

The intrinsic data structure of point clouds differs from that of 2D images, making it incompatible with the direct application of 2D convolution operations. To leverage the well-established architectures of convolutional neural networks (CNNs), it is essential to define appropriate point cloud representations, regardless of the specific tasks.

Raw point cloud

As shown in Figure 2.1a, the raw point cloud represents unprocessed data, structured as $(N, 3)$, where N denotes the number of points and the second dimension corresponds to the x, y, and z coordinates. Point clouds can also contain extra feature channels (e.g., intensity, RGB, and surface normal). For simplicity and clarity, unless otherwise stated, we only use the x, y, and z coordinates as our input channel. One key advantage of directly processing raw point clouds is the absence of transformation-induced information loss. However, there exists no explicit neighbourhood relationship in raw point clouds, as they form an unordered set of points. The unordered format impedes the application of convolution operations, which typically require an ordered data format (such as the structured



(a) Raw point cloud (b) Voxelizations (c) Multi-view

Figure 2.1: Three different point cloud representations: (a) raw point cloud, (b) voxelization, and (c) multi-view projection.

arrangement of pixels in 2D images). Current approaches [Qi et al. 2017b] address this issue by employing k-nearest neighbour (k-NN) searches or the ball query to define point relationships, though this can be computationally expensive, particularly for large-scale point clouds.

Voxelization

Alternatively, point clouds can be voxelized into discretized 3D grids, as shown in Figure 2.1b. Voxel grids have dimensions of $H \times W \times D$ (e.g., height, width, and depth correspondingly), which has one dimension more than 2D images ($H \times W$). In this process, points and their associated features are embedded into the corresponding voxels based on their x, y, and z coordinates, which are also discretized into the voxel grid. However, each voxel can only store a single point, even though multiple points may lie within the same voxel, leading to potential information loss by discarding important points (e.g., points describing fine geometries).

The primary advantage of voxelization is that it converts raw point clouds into a structured data format with clearly defined 3D neighbourhoods, allowing for seamless adaptation of convolution operations from 2D to 3D. This enables the direct use of well-established 2D CNN architectures [Çiçek et al. 2016] for tasks involving point clouds.

However, the extra memory consumption introduced by voxelization cannot be omitted in 3D. Compared to 2D images with $O(n^2)$ complexity, it is increased to $O(n^3)$ for 3D voxel grids, which makes voxelization computationally expensive, especially for high-resolution point clouds.

Multi-view projection

The core concept of multi-view projection is to convert 3D point clouds and their associated features, such as depth (as illustrated in Figure 2.1c), into 2D representations with dimensions of $H \times W$. This transformation is shown in Figure 2.1c. The advantage of the transformation is the dense nature of the 2D representation; for instance, a 2D image of 512×512 contains 262,144 pixels. Therefore, 2D representation helps preserve detailed information compared to voxelization, enabling CNNs to process the data efficiently [X. Chen et al. 2017].

However, multi-view projection introduces the loss of 3D spatial information, as critical geometric details may not be fully preserved. Additionally, the accuracy of these projections relies heavily on the precise alignment and calibration of cameras. Misalignment can cause degradation of the data quality. Furthermore, occlusions and overlapping of multi-view projections can cause confusion when merging prediction results, making it difficult for CNNs to understand underlying 3D geometry, especially in complex and large-scale scenes.

2.1.2 Semantic segmentation

Semantic segmentation refers to the task of assigning a class label to each point in a point cloud, enabling a detailed understanding of the scene by distinguishing between different objects or structures. In the context of infrastructure, this is crucial for identifying and classifying various components, such as beams, columns, or defects, within a 3D model.

Different approaches have been proposed based on the different point cloud representations. Apart from the multi-view projection, neural network architectures designed to directly process 3D data representations (e.g., raw point clouds and voxels) are all governed by a common design methodology:

(1) A proper definition of the neighbour relationship is required for further computations.

(2) A method for extracting local information from these defined neighbours must be established.

Point-based methods

Point-based approaches directly process raw point clouds without preceding transformation. However, two primary challenges must be addressed for this purpose:

(1) **Unordered**. Unlike images or voxels, which are structured on ordered grids, point clouds represent arrays without an inherent order. It means that a neural network, when taking N points as input, must be invariant to all possible permutations π, with $N!$ possible permutations of these points.

(2) **Interaction among points**. Points are not isolated, which means that the neighbour points form a meaningful geometry subset. The model should be able to exploit the local information of nearby points, and the interaction among the local geometries.

The pioneering research, PointNet [Qi et al. 2017a], addressed the first challenge by introducing a set function $f : \mathcal{X} \to \mathbb{R}$ to map a given point set $\boldsymbol{P} = \{\boldsymbol{p}_i \mid \boldsymbol{p}_i \in \mathbb{R}^{C_{in}}, i = 1, \ldots, n\}$, where $C_{in} = 3$ if it represents input with only coordinate information, to a vector:

$$f(\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_n) = g\left(h(\boldsymbol{p}_1), h(\boldsymbol{p}_2), \ldots, h(\boldsymbol{p}_n)\right) \tag{2.1}$$

where $h : \mathbb{R}^{C_{in}} \to \mathbb{R}^{C_{out}}$ represents a multi-layer perception network (MLP). $g : \mathbb{R}^{C_{out}} \times \cdots \times \mathbb{R}^{C_{out}} \quad (n \text{ times}) \to \mathbb{R}^{C_{out}}$ is a symmetric function which should be invariant to $\pi\left(h(\boldsymbol{p}_1), h(\boldsymbol{p}_2), \ldots, h(\boldsymbol{p}_n)\right)$. Qi et al. [2017a] explored three different symmetric functions (e.g., attention sum, average pooling, and max pooling) across different tasks, and experimentally concluded that max pooling yielded the best performance. Additionally, they introduced a T-Net, a learnable 3×3 matrix that applies a rigid transformation to the input point cloud, enabling PointNet to learn the underlying geometry regardless of rotations and translations.

However, PointNet lacks the ability to capture local context at different scales, as the max pooling operation is computed on the entire point cloud, leading to a loss of local geometric detail. PointNet++ [Qi et al. 2017b] proposes an encoder-decoder architecture designed to learn geometric features hierarchically. The encoder comprises several *set abstraction* layers, each containing two key components:

(1) **Sampling layer**. Given the input points of the size $N \times C_{in}$, farthest point sampling (FPS) is utilized to iteratively select a subset of points ($N^{'} \times C_{in}$) by measuring the most distance point from the source point cloud (before subsampling) w.r.t target point cloud (after subsampling). Compared to random sampling, FPS ensures better coverage of the entire point cloud, especially in areas with varying densities.

(2) **Group layer**. The computational region is defined locally by querying K neighbouring points from the input points ($N \times C_{in}$), using the subset of points ($N^{'} \times C_{in}$) as centroids. The output represents a group of point sets with the size of $N^{'} \times K \times C_{out}$. Unlike PointNet, where the computation is performed on the entire point cloud, PointNet++ applies the invariant operation to the local regions. To query the neighbours, both k-NN and ball query methods are used. k-NN selects the K nearest points relative to each centroid, whereas the ball query identifies all points within a predefined radius r, capturing neighbours within a fixed local scale.

The hierarchical structure of the encoder enables gradual expansion of the receptive field. By constructing local regions at multiple scales, the network aggregates local features from small neighbourhoods while progressively integrating larger spatial patterns, enhancing its ability to understand complex geometries within the point cloud.

To restore the subsampled point cloud to its original size, PointNet++ employs a hierarchical feature propagation strategy in the decoder. Point features from an upper layer $i+1$ (denoted as $N \times C_{i+1}$) are obtained by interpolating features from the lower layer i ($N' \times C_i$) using k-NN. The interpolation process uses the inverse distance weighted average strategy, where points closer to the centroid contribute more to the propagated feature, while points further away have a reduced influence. This method ensures smooth and accurate feature propagation, allowing the model to reconstruct the point cloud to the original size effectively. Subsequent approaches have refined neighbourhood relationships and enhanced the extraction of local information.

Inspired by the Scale Invariant Feature Transform (SIFT) [Lowe 1999], M. Jiang et al. [2018] proposed PointSIFT, which defines a stacked 8-neighbourhood structure

to find the nearest neighbours in each of the eight octants. Orientation-Encoding (OE) units are then introduced to capture local information, performing regular 2D convolution operations sequentially along the x, y, and z dimensions. PointSIFT also exploited the multi-scale representations by stacking the features extracted by OE from higher to lower levels. Building on the structure of PointNet++, Sheshappanavar and Kambhamettu [2020] refined the ball query into an ellipsoid query to better accommodate neighbourhoods with high-curvature surfaces or regions. The ellipsoid query is oriented w.r.t the principal axis of local neighbours, thereby capturing local geometry more effectively.

Alternatively, ShellNet [Zhiyuan Zhang et al. 2019] defines local regions using a set of multi-scale concentric spheres. The area between the two spheres forms a spherical shell, providing an unambiguous order as the shells are naturally organized from the innermost to the outermost. Separate convolution operations are applied to points within each shell, and max pooling aggregates the final features. To explicitly explore local representations, Yuelin Liu et al. [2019b] proposed constructing a low-level relation vector that leverages the coordinates of the centre and neighbouring points (e.g., $\boldsymbol{p}_c$ and $\boldsymbol{p}_i$, respectively). The resulting vector is defined as $(\|\boldsymbol{p}_c - \boldsymbol{p}_i\|, \boldsymbol{p}_c - \boldsymbol{p}_i, \boldsymbol{p}_c, \boldsymbol{p}_i)$ with 10 channels, where the first channel represents the Euclidean distance between $\boldsymbol{p}_c$ and $\boldsymbol{p}_i$. This low-level vector is then processed through an MLP and element-wise multiplied with high-level point features, adding geometric topology constraints among points. Inspired by the dense connection model [G. Huang et al. 2017], DensePoint [Yuelin Liu et al. 2019a] introduces an architecture that hierarchically aggregates point features at multiple scales (using varied ball query radii to control the receptive field). This dense connection design allows DensePoint to capture rich contextual information, making it highly effective for implicit shape recognition. W. Wu et al. [2019] introduced PointConv, a novel convolution operation that re-weights point features based on local point density and the Euclidean distance between the centre and neighbouring points.

However, previous convolution designs, including PointConv, still approximate 2D convolution and are not fully specialized for point clouds. Instead, Thomas et al. [2019] proposed kernal point convolution (KPConv), a 3D convolution method that represents the kernel with K points, denoted as $\{\tilde{\boldsymbol{x}}_k \mid k = 1, \ldots, K\}$. These kernel points are positioned as vertices of regular polyhedra within a ball of radius

r, interacting with points $\boldsymbol{p}_i$ (e.g., points acquired by ball query) via a kernel function g:

$$g(\boldsymbol{p}_i) = \sum_{k=1,\ldots,K} d_{linear}(\boldsymbol{p}_i, \tilde{\boldsymbol{x}}_k)\mathbf{W}_k \tag{2.2}$$

where d_{linear} defines a negative correlation between $\tilde{\boldsymbol{x}}_k$ and $\boldsymbol{p}_i$ based on their Euclidean distance. Here, $\mathbf{W}_k \in \mathbb{R}^{C_{in}} \times \mathbb{R}^{C_{out}}$ represents the associated weight matrices that map features from dimension C_{in} to C_{out}. KPConv provides an intuitive approach for extending 2D convolution to 3D, capturing local geometry robustly and efficiently.

By leveraging both PointConv and KPConv, M. Xu et al. [2021] developed Position Adaptive Convolution (PAConv), which dynamically constructs the convolution kernel. PAConv utilizes a weight bank $\mathcal{B} = \{\boldsymbol{B}_m \mid m = 1, \ldots, M\}$, in which each $\boldsymbol{B}_m \in \mathbb{R}^{C_{in} \times C_{out}}$ represents a learnable weight matrix. The kernel function is defined as a linear combination of all matrices in the weight bank, with coefficients determined by a proposed ScoreNet that leverages point positions (e.g., the low-level relation vector proposed by Yuelin Liu et al. [2019b]). Compared to PointConv and KPConv, PAConv is more flexible, free of hand-crafted design, and easily integrated with various architectures.

However, previous methods can only be trained on small-scale point clouds (thousands of points) since they rely on (1) expensive sampling methods (which are time-consuming) and (2) computationally heavy designs (which are memory-consuming). These limitations restrict their adaptation to large-scale infrastructure point clouds (millions of points). To address limitations (1) and (2), Hu et al. [2020] proposed RandLA-Net, which incorporates two innovative designs:

(1) **Random sampling (RS)**. Among existing sampling methods, FPS and inverse density importance sampling (IDIS) are the most commonly used for small-scale point clouds, both designed to ensure good coverage of the entire point cloud. However, the computational complexities of FPS and IDIS are approximate $O(N^2)$ and $O(N)$, respectively. In contrast, RS achieves the highest computational efficiency with a complexity of $O(1)$, regardless of the point cloud scale, making it suitable for large-scale point clouds.

(2) **Local feature aggregation**. RandLA-Net introduces a new local feature aggregation module to compensate for potentially important point features that may be discarded by RS. This module learns the complex local structure by progressively increasing the receptive field at each neural layer. Unlike previous methods, RandLA-Net does not perform direct convolution on neighbouring features. Instead, it employs an attentive pooling mechanism to learn to re-weight w.r.t feature channel via a shared MLP, significantly reducing GPU memory cost.

Inspired by the RandLA-Net, Kangcheng Liu et al. [2020] introduced FG-Conv to capture the feature correlations and model the local geometry of point clouds using a deformable kernel, which adopts a similar design to KPConv. To further reduce the memory consumption, the feature dimension of point features is downsampled from C_{in} to $C_{in}/8$ via a 1×1 convolution layer before being fed into FG-Conv. This approach achieves an 8.6% reduction in memory usage and a 274% increase in inference speed compared to RandLA-Net.

Recent advancement [R. Zhang et al. 2023] enables a novel non-parametric framework that avoids the reliance on learnable parameters in traditional deep neural networks. By focusing on geometric similarity and dynamic feature matching, their method achieves competitive performance while reducing training complexity. This paradigm shift challenges the conventional emphasis on heavily parameterized architectures and opens new avenues for efficient and interpretable point cloud processing.

Voxel-based methods

Inspired by 2D DL architectures, early 3D neural networks [Tchapmi et al. 2017; Çiçek et al. 2016] extended 2D convolution into 3D to process dense voxel grids directly. However, these 3D dense CNNs face significant computational challenges due to the additional dimension, as discussed in Section 2.1.1. Given that voxelization inherently results in a sparse representation (with most voxels being empty), various methods have been developed to leverage this sparsity and reduce memory costs.

One of the pioneering approaches, VoxelNet [Yinlong Zhou and Tuzel 2018], per-

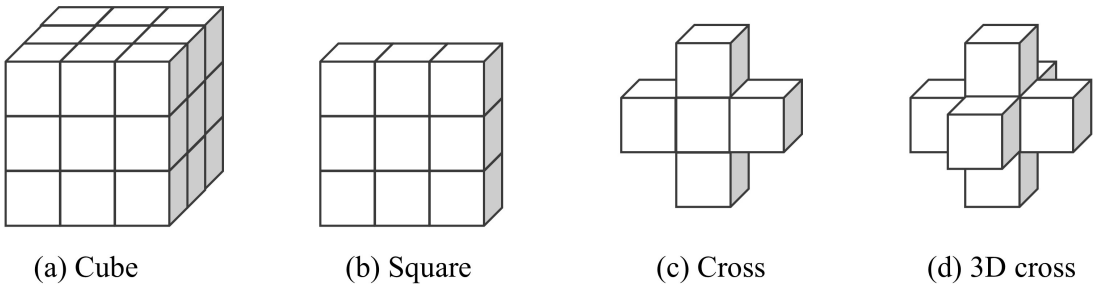


Figure 2.2: Visualization of various convolution kernels: (a) represents kernel on conventional dense 3D convolution, (b)-(d) show kernels used in Minkowski Engine.

forms feature encoding only on non-empty voxels, creating a uniform feature on all points within each voxel by using max pooling in the initial step. Similarly, Zhijian Liu et al. [2019a] introduced Point-Voxel CNN, which aggregates point features within each voxel using average pooling. However, despite the sparsity-aware feature encoding, both VoxelNet and Point-Voxel CNN continue to rely on dense 3D convolution for subsequent computations, limiting their computational efficiency.

To address this, Graham et al. [2018] proposed a method to perform 3D convolutions only on non-empty voxels, significantly reducing unnecessary computations. However, during the convolution process, empty voxels can inadvertently become activated by the convolution operation and be treated as non-empty, a phenomenon known as the dilation problem, which reduces sparsity. To mitigate this issue, Graham et al. [2018] proposed to supervise voxels covered by the convolution operation, which is to ensure only the originally non-empty voxels remain non-empty after each convolution operation.

However, previous studies have not fully explored the sparse representation of 3D voxelization. Choy et al. [2019a] introduced the Minkowski Engine, a comprehensive library for performing computational operations (such as sparse convolution, sparse max pooling, etc,) on sparse data. This sparse representation stores only the coordinates and associated features of non-empty voxels using the coordinate format (COO). In conventional 3D dense convolution, the convolution kernel (shown in Figure 2.2) is a direct extension of the 2D convolution kernel, resulting in an exponential increase in parameters as the kernel size grows. The Minkowski Engine proposes using a hybrid of square, cross, and 3D cross kernels (as shown in Figure 2.2) to perform sparse convolution, which significantly improves both ef-

ficiency and accuracy compared to the conventional kernel. A key difference in utilizing COO instead of dense voxels is that only non-empty voxels participate in the convolution computation. To convolve a non-empty voxel, a mapping is required to identify which input affects which outputs. This involves matching input coordinates with corresponding output coordinates where the resultant values are stored, which can be determined by the stride of the convolution operation. The Minkowski Engine is implemented in C++/CUDA, with self-defined forward and backward propagation to ensure efficient processing on the sparse data format.

Building on the Minkowski Engine, Komorowski [2021] proposed a simple architecture, MinkLoc3D, to learn global descriptors through a series of sparse convolution operations followed by average pooling. Y. Chen et al. [2022] addressed the dilation problem previously identified by Graham et al. [2018], which had been overlooked by the sparse convolution implementation in the Minkowski Engine. Instead of using a static mapping between input and output coordinates, Y. Chen et al. [2022] introduced a method to learn importance scores for the convolution kernel directly from the input features. The dilation of non-empty voxels, i.e., whether neighbouring voxels are activated or not, depends on these learned importance scores. A pre-defined threshold is applied to these scores: output positions with scores exceeding the threshold are dilated, while those below the threshold remain inactivated.

Additionally, Cabrera et al. [2024] proposed MinkUNeXt, a new architecture that adopts a typical encoder-decoder design inspired by UNet [Ronneberger et al. 2015]. MinkUNeXt incorporates a novel residual block, fully implemented using the Minkowski Engine, achieving SOTA performance across various tasks.

Projection-based methods

Projection-based methods convert the 3D point cloud $\boldsymbol{P}$ into a set of m 2D images $\{\boldsymbol{I}_i \mid i = 1, \ldots, m\}$, enabling well-established 2D CNNs for processing these planes. This projection is defined as:

$$\boldsymbol{I}_i = \mathcal{F}_{\text{proj}}^{(i)}(\boldsymbol{P}) \tag{2.3}$$

where $\mathcal{F}_{\text{proj}}^{(i)}$ represents a projection function for the i-th view. To ensure that

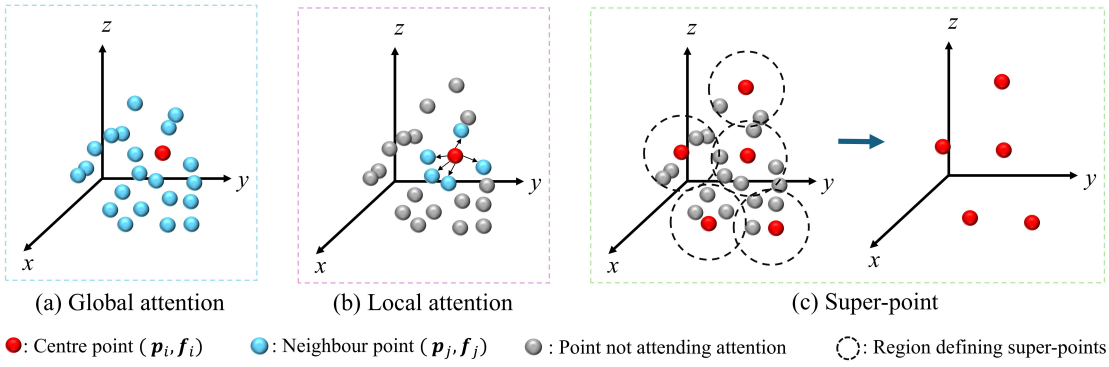


Figure 2.3: Visualization of different attention mechanisms: (a) Global attention in the vanilla Transformer, where relationships are computed between the centre point (red dots with input $\boldsymbol{p}_i$ and $\boldsymbol{f}_i$) and all points (blue dots with input $\boldsymbol{p}_j$ and $\boldsymbol{f}_j$); (b) local attention, focusing only on neighbouring points; (c) super-point attention, grouping points within local regions (dashed-line spheres) for subsequent global attention.

segmentation results on $\boldsymbol{I}_i$ can be accurately restored to the original 3D point cloud, the inverse function $\mathcal{F}_{\text{proj}}^{(i)\,-1}$ must exist, in which pixels in $\{\boldsymbol{I}_i\}$ can be back-projected to points in $\boldsymbol{P}$.

X. Chen et al. [2017] project $\boldsymbol{P}$ from both bird's eye view and front view (e.g., points are projected to a cylinder plane [B. Li et al. 2016]) to obtain height, density, and intensity maps. Features were extracted from these 2D maps using 2D CNNs, followed by a fusion step to produce feature maps, which could then be reversely projected back to $\boldsymbol{P}$

However, due to the inherently sparse nature of 3D point clouds, insufficient point density can lead to void regions in the 2D projections. To address this, Boulch et al. [2017] and Kundu et al. [2020] first convert $\boldsymbol{P}$ into mesh representation to allow the dense projection. Alternatively, Lawin et al. [2017] employs splatting-based rendering techniques to prevent void projections by distributing each point's projection over neighbouring pixels using a Gaussian point spread function. Each projected pixel contributes to neighbouring pixels according to a 2D Gaussian kernel, where weights decrease with distance from the projected object. Unlike methods that only use orthogonal views (e.g., bird view or front view), these techniques adopt varied camera positions and rotation angles to achieve more comprehensive scene coverage. However, segmentation performance remains sensitive to projection parameter choices (e.g., camera positions and angles), which limits their application to large, complex 3D scenes.

Transformer-based methods

Transformer, initially introduced for natural language processing tasks, has since become foundational in various applications due to their ability to capture long-range dependencies through self-attention mechanisms [Vaswani et al. 2017]. Unlike traditional convolutional architectures, which rely on localized convolution kernels, Transformers exploit relationships across an entire input sequence $\boldsymbol{F}_{input} = \{\boldsymbol{f}_i \mid \boldsymbol{f}_i \in \mathbb{R}^{C_{in}}, i = 1, \ldots, N\}$ by assigning attention scores that dynamically weigh the importance of each centre $\boldsymbol{f}_i$ as shown in Figure 2.3a. The standard self-attention mechanism in the vector form can be then represented as:

$$\boldsymbol{f}_{out} = \sum_{\mathbf{f}_j \in \boldsymbol{F}_{input}} \rho\left(\varphi(\boldsymbol{f}_i)^\top \psi(\boldsymbol{f}_j) + \boldsymbol{pos}\right) \alpha(\mathbf{f}_j) \tag{2.4}$$

where $\boldsymbol{f}_{out} \in \mathbb{R}^{C_{out}}$ represents the output feature. φ, ψ, and α are linear projections for acquiring the query, key, and value vectors. ρ is the normalization function $softmax$ (e.g., $softmax$ ensures $\sum_{\mathbf{f}_j \in \boldsymbol{F}_{input}} \rho\left(\varphi(\boldsymbol{f}_i)^\top \psi(\boldsymbol{f}_j) + \boldsymbol{pos}\right) = 1$). Since the Transformer architecture is inherently invariant to input sequence permutations $\pi\left(\boldsymbol{f}_1, \boldsymbol{f}_2, \ldots, \boldsymbol{f}_N\right)$, the positional encoding $\boldsymbol{pos}$ is provided to allow Transformer to understand the relative or absolute positions of the sequence. Such sequence-invariant property makes Transformers particularly suitable for point cloud processing, where point order is arbitrary. The architecture's ability to capture complex spatial patterns and contextual relationships is valuable in 3D point cloud tasks, where dependencies over large areas are essential. To distinguish the original design of Transformer from its adaptations in point clouds, the original Transformer is referred to as the vanilla Transformer in the following sections.

Vanilla Transformers operate on global relationships across all input sequences, resulting in $O(N^2)$ computational complexity. Initial implementations on 3D point cloud [Matveev et al. 2020] omit such cost which introduces an $N \times N$ attention matrix, significantly diminishing its feasibility on large-scale point clouds. To alleviate the computational consumption, two strategies, i.e., (i) **local attention mechanism** and (ii) **superpoints**, are adapted in the proposed architectures.

(i) Local attention mechanism

H. Zhao et al. [2021a] first proposed the Point Transformer which utilizes the local

attention mechanism as shown in Figure 2.3b. Instead of computing attention across all input points, Point Transformer focuses on local neighbourhoods defined via k-NN, where j is one of the k-nearest neighbours of i. Instead of utilizing scalar attention $w_{ij} = \varphi(\boldsymbol{f}_i)^\top \psi(\boldsymbol{f}_j)$ in the vanilla Transformer (shown in Equation 2.4), Point Transformer employs vector attention for feature aggregation defined as:

$$\boldsymbol{w}_{ij} = \gamma(\varphi(\boldsymbol{f}_i) - \psi(\boldsymbol{f}_j)) \in \mathbb{R}^{C_{in}} \tag{2.5}$$

where γ represents an MLP. The $\boldsymbol{pos}$ is redefined their positional encoding as follows:

$$\boldsymbol{pos} = \theta(\boldsymbol{p}_i - \boldsymbol{p}_j) \tag{2.6}$$

where $\boldsymbol{p}_i$ and $\boldsymbol{p}_j$ defines a pair of points i and j, and θ represents an MLP to encode the relative positional information. These designs have demonstrated superior performance compared to the vanilla Transformer experimentally.

However, the parameters for computing vector attention can increase dramatically as the network goes deeper and there are more feature channels. To overcome the limitations of vector attention, Point Transformer V2 [X. Wu et al. 2022] introduced a grouped vector attention mechanism. The grouped vector attention mechanism first evenly divides the C_{in} of $\boldsymbol{f}_i$ into g groups of vectors ($1 \leq g \leq C_{in}$). The γ, instead of the mapping of $\gamma : \mathbb{R}^{C_{in}} \to \mathbb{R}^{C_{in}}$, now represents the mapping of $\gamma : \mathbb{R}^{C_{in}} \to \mathbb{R}^{g}$. Therefore, the same scalar attention weights are shared across each group, significantly reducing parameters.

Local attention mechanism can suffer from a limited receptive field, restricting its ability to capture long-range contextual dependencies. However, increasing the receptive field in 3D space triggers a quadratic increase in memory consumption. To address this limitation, Lai et al. [2022a] proposed the Stratified Transformer, which defines local receptive fields at multiple scales, ranging from small to large. For smaller receptive fields, points are denser, while downsampling is applied in larger receptive fields to alleviate memory consumption.

Additionally, the Stratified Transformer introduces a memory-efficient strategy to avoid duplication in local attention mechanisms. Previous local attention frameworks typically duplicate $\boldsymbol{F}_{input}$ three times to initialize queries, keys, and value vectors. However, the computation of local attention can be determined directly using the indices of neighbours obtained from k-NN queries. The Stratified Transformer utilizes two index lists to directly retrieve $\boldsymbol{f}_i$ from original $\boldsymbol{F}_{input}$, avoiding duplication and reducing memory usage. However, the global dependencies of $\boldsymbol{P}$ remain unexplored when relying solely on the **Local attention mechanism**, which is crucial for distinguishing geometric similarities between different structural components in large-scale and complex infrastructure point clouds.

(ii) **Superpoints**

Alternatively, Pang et al. [2022] introduced Point-MAE, which groups points into superpoints $\boldsymbol{P}_s = \{\boldsymbol{p}_{s_i} \mid i = 1, \ldots, M\}$ using a PointNet-like architecture, as shown in Figure 2.3c. Points within each local region (represented by dashed-line spheres shown in Figure 2.3c) are embedded into $\boldsymbol{p}_{s_i}$, significantly downsampling the point cloud and enabling efficient global attention. However, Point-MAE adopts only an encoder design without considering the downsampling and upsampling architecture. Building upon Point-MAE, R. Zhang et al. [2022] proposed Point-M2AE, which introduces an encoder-decoder architecture with a pyramid structure. This design progressively models spatial geometries and captures both fine-grained and high-level semantics of 3D shapes, with the decoder following the same structure as PointNet++.

To jointly learn global and local geometric relationships, X. Yu et al. [2023a] proposed a hybrid approach by applying local attention to input features to learn local structures. The outputs from local attention are concatenated with the outputs of global attention in the vanilla Transformer to integrate both local and global geometries. However, extracting local structures from high-level features (e.g., features corresponding to $\boldsymbol{P}_s$) can lead to less discriminative geometries.

To address this limitation, Qin et al. [2022] introduced the Geometric Transformer, which explicitly encodes geometric information using coordinates of $\boldsymbol{p}_{s_i}$. Unlike previous works [H. Zhao et al. 2021a; Park et al. 2022] that use relative coordinates (e.g., $\theta(\boldsymbol{p}_i - \boldsymbol{p}_c)$), the Geometric Transformer ensures geometric information is

invariant to rigid transformations, achieving robust outputs regardless of the input pose. Given $\boldsymbol{p}_{s_1}$ and $\boldsymbol{p}_{s_2}$, Geometric Transformer proposed two transformation-invariant metrics for the further positional embedding, i.e., pair-wise distance ρ and triplet-wise angle α_i, where ρ is given by:

$$\rho = \|\boldsymbol{p}_{s_1} - \boldsymbol{p}_{s_2}\| \tag{2.7}$$

which represents the distance between $\boldsymbol{p}_{s_1}$ and $\boldsymbol{p}_{s_2}$. α_i can be computed by constructing triplets of $\boldsymbol{p}_{s_1}$ and $\boldsymbol{p}_{s_2}$:

$$\alpha_i = \arccos\left(\frac{(\boldsymbol{p}_{s_i} - \boldsymbol{p}_{s_1}) \cdot (\boldsymbol{p}_{s_2} - \boldsymbol{p}_{s_1})}{\|\boldsymbol{p}_{s_i} - \boldsymbol{p}_{s_1}\| \, \|\boldsymbol{p}_{s_2} - \boldsymbol{p}_{s_1}\|}\right), \quad i = 1, \ldots, k \tag{2.8}$$

where $\{\boldsymbol{p}_{s_i} \mid i = 1, \ldots, k\}$ represents the k nearest neighbours of $\boldsymbol{p}_{s_1}$. Both ρ and α_i are invariant to rigid transformations, enabling the Geometric Transformer to maintain geometric consistency across different point clouds.

Similar to the Stratified Transformer, Yahui Liu et al. [2023] proposed downsampling point clouds to increase the receptive field while operating on $\boldsymbol{P}_s$. Although downsampling significantly enhances computational efficiency and receptive fields, it risks inadvertently discarding critical points that represent key structural components, which can occupy only a small fraction of large-scale infrastructure.

2.1.3 Instance segmentation

Instance segmentation involves identifying and delineating each object instance in a 3D point cloud. Unlike semantic segmentation, which assigns a class label to every point without distinguishing between different instances of the same class, instance segmentation separates individual objects within the same class. This distinction makes it particularly useful for infrastructure point clouds, where extracting geometric or mechanical information from individual components is often required.

The pioneering work Similarity Group Proposal Network (SGPN) [W. Wang et al. 2018] introduces a PointNet-like backbone [Qi et al. 2017a; Qi et al. 2017b] and adds two extra branches to output similarity features $\boldsymbol{F}_{sim} \in \mathbb{R}^{N \times C_{out}}$ and

a confidence map $\boldsymbol{F}_{cf} \in \mathbb{R}^{N\times 1}$, alongside semantic features $\boldsymbol{F}_{sem} \in \mathbb{R}^{N\times C_{out}}$. A similarity matrix $\boldsymbol{S} \in \mathbb{R}^{N\times N}$ is used to formulate group proposals for instance segmentation. Each element S_{ij} of $\boldsymbol{S}$ is defined as:

$$S_{ij} = \|\boldsymbol{f}_{sim_i} - \boldsymbol{f}_{sim_j}\|^2 \tag{2.9}$$

where $\boldsymbol{f}_{sim_i}, \boldsymbol{f}_{sim_j} \in \mathbb{R}^{C_{out}}$ represent feature vectors of $\boldsymbol{F}_{sim}$. Since $\boldsymbol{S}$ generates N potential instance proposals, $\boldsymbol{F}_{cf}$ is used to filter out proposals with confidence scores below a threshold. The remaining points are assigned to instances based on $\boldsymbol{S}$.

X. Wang et al. [2019] and Lin Zhao and Wenbing Tao [2020] enhanced the presentations of $\boldsymbol{F}_{sim}$, by fusing it with $\boldsymbol{F}_{sem}$, allowing $\boldsymbol{F}_{sim}$ to benefit from the semantic information. Mean-shift clustering [Comaniciu and Meer 2002] is employed to group instances, replacing the reliance on confidence maps. Similarly, Lahoud et al. [2019] utilized mean-shift clustering, but adopted voxelized representations instead of raw point clouds for efficiency.

L. Jiang et al. [2020] introduced a dual-set point grouping method, where the network predicts a per-point offset vector $\Delta\boldsymbol{p}_i = (\Delta x_i, \Delta y_i, \Delta z_i)$ to shift each point toward the centroid of its corresponding object instance. This produces a shifted point cloud $\boldsymbol{P}_{shift}$, which helps distinguish nearby objects and improves clustering when combined with the original point cloud. Similarly, S. Chen et al. [2021] used $\boldsymbol{P}_{shift}$ for clustering points with the same predicted semantic labels and within a predefined spatial bandwidth.

Liang et al. [2021] proposed a tree-structured approach, initializing all points as leaf nodes. Each node contains features extracted from the backbone network, predicted semantic scores, and $\Delta\boldsymbol{p}_i$. By fusing three different pieces of information into a single nodal representation, the similarity is measured pairwisely on leaf nodes for hierarchically merging the tree, terminating when specific conditions are met. The resultant tree's leaf nodes correspond to object instances.

However, previous methods often rely on heuristic clustering algorithms to group instances. To overcome this, H. Jiang et al. [2020] introduced an end-to-end framework that directly predicts instance labels using an instance score matrix

$\boldsymbol{W} \in \mathbb{R}^{N \times K}$, in which K represents the number of instances candidates. Duplicate instances are eliminated using a predicted mask $M \in \{0,1\}^K$, where each dimension indicates whether a candidate is valid or not.

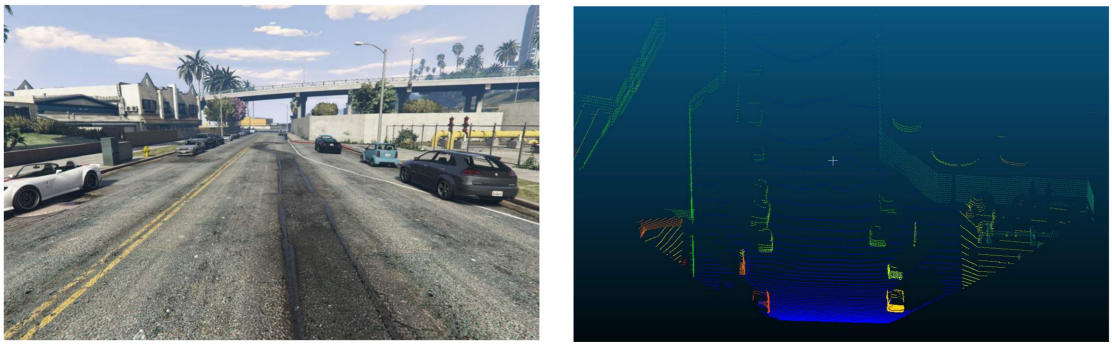
Schult et al. [2023] proposed Mask3D, leveraging the Transformer architecture. Mask3D uses K instance queries $\{\boldsymbol{q}_i \mid \boldsymbol{q}_i \in \mathbb{R}^{C_{in}}, i = 1, \ldots, K\}$ to compute the cross attention with the point feature $\boldsymbol{f}_j$ $(j = 1, \ldots, N)$. The output values are thresholded at 0.5 to produce a binary mask, indicating whether a point $\boldsymbol{p}_j$ belongs to the i-th instance or not. Although Mask3D achieves high accuracy in instance segmentation, its reliance on pairwise similarity computations by utilizing Transformer introduces significant computational overhead when applied to large-scale infrastructure point clouds, which restricts its feasibility for real-world scenarios.

2.1.4 3D point cloud synthetic dataset

Collecting and labelling 3D point cloud datasets are non-trivial compared to 2D image datasets. Point clouds require specialized equipment like LiDAR or scanners, which are costly and complex to operate in both pre- and post-processing. Annotating 3D data involves labelling individual points in an unstructured space, making it labour-intensive and error-prone, especially for large-scale scenes. This scarcity of labelled data limits development in 3D DL, necessitating techniques of synthetic data generation. This section reviews the development of synthetic point cloud datasets, initially introduced to address data scarcity in autonomous driving, and examines their adaptation for assisting the infrastructure point cloud segmentations.

Sythetic dataset in autonomous driving

Pioneering works [B. Wu et al. 2018; B. Wu et al. 2019] utilized the popular video game *Grand Theft Auto V* (*GTA-V*) to synthesize large-scale training data from urban environments. To generate synthetic point clouds, B. Wu et al. [2018] and B. Wu et al. [2019] implemented a LiDAR simulator within *GTA-V* that included automatic labelling capabilities, as illustrated in Figure 2.4a and b. Their experimental results showed a significant performance boost when training models on synthetic data and validating models on real-world datasets.



(a) *GTA-V* city scene (b) Synthetic point cloud

Figure 2.4: Visualizations of synthesizing point clouds via GTA-V[B. Wu et al. 2018]. (a) represents the game scene from GTA-V and (b) is the corresponding point cloud collected by the LiDAR simulator.

However, *GTA-V* employs only simple geometric models for pedestrians (e.g., cylinders), which fail to accurately represent real-world shapes. To address this limitation, researchers [Jeong et al. 2022; Griffiths and Boehm 2019; Fang et al. 2020] have explored using advanced simulators such as CARLA [Dosovitskiy et al. 2017] to generate realistic 3D point clouds under various conditions, including diverse weather, lighting, and traffic scenarios. In CARLA, object geometries (e.g., traffic cones, strollers, pedestrians, tricycles, and trees) are rendered via high-accuracy CAD models, ensuring accurate representations. Synthetic point clouds are collected by equipping simulated vehicles with virtual LiDAR sensors, mimicking real-world data acquisition processes.

Sythetic dataset in resolving data scarcity in infrastructure segmentation

While simulators such as CARLA have been widely adopted in autonomous driving for generating synthetic point clouds, their applicability to synthesize infrastructure point clouds is limited due to their focus on dynamic environments (e.g., terrestrial LiDAR operating in motion), which results in low-resolution point clouds that fail to capture the fine geometric details of the infrastructure.

Instead, researchers such as Pierdicca et al. [2019] and Morbidoni et al. [2020] have utilized Blender, an open-source 3D creation suite, to construct detailed 3D models by assembling basic structural components (shown in Figure 2.5a-c) into complex heritage structures (shown in Figure 2.5d-f). The corresponding point clouds of these heritage models are generated by extracting vertices from the Blender mod-

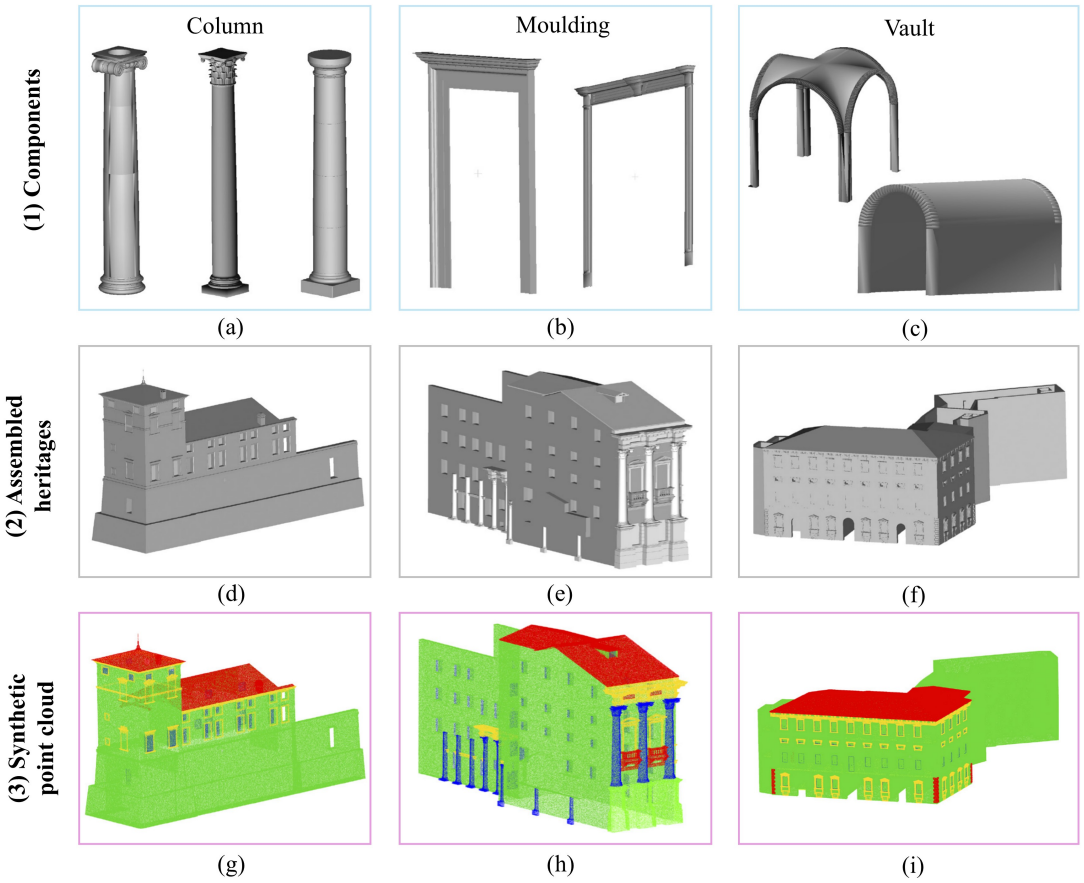


Figure 2.5: Visualizations of synthesizing point clouds [Pierdicca et al. 2019; Morbidoni et al. 2020] via Blender. (a)-(c) represents exemplified basic heritage elements; (d)-(f) show cultural heritage point clouds assembled from basic components, and (g)-(i) are the corresponding point cloud synthesised by the LiDAR simulator.

els. Building on this approach, Narazaki et al. [2021] developed synthetic datasets for reinforced concrete railway viaducts. To approach the realistic point clouds, textures (e.g., undamaged and damaged concrete, rail, fence, and pole) are mapped onto structure surfaces to provide RGB colour information rather than geometry.

However, these methods generate synthetic point clouds by uniformly sampling points from object surfaces, failing to capture features of real LiDAR scans such as scanning noise, uneven point distribution, and occlusions. To overcome these limitations, the LiDAR simulator Blensor [Gschwandtner et al. 2011], which integrates seamlessly with Blender, has been adopted to simulate real-world LiDAR characteristics. Recent studies [L. Yang et al. 2024; W. Lin et al. 2024c] have demonstrated that Blensor captures more realistic point clouds from various infrastructure scenarios by incorporating the features of real LiDAR into the Blensor.

2.1.5 3D point cloud segmentation in Infrastructure and their challenges

The techniques for point cloud segmentation can be broadly categorized into three main approaches: (1) traditional feature-based methods, (2) machine learning-based methods, and (3) DL-based methods. In this section, we review the evolution of these methods, highlighting their strengths, limitations, and applications in the segmentation of infrastructure point clouds.

Traditional feature-based methods

Traditional feature-based methods rely on handcrafted features and heuristic pipelines, often built upon presumptions about the geometry of infrastructures. For example, Riveiro et al. [2016] used normal vectors to hierarchically segment masonry bridge point clouds into individual components. By calculating the elevation angle and azimuth for each point based on its normal vector, the elevation angle is first used to distinguish between vertical walls (e.g., spandrel walls, piers, cutwaters) and non-vertical components (e.g., arches, pathways, cutwater hats). To further separate vertical walls, the azimuth values are embedded into a histogram, where the horizontal axis represents azimuths, and the vertical axis indicates the point counts. Peaks in the histogram are then analyzed, with the highest peak presumed to correspond to spandrel walls, as these typically constitute the largest proportion of points in vertical walls. The remaining segmentation is carried out using additional prior knowledge of masonry bridge geometry.

Similarly, R. Lu et al. [2019a] leveraged fundamental bridge design rules, such as topological constraints, to segment reinforced concrete (RC) bridge point clouds. Given an arbitrarily oriented bridge point cloud, the dataset is first aligned so that the horizontal alignment of the bridge is approximately parallel to the global x-axis. The bridge point cloud is then sliced along the x-axis into multiple pieces for further segmentation. The bridge point cloud is then sliced along the x-axis into multiple sections for segmentation. These slices are grouped into deck assemblies and pier assemblies based on the assumption that piers are significantly higher than decks. Pier caps are further extracted from pier assemblies by identifying elevations differing from the piers. The remained segmentation is performed using additional geometric assumptions specific to RC bridges.

While these heuristic pipelines achieve robust and accurate results for the specific types of infrastructure point clouds they are designed for, their reliance on fixed assumptions and geometric priors limits their generalizability to other types of infrastructure.

Machine learning-based methods

Machine learning-based methods leverage manually crafted features combined with statistical models or classifiers to segment 3D point clouds. Grilli et al. [2019] and Teruggi et al. [2020] trained multiple Random Forest (RF) classifiers [Breiman 2001] to perform hierarchical segmentation of cultural heritage point clouds with complex geometries, progressing from coarse- to fine-level detail. The geometric features used to train the RF classifiers are derived from combinations of eigenvalues (e.g., $\lambda_1 > \lambda_2 > \lambda_3$) of the covariance matrix [Chehata et al. 2009; Blomley et al. 2014] computed within the local neighbourhoods of each point. These features, which characterize the local geometry, include linearity, planarity, sphericity, and anisotropy, and are expressed as:

$$\begin{aligned}
&\text{Linearity:} && L_\lambda &&= && \tfrac{\lambda_1-\lambda_2}{\lambda_1}, \\
&\text{Planarity:} && P_\lambda &&= && \tfrac{\lambda_2-\lambda_3}{\lambda_1}, \\
&\text{Sphericity:} && S_\lambda &&= && \tfrac{\lambda_3}{\lambda_1}, \\
&\text{Anisotropy:} && A_\lambda &&= && \tfrac{\lambda_1-\lambda_3}{\lambda_1}, \\
&\text{Omnivariance:} && O_\lambda &&= && \sqrt[3]{\textstyle\prod_{j=1}^{3}\lambda_j}
\end{aligned} \tag{2.10}$$

To enhance segmentation accuracy, Grilli and Remondino [2020] also integrated RGB information into the RF training process, alongside geometric features. The integration of colour components significantly improved the differentiation of points with different materials, such as distinguishing windows from walls.

Alternatively, Xia et al. [2022] introduced the Signature of Histograms of Orientations (SHOT), a geometric descriptor, to capture the local geometry of RC bridge point clouds. The SHOT features were used to train a simple artificial neural network for segmentation, followed by the application of DBSCAN, an unsupervised clustering method, to filter noise and group points into individual components.

These approaches mark a shift from purely heuristic pipelines by learning patterns in the data, enabling better adaptability to variations in point cloud characteristics. However, their reliance on feature engineering still requires significant domain expertise and limits their scalability to more complex scenarios.

DL-based methods

DL-based methods are purely data-driven, providing an end-to-end approach for 3D point cloud segmentation by automating feature extraction. The rapid advancement of 3D DL techniques has significantly boosted research in infrastructure point cloud segmentation, leading to breakthroughs across various domains:

(1) **Tunnel:** W. Lin et al. [2024b] and W. Lin et al. [2024a] introduced the Seg2Tunnel dataset, comprising 1,300 annotated tunnel rings, to facilitate the development of 3D DL models for segmentation. By combining RandLA-Net [Hu et al. 2020] and an instance segmentation method [Kolodiazhnyi et al. 2024], W. Lin et al. [2024b] proposed a two-stage framework for fully automatic instance segmentation, achieving accurate segmentation results. Ji et al. [2022] proposed an encoder-decoder DL method for multi-class object segmentation, including seepage detection in tunnel point clouds. Additionally, Soilán et al. [2020] compared PointNet [Qi et al. 2017a] and KPConv [Thomas et al. 2019] for railway tunnel segmentation, with KPConv outperforming PointNet, achieving F1-scores over 97% for most classes.

(2) **Building:** Haznedar et al. [2023] investigated the implementation of PointNet for point cloud segmentation in heritage buildings. While PointNet struggled with accuracy due to deformations and deterioration in existing heritage structures, training the algorithm with synthetic data provided high accuracy. Pierdicca et al. [2020] utilized the Dynamic Graph CNN (DGCNN) [Yue Wang et al. 2019] to perform segmentation on 11 cultural heritage point clouds with complex geometries. The results are compared to the PointNet and PointNet++ [Qi et al. 2017b] which demonstrate that DGCNN achieves SOTA performance. Matrone et al. [2020] further improved DGCNN by incorporating RGB colours, normals, and geometric features (e.g., L_λ, P_λ) into the input, significantly boosting segmentation accuracy in heritage point clouds. J. Ma et al. [2020] demonstrated that augmenting small real point

cloud datasets with synthetic point clouds generated from Building Information Models can significantly improve segmentation performance on PointNet and DGCNN.

(3) **Bridge:** H. Kim et al. [2020] adopted PointNet for automatic bridge component recognition, effectively handling complex bridge geometries and background regions, and improving accuracy through consensus labeling. X. Yang et al. [2022] proposed a weighted superpoint graph method leveraging PointNet and graph neural networks. To reduce memory consumption in training on large-scale point clouds, points were grouped into superpoints based on similarities in geometric features (e.g., L_λ, P_λ, etc.), achieving superior performance compared to existing models. Furthermore, Lamas et al. [2024] introduced synthetic point cloud datasets for truss bridges to address the scarcity of training data. These synthetic datasets were used to train an adapted JSNet [Liangliang Zhao and Weidong Tao 2020], achieving high accuracy in semantic and instance segmentation.

Unlike traditional or machine learning-based approaches that rely on handcrafted features, DL methods learn hierarchical representations directly from raw data, capturing complex spatial patterns and relationships. With the advent of architectures like PointNet [Qi et al. 2017a] and its successors, these methods have achieved SOTA performance in processing unstructured point cloud datasets, making them highly adaptable to various infrastructure segmentation tasks. However, the adoption of 3D DL in infrastructure applications is still limited by high GPU computational demands and the scarcity of labelled datasets, posing significant challenges for further implementation in real-world cases.

2.2 Defect inspection in infrastructure: classification, localization, and quantification

Traditional feature-based computer vision (CV) methods have long been employed for defect detection in infrastructure, relying on edge detector algorithms [Abdel-Qader et al. 2003; Nishikawa et al. 2012; Yamaguchi et al. 2008], to extract superficial defects such as cracks, and corrosion, etc. While these methods offer interpretability and are computationally efficient, they often struggle with robustness in

complex and real-world scenarios. Variations in lighting, shadows, textures, and environmental noise significantly impact their performance. Additionally, these methods are limited in their ability to generalize across different defect types or infrastructure materials without extensive fine-tuning.

In recent years, the rapid development of DL neural networks in CV has transformed defect detection by shifting from handcrafted feature engineering to data-driven feature learning. These networks automatically extract hierarchical and discriminative features from raw data, enabling accurate classification, localization, and quantification of defects. In image-based applications, DL methods have outperformed traditional approaches, achieving remarkable accuracy and robustness across diverse tasks. However, for 3D point clouds, the robustness of features extracted by DL architectures remains a challenge due to the lack of high-volume datasets, such as ImageNet, which contains millions of images for training generalized feature extractors. As a result, both handcrafted and learned features are utilized to detect defects in 3D applications.

2.2.1 Supervised DL methods for defect detection in infrastructure

Supervised DL methods leverage annotated datasets to train models by learning patterns from labelled training data. These models are designed to predict labels that are consistent with the provided samples, enabling them to generalize to unseen structural defects with high accuracy. This section reviews the application of supervised DL methods in defect detection, focusing on their implementation in 2D images and 3D point clouds.

Defect detection in 2D images

The pioneering work by Cha et al. [2017] proposed training a simple CNN architecture as a binary classifier to assess concrete cracks using cracked or intact concrete images with a resolution of 256×256 pixels. The trained CNN achieved approximately 98% accuracy on the test dataset, significantly outperforming traditional edge detection methods (e.g., Canny [Canny 1986] and Sobel [Kanopoulos et al. 1988]). The main advantage is its automatic calculation of damage-sensitive features during the training process of the CNN. However, the method relied on

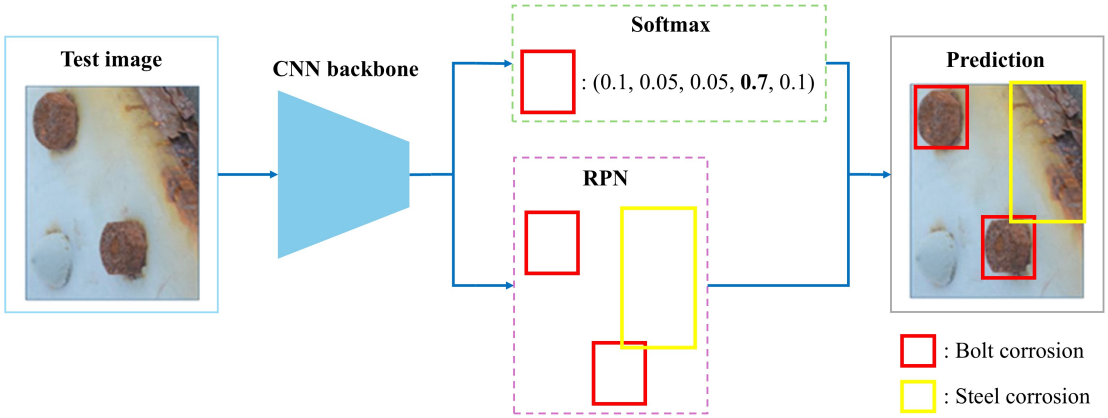


Figure 2.6: The adoption of Fast R-CNN in prediction structural defects [Cha et al. 2018]. Features of input images are extracted using a CNN backbone and then utilized to predict bounding boxes delineating defects through an RPN. Another branch simultaneously predicts the labels for these bounding boxes by processing the features through a softmax layer.

a sliding window approach for damage localization, which posed challenges in determining the optimal sliding window size, especially when testing images varied in size and scale.

To address these limitations, Cha et al. [2018] adopted the Fast Region-based CNN (Fast R-CNN) [Ren et al. 2016], which enables simultaneous object detection, localization, and classification within input images, as illustrated in Figure 2.6. Fast R-CNN uses a Region Proposal Network (RPN) branch to generate bounding boxes as object proposals. These proposals are then classified through another softmax branch to predict scores, achieving over 90% accuracy in detecting five types of structural defects on steel and concrete surfaces. Fast R-CNN has since been extended to other applications, including the inspection of welds in steel structures [Oh et al. 2020], pavement cracks [Ibragimov et al. 2022], steel surface defects [Wei et al. 2020], and building damages [K. Lee et al. 2020; Ghosh Mondal et al. 2020]. These applications demonstrate its versatility and robustness across diverse infrastructure types.

However, Fast R-CNN has a notable limitation: it does not provide pixel-level segmentation for precise defect size quantification, relying instead on bounding boxes to approximate defect delineation. To overcome this, Attard et al. [2019a] introduced Mask R-CNN [K. He et al. 2017], which modified Fast R-CNN by replacing the RPN with a new branch to generate pixel-level masks for each instance. This enhancement enables precise localization and quantification of concrete cracks.

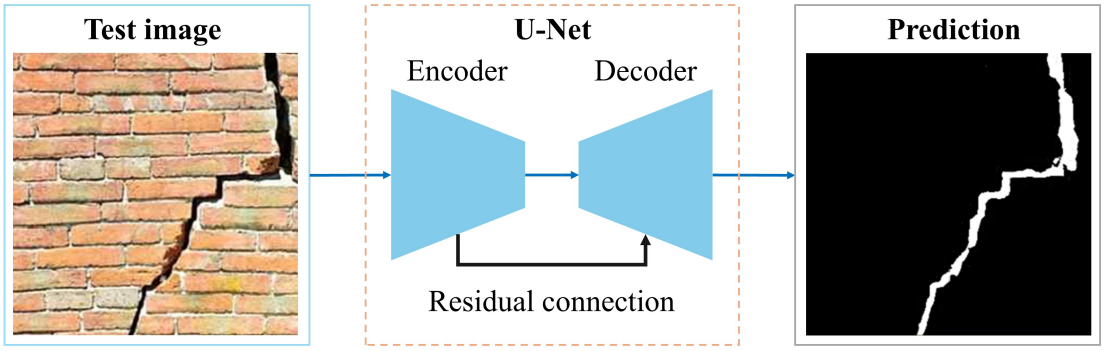


Figure 2.7: The adoption of U-Net in prediction structural defects [Huyan et al. 2020]. Images are input into U-Net with an encoder for downsampling and a decoder to upsample the image back to its original size for predicting labels at the pixel level. The residual connection is used to bridge high-level semantic features from the encoder with spatially detailed features in the decoder.

Subsequent studies have demonstrated the effectiveness of Mask R-CNN in identifying defects across various infrastructures, such as rail surfaces [Guo et al. 2021], road surfaces [Kulambayev et al. 2023; Y. He et al. 2022], concrete surfaces [C. Huang et al. 2024], and water tanks [Z. Wu et al. 2020]. Compared to Fast R-CNN, Mask R-CNN performs pixel-level segmentation, which allows for accurate defect quantification, a critical improvement over bounding-box-based methods.

In addition to Mask R-CNN, U-Net [Ronneberger et al. 2015] has emerged as a popular architecture for pixel-level segmentation in defect detection. U-Net employs an encoder-decoder structure, where the encoder extracts hierarchical features, and the decoder reconstructs spatial information to produce pixel-level masks as shown in Figure 2.7. Dang et al. [2022] adopted U-Net to perform automatic masonry crack detection, generating binary masks (Figure 2.7). Similarly, Shamsabadi et al. [2022] adopted TransUNet [J. Chen et al. 2021], a U-Net architecture integrating Transformer blocks [Vaswani et al. 2017] into the encoder and decoder, to enhance robustness against noisy signals such as Gaussian noise and luminance changes, achieving accurate predictions.

U-Net has also been widely applied to detect and segment defects in various infrastructure types, including concrete cracks [Zhenqing Liu et al. 2019; Huyan et al. 2020], road cracks [Arya et al. 2021; Augustaukas and Lipnickas 2019], steel corrosion [Das et al. 2024], and tunnel cracks [G. Li et al. 2020a]. Compared to Mask R-CNN, U-Net directly predicts pixel-level masks for the entire image without relying on region proposals, making it computationally efficient for dense

segmentation tasks.

Defect inspection in 3D point clouds

Despite the rapid development of DL for 2D image-based defect inspection, its adoption in 3D point clouds remains relatively limited. This limitation can be attributed to several factors:

(1) **Labour-intensive labelling**: Labelling 3D point clouds is significantly more labour-intensive and time-consuming compared to 2D images. It requires point-level labelling in large-scale datasets, often necessitating specialized domain expertise, which increases the cost and effort involved.

(2) **High computational costs**: Processing 3D point clouds demands substantial computational resources, especially for large-scale infrastructure datasets. Memory-intensive operations and high computational overhead hinder real-time or large-scale implementations for defect inspection.

(3) **Limited advantages over 2D methods**: Current applications of 3D point clouds in defect inspection predominantly focus on detecting superficial defects, offering little to no advantage over existing 2D methods in terms of accuracy or functionality. This overlap in capabilities diminishes the practical value of using point clouds for defect inspection.

Nevertheless, with the increasing availability of LiDAR and photogrammetry data, supervised learning methods have been extended from 2D image analysis to 3D point cloud processing for defect detection. For example, the early attempt of Nasrollahi et al. [2019] trained PointNet [Qi et al. 2017a] on a concrete bridge point cloud dataset to predict defects, achieving an accuracy of 65.9%. To improve accuracy, Bolourian et al. [2023] incorporated additional input features such as RGB, normal vectors, and elevation into the input and train by PointNet++ [Qi et al. 2017b], leading to significant performance improvements. Bahreini and Hammad [2024] utilized a DGCNN [Yue Wang et al. 2019] to identify cracks and spalling in concrete bridge point clouds. To address the challenge of unbalanced datasets, where only 14% of points correspond to defects, a weighted cross-entropy loss function was utilized, ensuring the model prioritized defect points during training.

By leveraging the attention mechanism, H. Wan et al. [2023] proposed the Bridge

Detection Transformers (BR-DETR) architecture, which combines 2D deformable convolutions and locally enhanced feed-forward layers for improved feature extraction in bridge defect detection. BR-DETR demonstrated superior performance, achieving higher accuracy than traditional CNN-based methods. Likewise, J. Fan et al. [2024] introduced the Point Attention Network (PAN) to address the irregularity and sparsity of point cloud data. By integrating dual-attention mechanisms, PAN enhanced feature representations and achieved SOTA results in detecting pavement cracks across large-scale datasets.

2.2.2 Overview of unsupervised learning for anomaly detection

However, the success of supervised learning methods depends on the availability of well-labelled datasets of infrastructure defects, which are often labour-intensive and costly to obtain. This reliance on labelled datasets remains a significant limitation, as it directly impacts the generalizability and performance of supervised models in diverse and real-world applications. Therefore, unsupervised DL anomaly detection, which does not require the labelling of datasets, has emerged as a promising alternative. This section first introduces the definition of the anomaly detection problem and how it can be resolved by unsupervised DL methods. Two main categories of unsupervised DL anomaly detection methods are reviewed with adaptions on 2D images and 3D point clouds in computer vision tasks. Finally, we reviewed the implementations of anomaly detection in infrastructure defect inspection.

Problem statement on anomaly detection

Given a dataset $\mathcal{D} = \{d_1, d_2, \ldots, d_n\}$, where most samples d_i belong to the normal class $\mathcal{N}$, the objective is to identify samples $d_j \in \mathcal{A}$, where $\mathcal{A}$ represents the anomalous data and $\mathcal{A} \cap \mathcal{N} = \emptyset$.

Unlike traditional classification problems, where predefined categories are assigned to inputs, anomaly detection methods focus on learning the statistical distribution of $\mathcal{N}$ and identify patterns or observations in $\mathcal{D}$ that deviate significantly from the normal data. This is especially relevant in scenarios where defects are rare or poorly defined, making it impractical to rely on fully supervised approaches due

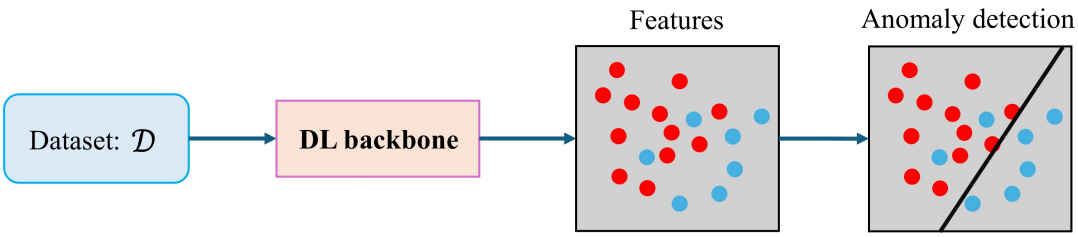


Figure 2.8: Framework of anomaly detection method by using DL backbones for only extracting features from the dataset $\mathcal{D}$. The extracted features are further classified by other unsupervised learning methods such as the support vector machine.

to the high data imbalance. Instead, anomaly detection leverages the assumption that the statistical distribution of normal data is different from the anomalous data. Therefore, data points that do not conform to the statistical distribution of normal data are flagged as anomalies.

Hybrid anomaly detection using DL for feature extraction

This category of methods leverages DL backbones to extract feature representations from data, which are then used for downstream anomaly detection, as illustrated in Figure 2.8. The anomaly detection step operates on these extracted features independently of the DL feature extraction step. This approach assumes that the feature representations obtained via DL backbones retain discriminative information for distinguishing anomalous data from normal data.

Tudor Ionescu et al. [2017] explored online anomaly detection in video data, introducing a binary classifier trained iteratively to differentiate between one set of video frames and its subsequent frames within a sliding window. The framework assumes the first set of video frames represents normal data and evaluates deviations in subsequent frames. High classification accuracy during training indicates anomalous frames, and vice versa. Features are extracted using a VGG network [Chatfield 2014], pre-trained on the ImageNet benchmark [Russakovsky et al. 2015], to ensure effective feature extraction. Similarly, Andrews et al. [2016] leveraged transfer learning by fine-tuning a pre-trained VGG model on the target dataset. Anomaly detection is performed on the features extracted by the fine-tuned VGG using a one-class support vector machine (SVM) [Hearst et al. 1998].

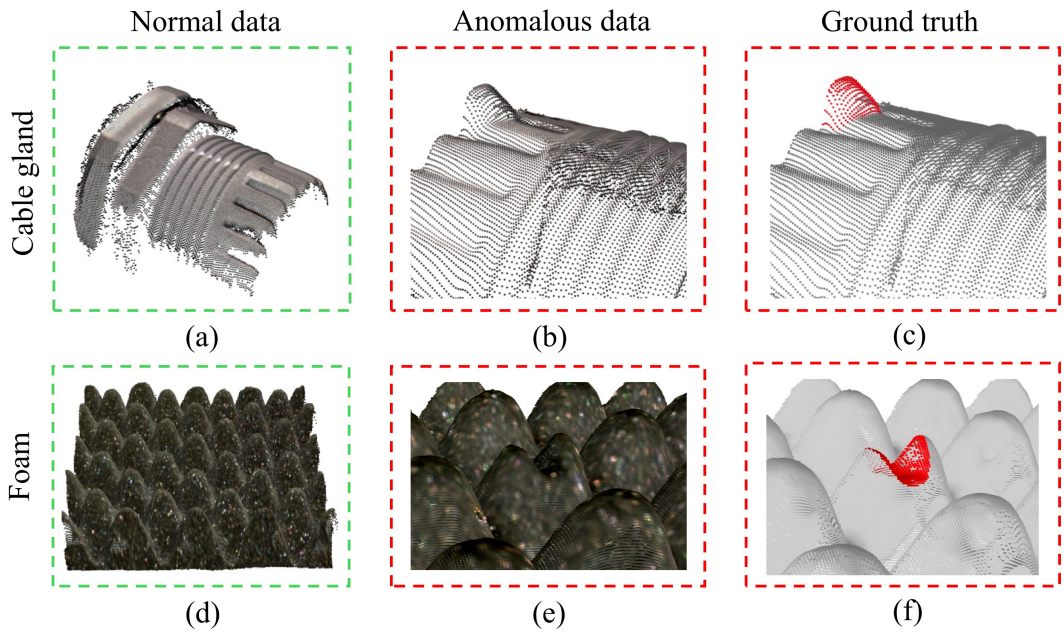


Figure 2.9: Examples from the cable gland and foam categories in the MVTec 3D-AD dataset [Bergmann et al. 2021]. (a) A normal point cloud of the cable gland with RGB values projected onto its surface. (b) and (c) represent an anomalous sample of the cable gland and the ground truth highlighting the anomalous region in red. (d) A normal foam sample. (e) and (f) corresponds to an anomalous sample and its ground truth.

Instead of relying solely on pre-trained models, D. Xu et al. [2015] proposed training three autoencoder networks to learn low-dimensional features corresponding to appearance, motion (e.g., normal flow derived from video frames), and the fusion of appearance and motion representations. An ensemble of three one-class SVMs was then employed to jointly compute anomaly scores on these learned representations.

Despite their robustness in anomaly detection, these methods face limitations. They struggle to capture image context across multiple scales and often suffer from inefficiencies in training and inference due to the extensive feature representations generated from normal data. To address these challenges, Roth et al. [2022] introduced *PatchCore*, an effective framework for anomaly detection in 2D images. *PatchCore* employs ResNet-like architectures [K. He et al. 2016], pre-trained on ImageNet, to extract multi-scale feature representations from images. These feature maps are aggregated to produce the output feature representations, effectively preserving localized information.

For all $d_i \in \mathcal{N}$, the pre-trained ResNet extracts feature representations that are stored in the *PatchCore* memory bank $\mathcal{M}$. To prevent $\mathcal{M}$ from growing exces-

sively large as the dataset size increases, *PatchCore* incorporates a coreset selection strategy to compress $\mathcal{M}$. This strategy aims to find a subset $\mathcal{M}_C$ such that computations over $\mathcal{M}$ are effectively approximated by those over $\mathcal{M}_C$ [Agarwal et al. 2005]. While the exact computation of $\mathcal{M}_C$ is an NP-hard problem [Wolsey and Nemhauser 2014], *PatchCore* employs an iterative greedy approach, as suggested by Sener and Savarese [2017], to approximate the ground truth optimization objective. Using $\mathcal{M}_C$, feature representations extracted from test images are matched to feature representations $\mathcal{M}_C$ by measuring minimum Euclidean distances pairwisely. These distances serve as anomaly scores s, where higher values indicate a greater degree of abnormality.

Horwitz and Hoshen [2023] adapted the *PatchCore* framework from 2D images to 3D point cloud data and validated its performance on the MVTec 3D-AD dataset [Bergmann et al. 2021]. The MVTec 3D-AD dataset, which was introduced to address data scarcity in point cloud anomaly detection, comprises over 4,000 high-resolution 3D scans of industrially manufactured products across 10 categories, as shown in Figure 2.9a-f. Five of the object categories in their dataset exhibit considerable natural variations from sample to sample, including bagel, carrot, cookie, peach, and potato. Three more objects, foam, rope, and tire, have a standardized appearance but can be easily deformed. The two remaining objects, cable gland and dowel, are rigid. Defects were devised and fabricated to simulate the real-world inspection scenarios as shown in Figure 2.9b and e.

Notably, the MVTec 3D-AD dataset provides RGB-D images, in which 3D information is represented as depth maps (organized data) rather than actual point clouds (unorganized data). The extracted features from the RGB-D images combine RGB and geometric information. Specifically, RGB features are extracted using neural networks pre-trained on ImageNet, leveraging their robust feature extraction capabilities.

Given the absence of 3D neural networks pre-trained on point cloud datasets with a similar scale as ImageNet, Horwitz and Hoshen [2023] incorporated the handcrafted Fast Point Feature Histograms (FPFH) [Rusu et al. 2009] to capture local geometric features, which involved converting depth maps into an unorganized data format by mapping each pixel's grid location to x and y coordinates and representing the depth value as z. The resulting framework, named *Back to the*

Feature (BTF), enabled a comprehensive analysis by integrating RGB (e.g., $\boldsymbol{f}_{2D}$) and geometric features (e.g., $\boldsymbol{f}_{3D}$). BTF achieved SOTA performance on pixel-wise anomaly detection on depth maps, demonstrating the effectiveness of combining pre-trained 2D neural networks with traditional 3D feature descriptors.

Cao et al. [2023] introduced the Complementary Pseudo Multimodal Feature (CPMF) method, a novel approach that combines $\boldsymbol{f}_{3D}$ (extracted by FPFH) and $\boldsymbol{f}_{2D}$ (extracted by pre-trained 2D neural networks [Cao et al. 2022; Q. Wan et al. 2021]). CPMF begins by generating the pseudo-2D modality through the multi-view rendering of 3D point clouds, producing 2D images from various angles. These images are processed using pre-trained 2D networks to extract rich semantic feature maps. Simultaneously, FPFH captures local geometrical details of the point cloud. $\boldsymbol{f}_{2D}$ and $\boldsymbol{f}_{3D}$ are aligned and concatenated to form a unified pseudo multimodal representation to replace the original feature representation in *PatchCore*. This method achieved SOTA performance on the MVTec 3D-AD dataset, demonstrating the advantages of fusing different modalities.

Chu et al. [2023] proposed a shape-guided dual-memory learning framework for anomaly detection, focusing on the fusion of colour and geometric modalities. This method leverages PointNet [Qi et al. 2017a], trained in an unsupervised manner on the MVTec 3D-AD dataset, utilizing signed distance functions (SDFs) to capture local 3D geometries. The framework constructs dual memory banks: the SDF memory bank $\mathcal{M}_{3D}$ (stores features extracted from PointNet) and the RGB memory bank $\mathcal{M}_{2D}$ (stores feature extracted from ResNet). The anomaly scores s are computed separately for $\boldsymbol{f}_{2D}$ and $\boldsymbol{f}_{3D}$ and subsequently fused using max pooling to enable pixel-wise anomaly detection on depth maps. This approach effectively integrates complementary information from both geometric and colour modalities, enhancing the robustness and accuracy of anomaly detection.

To further enhance feature fusion across different modalities, Yue Wang et al. [2023] proposed the Multi-3D-Memory (M3DM) framework, designed to learn the intrinsic relationships between 2D and 3D features ($\boldsymbol{f}_{2D}$ and $\boldsymbol{f}_{3D}$, respectively). M3DM leverages a Vision Transformer [Dosovitskiy et al. 2020] pre-trained on ImageNet [Russakovsky et al. 2015] to extract $\boldsymbol{f}_{2D}$ and a Point Transformer [H. Zhao et al. 2021a] pre-trained on ShapeNet [Chang et al. 2015] to extract $\boldsymbol{f}_{3D}$. Both feature types are trained in an unsupervised manner using contrastive loss

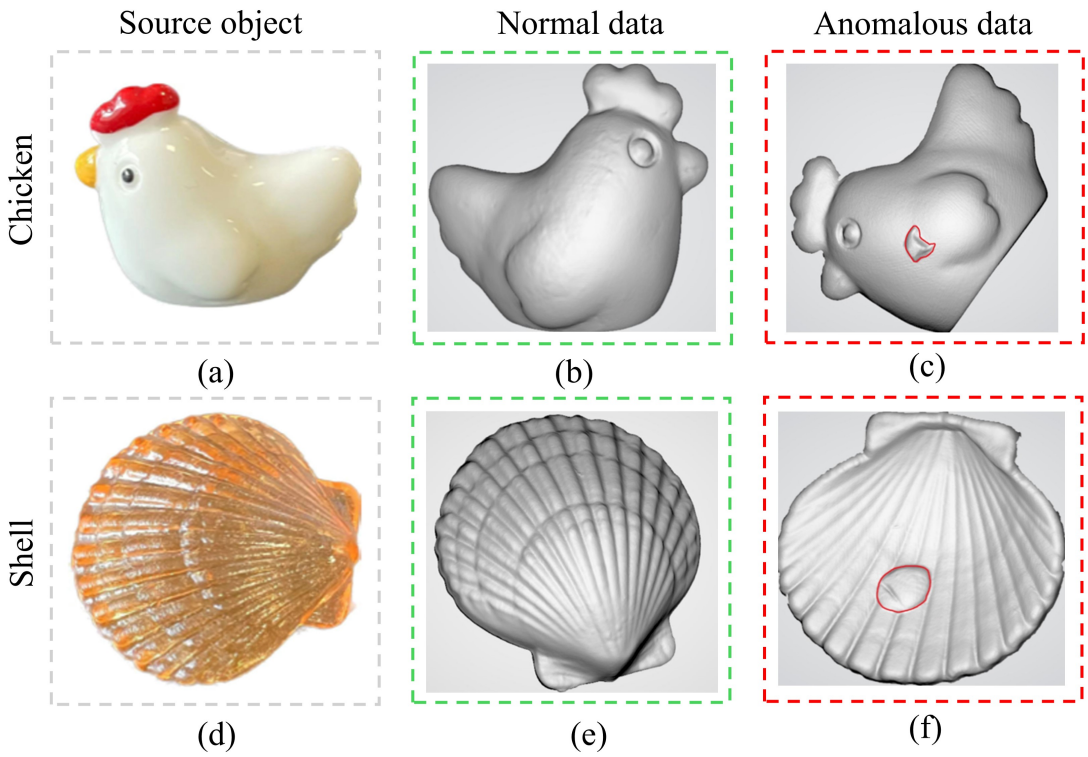


Figure 2.10: Examples from the chicken and shell categories in the Real3D-AD dataset [J. Liu et al. 2024]. (a) A product of the chicken. (b) and (c) represent the scanned normal data and a redundancy anomaly highlighted in a red inset. (d) A product of the shell. (e) and (f) corresponds to the normal data and an incompleteness defect highlighted in a red inset.

[Oord et al. 2018], encouraging $\boldsymbol{f}_{2D}$ and $\boldsymbol{f}_{3D}$ in the same locations to maximize mutual information. The resulting fused feature, $\boldsymbol{f}_{fs}$, is obtained by concatenating $\boldsymbol{f}_{2D}$ and $\boldsymbol{f}_{3D}$. Apart from $\mathcal{M}_{2D}$ and $\mathcal{M}_{3D}$ proposed by Chu et al. [2023], M3DM also constructs an additional memory bank $\mathcal{M}_{fs}$ to store $\boldsymbol{f}_{fs}$ from normal data. s are computed independently from each memory bank and combined into a vector $\boldsymbol{s}_{concat}$. The final anomaly detection is performed using a one-class SVM, leveraging $\boldsymbol{s}_{concat}$ derived from all three memory banks to improve detection accuracy.

However, the MVTec 3D-AD dataset relies on depth maps to represent geometric information, capturing only a single view of the object and potentially introducing blind spots during inspection. To address these limitations, J. Liu et al. [2024] introduced the Real3D-AD dataset, comprising 1,254 high-resolution 3D scans across 12 products from real-world applications, each containing up to millions of points. These categories include airplane, car, candybar, chicken, diamond, duck, fish, gemstone, seahorse, shell, starfish, and toffees. The anomalies introduced can be classified into incompleteness and redundancy of point clouds compared

to normal data, which are shown in Figure 2.10a-f. Unlike the MVTec 3D-AD dataset, Real3D-AD offers complete, multi-view scans, ensuring comprehensive coverage and eliminating blind spots.

Building upon the dual-memory method proposed by Chu et al. [2023], J. Liu et al. [2024] developed the Reg3D-AD method, which employs two memory banks (e.g., $\mathcal{M}_{local}$ and $\mathcal{M}_{global}$) to store local and global features extracted from point clouds. The global features are derived from the coordinate values of each point cloud (e.g., x, y, and z values), and a registration process aligns the test data with the corresponding normal data to ensure proper coordinate alignment in normal regions. The local features are extracted via PointMAE [Pang et al. 2022], pre-trained on the ShapeNet dataset [Chang et al. 2015]. Experiments demonstrate that Reg3D-AD achieves SOTA performance, outperforming BTF, which serves as the benchmark method.

To reduce the computational cost of the Reg3D-AD method [J. Liu et al. 2024], Baozhu Zhao et al. [2024] proposed PointCore, an efficient framework for unsupervised 3D point cloud anomaly detection. Rather than processing the entire point cloud, PointCore employs PointMAE to extract $\boldsymbol{f}_{3D}$ from a subsampled point cloud, while $\boldsymbol{f}_{3D}$ for the remaining points are derived through interpolation methods. Unlike Reg3D-AD, PointCore utilizes a single memory bank to fuse local (coordinate-based) and global (PointMAE-based) features, effectively lowering computational cost. Furthermore, a ranking-based normalization technique is introduced to eliminate the scale and distribution difference of s derived from local and global features, ensuring an effective fusion of memory banks. These improvements enable PointCore to outperform Reg3D-AD on the Real3D-AD dataset in both efficiency and accuracy.

End-to-end anomaly detection via normal data representation learning

The methods in this category couple feature learning with anomaly scoring by optimizing a generic feature learning objective function to learn the representations of normal data. The DL networks are forced to capture key underlying data normalities, which can be represented as:

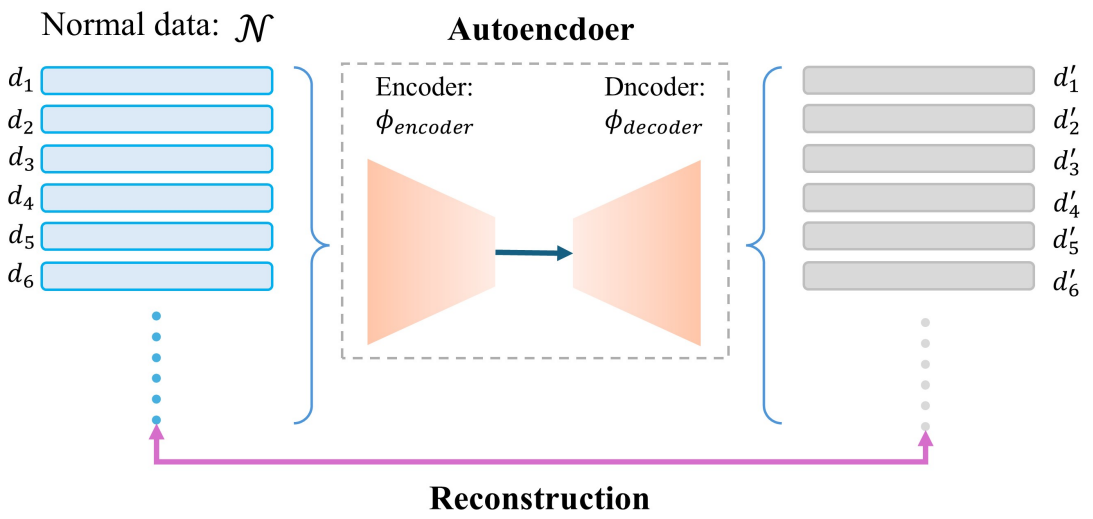


Figure 2.11: Architecture of the AE. The input normal data $\mathcal{N}$ is first mapped onto low-dimensional feature space by the encoder $\phi_{encoder}$ then recovered to the original input size by decoder $\phi_{decoder}$. The loss function can be defined by the reconstruction between input and output pairs d_i and $d_i^{'}$.

$$\mathcal{W}^* = \arg\min_{\mathcal{W}} \sum_{d_i \in \mathcal{N}} \mathcal{L}\left(\phi(d_i, \mathcal{W})\right) \tag{2.11}$$

$$s_{d_i} = f_{score}\left(d_i, \phi_{\mathcal{W}^*}\right) \tag{2.12}$$

where ϕ parameterized by $\mathcal{W}$ represents an unsupervised learning method for learning the statistical distribution and feature representations of normal data, $\mathcal{L}$ is a loss function related to different modelling approaches, and f_{score} is a scoring function which utilizes the $\phi_{\mathcal{W}^*}$ to calculate the anomaly score s_{d_i} from each input data $d_i \in \mathcal{D}$.

By leveraging the governing methodology, autoencoder (AE) [Liou et al. 2014], which aims to learn the low-dimensional feature representation on the given dataset via reconstructions, is used for anomaly detection. The architecture of the AE is shown in Figure 2.11, in which the input data $\mathcal{D}$ is first projected to low-dimensional feature space by the encoder $\phi_{encoder}$ and then projected back to original data dimension via decoder $\phi_{decoder}$. The model is trained to retain the information that is important in reconstructing the input data samples, in which the model can be optimized by the reconstruction loss:

$$\{\mathbf{\Theta}_d^*, \mathbf{\Theta}_e^*\} = \arg\min_{\mathbf{\Theta}_e, \mathbf{\Theta}_d} \sum_{d_i \in \mathcal{N}} \| d_i \; - \; \phi_{decoder} \left(\phi_{encoder}(d_i; \mathbf{\Theta}_e); \mathbf{\Theta}_d \right) \|^2 \tag{2.13}$$

where $\mathbf{\Theta}_e$ and $\mathbf{\Theta}_d$ represent the parameters of $\phi_{encoder}$ and $\phi_{decoder}$ correspondingly, and $\mathbf{\Theta}_d^*$ and $\mathbf{\Theta}_e^*$ are the optimized parameters. s_{d_i} can be regarded as reconstruction loss to indicate the anomaly scale, which is given by:

$$s_{d_i} = \| d_i - \phi_{decoder} \left(\phi_{encoder}(d_i; \mathbf{\Theta}_e^*); \mathbf{\Theta}_d^* \right) \|^2 \tag{2.14}$$

Hasan et al. [2016] implemented AE with CNN (AE-CNN) to predict the irregular motions on video frames. The input to AE-CNN is a handcrafted motion feature extracted from the frames, where the features correspond to normal motion patterns at each pixel. Anomaly scores are derived by summing the reconstruction loss across all pixels, enabling the detection of anomalous motion patterns.

However, relying on per-pixel reconstruction loss assumes that neighbouring pixel values are mutually independent, which is an unrealistic simplification. This approach is particularly sensitive to minor reconstruction inaccuracies, especially along edges, which can lead to disproportionately high residuals in the final loss calculation. To address this limitation, Bergmann et al. [2018] proposed using the Structural Similarity Index Measure (SSIM) [Z. Wang et al. 2004] to evaluate reconstruction accuracy. SSIM assesses deviations in statistical features (e.g., mean and covariance) between corresponding patches (subdividing images into regular grids) of the input and reconstructed images, rather than evaluating pixels independently. By leveraging SSIM, this method significantly improved anomaly detection performance in challenging real-world scenarios, demonstrating its effectiveness over per-pixel reconstruction metrics.

To further minimize the reconstruction loss, Ristea et al. [2022] proposed the Self-Supervised Predictive Convolutional Attentive Block (SSPCAB) for anomaly detection in images and videos. SSPCAB uses masked convolutions, where central regions of convolution kernels are intentionally masked, and channel attention mechanism to predict missing areas based on contextual information. SSPCAB block improves reconstruction-based anomaly detection by capturing both global and local structures. Integrated into AE-CNN, SSPCAB achieves superior detec-

tion and localization performance on the MVTec AD dataset [Bergmann et al. 2019], showing significant improvements.

By adapting autoencoders from 2D to 3D, the pioneering work [Bergmann et al. 2021] proposed to utilize a Voxel AE [Bengs et al. 2021] as a benchmark on the MVTec 3D-AD dataset. In this approach, RGB-D images are converted into the voxel data format to enable 3D reconstruction by the Voxel AE, with reconstruction loss quantified using the voxel-wise Mahalanobis distance [Mahalanobis 2018].

Despite its innovation, the Voxel AE achieved low detection accuracy and failed to preserve fine geometric details due to the limitations inherent in voxelization. Therefore, Bergmann and Sattlegger [2023] introduced the 3D Student-Teacher (3D-ST) framework to perform unsupervised anomaly detection directly in 3D point clouds. 3D-ST utilizes a RandLA-like [Hu et al. 2020] architecture to directly reconstruct 3D point clouds, in which the reconstruction loss is measured by the Chamfer distance [Pang et al. 2022] on local regions in corresponding locations. The 3D-ST framework uses a student-teacher paradigm [Hinton et al. 2015; Abbasi et al. 2020] where a teacher network (RandLA-like architecture), pre-trained in a self-supervised manner, extracts local geometric descriptors for 3D points by reconstructing their receptive fields. A student network, which shares the same architecture as the teacher network is subsequently trained to mimic these descriptors from normal data. During inference, anomalies are localized by analyzing regression errors between the teacher and student descriptors.

W. Li et al. [2024] introduced a novel synthetic dataset, Anomaly-ShapeNet, to advance 3D point cloud anomaly detection. This dataset was constructed by applying a 3D anomaly synthesis pipeline to ShapeNet [Chang et al. 2015] and includes 1,600 samples across 40 object categories with six typical anomaly types (e.g., bulge, concavity, hole). Unlike Real3D-AD [J. Liu et al. 2024], Anomaly-ShapeNet offers a broader variety of normal and anomalous data with more diverse anomaly types, enhancing its applicability for comprehensive anomaly detection tasks. To avoid potential information loss of anomalous data points, W. Li et al. [2024] proposed a geometry-aware sampling method that preserves regions with high variation in normal vectors and curvatures (such high variations can indicate potential anomalies) during the subsampling process. PointMAE [Pang et al. 2022] was then adopted to iteratively reconstruct the subsampled point clouds, with the

reconstruction loss measured by the Chamfer distance. The proposed framework demonstrated SOTA performance on both the Anomaly-ShapeNet and Real3D-AD datasets, setting a new benchmark for 3D point cloud anomaly detection.

2.2.3 Anomaly detection for defect inspection on infrastructure

Although unsupervised learning methods have shown promising results in anomaly detection tasks of CV, their application to infrastructure defect detection remains limited, which can be attributed to:

(1) Features extracted by DL backbones generally cannot achieve the level of accuracy compared to supervised methods, as they lack explicit training on labelled data. Moreover, structural defects, such as cracks, are small-scale anomalies compared to the vast size of raw data representing the whole infrastructure. This gap in scale often results in insufficient resolution for DL backbones, hindering their ability to capture the fine-grained details necessary for robust anomaly detection.

(2) Unlike images, collecting infrastructure point clouds is labour-intensive, time-consuming, and costly in real-world applications. This scarcity of labelled and high-quality point cloud datasets makes it infeasible to train DL backbones which can extract robust and high-resolution feature representations of local geometries.

Nonetheless, unsupervised methods offer a viable alternative in scenarios where labelled datasets are difficult or impractical to obtain. A pioneering study by Chow et al. [2020] adopted an AE-CNN for anomaly detection on concrete surfaces. The AE-CNN was trained exclusively on images of defect-free concrete surfaces. Test samples containing defects, such as cracks and spalling, were reconstructed poorly, resulting in high reconstruction errors that served as indicators of anomalies. This approach highlighted the feasibility of anomaly detection for efficiently identifying concrete defects from images without relying on pixel-level labelled datasets. Similarly, Iftikhar et al. [2022] applied an AE-CNN for unsupervised detection of railway surface defects. By training the model solely on defect-free railway surface images, the framework achieved satisfactory results in identifying anomalies

without huge costs (both financially and computationally).

Yanyan Wang et al. [2022] proposed the Patch-aware Mutual Reasoning Network (PMRN) for unsupervised anomaly detection on concrete cracks. Unlike previous AE-CNN models, PMRN incorporates a second encoder, $\phi_{encoder_2}$, following $\phi_{decoder}$ to generate reconstructed latent vectors. The reconstruction loss is computed on both the image (pixel-wise) and the latent vectors, i.e., outputs from $\phi_{encoder}$ and $\phi_{encoder_2}$. This dual-level loss encourages the model to better capture normal data representations in both the image and latent space. s are calculated from the reconstruction losses in both domains. Experiments on the AD-Crack500 dataset (containing images of pavement cracks) demonstrated PMRN's effectiveness in detecting cracks under complex backgrounds and with weak textures.

However, to the best of the authors' knowledge, only one study has applied unsupervised learning methods for anomaly detection in 3D point clouds. The only example is that Kasireddy and Akinci [2023] developed a self-supervised learning framework for detecting concrete spalling in 3D bridge point clouds. Geometric features such as L_λ and P_λ are combined to represent the local geometry of each point as input to the AE model. The AE is trained exclusively on non-defective regions, enabling it to identify anomalies by analyzing reconstruction errors. Detected anomalies are then classified through a one-class SVM, effectively distinguishing defect points from normal ones.

The limited application of unsupervised learning methods for 3D point cloud anomaly detection can be attributed to the challenges outlined at the start of Section 2.2.1. Nevertheless, these methods offer a sound theoretical foundation and adaptable frameworks that can be adjusted to address infrastructure defect inspection using point clouds.

2.3 Summary

LiDAR scanner has become a popular tool for documenting the geometries of infrastructure through 3D point clouds. Unlike 2D images, point clouds provide rich geometric information about infrastructure, enabling detailed analysis of structural components such as the span length of a bridge arch and mechanical deformations related to geometric distortions. However, their unstructured nature complicates

graphical processing (e.g., segmentation for retrieving individual structural components), requiring extensive manual effort and domain expertise, especially when dealing with large and complex datasets.

The recent advancements in DL have enabled point cloud processing to be automated in the civil engineering domain. Despite these promising developments, the computational cost of applying DL to large-scale infrastructure point clouds remains a significant barrier. Furthermore, the successful application of DL relies on well-labelled datasets, which are often scarce in civil engineering. These challenges collectively limit the adoption of DL techniques in practical applications on infrastructure inspections. The review in this section investigates the DL in CV for processing both images and point clouds, providing a framework from segmentation to defect inspection, which provides potential solutions for the utilization of point clouds in infrastructure inspections.

This review first explored advancements in DL segmentation technologies in computer vision, focusing on various 3D data representations. Several methods have been developed to enhance the efficiency of DL architectures to address the computational demands of processing large-scale point clouds. Additionally, synthetic point cloud generation, facilitated by graphical simulators like CARLA, has emerged as a solution to tackle data scarcity in DL segmentation tasks. These approaches have been successfully applied to infrastructure point clouds, demonstrating the superiority of DL in automating point cloud processing compared to traditional feature-based and machine learning-based methods. However, challenges such as high computational costs, scalability, and the generalizability of DL segmentation technologies remain unexplored, particularly when applied to large-scale infrastructure datasets.

The review then examined supervised and unsupervised learning methods for defect inspection across both 2D images and 3D point clouds, highlighting the strengths and limitations of each approach. Supervised methods have achieved remarkable success by utilizing labelled datasets to detect and classify defects with high accuracy. These methods are particularly effective in 2D image-based inspections, leveraging well-established architectures like CNNs. In the context of 3D point clouds, supervised learning has been extended using architectures such as DGCNN and RandLA-Net, enabling the detection of structural defects. However,

the reliance on labelled datasets remains a significant limitation, particularly for point clouds, where acquiring datasets is labour-intensive and costly.

Unsupervised learning methods, which significantly alleviate the dependency on labelled data, present an alternative approach to infrastructure defect detection. This review categorizes unsupervised methods for anomaly detection into two main types. The first type leverages DL architectures exclusively for feature extraction, with anomaly detection performed separately on the extracted features. The second type employs an end-to-end framework that models the underlying data distribution of normal samples, allowing the trained DL network to directly identify anomalies. While these methods have shown robust performance in computer vision tasks, their application to 3D point clouds remains underexplored due to the scarcity of datasets.

In defect detection for 2D images, the adaptions of unsupervised methods are limited by their reduced accuracy compared to supervised approaches and their inability to explicitly classify defect types. Furthermore, the advantages of utilizing 3D point clouds for defect inspection over 2D images have not been adequately explored. Existing anomaly detection methods in 3D point clouds primarily focus on identifying superficial defects, offering functionality similar to 2D image-based methods without introducing extra advancements.

In summary, while point clouds show strong potential for enabling automatic and advanced infrastructure inspection and analysis, current techniques face significant technical and practical challenges. Addressing these issues requires the development of more efficient algorithms, methods to address data scarcity, and innovations in unsupervised learning to fully leverage the capabilities of point clouds in civil engineering.

Chapter 3

Segmentation of masonry arch bridge point clouds

The paper has been accepted for publication at the Automation in Construction, 2022.

Segmentation of large-scale masonry arch bridge point clouds with a synthetic bridge simulator and the BridgeNet neural network

Yixiong Jing*
University of Oxford
Department of Engineering Science,
University of Oxford,
Parks Road, Oxford, UK
yixiong.jing@wolfson.ox.ac.uk

Brian Sheil
University of Cambridge
Construction Engineering,
University of Cambridge,
Cambridge, UK
bbs24@cam.ac.uk

Sinan Acikgoz
University of Oxford
Department of Engineering Science,
University of Oxford, Parks Road, Oxford, UK
sinan.acikgoz@eng.ox.ac.uk

May 26, 2025

Abstract

Masonry arch bridges constitute the majority of the European bridge stock and feature a wide range of geometric characteristics. Due to a general lack of construction drawings, their geometry is difficult to parametrize. Laser scanning devices are commonly used to capture bridge geometry. However, this requires time-consuming segmentation of point clouds into their constituent components to extract key geometric parameters. To increase efficiency, a 3D deep learning neural network called BridgeNet is proposed that can automate the segmentation. To tackle the scarcity of labelled point cloud data, a synthetic dataset is created to train BridgeNet. BridgeNet is subsequently tested on real point clouds and achieves state-of-art performance, demonstrating the utility of synthetic training data and the advantages of the new network design. Segmented components are then fitted with primitive shapes by using Random Sample Consensus based algorithms to characterize key geometric parameters to assist assessments and inspections.

3.1 Introduction

Around 60% of the railway bridge stock in Europe is constructed of masonry. These bridges were built during the 19th and early 20th centuries and still play an important role in the railway network. In the UK, there are approximately 18,000 operational masonry railway bridges [Orbán 2004]. These bridges feature ageing materials and are subjected to increasing traffic demands. It is therefore necessary to inspect and assess their safety regularly [Acikgoz et al. 2018]. These activities require detailed knowledge of the geometry of the structure. However, construction drawings are often unavailable for masonry arch bridges.

Laser scanning is an efficient and non-contact technology to capture the in-situ 3D geometry of large-scale civil engineering infrastructure with point clouds [Luhmann et al. 2020b]. The obtained point clouds can help assessments by providing information on bridge geometry, such as the radius of curvature and generatrix direction of the arch, which is difficult to measure in the field. Parametric [Riveiro et al. 2011] and non-parametric [Armesto et al. 2010; Schnabel et al. 2007] shape fitting methods for point cloud components can be used to automate the identification of such geometric parameters.

Point clouds can also be used for inspecting geometric distortions and defects in structural components. C. Ye et al. [2018] have developed algorithms to fit segmented arch and pier point clouds with idealized 3D cylinders and 2D planes to reveal geometric distortions introduced by past settlements. Other studies [Valero et al. 2019; Armesto-González et al. 2010] used irregularities in surface normal vectors, roughness and colour to identify and classify defects such as cracking and material loss. Separately, point clouds have been used to quantify the size of structural defects [Laefer et al. 2014] and track deformations occurring over time [Acikgoz et al. 2017a; Acikgoz et al. 2017b; Acikgoz et al. 2022].

These previous studies demonstrate the potential for point clouds to help assess, inspect, and monitor masonry arch bridges. However, the large size of typical point clouds often requires significant amounts of time for semantic and instance segmentation of constituent components. To recognize the potential of laser scanning in structural engineering practice, it is necessary to develop automated segmentation algorithms which can be used by practising engineers and asset managers.

Semantic and instance segmentation methods adopted in the literature can be roughly classified into three categories: feature-based, machine learning, and deep learning methods. Feature-based methods consider infrastructure components as primitive shapes organized according to an idealized topology. Segmentation is then conducted on the basis of these assumptions [Riveiro et al. 2016; R. Lu et al. 2019a]. Machine learning methods consist of unsupervised methods that cluster points with similar geometric features followed by supervised learning methods that classify the clustered points [Grilli and Remondino 2020; Teruggi et al. 2020]. Although these methods can achieve remarkable segmentation accuracy, the algorithms are designed for specific topologies with hand-crafted geometric features, which are difficult to generalize to other cases. In contrast, deep learning (DL) based segmentation techniques do not require the prior definition of features since they can be learned latently by the neural network. Previous applications of DL to the segmentation of heritage buildings [Pierdicca et al. 2020; Matrone et al. 2020] and bridge point clouds [Xia et al. 2022] have demonstrated competitive performance relative to established machine learning methods.

There are two significant impediments to using DL techniques to segment masonry arch bridge point clouds: (i) lack of a sufficiently large and labelled point cloud dataset, which is often required for achieving good segmentation accuracy, and (ii) lack of neural networks for conducting efficient semantic segmentation on large-scale point clouds.

To tackle challenge (i), researchers in the autonomous driving field augment real datasets with synthetic data obtained from LiDAR simulators [B. Wu et al. 2018; B. Wu et al. 2019; Griffiths and Boehm 2019; Fang et al. 2020]. The mixed (real+synthetic) training dataset enables significant performance improvement compared to training neural networks on the real dataset alone [Geiger et al. 2012]. In the infrastructure domain, J. Ma et al. [2020] and Morbidoni et al. [2020] train the network with synthetic data and show its potential to address data scarcity (see also Narazaki et al. [2021]).

Recent research efforts using point-based DL architectures show some promise in tackling challenge (ii). Voxel [Yinlong Zhou and Tuzel 2018; Çiçek et al. 2016; B. Li 2017a; Choy et al. 2019a; Renjie Cheng et al. 2021; Riegler et al. 2017; Zhijian Liu et al. 2019b] and graph-based [Simonovsky and Komodakis 2017; Yue Wang

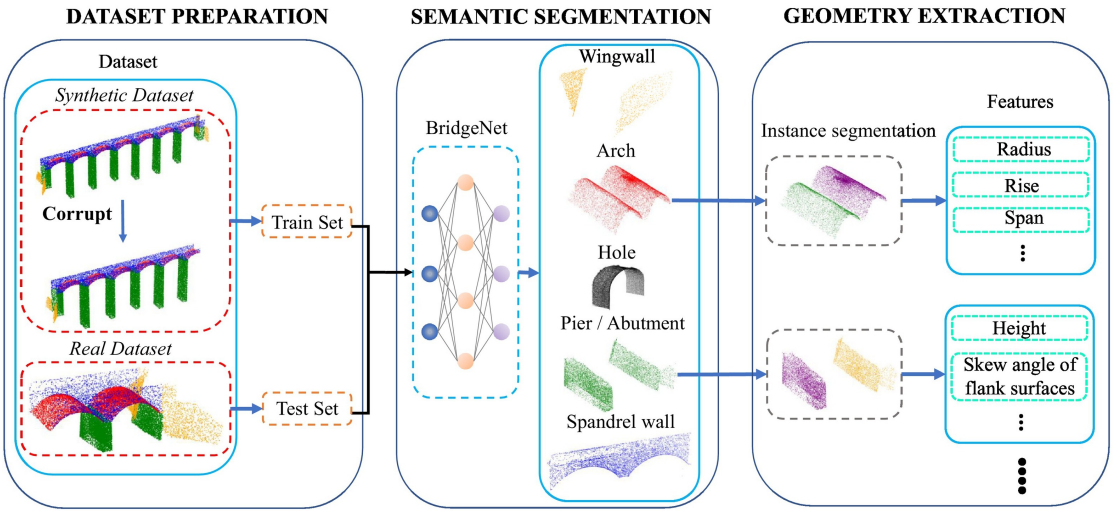


Figure 3.1: Research pipeline for automating segmentation of masonry arch bridge point clouds for geometry parametrization.

et al. 2019; L. Wang et al. 2019; Landrieu and Simonovsky 2018; Landrieu and Boussaha 2019] neural networks are memory-inefficient since raw point clouds are transformed into other data representations. In contrast, point-based methods [Qi et al. 2017a; Qi et al. 2017b; M. Jiang et al. 2018; Yuelin Liu et al. 2019b; Thomas et al. 2019; Yuelin Liu et al. 2019a; W. Wu et al. 2019; Sheshappanavar and Kambhamettu 2020; Zhirong Zhang et al. 2019; Hu et al. 2020] can take raw point clouds as direct input. However, pioneering point-based architectures like PointNet++ [Qi et al. 2017b] are memory-intensive and unsuitable for large-scale point clouds due to the complicated design of local feature extractors. The recently proposed RandLA-Net [Hu et al. 2020] can perform semantic segmentation on millions of points in real-time by adopting a lightweight architecture. However, this network can demonstrate poor performance in segmentation tasks which require the neural network to be aware of complex local features. FG-Net [Kai Liu et al. 2020] applies a feature condensing mechanism to alleviate memory consumption while managing to retain local information. As a consequence, FG-Net [Kai Liu et al. 2020] outperforms other semantic segmentation neural networks on the public dataset Shape-Net [Chang et al. 2015]. To address instance segmentation within the DL framework, the similarity matrix [W. Wang et al. 2018; B. Zhang and Wonka 2021; Liangliang Zhao and Weidong Tao 2020] and a top-down hierarchical structure [F. Yu et al. 2019] have been proposed. However, these approaches require heavy computations and are not suitable for large-scale point clouds. Instead, a memory-efficient clustering machine learning method called DBSCAN [Ester et al.

1996] may be used to perform instance segmentation.

To enable the application of DL to bridge segmentation, this paper develops a synthetic masonry arch bridge simulator that can overcome the data scarcity challenge. In addition, the paper develops a computationally efficient semantic and instance segmentation pipeline to segment large-scale point clouds. Figure 3.1 demonstrates the workflow adopted in this paper. To this end, Section 2 describes the development of a new neural network called BridgeNet for semantic segmentation. Section 3 discusses the formulation of synthetic data used to train the network and introduces the real point cloud data used for testing. Section 4 evaluates BridgeNet semantic segmentation accuracy and compares it with state-of-art techniques. Section 5 discusses post-processing methods for efficient instance segmentation and the extraction of useful geometric parameters from instances for engineering purposes.

3.2 BridgeNet

As shown in Figure 2, the architecture of BridgeNet adopts the popular encoder-decoder structure to effectively extract features of point clouds at different resolutions. The input size of our neural network is $N \times C_{input}$, where N is the number of points and C_{input} is the input channel size; $C_{input} = 3$ since we only provide the raw x, y, and z point coordinates as input. This input is first sent to a 1×1 convolution layer to extract the feature representations of each point. It is then passed through four Residual Feature Encoder (Residual FE) layers to learn local features at different resolutions.

We apply Random Sampling (RS) to downsample the point clouds with a ratio of 4, achieving time efficiency for large-scale point clouds [Hu et al. 2020]. The downsampling method is adopted from PointNet++ [Qi et al. 2017b], which retains 25% of the points at each step. In our experiments, we found that random sampling can prevent overfitting compared to other sampling methods, such as farthest point sampling, which is widely used in 3D neural networks.

The downsampled features are sent to a Global Feature Extractor adapted from [Kai Liu et al. 2020] to enhance the global point-wise relationships. Four decoder layers are used to scale the number of points back to the original input by using the

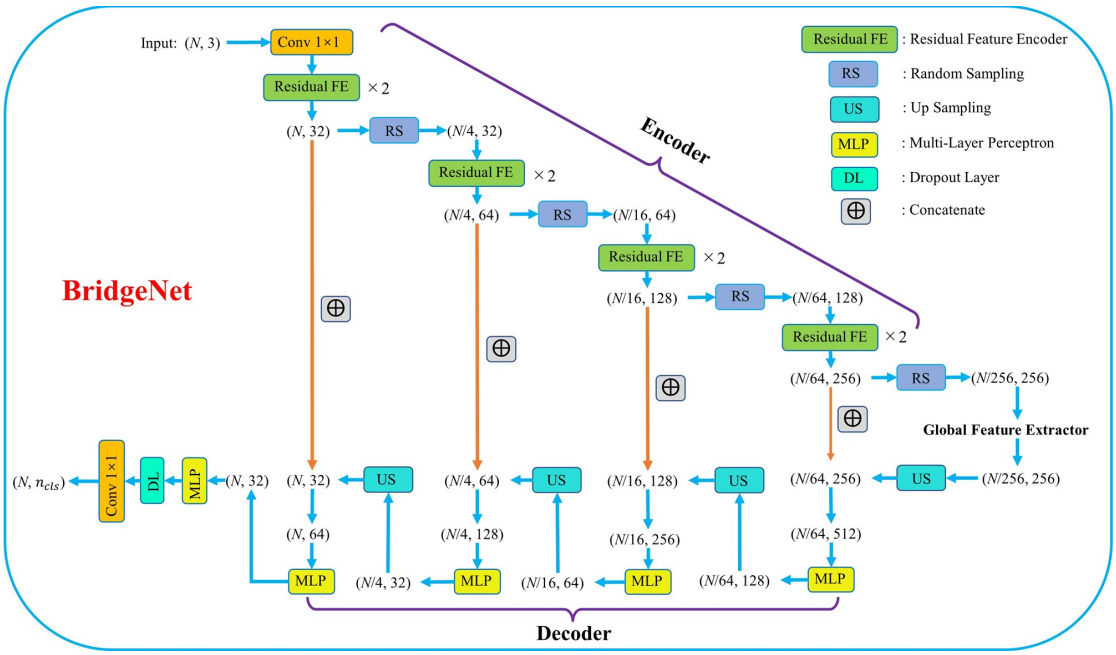


Figure 3.2: The input point cloud has dimensions $(N, 3)$, where N is the number of points and 3 represents the x, y, and z coordinates. Features are extracted by the Encoder, which contains four Residual Feature Encoder (Residual FE) layers and subsamples the points through random sampling. A Global Feature Extractor is then employed to correlate pairwise global relationships. Finally, a Decoder upsamples the point cloud back to the original point size using interpolation and convolutional layers, providing the output with dimensions (N, n_{cls}), where n_{cls} represents the semantic label classes.

nearest interpolation method [Qi et al. 2017b]. For each upsampling layer in the decoder, interpolated features are concatenated with the corresponding encoding layer features, as shown by the orange arrows. This is to enhance representations of feature patterns. The output features from the decoder are then passed through a shared Multi Layer Perceptron (MLP) followed by a dropout layer to alleviate the overfitting [Hu et al. 2020]. Three different drop ratios (0.2, 0.5, and 0.8) were tested, where the test accuracy for 0.5 outperformed 0.2 by a small margin, while 0.8 was shown to cause underfitting. Therefore, a drop ratio of 0.5 is used, consistent with other neural networks such as RandLA-Net [Hu et al. 2020]. Finally, a 1×1 convolution layer is used to learn the semantic labels. The Residual FE and Global Feature Extractor [Kai Liu et al. 2020] blocks are explained in the following sections.

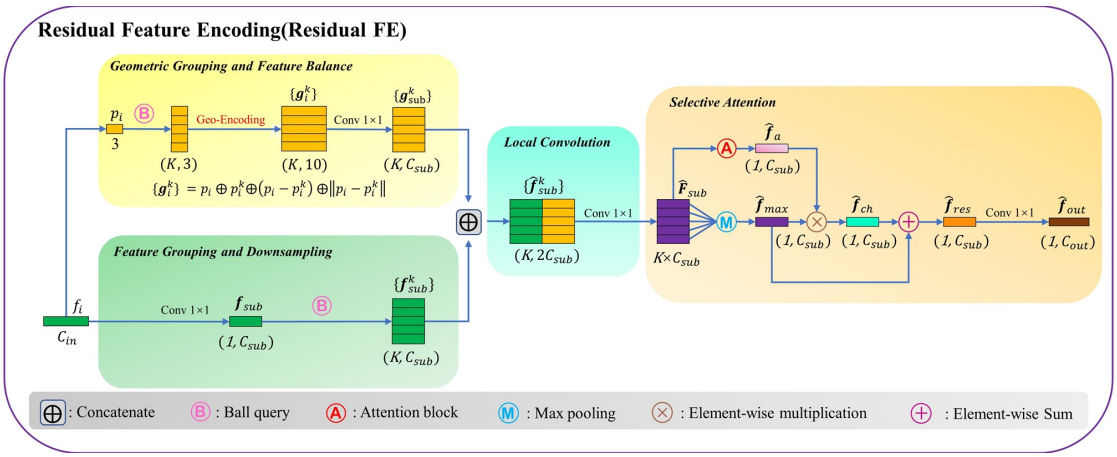


Figure 3.3: The modules in the Residual Feature Encoding (Residual FE) layer. The learned latent features are downsampled before applying ball query to save memory in Feature Grouping and Downsampling module. Geometric Grouping and Feature Balance explicitly encodes and balances grouped geometric features in feature dimensions.

3.2.1 Residual Feature Encoder

The original local feature aggregation module from [Hu et al. 2020] was not designed for learning complex local features. Therefore, as shown in Figure 3.3, the local feature aggregation module [Hu et al. 2020] was redesigned to enable it to capture local characteristics. The module contains four basic components: (1) Feature Grouping and Downsampling, (2) Geometric Grouping and Feature Balance, (3) Local Convolution, and (4) Selective Attention, which are explained as follows:

(1) *Feature Grouping and Downsampling:* Let p_i be the x, y, and z coordinates of a given point within a point cloud $\boldsymbol{P}$, where $p_i \in \mathbb{R}^3$. The corresponding feature vector for p_i is denoted as f_i, where $f_i \in \mathbb{R}^{C_{in}}$. For all points belonging to point cloud P, features are first downgraded from C_{in} to $C_{sub} = C_{in}/M$ by a 1×1 convolution before applying the neighbor query to save memory [Kai Liu et al. 2020]. $M = 8$ is adopted here, similar to FG-Net [Kai Liu et al. 2020], to achieve computational efficiency. According to [Kai Liu et al. 2020], a larger M is expected to reduce accuracy, while a smaller M increases GPU memory consumption. Ball query is then applied to gather the K nearest neighboring points within a sphere of radius r, where the k-th neighbor is mathematically defined as $\{p_i^k \in \mathbb{R}^3 \mid \|p_i - p_i^k\| \leq r\}$, for $k = 1, 2, 3, \ldots, K$. The local feature $\{\boldsymbol{f}_{sub}^k\}$ is then derived by stacking all

sub-sampled neighbour features together.

(2) *Geometric Grouping and Feature Balance:* The relative point position encoding is applied to explicitly provide geometric features to assist the network in achieving better performance. This procedure also compensates for the lost information by downgrading feature vectors in the Feature Grouping and Downsampling module. More details can be found in RandLA-Net [Hu et al. 2020]. The feature dimension of relative point position $\{\boldsymbol{g}_i^k\}$ is balanced by a 1×1 convolution from 10 to C_{sub} to derive $\{\boldsymbol{g}_{sub}^k\}$, which makes the feature dimension equivalent to the local feature $\{\boldsymbol{f}_{sub}^k\}$.

(3) *Local Convolution*: Different from FG-Net [Kai Liu et al. 2020], after concatenating the grouped features from Feature Grouping and Downsampling and Geometric Grouping and Feature Balance layers, $\left\{\hat{\boldsymbol{f}}_{sub}^k\right\}$ is encoded through a 1×1 convolution to generate local features $\hat{\boldsymbol{F}}_{sub}$. It was observed that this step improves the overall performance significantly by explicitly learning from local structures. It also requires considerable memory but this is compensated by feature downsampling steps. Alternative convolutional kernels like KPConv [Thomas et al. 2019] were also tested but these led to severe overfitting.

(4) *Selective Attention*: A self-attention mechanism is applied to weight $\hat{\boldsymbol{F}}_{sub}$ with respect to their importance in the feature dimension. The channel-wise weighted feature $\hat{\boldsymbol{f}}_a$ is learned from grouped features which represent the importance of each feature channel from a statistical perspective and are defined as:

$$\hat{\boldsymbol{f}}_a = softmax\left(\boldsymbol{w}_{att}\hat{\boldsymbol{F}}_{sub}\right) \tag{3.1}$$

where $\boldsymbol{w}_{\text{att}} \in \mathbb{R}^K$ is the attentive score which is learnable for aggregation and *softmax* is a function that normalizes all K scores in each channel so that they sum to 1. Even though both [Hu et al. 2020; Kai Liu et al. 2020] adopt summation instead of maxpooling to avoid information loss, it was observed in our dataset that the summation operation can lead to severe overfitting since it groups relatively irrelevant information. Therefore, maxpooling was applied here to select the largest feature from $\hat{\boldsymbol{F}}_{sub}$ in each channel. The selected feature vector $\hat{\boldsymbol{f}}_{max}$ is multiplied element-wisely with $\hat{\boldsymbol{f}}_a$ and a residual connection is used to add the

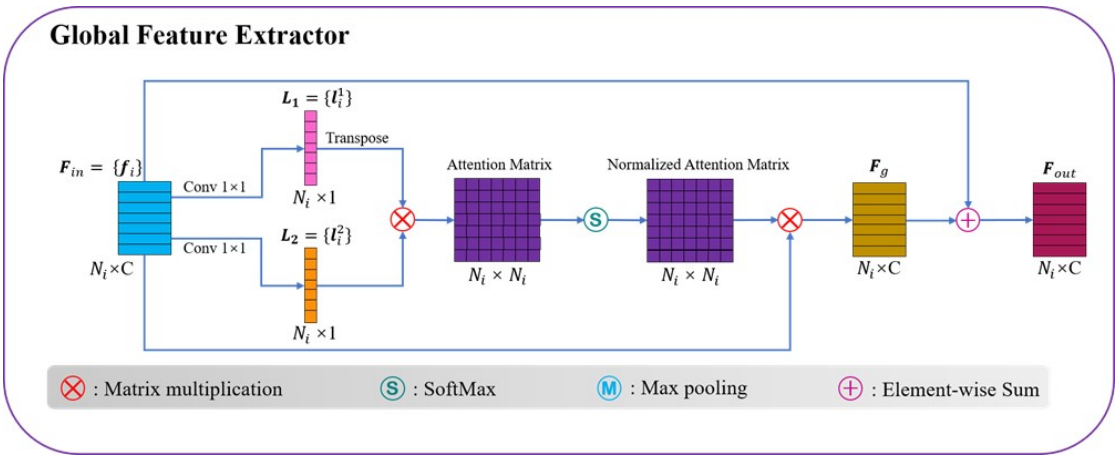


Figure 3.4: Graphical explanation of Global Feature Extractor shown in Figure 3.2. The global pairwise points relationship is learned and represented by the normalized attention matrix.

original feature $\hat{\boldsymbol{f}}_{max}$ to the learned feature $\hat{\boldsymbol{f}}_{ch}$. Another 1×1 convolution is used in the final step to recover the feature dimension C_{sub} of the residual feature $\hat{\boldsymbol{f}}_{res}$ to C_{out} of the output $\hat{\boldsymbol{f}}_{out}$.

3.2.2 Global Feature Extraction

Global Feature Extraction is proven to be effective in the ablation test of FG-Net [Kai Liu et al. 2020]. The same structure is also adopted in our network to exploit and enhance globally the relationship of closely related points. As shown in Figure 3.4, the input feature $\boldsymbol{F}_{in} \in \mathbb{R}^{N_i \times C}$ ($N_i = N/C$ and $C = 256$ in our case) from the encoder layer is first transformed by two 1×1 convolution layers into latent representations $\boldsymbol{L}_1$ and $\boldsymbol{L}_2$. The matrix multiplications of $\boldsymbol{L}_1$ and $\boldsymbol{L}_2$, followed by normalization, give the pointwise similarity map which is further multiplied pair-wisely with input features to capture related regions more effectively. Finally, the local feature map $\boldsymbol{F}_{in}$ is summed element-wise with learned global contextual representation $\boldsymbol{F}_g$ to derive both locally and globally relation-aware features $\boldsymbol{F}_{out}$. More details can be found in [Kai Liu et al. 2020].

3.3 Dataset

The dataset is composed of real and synthetic components, which are used for test and training purposes, respectively.

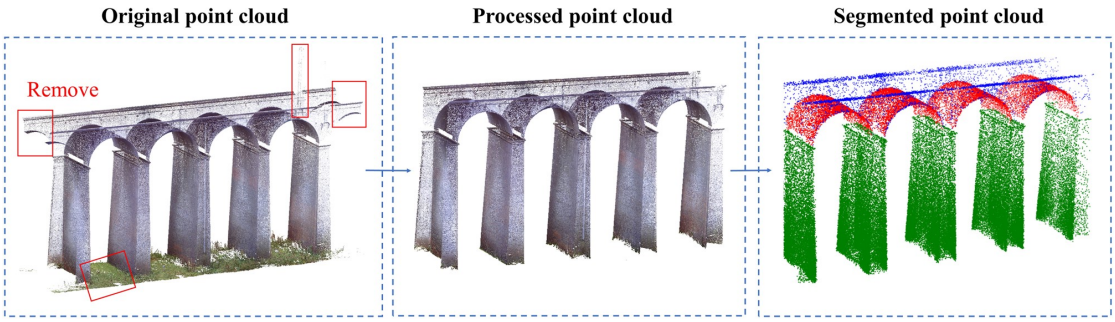


Figure 3.5: Graphical representation of real point cloud processing including pre-processing (e.g. removal of irrelevant environment and components) and manual segmentation.

3.3.1 Real dataset

The real dataset comprises the point clouds of seven multi-span masonry arch railway bridges from the UK. Before segmentation, these point clouds need to be pre-processed to remove parts that are not modelled by the synthetic dataset, such as the ground and the environment surrounding the bridge. Some bridge components, such as the parapet, were not of interest and were also removed. Poorly scanned areas, which were not intended for detailed analysis were also deleted. Finally, bridge components were manually segmented into 5 classes which are the spandrel wall, pier and abutment, arch, hole (only in the Marsh Lane viaduct), and wingwall (only in the Stapleton Road viaduct). Figure 3.5 demonstrates this procedure. The real dataset contains the point cloud of 7 masonry railway bridges as shown in Figure 3.6. Marsh Lane and Stapleton Road viaduct data did not feature colour information and were therefore visualized with black points in Figure 3.6. Each point was classified as a member of one of the 5 following classes:

Spandrel wall: The two side surfaces of the bridge superstructure are known as the spandrel wall. The lower spandrel wall boundary is defined at the top of the piers at the arch 'springing'. Points in this class are shown in blue in Figure 3.5.

Pier and abutment: The piers are the columns beneath the bridge superstructure. They can be tapered (see Digswell viaduct in Figure 3.6) and can also have a non-rectangular footprint due to track curvature (see Marsh Lane and Chelmsford viaducts). Bridges with a tapering pier are listed in Table 3.1. Abutments are located at the end of a bridge. They share similar geometric features to piers and are therefore classified in the same category. Piers and abutments are shown in

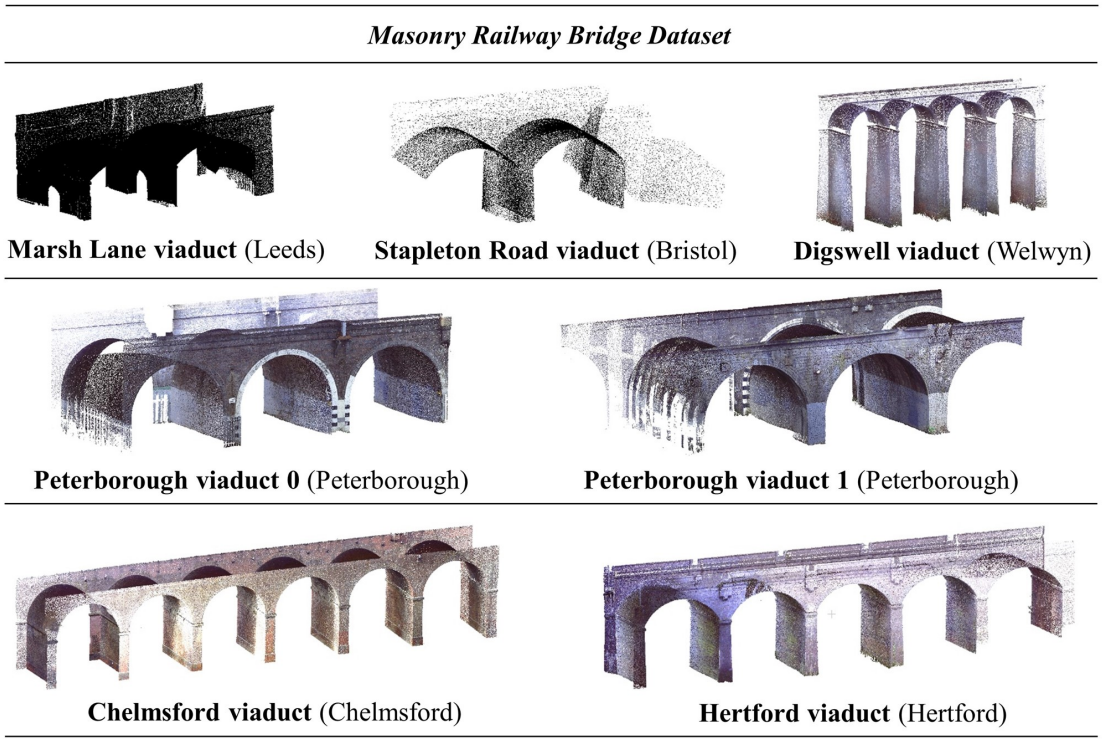


Figure 3.6: Real point clouds of seven masonry bridges that constitute in the test dataset.

Table 3.1: Characteristics of the real point cloud dataset.

		Span	Classes					
			Pier taper	Spandrel	Pier	Arch	Wingwall	Hole
Curved bridge	Chelmsford viaduct	6	X	✓	✓	✓	X	X
	Marsh Lane viaduct	2	X	✓	✓	✓	X	X
	Stapleton Road viaduct	2	X	✓	✓	✓	X	X
Straight bridge	Digswell viaduct	4	X	✓	✓	✓	X	X
	Peterborough viaduct 0	3	X	✓	✓	✓	X	X
	Peterborough viaduct 1	3	X	✓	✓	✓	X	X
	Hertford viaduct	5	✓	✓	✓	✓	X	X

green in Figure 3.6.

Arch: Arches are shown in red points and their shape can typically be approximated by a partial cylinder. Elliptical arches exist, but Brencich and Morbiducci [2007] note that they are rare. Only square spans are considered in this study.

Hole: Holes exist in piers to minimize the use of materials. These holes penetrate

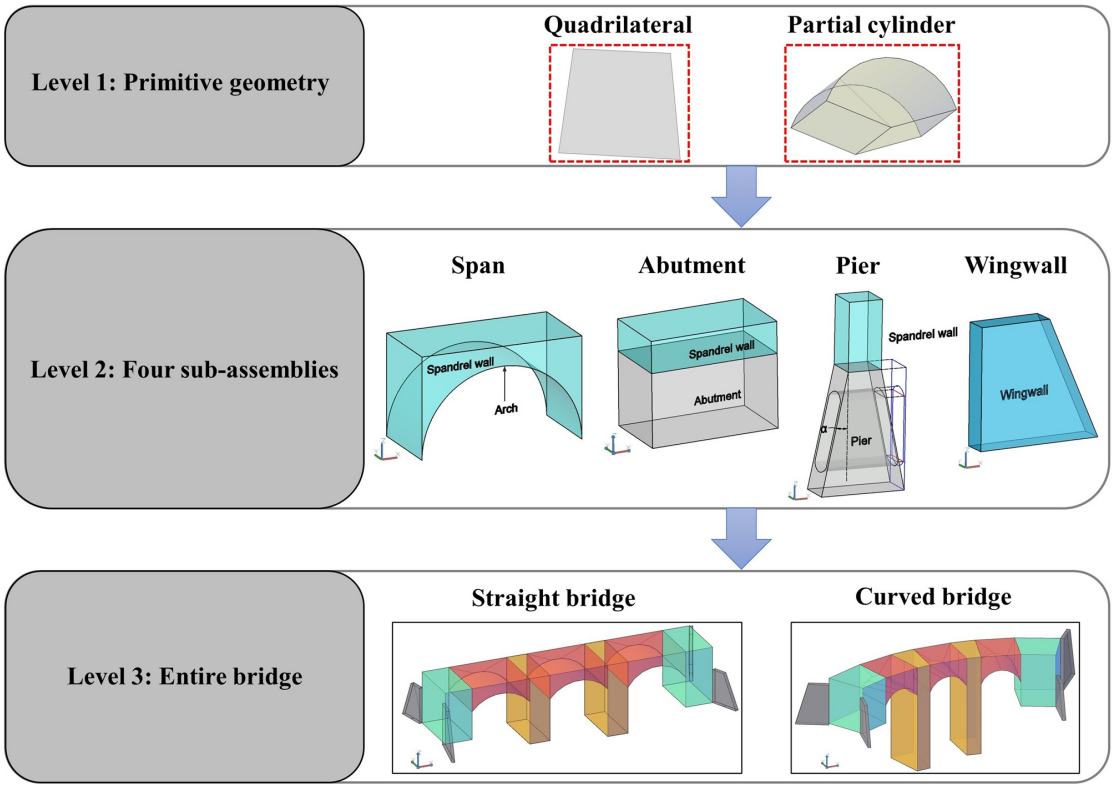


Figure 3.7: Flowchart illustrating the generation of synthetic bridges hierarchically from primitive geometry to the complete bridge.

the pier in the longitudinal direction of the bridge and are often symmetric with respect to the vertical centerline of the pier. The holes only exist in the scanning of the Marsh Lane viaduct as shown in Figure 3.6.

Wingwall: Wingwalls attach to the side surfaces of the abutments. The wingwalls of the Stapleton Road viaduct, shown in Figure 3.6, have a triangular shape on one side and a pentagon on the other side. The shape of the wingwall varies drastically according to the local topography of the embankment.

The basic geometric information and the classes included in each bridge are summarised in Table 3.1.

3.3.2 Synthetic dataset

The synthetic dataset aims to approximate the geometric properties of two types of real masonry arch bridges: straight and curved bridges (see Table 3.1). A bottom-to-top methodology is developed to automate synthetic bridge geometry generation by assembling entire bridges from two primitive shapes. As shown in Figure 3.7, (1) the two primitive shapes in level 1 are quadrilaterals and partial

cylinders; (2) four subassemblies, namely the span, abutment, pier and wingwall, are then derived from the combinations of primitive geometries; (3) different 3D masonry bridges are generated by iteratively producing those components based on bridge topologies. The noiseless synthetic point cloud produced this way is then corrupted randomly to simulate geometric distortions and laser scanning errors that are commonly encountered in practice.

Since the mathematical operations involved in Level 1 and Level 3 (Figure 3.7) are well-established, the following sections discuss the definition of the subassembly data, the range of parameters used to specify the size of components, and the corruption of the resulting geometries.

Component generator

The 3D drawings of the span, pier, abutment and wingwall subassemblies are shown in Level 2 of Figure 3.7.

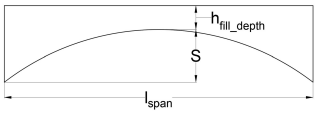
Span subassembly: The span subassembly can be considered as the combination of one arch surface and two spandrel wall surfaces. The arch is expressed by a segmental cylinder, for which the range is defined by the minimum and maximum circular angles. The spandrel wall can be defined by first generating a quadrilateral and then filtering the part under the arch.

Pier subassembly: The pier subassembly includes a part of the superstructure that connects the neighbouring spans. It can be defined in the same way as in the span subassembly but without filtering. The lower part of the pier is composed of four quadrilateral surfaces. Two surfaces are coplanar with the spandrel walls and the other two surfaces have a taper angle α relative to the ground. The pier sometimes features a hole that can be built by filtering areas inside the pier that intersect with two partial cylinders sandwiching a rectangular prism.

Abutment subassembly: The abutment subassembly can be represented by a slightly modified version of the pier subassembly.

Wingwall subassembly: The wingwall subassembly takes a variety of shapes. For simplicity, this study represents the wingwall with a 3D right-angled trapezoid. Since the rear surface in contact with the soil is not visible, only the front, top and outward side surfaces are generated by quadrilaterals.

Table 3.2: Schematic descriptions and the range of geometric parameters used to define span subassemblies.



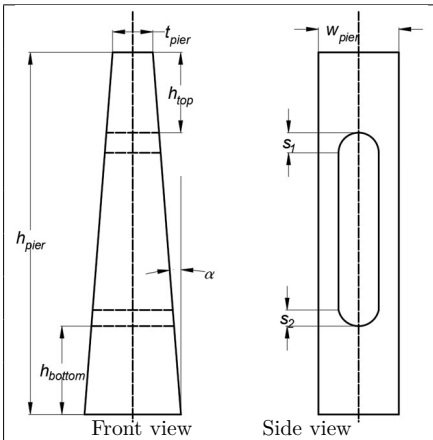
Front view



Top view

Span			
Parameter	Range	Probability	Reference
l_{span}	[8m, 15m]	70%	[Brencich and Morbiducci 2007]
	[15m, 26m]	30%	
w_{span}	[3.5m, 5m] for one lane	50%	
	[7m, 10m] for two lanes	50%	
s/l_{span}	[0.05, 0.50]	100%	
h_{fill_depth}/l_{span}	[0.02, 0.14]	100%	

Table 3.3: Schematic descriptions and the range of geometric parameters used to define pier subassemblies and the hole.



Pier			
Parameter	Range	Probability	Reference
t_{pier}/l_{span}	[0.1, 0.25]	100%	Empirical and [Melbourne et al. 2006]
h_{pier}	[5m, 20m]		Empirical
α	$[0°, 30°]$		
m/w_{pier}	[0.1, 0.4]		
s_1/m	(0, 0.5]		
s_2/m	(0, 0.5]		
h_{top}/h_{pier}	[0.15, 0.35]		
h_{bottom}/h_{pier}	[0.15, 0.35]		

Bridge parameter definition

The range of parameters adopted to describe masonry arch railway bridges refers to the statistical data collected in Italy [Brencich and Morbiducci 2007] and the UK [Melbourne et al. 2006]. Due to the limited information available, some of the parameters had to be defined empirically. Following Figure 3.7, the parameters were defined to describe the four subassemblies:

(1) *Span subassembly:* The number of spans is assumed to range from 5 to 20. The top and front views in Table 3.2 indicate parameters to define each span subassembly. The parameters include the clear length of the span l_{span}, bridge width w_{span}, rise of the arch s, and the fill depth h_{fill_depth}. The synthetic bridge generator uses the specified range for the length of the span l_{span} and determines s and h_{fill_depth} accordingly, following the ratios presented in Table 3.2. All values are sampled uniformly within the specified parameter ranges. Single track railway bridges were assumed to have a width



Figure 3.8: Example of a pier hole at Stanway Bridge, UK, where the upper and lower arch surfaces are partial cylinders sandwiching a rectangular prism.

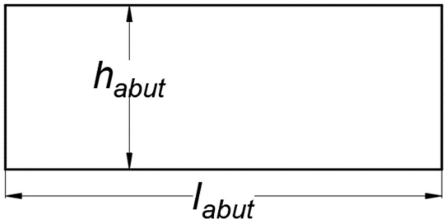
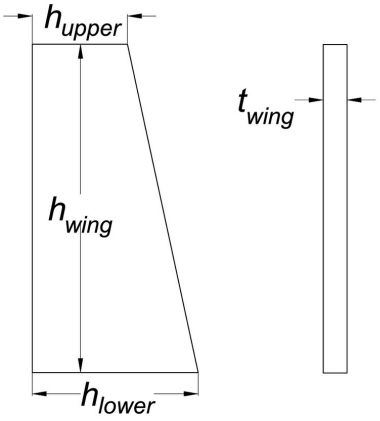


Figure 3.9: Top view of the geometric parameters used to define abutments.



(a) Side view (b) Front view

Figure 3.10: Schematic descriptions (a), (b) of geometric parameters used to define wingwalls in the synthetic data.

ranging from 3.5m to 5m, and double track bridges, a width of 7m to 10m. Either of the two cases have a 50% possibility of being sampled, as indicated in the third column of Table 3.2.

(2) *Pier subassembly:* Parameters are needed to describe the overall pier geometry and the position and size of the hole. Table 3.3 specifies the thickness t_{pier}, height h_{pier}, and width w_{pier} of the pier. The vertical distance between

the crown of the hole to the upper boundary of piers h_{top}, the corresponding bottom distance h_{bottom}, taper angle α, width of the hole m, the rise of the upper and lower arch for the hole s_1 and s_2 are the parameters used to determine the geometry of the hole. The hole geometry is not covered by either [Brencich and Morbiducci 2007] or [Melbourne et al. 2006], therefore it was defined empirically by the authors based on example bridges (see Figure 3.8).

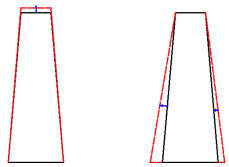
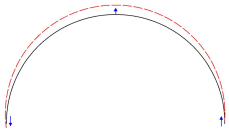
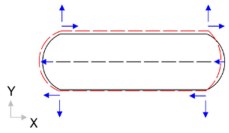
(3) *Abutment subassembly:* The width of abutments is assumed to be the same as w_{span} for geometric consistency. The length of the abutment is also not defined in the literature. Therefore, the height h_{abut} and the length l_{abut}, are defined empirically. The range of the abutment length is defined from [0.2*l_{span}, 0.5*l_{span}] which is thicker than the pier since it is responsible for dispersing the loading to the supporting soil. The height of abutments is defined in the range from $0.3h_{pier}$ to h_{pier} (see Figure 3.9).

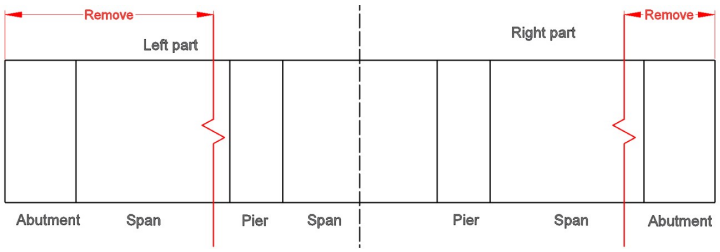
(4) *Wingwall:* 4 wingwalls are attached to the lateral surfaces of the abutments. These are assumed to be 3D trapezoids. From Figure 3.10, the length of the upper-line h_{upper}, lower-line h_{lower}, the thickness of the wingwall t_{wing}, and the height of the wingwall h_{wing} need to be determined. Due to limited information on the geometry of the wingwalls, the parameter ranges used in the synthetic simulator were based on the authors' experience. h_{upper} is sampled from a uniform distribution with the range of [2m, 4m] and h_{lower} with [4m, 8m]. h_{wing} is sampled from 0.3*($h_{pier} + h_{span}$) to ($h_{pier} + h_{span}$). The angle between the wingwall and the side surfaces of the abutment is sampled from $[-60°, 60°]$. t_{wing} is defined to be in the range of 0.25m to 0.75m.

Data corruption and truncation

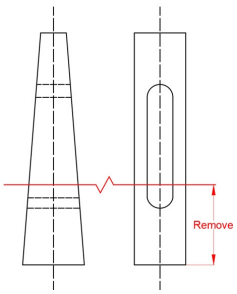
Geometric irregularities are common in masonry arch bridges. Spans may be of different lengths, pier top elevations may vary across the bridge due to topographical reasons and piers and holes may feature distorted geometries due to support settlements. To simulate these aspects, the geometric properties of the subassemblies need to be randomised, as shown in Table 3.4. For instance, according to the first row, the span length specified for the bridge will vary uniformly in the range

Table 3.4: Schematic description of the data corruption introduced to each component of the synthetic bridge point clouds. The deformations are indicated by red dotted lines which depart from black lines that represent the initially generated regular bridge geometry.

	Component	Randomness	Range (Uniform distribution)
	Front view of a span and a pier	Length	[-0.2m, 0.2m]
	Pier	Height and skew angle of flank surface	Height: [-0.1, 0.1]*h_{pier} Skew angle:[$-0.5°$, $0.5°$]
	Arch	Rise and elevations for two ends of the arc	Rise: [-0.1m, 0.1m] Elevation: [-0.1m, -0.1m]
	Hole	Rise of both lower and upper arches and two directions for four corner points	Rise: [-0.02m, 0.02m] X axis: [-0.005m, 0.005m] Y axis: [-0.005m, 0.005m]



(a) Bridge top view



(b) Pier front and side

Figure 3.11: Data truncation in synthetic bridge models. The first example (a) shows the plan view of a bridge, whose point cloud is truncated from two sides (see zigzagged red lines), while the second example (b) shows the side view of a pier, which is truncated along its height.

[-0.2m, 0.2m] in synthetic point clouds.

Measurement imperfections in point clouds occur due to measurement errors and occlusions. Laser scanning measurement noise was approximated simply by adding a zero-mean Gaussian noise distribution with a standard deviation of 0.005m to all x, y and z coordinates [Griffiths and Boehm 2019]. The occlusions in real

point clouds are case-specific and impossible to capture without detailed models of scanning geometry and the surrounding environment. To approximately capture the incomplete information that can arise as a result of occlusions, data truncation is performed. The synthetic point cloud is truncated from both ends as shown in the plan view in Figure 3.11a. The truncation ratio for the left and right parts does not exceed 50% of the bridge length. Since the Marsh Lane viaduct used in the testing dataset contains a truncated hole, some piers were also truncated, as shown in Figure 3.11b. This involves first setting a threshold sampled from the uniform distribution with the range of $[0.15 * h_{pier}, 0.5 * h_{pier}]$ and subsequently filtering out the points with z coordinates lower than the threshold.

3.4 Training results

3.4.1 Implementation details

Our framework is implemented using Pytorch. Adam is used as an optimizer with an initial learning rate of 10^{-3} and a decay rate of 10^{-4}. The x, y, and z coordinates of point clouds are first normalized within the range $[-1, 1]$. The coordinates are then randomly rotated around the z-axis with an angle between 0 to 2π. The normalized point cloud is also rescaled with a scalar between 0.8 and 1.25 and shifted from 0 to 0.1 relative to the x, y, and z directions for data augmentation. The batch size is 4 for training, so the training data has the size $(4, 65536, 3)$, where 4 represents the batch size and 65536 is the number of points for each batch with 3 referring to the x, y, and z coordinates of each point.

The training dataset contains only synthetic data in which up to 2000 point clouds are sampled by using the data generator introduced in Section 3.3. To be compatible with the real dataset, the ratio between the number of synthetic curved and straight bridges is set to 2/5. The test dataset includes all 7 real bridges. Since most of the test point clouds have millions of points while the default number of input points in the network is 65536, each real bridge point cloud data is randomly sampled into non-repeated sets with the default number of points. For example, the Hertford viaduct point cloud contains 5 million points which can be split into $5000000/65536 \approx 76$ sets. The residual points which are not included in the sets are discarded. 70 epochs are trained for each case. The experiments are conducted

on an RTX 3080 GPU with 10GB of memory.

3.4.2 Evaluation metrics

Three different metrics are adopted in the experiments to assess BridgeNet performance and benchmark against existing networks PointNet++ and RandLA-Net. The overall accuracy is defined as:

$$Accuracy = \frac{TP}{N} \tag{3.2}$$

in which N is the number of points and TP (true positive) represents the points that are correctly labelled. Equation 3.2 gives overall measurements of the neural network performance though it cannot reflect the segmentation accuracy for each class. Therefore, the mean precision, denoted by $mean_Pr$, and mean intersection over union, $mIoU$, metrics are also used. These are defined in Equations 3.3 and 3.4,

$$mean_Pr = \frac{\sum_{i=1}^{n_{cls}} \left(\frac{TP_i}{(TP_i + FP_i)} \right)}{n_{cls}} \tag{3.3}$$

$$mIoU = \frac{\sum_{i=1}^{n_{cls}} \left(\frac{TP_i}{(TP_i + FP_i + FN_i)} \right)}{n_{cls}} \tag{3.4}$$

where TP_i is the true positive for class i, FP_i (false positive) is the number of points being misclassified as class i and FN_i (false negative) is the number of misclassified points belonging to class i. Equations (3.3) and (3.4) can highlight poor segmentation of a certain class even when the class occupies a small proportion of the entire point clouds. In particular, the metric $mIoU$ presents the most rigorous evaluation of accuracy, as it also considers the influence of misclassified points FN_i. If the total number of classes is under 5, which is possible since some bridges do not contain hole or wingwall classes (see Table 3.1), the n_{cls} is adjusted to remove non-existing classes.

Table 3.5: Comparison of the performance of PointNet++, RandLA-Net and BridgeNet for semantic segmentation on test dataset with increasing training samples in the synthetic dataset using accuracy, mean precision and mean intersection over union metrics.

Metric	**Model**	**Number of synthetic bridges used for training**						
		100	**400**	**500**	**600**	**1000**	**1500**	**2000**
Accuracy	**PointNet++**	0.874	0.915	0.913	0.912	0.895	0.908	0.896
	RandLA-Net	0.845	0.911	0.914	0.911	0.932	0.920	0.927
	BridgeNet	0.875	0.938	0.947	0.941	0.951	0.954	**0.957**
mean_Pr	**PointNet++**	0.545	0.605	0.641	0.645	0.618	0.623	0.627
	RandLA-Net	0.568	0.678	0.665	0.695	0.666	0.716	0.713
	BridgeNet	0.532	0.685	0.743	0.752	0.785	0.861	**0.867**
mIoU	**PointNet++**	0.466	0.544	0.586	0.582	0.513	0.564	0.552
	RandLA-Net	0.438	0.562	0.574	0.581	0.583	0.585	0.598
	BridgeNet	0.461	0.629	0.654	0.680	0.684	0.751	**0.779**

Table 3.6: Mean intersection over union ($mIoU$) of each semantic class for each real bridge.

	Chelmsford	**Digswell**	**Hertford**	**Peter 0**	**Peter 1**	**Stapleton**	**Marsh Lane**
Spandrel	0.837	0.855	0.888	0.875	0.874	0.820	0.962
Pier	0.927	0.980	0.973	0.628	0.756	0.926	0.924
Arch	0.918	0.877	0.966	0.782	0.878	0.973	0.967
Hole	0	0	0	0	0	0	0.513
Wingwall	0	0	0	0	0	0.704	0
Average mIoU	0.894	0.904	0.942	0.762	0.836	0.856	0.841

3.4.3 Semantic segmentation results

Networks are trained on the purely synthetic dataset with an increasing number of bridge point clouds; 100, 400, 500, 600, 1000, 1500 and 2000 point clouds are considered. Once trained, the networks are tested on the real bridge dataset containing the 7 bridges. To ensure like-for-like comparisons, the sample ratio, number of neighbours, input and output channel for each feature extraction block, and number of feature extractors used in RandLA-Net and BridgeNet are set to be the same.

Table 3.5 presents the evaluation metrics obtained for PointNet++, RandLA-Net

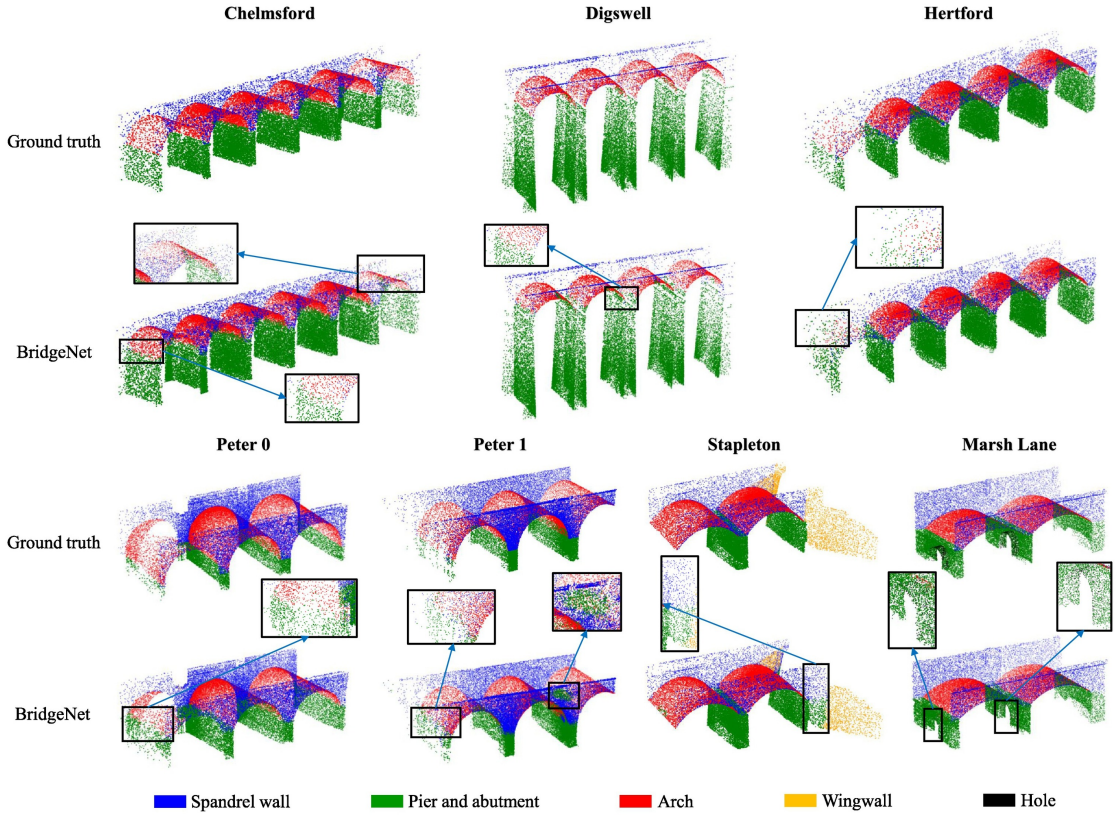


Figure 3.12: Visualizations demonstrating the semantic segmentation results from BridgeNet compared with the 'ground truth' for the real bridge test dataset. Misclassifications are highlighted and shown inset with a zoomed-in view within black bounding boxes.

and BridgeNet. The best results for each metric are marked with **bold numerals** and underlined. BridgeNet outperforms the other two networks by a large margin except for the case of 100 point clouds. BridgeNet achieves the best results with 2000 point clouds in the training dataset. As the training data size is increased, the $mIoU$ improves steadily, indicating that the synthetic dataset provides valuable new information that helps to improve the performance of the network in segmenting real point clouds. To better understand how BridgeNet performs on real bridge point clouds, the $mIoU$ for each component is shown in Table 3.6. Values of 0 mean that the point cloud does not contain the corresponding class. Both Peter 0 and Peter 1 are poorly classified with average $mIoU$ of 0.762 and 0.836, mainly due to the poor classifications of the pier class. For the wingwall and hole, the network achieves satisfactory performance for the wingwall class by achieving a $mIoU$ value of 0.704. However, the hole is not well segmented with a $mIoU$ value of 0.513.

Figure 3.12 visualizes the segmentation results and the ground truth labelling of

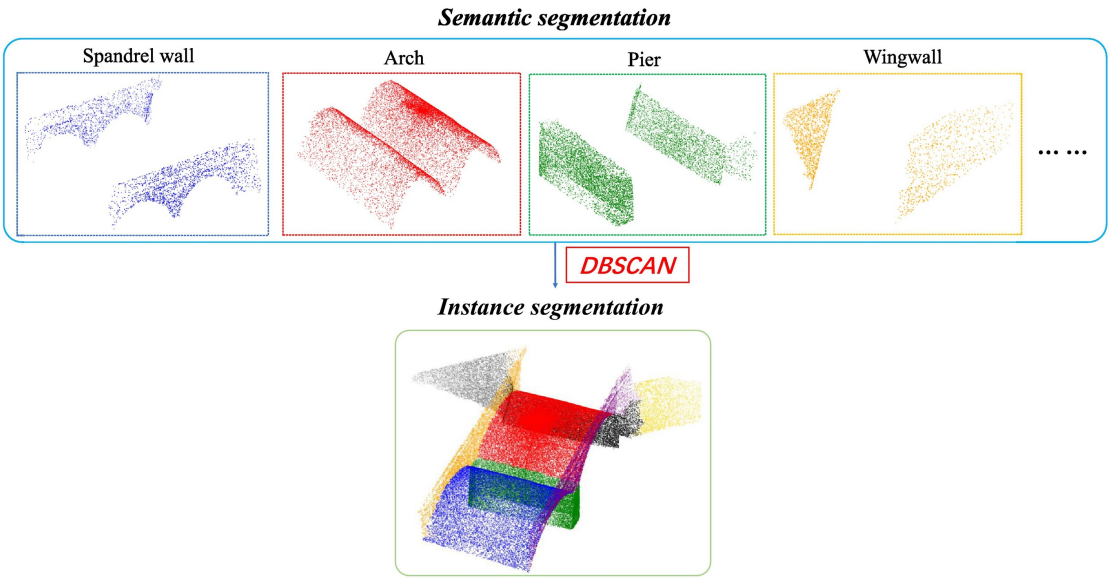


Figure 3.13: Example of instance segmentation for all bridge components of the Stapleton Road viaduct by using DBSCAN.

each bridge. Bridge components could be identified correctly, despite significant differences in bridge geometry and point cloud density amongst the real dataset examples. The misclassifications are highlighted with inset figures, showing close-ups of the misclassified areas. The piers of Peter 0 and Peter 1 on the leftmost side are misclassified with an arch class, likely due to the low density of points in this region. Similar misclassification is observed for other test cases near the arch springing points, indicating that the edge between the objects is not well captured by the neural network. This could be due to the downsampled size of the point cloud, which is used for evaluation. From Figure 3.12, it can be seen that the hole in the Marsh Lane viaduct is partially misclassified as a pier element. Nonetheless, despite its truncated shape, the outlines of the hole feature can still be identified. According to Table 3.6, the wingwall class has a relatively low average class $mIoU$ compared with the arch, pier and spandrel wall classes. This discrepancy arises as synthetic point clouds do not model the curved transition between the abutment and the wingwall. Adapting more detailed geometry in synthetic models, or including real point clouds with such features in the training dataset, is likely to improve accuracy further.

Table 3.7: Two parameters of DBSCAN used for instance segmentation of each bridge after semantic segmentation.

		Chelmsford	Digswell	Hertford	Peter 0	Peter 1	Stapleton	Marsh Lane
Arch	*eps*	0.03	0.03	0.03	0.04	0.04	0.03	0.03
	min_samples	50	40	50	40	40	40	40
Pier	*eps*	0.1						
	min_samples	100						

3.5 Post-processing

3.5.1 Instance segmentation

Since the network only gives the semantic segmentation results, the unsupervised learning method DBSCAN [Ester et al. 1996] is used to perform instance segmentation to cluster points into separate objects as shown in Figure 3.13. The performance of the algorithm depends on two parameters *eps* and *min_samples*, which are the maximum distance between two samples for one to be considered in the neighbourhood of the other and the number of samples in a neighbourhood for a point to be considered as a core point, respectively. Two parameters used in the algorithm had to be adjusted following a trial and error approach to achieve the best instance segmentation performance as shown in Table 3.7. The outliers were automatically removed by filtering out clusters that contain points lower than the defined threshold.

3.5.2 Geometric feature extraction

Two difficult to obtain geometric parameters, the generatrix and the arch radius, are extracted from the manually segmented 'ground truth' point clouds as true values. These are then compared to geometric parameters extracted from the segmented arch and pier point clouds by BridgeNet and DBSCAN.

The generatrix and radius of arches can be calculated using a Random Sample Consensus (RANSAC) based iterative geometric extraction algorithm. The algorithm first fits a 3D cylinder function to a random selection of a small set of points (20 points in our case) from an arch class. The shortest distance between each remaining point and the fitted generatrix (uniquely determined by the centre and the direction of the generatrix) is then calculated. By subtracting the fitted radius

Table 3.8: RMSE is associated with the radius and the generatrix angle obtained by fitting cylinders to arch point clouds of the Chelmsford viaduct segmented from BridgeNet. The errors are relative to the radius and generatrix estimations from cylinders fitted manually sampled point clouds ('ground truth').

Arch num	**Radius (m)**	**Generatrix (°)**
6	$8.43 * 10^{-3}$	$1.70 * 10^{-1}$

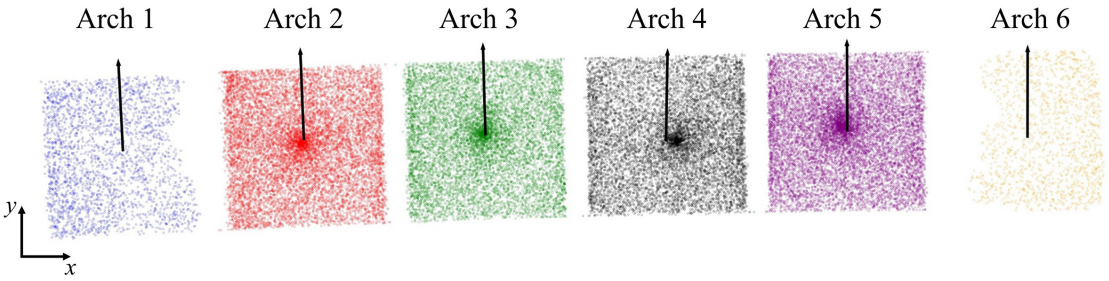


Figure 3.14: Arches segmented by BridgeNet are projected on the xy plane. The extracted generatrix from predicted arches is shown in black arrows.

of the cylinder e_{radius} from this value, the residual error of the radius is obtained for each point. The number of points on the cylinder, n_{c_fit}, can then be determined by filtering points with residual errors larger than a pre-defined threshold. Then, RANSAC is applied to iteratively select points to fit the 3D cylinder and output two metrics: n_{c_fit} and e_{radius}. Those metrics are evaluated at every loop until the residual error becomes smaller than a pre-defined error threshold or the maximum iteration number is reached.

The extracted geometric parameters in the final step of the algorithm for the segmented arches are compared to the ground truth results in Table 3.8 for the Chelmsford Viaduct. Root mean squared errors(RMSE) are specified since the geometric extraction was conducted for multiple arches. The RMSE for the radius is smaller than 1cm, indicating remarkable accuracy. The direction of the generatrix is captured with an RMSE of less than 0.2°, which shows the robustness of the segmentation results. This is demonstrated visually with a top view of the arch point clouds and their generatrix (represented by black arrows) in Figure 3.14. These results demonstrate that the developed algorithm is capable of performing geometric extraction accurately even for the incomplete scans of arch 1 and arch 6.

3.6 Conclusion

The development of automated and generalisable techniques for semantic and instance segmentation can facilitate the assessment, inspection and monitoring of masonry railway bridges. In this work, a new deep learning network called BridgeNet was developed to perform partial segmentation on large-scale masonry point clouds. This network was inspired by RandLA-Net [Hu et al. 2020] and FG-Net [Kai Liu et al. 2020] and modified based on their architecture to conduct semantic segmentation on large-scale point clouds. Instead of deformable convolution kernels, BridgeNet applies a 1×1 convolution layer to capture local feature correlations; this approach was observed to be more robust in the experiments. Furthermore, maxpooling is used rather than summations to aggregate neighbour point features, which alleviated overfitting issues.

Due to the scarcity of real data for training, a masonry arch bridge point cloud simulator was developed. BridgeNet was then trained with a solely synthetic dataset from this simulator and tested on a dataset composed of real point clouds from 7 railway bridges in the UK, each with significantly different geometric properties. The results were used to evaluate the performance of BridgeNet compared with two state-of-art neural networks PointNet++ and RandLA-Net. The results demonstrate that BridgeNet segments the point clouds with remarkable accuracy, and outperforms other networks by a significant margin. This result also highlights the success of the synthetic point cloud simulator in capturing global geometric properties of real bridges and justifies the research methodology developed in this paper. Segmentation errors occasionally arise due to the simplistic representation of geometry in the bridge simulator.

An unsupervised method called DBSCAN was used for instance segmentation. This enabled the investigation of key geometric parameters of the bridge. In particular, by fitting cylinders to arch instances using a RANSAC based approach, two key geometric parameters, arch radius of curvature and generatrix direction, were obtained. The geometric parameters obtained from point clouds segmented by BridgeNet, were compared to ground truth values, demonstrating a high accuracy. This demonstration emphasized the robustness of segmentation and highlighted the useful geometric information that can be gained from segmented point clouds.

Currently, our synthetic dataset simplifies the geometry of real masonry arch bridges (e.g. by excluding components, such as parapets) and the simulation of laser scanning; these aspects may be limiting the segmentation accuracy. Furthermore, in the current pipeline, BridgeNet performs semantic segmentation only, and unsupervised clustering methods, such as DBSCAN, are necessary for instance segmentation. Since hyper-parameters for DBSCAN needs to be adjusted for different cases, an end-to-end network might be proposed in the future to achieve a more efficient pipeline.

Acknowledgement

The authors acknowledge Dan Brackenbury and Robert Gayer (University of Cambridge) and Matthew DeJong (University of Berkeley) who collected and shared the masonry arch bridge point cloud data used in this study.

Statement of Authorship in Chapter 3

Paper title	Segmentation of large-scale masonry arch bridge point clouds with a synthetic bridge simulator and the BridgeNet neural network
Authors	Yixiong Jing, Brian Sheil, Sinan Acikgoz
Publication status	Published
Publication details	Automation in Construction, 2022.

Student Confirmation

Student name	Yixiong Jing	
Contribution to the paper	First-author contribution: • conception of research ideas • design and implementation of models • writing and presentation of the paper	
Signature and Date		Jan. 15th 2025

Supervisor Confirmation

By signing the Statement of Authorship, the supervisor is certifying that the candidate made a substantial contribution to the publication, and that the description above is accurate.

Supervisor name	Prof. Sinan Acikgoz	
Supervisor comments		
Signature and Date		Jan. 15th 2025

Chapter 4

Transformer-based segmentation of masonry arch bridge point clouds

The paper has been accepted for publication at the Computer-Aided Civil and Infrastructure Engineering, 2024.

A lightweight Transformer-based neural network for large-scale masonry arch bridge point cloud segmentation

Yixiong Jing*
University of Oxford
Department of Engineering Science,
University of Oxford,
Parks Road, Oxford, UK
yixiong.jing@wolfson.ox.ac.uk

Brian Sheil
University of Cambridge
Construction Engineering,
University of Cambridge,
Cambridge, UK
bbs24@cam.ac.uk

Sinan Acikgoz
University of Oxford
Department of Engineering Science,
University of Oxford, Parks Road, Oxford, UK
sinan.acikgoz@eng.ox.ac.uk

May 26, 2025

Abstract

Transformer architecture based on the attention mechanism achieves impressive results in natural language processing (NLP) tasks. This paper transfers the successful experience to a 3D point cloud segmentation task. Inspired by newly proposed 3D Transformer neural networks, this paper introduces a new Transformer-based module, which is called **Local Geo-Transformer**. To alleviate the heavy memory consumption of the original Transformer, Local Geo-Transformer only performs the attention mechanism in local regions. It is designed to alleviate the information loss caused by the sub-sampling of point clouds for segmentation. **Global Geo-Transformer** is proposed to exploit the global relationships between features with the lowest resolution. The new architecture is validated on a masonry bridge dataset developed by the authors for their earlier work on a previous segmentation network called BridgeNet. The new version of the network with transformer architecture, BridgeNetv2, outperforms BridgeNet in all metrics. BridgeNetv2 is also shown to be lightweight and memory efficient, well-adapted to large-scale point cloud segmentation tasks in civil engineering.

4.1 Introduction

Laser scanning has been widely adopted for 3D data collection in the civil engineering industry. It allows rapid collection of accurate and rich 3D data. The convenience of portable laser scanners has facilitated their widespread use to document buildings and infrastructure Yuhui Zhou et al. [2017] and Kang et al. [2014]. The generated data points are collectively called a 'point cloud' which contains detailed shape information of the scanned objects. To enable structural analysis and inspection using point cloud data, via for instance, geometry extraction Y. Jing et al. [2022] or defect inspection C. Ye et al. [2018] and Y. Zhang et al. [2024], it is essential to segment the point cloud into different structural components. In particular, point cloud data may provide useful information in the assessment, maintenance, and retrofit of masonry arch bridges, which often lack construction drawings.

However, the typically large size of point clouds of civil engineering structures renders manual segmentation time-consuming. Deep learning (DL) technologies can alleviate this problem by automating the segmentation task J. Lin et al. [2022] and García-Aguilar et al. [2023]. Compared to traditional feature-based segmentation algorithms R. Lu et al. [2019b], Riveiro et al. [2016], and Walsh et al. [2013], DL does not require apriori definition of features since they can be implicitly generated by the neural network. As such, DL can provide a general framework to address a wide variety of segmentation tasks. Its competitive point cloud segmentation performance has been demonstrated by J. Ma et al. [2020] and Matrone et al. [2020] for heritage buildings and by Xia et al. [2022] for reinforced concrete slab and beam-slab bridge structures.

In contrast to well-established DL architectures in the 2D image domain, it was not until recently that researchers started to focus on point cloud processing. Early attempts [Çiçek et al. 2016; B. Li 2017b] to leverage mature methods of image processing tasks involved point cloud data voxelization. This approach encodes the spatial organization information and enables the use of direct extensions of 2D neural networks such as U-net [Ronneberger et al. 2015] to 3D. However, voxelization is computationally intensive; computational demands grow cubically with increasing voxel resolution. Sparse data representation [Riegler et al. 2017] and

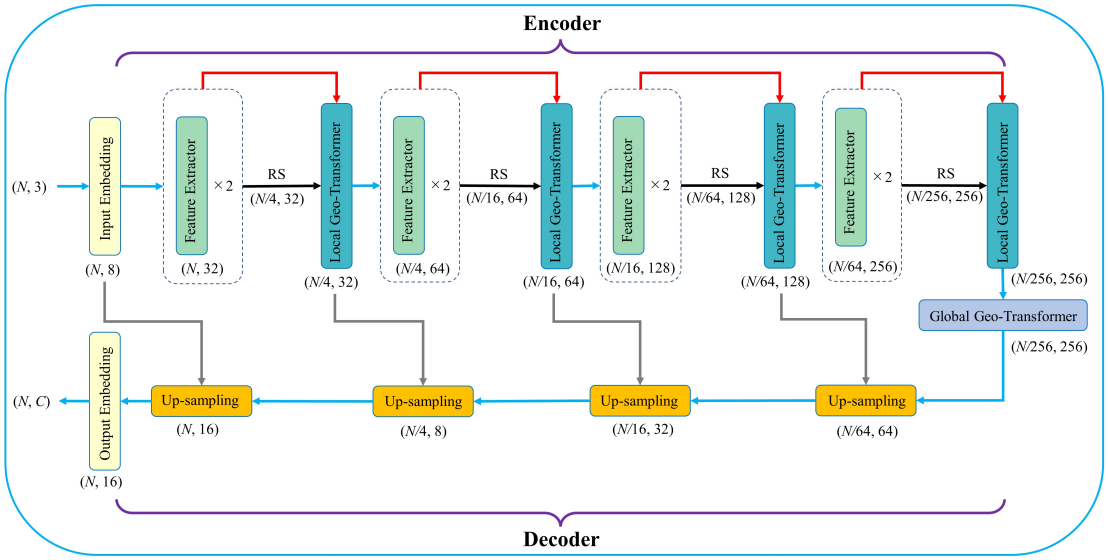


Figure 4.1: BridgeNetv2 proposes Local Geo-Transformer and Global Geo-Transformer to perform attention mechanisms on local and global scales for point cloud segmentation. They are inserted into the original BridgeNet architecture.

sparse 3D convolution [Choy et al. 2019b; Tang et al. 2020; M. Ye et al. 2021] techniques can be exploited to reduce the computational requirements. Regardless, voxelization may result in information loss by preserving only a single point for every voxel.

PointNet [Qi et al. 2017a] and PointNet++ [Qi et al. 2017b] directly take raw and unstructured point clouds as input. They have been demonstrated to be successful in many point cloud processing tasks, including segmentation. The architectures have been subsequently adapted and improved by successive research. For instance, M. Jiang et al. [2018] designed a new feature descriptor based on the famous 2D shape descriptor SIFT to be adaptive to the shape scale. 3D convolution kernels [Yuelin Liu et al. 2019b; Thomas et al. 2019; W. Wu et al. 2019] which can be applied directly on point clouds have also been proposed. Yuelin Liu et al. [2019a] extends the regular grid-based convolution to 3D to capture contextual semantic information. Zhirong Zhang et al. [2019] defined an efficient neural network by aggregating features of nearby points within spherical neighborhoods of different radii.

The networks developed in early studies suffer from heavy memory consumption and slow inference on large-scale point cloud datasets, such as the KITTI dataset [Menze and Geiger 2015], which has millions of points per scan. The inability of the algorithms in these earlier studies to deal with large-scale point clouds has impeded

their implementation for civil engineering applications. Hu et al. [2020] proposed RandLA-Net with a lightweight architecture that can process large-scale point clouds efficiently. FG-Net [Kangcheng Liu et al. 2020] inherits the lightweight feature of RandLA-Net and incorporates a more complex feature extractor. It demonstrated state-of-the-art segmentation performance on the Shape-Net dataset [Chang et al. 2015]. The BridgeNet architecture combines aspects of the RandLA-Net and FG-Net to conduct semantic segmentation in a large-scale masonry arch bridge dataset [Y. Jing et al. 2022]. BridgeNet outperforms PointNet++ and RandLA-Net for this dataset. However, BridgeNet's performance is not entirely satisfactory when it comes to dealing with imbalanced classes in the masonry bridge dataset; the segmentation of small classes (such as holes in bridge piers which existed only in a few cases) tends to be more error-prone. A potential reason for this is the adoption of sub-sampling after each feature extractor block in the BridgeNet architecture; this step may be discarding important features and exacerbating information loss, particularly for small classes.

The Transformer architecture [Vaswani et al. 2017] has achieved success in various NLP [Devlin et al. 2018; Z. Yang et al. 2019; Yiheng Liu et al. 2023] and image processing [Carion et al. 2020; H. Fan et al. 2021; Peh et al. 2022] tasks. It is particularly suitable for point cloud processing since the attention mechanism of the Transformer is invariant to the input point cloud permutation [H. Zhao et al. 2021b]. Transformer was initially introduced for feature aggregations in raw point clouds to highlight important features in local regions [Xie et al. 2018; X. Liu et al. 2019]. J. Yang et al. [2019] modified the Multi-Head attention of Transformer [Vaswani et al. 2017] by using nonlinear activation rather than a linear layer while producing 'value' matrices; this modification makes the networ more parameter-efficient. J. Lee et al. [2019] utilizes Transformer for encoding and feature extraction, showing promising results.

However, early attempts to use Transformer architecture perform the attention mechanism globally on the point cloud. This leads to increased computational complexity and memory cost are $O(n^2)$. Different methods [Rabe and Staats 2021; Dao et al. 2022] have been proposed to mitigate the increased memory cost, but they increase the computation times. S. Wang et al. [2020] proposes an approximation which manages to achieve $O(n)$ in time and memory costs.

However, this approximation may potentially compromise the overall performance of the Transformer due to information condensing operations.

To minimize the computation and memory costs while preserving good accuracy, two different methods are proposed for applying the attention mechanism on point clouds: (i) Instead of computing attention globally, local attention can be performed on raw point clouds where neighbor regions are defined by k-nearest neighbors algorithm (k-NN) [Matveev et al. 2020; H. Zhao et al. 2021b; Fu et al. 2023], or on neighbor voxels [Park et al. 2022; Lai et al. 2022b], (ii) points in neighboring regions with similar features are clustered into super points to substantially decrease the data size. Attention mechanism can be then performed on super points [Qin et al. 2022; C.-K. Yang et al. 2022; Yahui Liu et al. 2023] globally. Super points can be also extracted through a neural network to represent features of local regions [X. Yu et al. 2023b]. Both methods dramatically reduce the computational complexity of the vanilla Transformer from $O(N^2)$ to $O(N)$ while preserving the Transformer architecture.

Inspired by the cited studies, we propose BridgeNetv2 which adapts the Transformer on the local scale for feature extraction from raw point clouds. The attention mechanism is performed only on points that are defined by k-NN, which alleviates the memory consumption by directly adopting the vanilla Transformer. Positional encoding is commonly used to provide positional information of words in NLP. Similar encoding information is implicitly contained in point clouds, and different types of positional encoding are tested. Global attention is performed to explore point-wise relationships in the feature maps with the lowest resolution.

BridgeNetv2 is trained on a synthetic masonry arch bridge point cloud dataset developed in the authors' earlier work [Y. Jing et al. 2022]. That study demonstrated how a well-designed synthetic dataset may be able to replace real data in training. BridgeNetv2 is then tested on 7 real bridge point clouds. A similar strategy regarding the use of synthetic data for training has been adopted in other recent studies [Lamas et al. 2024; Korus et al. 2023], due to the lack of suitable open-source point cloud data for conducting segmentation training of civil engineering assets. Due to this data scarcity issue, BridgeNetv2 is only applied to the masonry arch bridge dataset available to the authors. However, the network can be readily customized to segment other large-scale civil engineering structure

point clouds.

This paper develops a new Transformer-based block that is implemented in BridgeNet architecture. The block is designed such that it may be easily inserted into different deep learning architectures without significant modifications. The details of the architecture and block are explained in Section 2. In section 3, the performance of BridgeNetv2 is evaluated quantitatively. An ablation test is also performed to validate the selection of structures and hyper-parameters. Section 4 highlights the contributions of this paper.

4.2 Methodology

The architecture of BridgeNetv2 is shown in Figure 4.1. It has a classic encoder-decoder architecture. For the feature encoder, there are a total of four stages for extracting and downsampling features progressively. We adopt random sampling (RS), with a sampling rate of 4, similar to RandLA-Net [Hu et al. 2020]. The input point cloud, which has a size of $(N, 3)$, is first encoded by Input Embedding to the higher dimension of $(N, 8)$. At each stage, two feature extractor (FE) layers are applied to extract and aggregate local geometric features. The FE layers are identical to those in BridgeNet [Y. Jing et al. 2022].

In BridgeNet, max pooling was used for aggregating neighbor features. This operation is known to typically extract high-frequency features [Yahui Liu et al. 2023]. As a result, low-frequency geometric information may be lost. In BridgeNetv2, a Local Geo-Transformer is inserted between each FE block for computing local attention on the point clouds before and after RS. This new feature is designed to alleviate information loss; Local Geo-Transformer aggregates information from points that are abandoned by RS and preserves the geometric information from the shallow layers. The schematic in Figure 4.2 illustrates this. In BridgeNet, the original point cloud (represented by silver points) is sub-sampled such that blue points are abandoned, and only the center red point moves on to the next FE block. In BridgeNetv2, k-NN is used to search for neighbors (green points) of the center red point amongst the abandoned points, so that the center point can aggregate features of its neighbors with the new Local Geo-Transformer module. Global Geo-Transformer is designed to compute self-attention globally on the

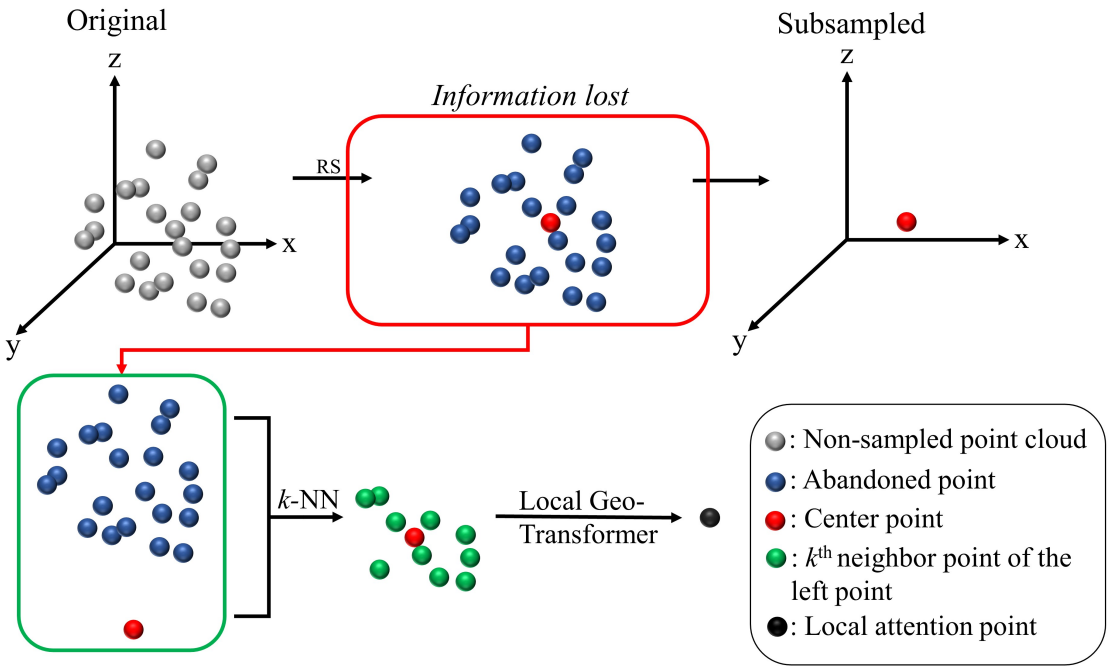


Figure 4.2: Illustration of aggregating features from abandoned points into the remaining center point by using the Local Geo-Transformer layer.

lowest resolution features, to capture long-range feature dependencies [Yahui Liu et al. 2023].

The architecture and implementation of BridgeNetv2 differs from BridgeNet only in the Local Geo-Transformer and Global Geo-Transformer. Therefore, details regarding the general architecture are not presented in this paper; only the new components are discussed below. Readers interested in further details regarding the BridgeNet architecture and layers are referred to Y. Jing et al. [2022].

4.2.1 Local Geo-Transformer

The specific implementation of the Local Geo-Transformer in BridgeNetv2 is shown in the orange box of Figure 4.3. The input $\boldsymbol{p}_1$:(N, 3) and $\boldsymbol{p}_2$:(M, 3) are point clouds before and after the sub-sampling respectively, in which N and M ($N > M$ in this case) represent the point cloud size and 3 indicates the x, y, and z coordinates. $\boldsymbol{f}_1$ and $\boldsymbol{f}_2$ are the corresponding features with feature dimensions C_{in_1} and C_{in_2} respectively. The attention is computed locally on $\boldsymbol{f}_1$ and $\boldsymbol{f}_2$. This implies that input vectors for the attention are computed locally on k neighbors in each point cloud. For instance, neighbors are searched from $\boldsymbol{p}_1$ which are the k nearest points of each point in $\boldsymbol{p}_2$. The index of k neighbors is given by $\boldsymbol{index}_K \in \mathbb{R}^{M \times K}$.

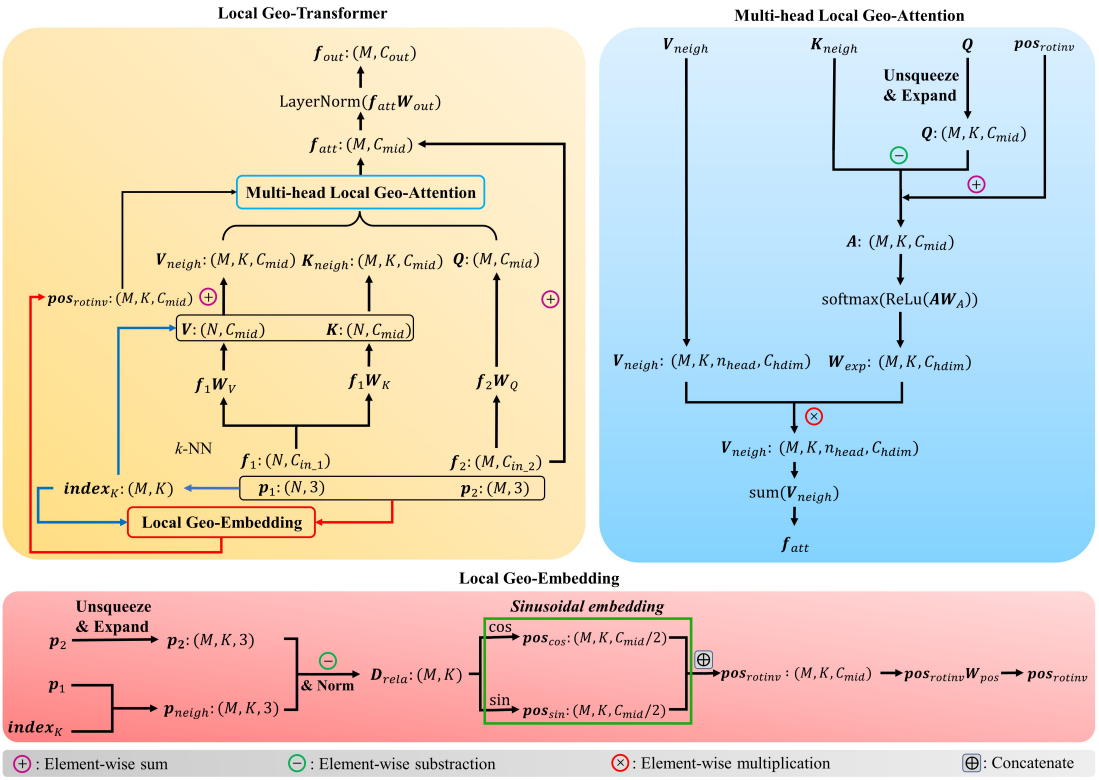


Figure 4.3: The general architecture of the Local Geo-Transformer is shown in the orange box. A new $\boldsymbol{pos}_{rotinv}$, which is computed from the Local Geo-Embedding module, is utilized as shown in the bottom red box. Multi-head Local Geo-Attention is used for computing attention and is described in the blue box.

Similar to the vanilla Transformer, $\boldsymbol{f}_1$ and $\boldsymbol{f}_2$ are first linearly projected by $\boldsymbol{W}_V$, $\boldsymbol{W}_K$, and $\boldsymbol{W}_Q$ to value $\boldsymbol{V} \in \mathbb{R}^{M \times C_{mid}}$, key $\boldsymbol{K} \in \mathbb{R}^{M \times C_{mid}}$, and query $\boldsymbol{Q} \in \mathbb{R}^{N \times C_{mid}}$, where C_{mid} represents the feature dimension. $\boldsymbol{index}_K$ is used to extract the K features from $\boldsymbol{V}$ and $\boldsymbol{K}$, which are defined by the k-NN in the Euclidean space of $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$. This operation yields $\boldsymbol{V}_{neigh} \in \mathbb{R}^{M \times K \times C_{mid}}$ and $\boldsymbol{K}_{neigh} \in \mathbb{R}^{M \times K \times C_{mid}}$. For each row of features in $\boldsymbol{Q}$, the corresponding row in $\boldsymbol{V}_{neigh}$ and $\boldsymbol{K}_{neigh}$ contains K features from the corresponding neighbor points defined by k-NN. This ensures that the attention is computed on point sets within the same segmentation class. Before the attention computation, a rotationally invariant positional embedding (denoted by $\boldsymbol{pos}_{rotinv} \in \mathbb{R}^{M \times K \times C_{mid}}$ and computed by the submodule Local Geo-Embedding which will be discussed later in Section 2.3) is added to $\boldsymbol{V}_{neigh}$ (element-wise) to explicitly provide geometric information. Local Geo-Embedding is also performed on the local regions which are defined by the $\boldsymbol{index}_K$ to be consistent with $\boldsymbol{V}_{neigh}$ and $\boldsymbol{K}_{neigh}$. The $\boldsymbol{pos}_{rotinv}$ is also used in the computation of Multi-head Local Geo-Attention submodule to derive the attention score. The attentive feature $\boldsymbol{f}_{att} \in \mathbb{R}^{M \times K \times C_{mid}}$ is then summed with the $\boldsymbol{f}_2$ element-wisely and sent

to a linear layer followed by a LayerNorm layer to obtain the final output $\boldsymbol{f}_{out} \in \mathbb{R}^{M \times K \times C_{out}}$ where C_{out} is the output feature dimension.

4.2.2 Multi-head Local Geo-Attention

The structure of Multi-head Local Geo-Attention is shown in the blue box of Figure 4.3. A new dimension is first unsqueezed on $\boldsymbol{Q}$ for it to attend the element-wise computation. The feature is then replicated K times in the newly created dimension to expand it to the same size as the rest of the inputs, i.e., (M, K, C_{mid}).

The multi-head mechanism, which is also used in the vanilla Transformer, is adopted to allow joint attention of the information from different representation subspaces [Vaswani et al. 2017]. In practice, the feature dimension of $\boldsymbol{V}_{neigh}$ is reshaped from a single dimension of C_{mid} to the dimensions (n_{head}, C_{hdim}), where $C_{mid} = n_{head} \times C_{hdim}$. The n_{head} represents the number of subspaces and the C_{hdim} is the dimension of the feature channel. It is expected that each subspace can focus on different geometric features to improve the general performance. Different from the dot product operation used in the vanilla Transformer, the vector based attention is used, as proposed by [H. Zhao et al. 2021b].

$$\boldsymbol{W}_{exp} = \text{softmax}(\text{ReLu}((\boldsymbol{K}_{neigh} - \boldsymbol{Q} + \boldsymbol{pos}_{rotinv})\boldsymbol{W}_A)) \tag{4.1}$$

Equation (4.1) describes the calculation of the attention score. In this equation, $\boldsymbol{pos}_{rotinv}$ explicitly provides relative position information. The resultant intermittent attention score is represented as $\boldsymbol{A} \in \mathbb{R}^{M \times K \times C_{mid}}$ in Figure 4.3. We propose to also use linear $\boldsymbol{W}_A \in \mathbb{R}^{C_{mid} \times C_{hdim}}$ and non-linear activation ReLU layers, which are shown in Equation (4.1) and Figure 4.3, to make the attention score learnable. The softmax layer is followed by the attention score $\boldsymbol{W}_{exp} \in \mathbb{R}^{M \times K \times C_{hdim}}$. For parameter efficiency, the feature dimension C_{mid} of $\boldsymbol{A}$ is projected to C_{hdim} by the same linear layer in different heads, which results in the attention score for features in subspaces being shared.

After computing the element-wise multiplication of $\boldsymbol{V}_{neigh}$ and $\boldsymbol{W}_{exp}$, the feature channel of $\boldsymbol{V}_{neigh}$ is re-weighted. The last step, which is shown in Figure 4.3, adds the re-weighted features of k neighbors to obtain the output $\boldsymbol{f}_{att} \in \mathbb{R}^{M \times C_{mid}}$.

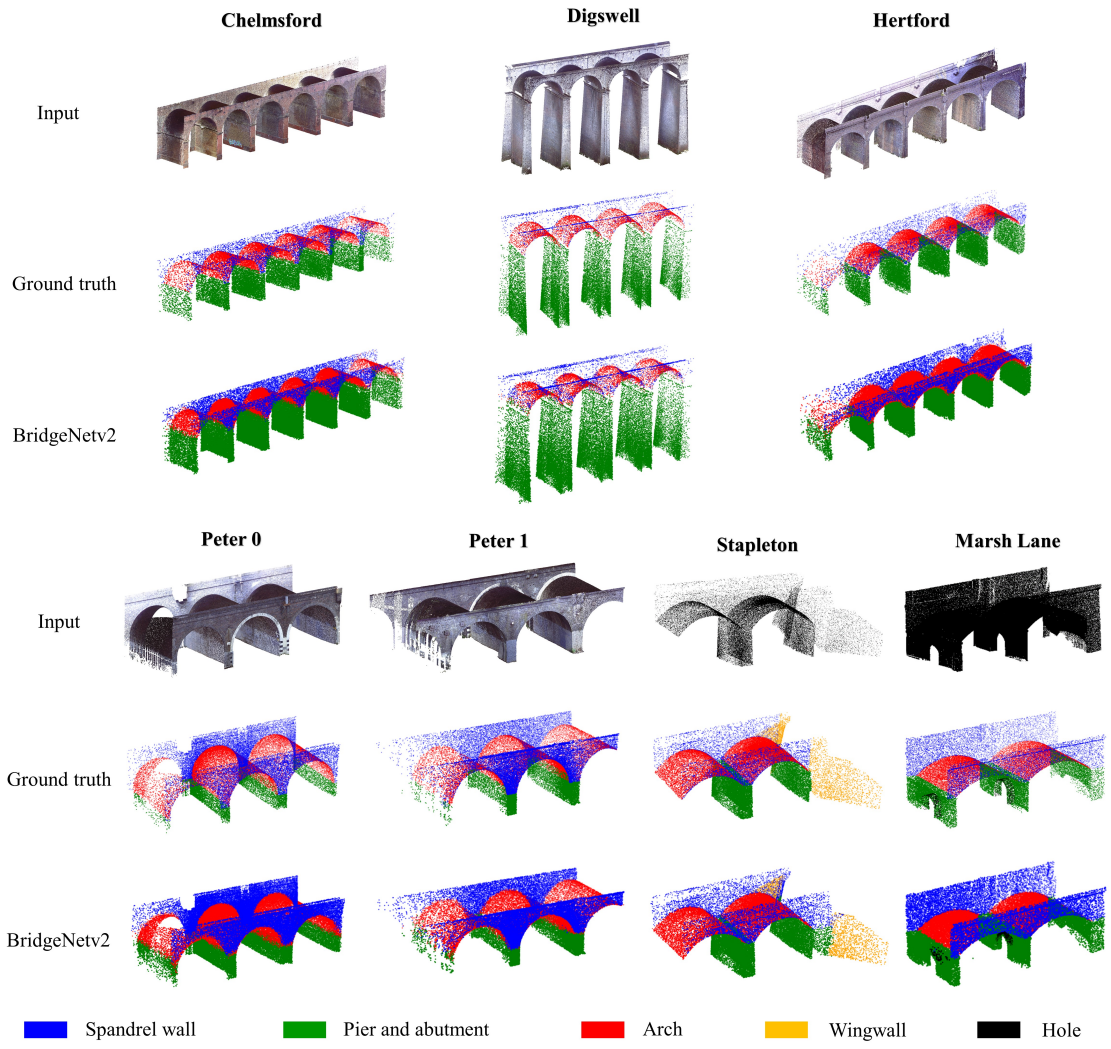


Figure 4.4: Visualization of the BridgeNetv2 semantic segmentation results on 7 masonry bridges from the test dataset. The raw input and ground truth data are also provided for comparison.

4.2.3 Local Geo-Embedding

Positional embeddings are commonly used in NLP for a unique representation of each word in a different position. The sinusoidal embedding is used in the vanilla transformer [Vaswani et al. 2017] as follows:

$$pos_{i,2j} = \sin\left(\frac{i}{10000^{2j/d_{model}}}\right) \tag{4.2}$$

$$pos_{i,2j+1} = \cos\left(\frac{i}{10000^{2j/d_{model}}}\right) \tag{4.3}$$

where $pos_{i,2j}$ and $pos_{i,2j+1}$ represent values of $\boldsymbol{pos}$ at indices $(i, 2j)$ and $(i, 2j+1)$.

Here, i and j represent the row and column index with $0 < i \leq L/2$ and $0 < j \leq (d_{model} - 1)/2$ for L. L and d_{model} represent the length and feature dimension of the positional embedding. i can be regarded as the representation of the position of word embeddings. j is defined over the feature dimension which generates a repeated feature pattern since periodic sine and cosine functions are used.

The Local Geo-Embedding($\boldsymbol{pos}_{rotinv}$) is introduced to the attention score computation in Equation (4.1), as it plays an important role in highlighting discriminant geometrical features [Qin et al. 2022] in point clouds.

Instead of the approach presented in Equations (4.2) and (4.3), point coordinates may be directly adopted as embeddings [J. Yang et al. 2019]. However, this approach is not rotationally invariant. Most point cloud data (including the datasets used for training and testing BridgeNet) feature rigid transformations employed for georeferencing. The attention score will be dependent on rigid rotation if positional embeddings are not rotationally invariant. Therefore, a simple and memory-efficient method is proposed based on the *pair-wise distance embedding* [Qin et al. 2022]. As shown in the red box at the bottom of Figure 4.3, $\boldsymbol{p}_2$ is unsqueezed and expanded in the second dimension to compute the relative coordinate difference between each point in $\boldsymbol{p}_2$ and the corresponding k neighbor points from $\boldsymbol{p}_{neigh}$. $\boldsymbol{p}_{neigh}$ is extracted by gathering points from $\boldsymbol{p}_1$ based on $\boldsymbol{index}_K$. The second norm is then computed on the last dimension of the coordinate difference for the rotationally invariant feature $\boldsymbol{D}_{rela} \in \mathbb{R}^{M \times K}$. $\boldsymbol{D}_{rela}$ measures the relative distance between the center and its neighbor points, which remain constant for any rigid transformation.

In Equations (4.2) and (4.3), i is the indicator of the location. In BridgeNetv2, i is substituted by the relative distance in each entry of $\boldsymbol{D}_{rela}$. Both i and $\boldsymbol{D}_{rela}$ represent the location of elements in sequences; the former operates in 1D while the latter operates in 3D space. For each element in $\boldsymbol{D}_{rela}$, sin and cos functions encode the element into a periodic feature pattern $\boldsymbol{pos}_{cos} \in \mathbb{R}^{M \times K \times C_{mid}/2}$ and $\boldsymbol{pos}_{sin} \in \mathbb{R}^{M \times K \times C_{mid}/2}$ respectively. The resultant $\boldsymbol{pos}_{rotinv} \in \mathbb{R}^{M \times K \times C_{mid}}$, which is obtained by concatenating $\boldsymbol{pos}_{cos}$ and $\boldsymbol{pos}_{sin}$, is passed through a linear layer for the output. Compared to coordinate embedding, rotational invariant embedding achieves better performance for independent on rigid transformations, which will be demonstrated in the ablation tests presented in Section 3.3.

Table 4.1: Comparison of the performance of PointNet++, RandLA-Net, BridgeNet, and BridgeNetv2 for semantic segmentation on the test dataset in *Accuracy*, *mean_Pr*, and *mIoU*. The *IoU* for each of the segmentations is shown in the last five columns.

Method	***Accuracy***	***mean_Pr***	***mIoU***	**Spandrel**	**Pier**	**Arch**	**Wingwall**	**Hole**
PointNet++	0.913	0.641	0.586	0.830	0.842	0.821	0.412	0.025
RandLA-Net	0.914	0.665	0.574	0.806	0.825	0.840	0.388	0.011
BridgeNet	0.945	0.741	0.635	0.849	0.870	0.842	0.541	0.073
BridgeNetv2	**0.948**	**0.796**	**0.726**	0.850	0.933	0.922	0.667	0.258

4.2.4 Global Geo-Transformer

To enable the network to capture global context relationships, the Global Geo-Transformer block is added between the encoder and decoder of the network as shown in Figure 4.1. Global Geo-Transformer is only performed on the feature with the lowest resolution for memory efficiency. Global Geo-Transformer broadly shares the same architecture with the Local Geo-Transformer, except for two key differences. First, the Global Geo-Transformer computes self-attention, which means that the input $\boldsymbol{f}_1$ and $\boldsymbol{f}_2$ shown in Figure 4.3 represent the same feature. Second, the attention is performed globally instead of locally, such that each feature will depend on the rest of the features. Global Geo-Transformer is designed to substitute the role of Global Feature Extractor, which was used in BridgeNet.

4.3 Experiments and Results

4.3.1 Implemention details

BridgeNetv2 is implemented using Pytorch. We use Adam as the optimizer with an initial learning rate of 10^{-3} and a decay rate of 10^{-4}. The cross-entropy loss is used as the objective function. These parameters and functions were observed to work well for BridgeNet and were adopted for BridgeNetv2. The x, y, and z coordinates of input data are normalized to the range of $[-1, 1]$. The coordinates are then randomly rotated around the z-axis with an angle sampled from 0 to 2π. The normalized point cloud is also rescaled with a scalar between 0.8 and 1.25 and shifted from 0 to 0.1 relative to the x, y, and z directions for data augmentation. The mini-batch size is set to 4 and the size of each point cloud is $(65536, 3)$. We use

a 0.2 dropout ratio for both the output of Local and Global Geo-Embedding and 0.8 for the output layer to alleviate overfitting. We use $k = 16$ (in k-NN) for all Local Geo-Embedding layers. Ball query is used in FE for gathering local features that can cover wider regions. To make a fair comparison, we use the same random seed as BridgeNet. The hyperparameters of the FE and Up-sampling blocks can be found in [Y. Jing et al. 2022].

The BridgeNet masonry dataset [Y. Jing et al. 2022] is used for measuring the performance of BridgeNetv2, in which 500 synthetic datasets are used for training and 7 real datasets are used for testing. Except for the Marsh Lane and Stapleton Road viaduct point clouds, all five real datasets feature color information as shown in the input row of Figure 4.4. This color data is only used for visualization. Since most of the test data contains over a million points which is beyond the input size of the network, each real bridge point cloud is randomly sampled into non-overlapping sets with the default input size. Five different classes were defined for semantic segmentation: (i) spandrel wall, (ii) pier, (iii) arch, (iv) wingwall, and (v) hole.

We use the following metrics to indicate the performance: overall accuracy ($Accuracy$), mean precision ($mean_Pr$), and mean intersection over union ($mIoU$). Mathematical definitions of these metrics can be found in [Y. Jing et al. 2022]. The performance of BridgeNetv2 is compared to PointNet++, RandLA-Net, and BridgeNet under the same training settings and environment. BridgeNetv2 is trained for 200 epochs to reliably achieve convergence which takes an average of 11.43h for each training. The experiments are run on an RTX 3080 GPU with 10GB of memory.

4.3.2 Semantic segmentation results

The results are presented in Table 4.1. BridgeNetv2 outperforms PointNet++, RandLA-Net, and BridgeNet according to all metrics. Compared to BridgeNet, BridgeNetv2 achieves 5.5% and 9.1% performance boost on the $mean_Pr$ and $mIoU$ metrics. The IoU for each segmentation class is also presented in Table 4.1. These results indicate that the increased accuracy is mainly due to improved segmentation in classes other than the spandrel wall. The hole class, which only occupies a small portion of the whole dataset (less than 1% of the input points), was

poorly captured in BridgeNet due to the data imbalance. However, BridgeNetv2 improves the segmentation accuracy of the hole by a significant margin. This demonstrates that the newly designed Transformer block is able to capture and aggregate the features even for small-size classes in unbalanced data.

Figure 4.4 visualizes the predictions of BridgeNetv2 on the test dataset. It can be seen that the predictions are close to the ground truth. The boundary of each component is clearly defined, this is evident from the transition from blue to green colors between the spandrel walls and the piers. Incorrect classifications can be seen on the wingwall of the Stapleton Road viaduct and the hole of the Marsh Lane bridge. The first is due to the inconsistency between the geometries of the wingwall components used in the synthetic training and real test datasets. The second arises as a result of data imbalance as described earlier. These observations indicate that the bottleneck in the performance is related to the synthetic test dataset: by adopting an improved simulator that can address data imbalance in augmentation and better approximate real point cloud data, further improvements in accuracy may be achieved.

4.3.3 Ablation tests

To investigate the impact of design choices made in our architecture and evaluate the selection of hyperparameters used to process the data, a series of experiments are conducted by varying the design and hyperparameters of BridgeNet2. The training and test datasets presented earlier are also used for the ablation tests.

Table 4.2: Ablation test on the impact of k local neighborhood sizes used in Local Geo-Transformers on performance metrics.

k	$Accuracy$	$mean_Pr$	$mIoU$
$[16, 16, 32, 64]$	0.923	0.739	0.693
$[64, 32, 16, 16]$	0.900	0.767	0.711
$[32, 32, 32, 32]$	0.932	0.753	0.707
$[16, 16, 16, 16]$	**0.948**	**0.796**	**0.726**
$[8, 8, 8, 8]$	0.939	0.780	0.703

The influence of k, which is used in the Local Geo-Transformer to define the number of neighbors, is explored first. Three different parameter sets in which the k increases, decreases, or is maintained the same with increasing layer depth,

Table 4.3: Ablation test on the impact of different positional embeddings and locations on performance metrics.

Positional embeddings	***Accuracy***	***mean_Pr***	***mIoU***
None	0.933	0.773	0.702
Relative coordinate	0.920	0.763	0.692
Relative distance	0.937	0.775	0.701
Local Geo-Embedding for all	**0.948**	**0.796**	**0.726**
Local Geo-Embedding for attention	0.944	0.777	0.720
Local Geo-Embedding for value	0.945	0.763	0.715

is explored. The results are shown in Table 4.2. The best results are achieved when the k is set to be consistent for all Local Geo-Transformer layers. $k = 16$ yields better performance metrics than $k = 8$ and $k = 32$. The classic strategy of adopting $[16, 16, 32, 64]$, to increase the reception field while sub-sampling features does not improve performance. A larger reception field, which is indicated by a larger k, may aggregate information from points that are less relevant to the center point. As the layer goes deeper, the spatial distance between points is enlarged due to sub-sampling, which means that the same k can cover a much larger reception field in a feature with low resolution compared to a feature with high resolution. This may have introduced extra noise in the segmentation and led to poorer performance.

We also test the performance on different positional embeddings. The results are shown in Table 4.3. The network is first trained on three cases. In case 1, positional embedding is not considered. In case 2, positional embedding is described by the relative coordinates of neighbor and center points, In case 3, relative distances are used instead. Since the resultant feature dimension of the relative coordinate and relative distance metrics are 3 and 1, a linear layer is used to project them to the same size as the $\boldsymbol{pos}_{rotinv}$ of Local Geo-Embedding.

Table 4.3 shows that the performance is notably poorer when no positional embedding is used or relative coordinates are adopted. This result indicates the importance of using rotational invariant $\boldsymbol{pos}_{rotinv}$. Local Geo-Embedding positioning for all, which is the fourth row in Table 4.3, yields the best performance. Different from the relative distance (3rd row), Local Geo-Embedding encodes the information from the relative distance by using sinusoidal embedding and making it learnable. This may effectively maintain the density of the positional informa-

tion compared to the direct linear projection. When $\boldsymbol{pos}_{rotinv}$ is added to only the attention (as shown in Equation (4.1)) or the value $\boldsymbol{V}_{neigh}$ (as shown in Figure 4.3), a performance drop is also observed.

4.3.4 Efficiency analysis

To investigate whether BridgeNetv2 preserves the lightweightness and memory efficiency of BridgeNet, we measure their parameter size and the memory use in Table 4.4. The parameter size is measured in millions and memory usage is in Gigabytes. The parameter size is kept constant in BridgeNet2 since the network architecture remains unchanged.

Memory consumption of BridgeNet and BridgeNetv2 is mainly determined by the largest tensor in each block of the neural network. The size of the tensor is (B, N, k, f), where B, N, and f represent the batch size, number of points, and feature dimension separately. The same B, N, and f are utilised in BridgeNet and BridgeNetv2. The parameter k is used in Local Geo-Transformer of BridgeNetv2 and introduces extra memory consumption. Compared to BridgeNet, the parameter size of BridgeNetv2 is increased by 51.7%. This represents a reasonable increase, which is lightweight compared to other Transformer-based point cloud neural networks. For instance, the parameter size of Fast Point Transformer [Park et al. 2022] is 37.9 M which is 25 times larger than BridgeNetv2.

Table 4.4: Comparison of the parameter size and memory use in BridgeNet and BridgeNetv2 for different neighborhood sizes k.

Method	**Param. (M)**	**Memory (GB)**
BridgeNet	0.989	4.147
BridgeNetv2 ($k = 8$)	1.500	4.248
BridgeNetv2 ($k = 16$)	**1.500**	**4.688**
BridgeNetv2 ($k = 32$)	1.500	5.511

In terms of memory use, BridgeNetv2 with $k = 8$ and $k = 16$ have similar requirements to BridgeNet. The larger size of the resultant tensor in BridgeNetv2, therefore, does not lead to significant additional memory consumption in the Local Geo-Transformer. However, the increase in memory consumption becomes more significant when k is raised to 32 (from 8 and 16) as shown in Table 4.4. The results indicate that the extra memory consumption of BridgeNetv2 compared to

BridgeNet is influenced primarily by k which is utilized to define the neighbors in Local Geo-Transforer.

The inference time of BrdigeNetv2 for processing 1 million points is 25.45s. The size of input tensor is $(4, 65536, 3)$ which corresponds to B, N, and coordinates$(x,\ y\ , z)$. This renders BridgeNetv2 34.4% slower compared to BridgeNet. The bottleneck of inference time is the neighbor searching algorithms(k-NN and ball query). For both BridgeNet and BridgeNetv2, the ball query, which is used in the FE module, remains unchanged. The added time cost is therefore attributed to the k-NN on the Local Geo-Transformer. Despite both neural networks inheriting the efficiency features from RandLA-Net [Hu et al. 2020], they have not been optimized for inference speed. Since inference time can vary significantly on different machines, further investigations were not devoted to this aspect. While the increased memory consumption and inference time may be considered a limitation for BridgeNetv2, it is judged that these aspects are compensated by increased performance.

4.4 Concluding remarks

This paper has described a novel transformer-based DL architecture called BridgeNetv2 to perform semantic segmentation on large-scale masonry arch bridge point clouds. Two Transformer-based modules, called the Local and Global Geo-Transformers, are designed to be inserted into the original BridgeNet. Local Geo-Transformer utilizes an attention mechanism to minimize the information loss during sub-sampling. Global Geo-Transformer exploits contextual correlations of features. A new rotational invariant positional encoding method, which is inspired by Qin et al. [2022], is proposed to enhance the performance.

BridgeNetv2 outperforms PointNet++, RandLA-Net, and BridgeNet in all performance metrics for the segmentation of real masonry bridge point clouds. Ablation tests are conducted on the selection of k neighborhood size and positional embedding parameters, which support the design choices adopted for the network and its parameters. The memory consumption and parameter size analysis show that BridgeNetv2 is still lightweight and can be trained under the same conditions as the original BridgeNet.

Considering its superior performance, it is expected that BridgeNetv2 can inspire

new adaptions of Transformer-based design on the point cloud processing tasks in civil engineering. In the future, the network may be generalized further to tackle segmentation in a wider range of structures, including other types of bridges or structures, such as buildings and tunnels. This will require the compilation of large scale datasets, which may benefit from recent improvements in realistic synthetic data generation. The code will be published on GitHub once the present manuscript is accepted.

Acknowledgements

The authors acknowledge Dan Brackenbury and Robert Gayer (University of Cambridge) and Matthew DeJong (University of Berkeley) who collected and shared the masonry arch bridge point cloud data used in this study.

Statement of Authorship in Chapter 4

Paper title	A lightweight Transformer-based neural network for large-scale masonry arch bridge point cloud segmentation
Authors	Yixiong Jing, Brian Sheil, Sinan Acikgoz
Publication status	Published
Publication details	Computer-Aided Civil and Infrastructure Engineering, 2024.

Student Confirmation

Student name	Yixiong Jing	
Contribution to the paper	First-author contribution: • conception of research ideas • design and implementation of models • writing and presentation of the paper	
Signature and Date		Jan. 15th 2025

Supervisor Confirmation

By signing the Statement of Authorship, the supervisor is certifying that the candidate made a substantial contribution to the publication, and that the description above is accurate.

Supervisor name	Prof. Sinan Acikgoz	
Supervisor comments		
Signature and Date		Jan. 15th 2025

Chapter 5

Anomaly detection of cracks in synthetic bridge point cloud

The paper has been accepted for publication at the Automation in Construction, 2024.

Anomaly detection of cracks in synthetic masonry arch bridge point clouds using fast point feature histograms and PatchCore

Yixiong Jing*
University of Oxford
Department of
Engineering Science,
University of Oxford,
Parks Road, Oxford, UK
yixiong.jing@wolfson.ox.ac.uk

Jia-Xing Zhong
University of Oxford
Department of
Computer Science,
University of Oxford,
Parks Road, Oxford, UK
jiaxing.zhong@cs.ox.ac.uk

Brian Sheil
University of Cambridge
Construction Engineering,
University of Cambridge,
Cambridge, UK
bbs24@cam.ac.uk

Sinan Acikgoz
University of Oxford
Department of
Engineering Science,
University of Oxford,
Parks Road, Oxford, UK
sinan.acikgoz@eng.ox.ac.uk

May 26, 2025

Abstract

Management of ageing masonry arch bridges entails periodic site inspections to identify signs of potential structural degradation. Previous research has focused on detecting surface cracks from images. This paper develops an alternative approach where cracks are identified from point clouds via geometric distortions. An image-based anomaly detection method called *PatchCore* is customised for 3D applications for this purpose. First, Fast Point Feature Histograms (FPFH) are used to extract geometric features. Then *PatchCore* is applied on synthetic point clouds with crack labels, generated using 3D finite element modelling (FEM) and graphical modelling. Results show that the proposed method can capture surface and internal cracks in arches. Analyses show that the method is robust against measurement noise, initial damage and masonry surface roughness, and can be applied to other bridge components. Limitations of the method in detecting small changes in curvature and in-plane geometric distortions are highlighted for further improvements.

5.1 Introduction

Masonry arch bridges constitute around 60% of the railway bridge stock in Europe [Orbán 2004]. Most of these bridges are over 100 years old. Due to changes in rail and road vehicles since their construction, these bridges are now subjected to increased load demands [Bozyigit and Acikgoz 2022]. For these reasons, frequent visual inspections are necessary to track their condition [Acikgoz et al. 2018]. Such inspections are considered labour intensive and subjective [Daniel Brackenbury 2022b].

The success of deep learning (DL) in various image processing tasks, has enabled the automated classification, localization, and quantification of structural defects in image data (e.g. [Cha et al. 2017; F.-C. Chen and Jahanshahi 2017; Cha et al. 2018; Attard et al. 2019b; Zhenqing Liu et al. 2019; G. Li et al. 2020b]. DL-based crack detection techniques have been recently applied to masonry structures with promising results [Dais et al. 2021; Hallee et al. 2021; Loverdos and Sarhosis 2022b]. These studies used clean masonry surfaces to develop and assess the algorithms. However, Daniel Brackenbury [2022b] observed that features such as vegetation, graffiti, surface deposits (e.g. due to efflorescence), shadows (e.g. due to lighting), repointing or rendering are often present in masonry arch bridges and complicate the visual detection of defects.

Image-based defect detection algorithms do not make use of geometry information. However, cracking of structural interest is often accompanied by large-scale geometric distortions. In this paper, we explore the possibility of detecting cracks using these distortions. This new approach offers several potential advantages over visual defect detection. For instance, internal or covered cracks may have no visual markers but may be associated with surface geometric distortions. Furthermore, large-scale geometric distortions may be more indicative of structural issues, rather than local defects (e.g. due to surface deposits).

Point clouds acquired by laser scanning or photogrammetry can be used to identify geometric distortions. Laser scanners are widely adopted in the civil engineering industry, where they are used to document 3D geometry for construction management [C. Wang and Cho 2015; Dai and M. Lu 2013] and documentation [Lubowiecka et al. 2009; Shanoer and Abed 2018] purposes. Several studies have

explored the potential use of this data to identify structural defects. Some methods measure geometric deviations between point clouds and pre-defined primitive geometries (such as a plane or a cylinder) to identify the distortions [Pesci et al. 2011; W. Liu et al. 2011; C. Ye et al. 2018; Dong et al. 2023; del Río-Barral et al. 2022]. Distortions are associated with abrupt changes in elevations, normal vectors, and/or curvature. Without relying on a reference geometry, Mohammadi et al. [2019] analyzes the probability distribution function of geometric features to identify damage in smooth concrete surfaces. E. T. Lee et al. [2023] extracts defect-induced geometric features from normal vectors of point clouds. Stałowska et al. [2022] proposes to capture cracks on concrete surfaces by investigating changes in the intensity of the laser beam. However, these traditional methods use case-specific workflows, which are hard to generalise. Identifying distortions from point clouds using change detection algorithms may hold promise in delivering generalised techniques [Acikgoz et al. 2017b; Yiyan Liu et al. 2024]. However, further research is required in this area.

Anomaly detection techniques provide a well-established and generalized framework for detecting cracks. The state-of-the-art in 2D anomaly detection involves learning the distribution of 'normal' (anomaly-free) data using generative adversarial neural networks (GANs) [Schlegl et al. 2019; Akcay et al. 2019; Beggel et al. 2020], autoencoding models [Sakurada and Yairi 2014; Nguyen et al. 2019] or other unsupervised adaption methods [Yi and Yoon 2020]. Anomalies are then indicated as outliers in the test data. However, in industrial anomaly detection problems, data tends to be scarce and heavily imbalanced (e.g. often only a tiny proportion of data is anomalous). Researchers have proposed techniques to deal with data imbalance by customising loss functions, such as focal loss [T. Lin et al. 2017]. Recently, Roth et al. [2022] proposed *PatchCore* which uses a memory bank to store compressed features (extracted hierarchically by a neural network) of normal samples. *PatchCore* achieves the state-of-the-art in a 2D image anomaly detection dataset called MVTec AD [Bergmann et al. 2019]. The *PatchCore* framework has been widely adapted to varied industrial datasets [Hattori et al. 2023; X. Jing 2023] where it has demonstrated good accuracy and robustness.

Research on 3D anomaly detection has received considerably less attention than 2D anomaly detection. Most studies utilize the dataset MVTec 3D-AD, which

features normal and anomalous RGB-D images of industrial objects. The study which introduced the dataset, Bergmann et al. [2021], explored voxel-based auto-encoding and GANs to perform anomaly detection. Since voxelization degraded anomaly detection performance, the authors' next study, Bergmann and Sattlegger [2023], proposed a 'student-teacher' network which reduced the reconstruction errors by preserving local geometric features. The student network [Hinton et al. 2015] was trained using the outputs of the teacher network, and the regression errors between the teacher and student networks were used to identify anomalies. Rudolph et al. [2023] adapted the same student-teacher network while utilizing an invertible neural network [Dinh et al. 2016; Kingma and Dhariwal 2018] to transform normal data into a normal distribution. In an alternative approach, Cao et al. [2023] and Yue Wang et al. [2023] fused features from 2D and 3D to improve the detection accuracy. Horwitz and Hoshen [2023] noticed that classical 3D geometric descriptors, such as Fast Point Feature Histograms (FPFH) [Rusu et al. 2009] perform better than learned features. They adapted the *PatchCore* algorithm [Roth et al. 2022] to RGB-D images and calculated geometric features using FPFH. This approach is not learning based; it uses hand-crafted geometric features which can be calculated directly from the data. Furthermore, *PatchCore* requires only two sets of data (the first set representing normal and the second the test data) to perform anomaly detection. It is ideally suited for applications where data for training algorithms is limited.

There is a scarcity of publicly available datasets that describe the 3D geometry of civil engineering assets. We have overcome this challenge by crafting large-scale synthetic datasets in our previous work. For instance, to inform the development of the BridgeNet semantic segmentation network, we have developed a dataset to simulate multi-span masonry arch bridge geometry [Y. Jing et al. 2022; Y. Jing et al. 2024a]. More recently, we have integrated nonlinear finite element simulations and 3D graphics software (including a laser scan simulator [Reitmann et al. 2021b]) to automate the development of synthetic point clouds of single masonry arch bridges in Y. Jing et al. [2023]. This approach ensures consistency between deformations and surface texture changes in the resulting point clouds. In previous studies, either only the geometry (using ad-hoc cubic surface equations [Mohammadi et al. 2019]) or textures (using modified UV maps [Narazaki et al. 2021])

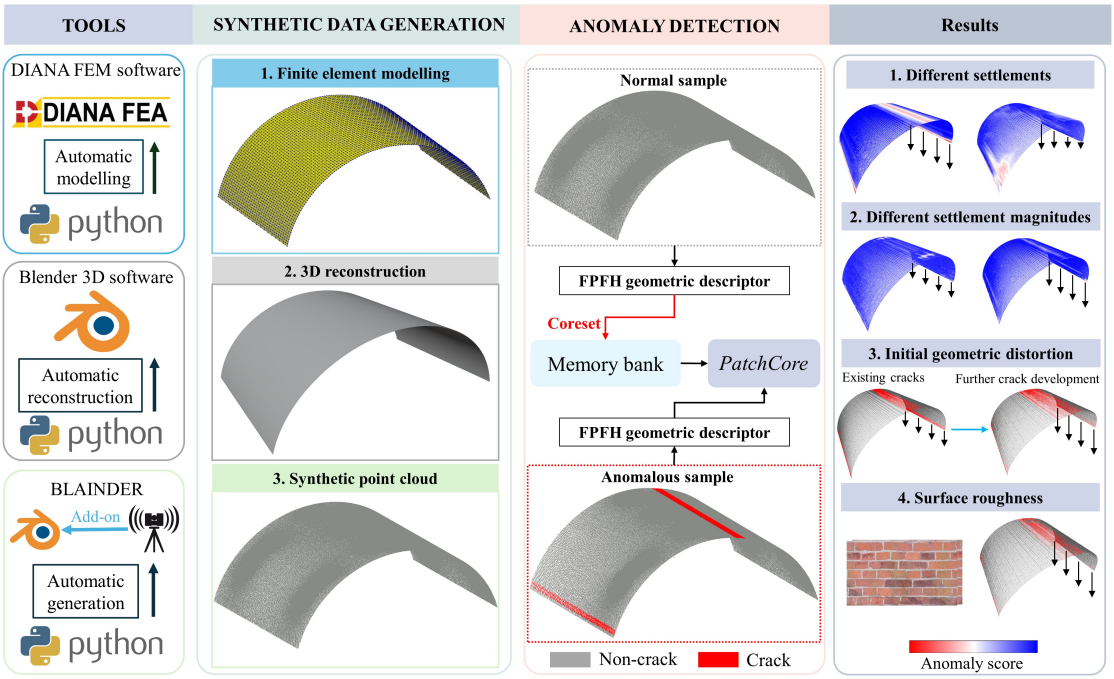
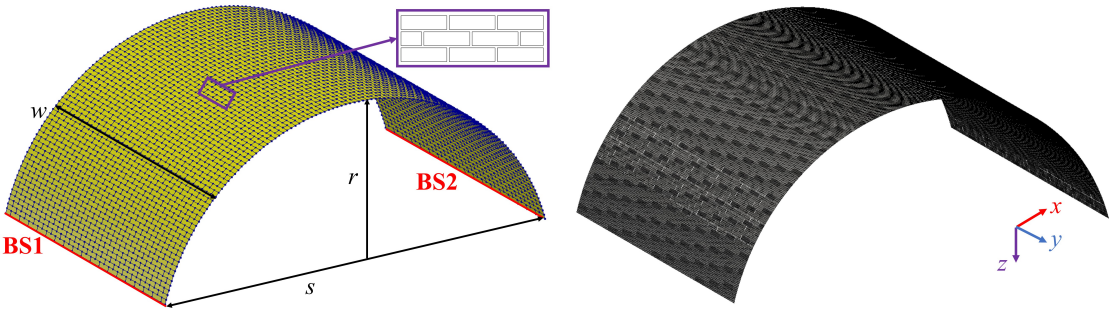


Figure 5.1: Research overview of anomaly detection on synthetic masonry arch point clouds.

were simulated to describe the damage. It should also be noted that the synthetic modelling framework enables the automatic assignment of damage labels to point clouds using finite element model data outputs. Similar labelling is unlikely to be available when working with real point clouds, especially concerning internal cracks.

In the current study, we propose a new anomaly detection framework for the geometric identification of cracks in masonry arch bridge point clouds, which is demonstrated in Figure 5.1. Due to a lack of suitable data to validate the new framework, we synthesize realistic masonry arch clouds in Section 2, using a parametric and automated generation pipeline. In particular, we simulate crack-related distortions due to support movements, which are one of the primary drivers of damage in masonry arch bridges [Orbán 2004]. The undeformed and deformed finite element model geometries are then imported into 3D computer graphics software, where textures are added and synthetic point clouds with automatic damage labels are generated. The subsequent anomaly detection phase is described in Section 3. Different from previous work which is confined to RGB and RGB-D images [Horwitz and Hoshen 2023], we customize the *PatchCore* anomaly detection framework so that it can operate on 3D points clouds. Furthermore, instead of calculating FPFH within a point neighbourhood of fixed radius [Horwitz and Hoshen 2023],



(a) 3D FEM geometry (b) FEM mesh

Figure 5.2: Visualization of (a) the arch geometry and (b) FEM mesh.

we utilize multi-scale (e.g. multiple radii) feature calculations to enhance anomaly detection performance. Our work also deals with new challenges that arise due to non-rigid deformations; such deformations were not considered in the MVTec 3D-AD dataset. In Section 4, we quantify the accuracy of crack detection in a simple arch model for a range of settlement scenarios. Anomaly detection robustness to the presence of initial damage and masonry texture are investigated. We then extend the investigation to a complete arch bridge point cloud, where damage is assessed in other structural components including spandrel walls. All procedures are parameterized and run in batch mode using custom Python scripts as shown in Figure 5.1. Finally, the limitations of *PatchCore* and FPFH are briefly discussed to identify potential areas for improvement.

5.2 Generation of synthetic data

5.2.1 Simulation of the masonry arch model

To generate synthetic data automatically, code was developed which can parametrically generate meso-scale 3D finite element models (FEM) of arches using the commercial software DIANA 10.6. Due to the high computational costs associated with modelling the entire masonry arch bridge, only the arch is modelled at first. For the same reason, the exact geometry of the mortar is not modelled explicitly. Instead, mortar is represented by zero-length 3D interface elements and masonry units are extended to ensure continuity.

The geometry of the arch is shown in Figure 5.2a. It is a segmented cylinder with a 'stretcher bond' brick pattern shown inset. Two types of elements are included in this model (Figure 5.2b): eight-node quadrilateral shell elements (code CQ48S)

Table 5.1: Elastic properties of the masonry units and mortar

$E_{unit}(N/m^2)$	$G_{unit}(N/m^2)$	$E_{mortar}(N/m^2)$	$G_{mortar}(N/m^2)$	ν	$\rho(kg/m^3)$
1×10^{10}	4.2×10^9	1×10^9	4.2×10^8	0.2	2200

Table 5.2: Constitutive parameters describing interface plasticity

$\sigma_t(MPa)$	$c_0(MPa)$	$\phi_0(^\circ)$	$\psi(^\circ)$
0.15	0.35	35	0

and six-node interface elements (code CL24I). A suitably refined mesh was adopted with a size of 0.02m in all dimensions. Boundary conditions were defined on the arch spring lines which are denoted as BS1 and BS2 in Figure 5.2a. All relevant translational and rotational degrees of freedom are restrained.

The in-plane dimensions of the extended units are taken as 200mm×60mm. The thickness of the quadrilateral and interface elements are taken as $t = 0.5$m to represent the arch thickness. The span length s and rise r, which are shown in Figure 5.2, are 5.31m and 2.04m respectively. The bridge is assumed to be one-lane wide and its width w is taken as 3.6m.

For simplicity, units are defined as elastic and damage is confined to the nonlinear interface elements. The units are characterised by Young's modulus E_{unit}, shear modulus G_{unit}, Poisson's ratio ν, and density ρ listed in Table 5.1. These values are considered representative of soft historical solid clay bricks [Lourenço and Gaetani 2022].

The elastic behaviour of the interface elements is controlled by the normal and shear stiffness (k_n and k_s). These can be calculated using the elastic properties of the units and mortar and the joint thickness [Lourenço and Rots 1997]:

$$k_n = \frac{E_{unit}E_{mortar}}{t(E_{unit} - E_{mortar})}; k_s = \frac{G_{unit}G_{mortar}}{t(G_{unit} - G_{mortar})} \tag{5.1}$$

where Young's modulus E_{mortar} and shear modulus G_{mortar} of mortar is shown in Table 5.1. Poisson's ratio is identical for the units and mortar.

To ensure stable nonlinear models which do not suffer convergence issues, conventional plasticity models are used. Contact force in the interface shear direction is modelled using Coulomb friction. Cracking in the interface's normal direction is controlled by a Rankine-type model, with a cut-off tensile strength σ_t. The param-

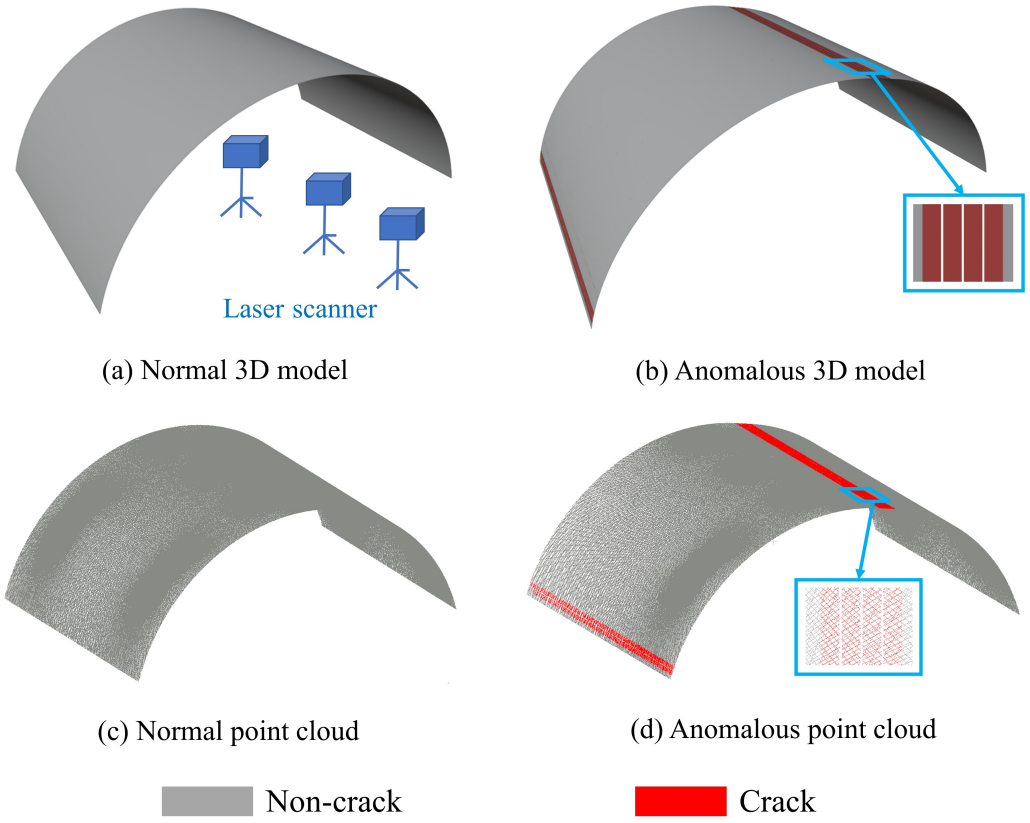


Figure 5.3: Arch reconstructed in Blender for synthetic point cloud generation. (a) and (b) represent the reconstructed 3D model from FEM output before and after support movements. Crack labels are automatically assigned in the anomalous geometry in (b) using FEM output. (c) and (d) are the synthesized point clouds from 3D models.

eters governing the Coulomb friction behaviour include the cohesion c_0, friction angle ϕ_0, and dilatancy angle ψ. The adopted parameters are listed in Table 5.2.

Prescribed displacements are applied on the boundary BS2 (see Figure 5.2). Four support movement patterns are considered: x translation only, z translation only, combined $x + z$ translation and combined x rotation and z translation. The first two cases represent span opening and pier settlement. The third case represents a combination of the first two. These three cases all lead to well-studied 2D response mechanisms [Acikgoz et al. 2017b]. The fourth case represents differential settlements of piers, commonly encountered during foundation scour [Tubaldi et al. 2018]. It is expected to lead to a more complicated 3D response.

The magnitude of translational displacements in the first three cases is limited to $0.2r = 0.4$m which falls within the range previously investigated by Acikgoz et al. [2017b]. For the $x + z$ case, the total displacements are partitioned equally

to $x = 0.283$m and $z = 0.283$m. For the combined x rotation and z translation case, a linearly varying vertical displacement field is defined over the length of BS2 ranging from 0 to 0.1m.

5.2.2 Synthetic labelled point cloud generation

The nodal coordinates from FEM are exported to Blender for the 3D reconstruction of the undeformed ('normal') arch (Figure 5.3a). Displacements of extended units and the interface normal direction relative displacements are then mapped on the deformed ('anomalous') 3D model as shown in Figure 5.3b. According to Burland et al. [1978], open cracks with a width greater than 1mm (positive values indicate tension) can be spotted with the naked eye and may be indicative of structural damage. To represent the presence of damage, the largest interface normal cracks (obtained as the maximum of the three layers where integrations are done) are attributed to the neighbouring units which share the same nodes and are assigned 'crack' labels. All geometric elements are classified with binary labels where crack and non-crack labels are indicated in red and grey correspondingly in Figure 5.3b. It should be emphasised that this classification focuses on normal (Mode-I) cracks and neglects shear (Mode-II and III) cracks. This approach is consistent with the visual classification of damage in Burland et al. [1978].

To obtain synthetic point clouds, the laser scanning simulator BLAINDER [Reitmann et al. 2021b] is used. It simulates important features of laser scan data collection, including measurement noise and distance-varying point density. Such features enable the evaluation of the anomaly detection algorithm under realistic conditions. Since cracks are often localized, dense point clouds are required to accurately capture the geometries of crack regions. The laser scanner is therefore deployed underneath the arch in 3 positions spaced evenly along the direction of the generatrix (two at either end and one in the middle). Random measurement noise is sampled from a zero-mean 3D Gaussian distribution with a standard deviation of 0.005m [Y. Jing et al. 2022]. This represents a high level of noise for modern scanners and presents a rigorous test for anomaly detection algorithms. Exemplar point clouds are shown in Figure 5.3c and d. In case of damage, points from the units surrounding crack regions are labelled red (see Figure 5.3d) as they inherit the 'crack' labels from the Blender geometry (see Figure 5.3b).

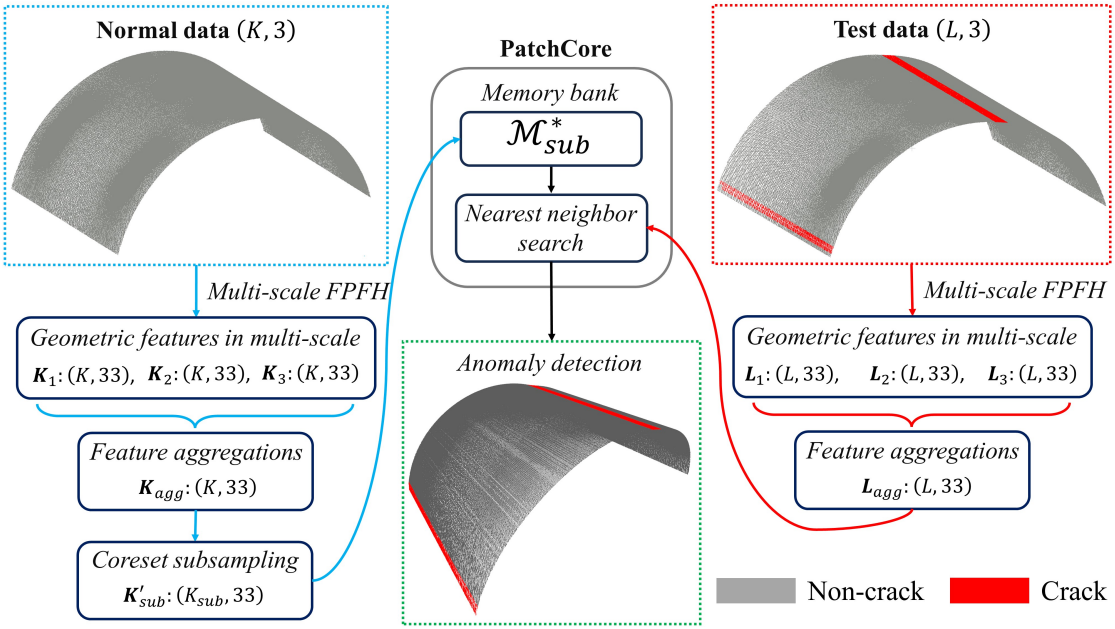


Figure 5.4: Overview of anomaly detection using *PatchCore*. Geometric features of normal and test point cloud data are extracted. Normal features are compressed via greedy coreset subsampling, and stored in a memory bank. Test data features are compared to the memory bank to detect the anomalies, which are highlighted in red with 'Crack' labels.

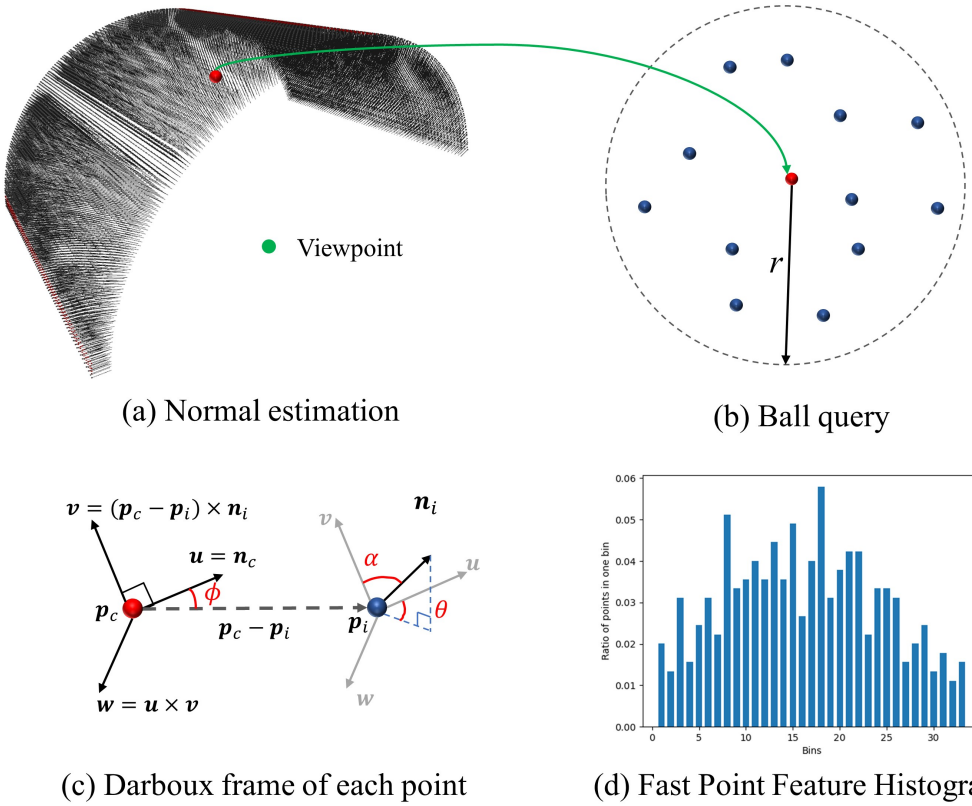
Figure 5.3c shows the 'normal' point cloud data. This typically corresponds to the arch without deformations and damage. The deformed model in Figure 5.3c is considered the test data where the anomaly is associated with the crack regions and is highlighted in red. The grey areas feature elastic deformations but since they do not feature cracking, they are highlighted in grey.

5.3 Methodology for anomaly detection

The anomaly detection framework is shown in Figure 5.4 which includes three key parts: (1) the FPFH computation and feature aggregation; (2) the coreset subsampling and (3) anomaly detection via *PatchCore*.

5.3.1 FPFH computation and feature aggregation

To characterise local geometry, FPFH is adopted. Its computation requires surface normal estimates $\boldsymbol{n}$ at each point $\boldsymbol{p}$ (Figure 5.5a). In this paper, surface normals are obtained using the point cloud data processing package Open3D [Q.-Y. Zhou et al. 2018], which calculates the principal axis of the adjacent points (ball query is used for neighbour searching) using covariance analysis. Features are first com-



(a) Normal estimation

(b) Ball query

(c) Darboux frame of each point

(d) Fast Point Feature Histograms

Figure 5.5: The steps for extracting FPFH descriptors. (a) shows the surface normal vector estimation for each point. (b) and (c) visualize the neighbour search and the construction of the Darboux frame. (d) shows a sample fast point feature histogram that describes the local geometry of a point.

puted within a radius r, where the points $\boldsymbol{p_i}$ neighbouring a centre point $\boldsymbol{p_c}$ are identified using a ball query algorithm as shown in Figure 5.5b. A Darboux frame is then defined using the centre point normal vector $\boldsymbol{n_c}$ and the direction vector $\boldsymbol{p}_c - \boldsymbol{p}_i$. The frame and the neighbour normal vector $\boldsymbol{n}_i$ are used to compute the angles θ, α, and ϕ as shown in Figure 5.5c. After repeating this process for each neighbouring point, a simplified point feature histogram is generated, where the distribution of angles within the neighbourhood is represented with 33 bins as shown in Figure 5.5d. The distribution of angles describes the normal, geodesic and torsional curvature around each point. Simplified point feature histograms from the neighbourhood are then subjected to weighted averaging to obtain an FPFH value for each centre point. More details regarding FPFH can be found in Rusu et al. [2009].

FPFH may be computed using a fixed r. However, since the length and width of cracks can vary significantly, it is advisable to compute FPFH for different r values. Larger values of r are useful to detect large-scale distortions which are

not influenced by local defects. At the same time, smaller values of r are more sensitive to localized geometric changes. To achieve robust features, multi-scale FPFH is used in this paper as follows:

$$\boldsymbol{f}_{agg} = \text{AveragePooling}(\{\boldsymbol{f}_i | i \in \boldsymbol{r}_{FPFH}\}) \tag{5.2}$$

where $\boldsymbol{r}_{FPFH}$ is a list of all radius values where FPFH vectors $\boldsymbol{f}_i$ (with a feature dimension of 33) are calculated. FPFH features from different scales are aggregated through AveragePooling (averaging FPFH in different radii in feature dimension) which generates a single representation $\boldsymbol{f}_{agg}$ [Roth et al. 2022] with the original feature dimension of 33.

5.3.2 Coreset subsampling

The multi-scale FPFH descriptors of normal data are saved in the memory bank $\mathcal{M}$ [Cohen and Hoshen 2020]. However, storing the features of a dense point cloud (millions of points for a masonry bridge) in the memory bank is memory intensive. As the normal data increases, the size of the memory bank increases exponentially. Patchcore compresses these features using coreset subsampling, which enables working with the large point clouds typically encountered in civil engineering applications.

The objective of the coreset subsampling is to find a subset $\mathcal{M}^*_{sub} \subseteq \mathcal{M}$ such that the solution being sought for $\mathcal{M}$ can be closely approximated by $\mathcal{M}^*_{sub}$. In the case of anomaly detection with *PatchCore*, the *minimax facility location* coreset selection [Farahani and Hekmatfar 2009] is adopted to guarantee maximum similarity between $\mathcal{M}^*_{sub}$ and $\mathcal{M}$. The optimization function is given by Equation 5.3 [Roth et al. 2022]:

$$\mathcal{M}^*_{sub} = \underset{\mathcal{M}_{sub} \subset \mathcal{M}}{\arg\min} \max_{\boldsymbol{f}_{agg} \in \mathcal{M}} \min_{\boldsymbol{f}'_{agg} \in \mathcal{M}_{sub}} \|\boldsymbol{f}'_{agg} - \boldsymbol{f}_{agg}\|_2 \tag{5.3}$$

Intuitively, Equation 5.3 can be regarded as finding a set of $\boldsymbol{f}'_{agg} \in \mathcal{M}^*_{sub}$ in which the maximum Euclidean distance of any feature in $\mathcal{M}$ to $\boldsymbol{f}'_{agg}$ is minimized.

However, the exact optimization of $\mathcal{M}^*_{sub}$ is NP-Hard as explained by Wolsey and Nemhauser [1999]. The iterative greedy algorithm suggested by Sener and Savarese

Algorithm 1 *Coreset subsampling*

Require: FPFH features $\boldsymbol{F}_{agg}$, coreset target number c.
Ensure: Optimal memory bank $\mathcal{M}^*_{sub}$

$\mathcal{M}^*_{sub} \leftarrow [\,]$
$\boldsymbol{f}^{'}_{agg} = Random_Sampling(\boldsymbol{F}_{agg})$ ▷ Sample a row of feature
$min_dists = Dists(\boldsymbol{F}_{agg}, \boldsymbol{f}^{'}_{agg})$ ▷ Initialize pair-wised distances
for $i \in [0, c-1]$ **do**
    $dists = Dists(\boldsymbol{F}_{agg}, \boldsymbol{f}^{'}_{agg})$
    $min_dists = eleMin(dists, min_dists)$ ▷ Element-wise mininum
    $_, max_idx = Argmax(min_dists)$
    $\boldsymbol{f}^{'}_{agg} = \boldsymbol{F}_{agg}[max_idx]$ ▷ Feature with maximum distance
    Set the distance of $\boldsymbol{f}^{'}_{agg}$ in min_dists to 0
    Add $\boldsymbol{f}^{'}_{agg}$ to $\mathcal{M}^*_{sub}$
return $\mathcal{M}^*_{sub}$ ▷ Memory bank

[2017] is adopted (see Algorithm 1) to compress $\boldsymbol{F}_{agg}$. Here, min_dists records the minimum distances between features in $\mathcal{M}^*_{sub}$ and the other features in $\boldsymbol{F}_{agg}$. In every iteration, min_dists is updated by an element-wise comparison between $dists$ (Euclidean distance between $\boldsymbol{F}_{agg}$ and $\boldsymbol{f}^{'}_{agg}$) and min_dists. $\boldsymbol{f}^{'}_{agg}$ is then selected as the feature with the maximum Euclidean distance in min_dists. The coreset target number c indicates the size of the memory bank. c is often set to 1-2% of the original normal data size, which refers to the number of points in the point cloud.

5.3.3 Anomaly detection

Extracted and aggregated features of normal and test data are denoted by $\boldsymbol{K}_{agg}$ and $\boldsymbol{L}_{agg}$ respectively as shown in Figure 5.4. Algorithm 1 is then used to compress the aggregated FPFH features of normal data $\boldsymbol{K}_{agg}$ into $\boldsymbol{K}^{'}_{sub}$, where $\boldsymbol{K}^{'}_{sub}$ refers to the memory bank $\mathcal{M}^*_{sub}$ returned by the algorithm. The maximum distance s^*, which defines the largest anomaly, can be computed using Equation 5.4:

$$\boldsymbol{l}^*_{agg},\ \boldsymbol{k}^{'*}_{sub} = \underset{\boldsymbol{l}_{agg} \subset \boldsymbol{L}_{agg}}{\arg\max}\ \underset{\boldsymbol{k}^{'}_{sub} \subset \mathcal{M}^*_{sub}}{\arg\min}\ \|\boldsymbol{l}_{agg} - \boldsymbol{k}^{'}_{sub}\|_2$$
$$s^* = \|\boldsymbol{l}^*_{agg}, -\boldsymbol{k}^{'*}_{sub}\|_2 \tag{5.4}$$

where $\boldsymbol{l}^*_{agg}$ and $\boldsymbol{k}^{'*}_{sub}$ are the candidate results for optimizing Equation 5.4 on $\mathcal{M}^*_{sub}$ and $\boldsymbol{L}_{agg}$ respectively. Lines 1, 2 and 3 of Algorithm 2 show the computational

Algorithm 2 *PatchCore* anomaly detection

Require: Memory bank $\mathcal{M}^*_{sub}$, FPFH features of test data $\boldsymbol{L}_{agg}$, number of neighbors for re-weighting k_{neigh}.

Ensure: Anomaly score s, minimum matching distances min_val

1: $min_val, min_idx = NearestSearch(\mathcal{M}^*_{sub}, \boldsymbol{L}_{agg})$ ▷ Best matching

2: $s^*, s^*_idx = Argmax(min_val)$

3: $\boldsymbol{k}^{'*}_{sub}$, $\boldsymbol{l}^*_{agg} = \mathcal{M}^*_{sub}[s^*_idx]$, $\boldsymbol{L}_{agg}[min_idx[s^*_idx]]$

4: $\boldsymbol{L}^{k,*}_{agg} = k_NearestFeatures(\boldsymbol{l}^*_{agg}, \mathcal{M}^*_{sub}, k)$

5: $s = \left(1 - \frac{exp\|\boldsymbol{l}^*_{agg} - \boldsymbol{k}^{'*}_{sub}\|_2}{\sum_{\boldsymbol{l}^{k,*}_{agg} \in \boldsymbol{L}^{k,*}_{agg}} exp\|\boldsymbol{l}^*_{agg} - \boldsymbol{l}^{k,*}_{agg}\|_2}\right) \cdot s^*$ ▷ Re-weighting

6: **return** s, min_val

steps required for optimising Equation 5.4. Line 1 of Algorithm 2 determines a correspondence between FPFH features in $\boldsymbol{L}_{agg}$ and $\mathcal{M}^*_{sub}$ by minimizing the Euclidean distances, where the minimum distance min_val and matched index min_idx (e.g. the location of FPFH feature in $\mathcal{M}^*_{sub}$ paired with $\boldsymbol{L}_{agg}$) is achieved. s^* and s^*_idx which are shown in line 2 of Algorithm 2 represent the maximum value of min_val and its corresponding index. $\boldsymbol{l}^*_{agg}$ and $\boldsymbol{k}^{'*}_{sub}$ are extracted from $\mathcal{M}^*_{sub}$ and $\boldsymbol{L}_{agg}$ with the index s^* and s^*_idx respectively, which is shown in line 3 of Algorithm 2.

Roth et al. [2022] suggest that re-weighting s^* in the k_{neigh} neighborhood of $\boldsymbol{l}^*_{agg}$ in $\mathcal{M}^*_{sub}$ can provide a more robust indicator for anomalies. Therefore, the final anomaly score $s \in \mathbb{R}$ is obtained from lines 4 and 5 in Algorithm 2. Binary anomaly classification can be conducted with s defined as a threshold on the returned distances min_val, where the point with a matched distance larger than s is defined as an anomaly.

5.4 Results

5.4.1 Implementation details and accuracy metrics

To speed up the calculation of FPFH features, point clouds are downsampled using a voxel of size 0.02m. The ball query for the nearest neighbourhood calculation uses a radius of 0.12m to estimate surface normal vectors. Three radii are selected to retrieve multi-scale FPFH features for feature aggregation: 0.4m, 0.8m, and 1.2m. The smallest radius corresponds to the length of two bricks, whilst the largest radii corresponds to six bricks. The size of $\mathcal{M}^*_{sub}$ is set to 2000 which

occupies approximately 2% of the normal point cloud size. The experiments were conducted on AMD Ryzen 9 5900 CPU with 128GB memory. The algorithm does not require GPU to accelerate the computation.

Precision, *Recall*, and $F1$-*score* are used as metrics to evaluate the performance of the anomaly detection approach:

$$\begin{aligned} Precision &= \frac{TP}{TP+FP} \\ Recall &= \frac{TP}{TP+FN} \\ F1\text{-}score &= 2 \times \frac{Precision \times Recall}{Precision + Recall} \end{aligned} \tag{5.5}$$

where TP, FP, and FN are the true positive, false positive, and false negative predictions respectively. FP and FN represent normal and abnormal points which are wrongly classified by the algorithm. *Precision* only considers FP while *Recall* only considers FN. For this reason, these measures cannot be used for a comprehensive evaluation of anomaly detection performance. For instance, a poorly performing crack detection algorithm may have a high *Precision* if it does not misclassify normal points. To provide a balanced measurement, $F1$-*score* is adopted which symmetrically represents both *Precision* and *Recall* in one metric by using the harmonic mean of *Precision* and *Recall*.

5.4.2 Anomaly detection on synthetic masonry arch data

Different support movement patterns

The masonry arch FEM model is subjected to BS2 support translation in x, z and $x+z$ and combined x rotation and z translation as shown by black arrows in Figure 5.6a-d. The support movement patterns, the final displacements and the resulting (diffuse) hinge locations are summarized in Table 5.7. The maximum open crack widths are visualized in Figure 5.6a-d with contours which indicate the size of cracks at each interface in the final state. Anomaly detection is performed on point clouds of the arch in Figure 5.7a-d, after synthesizing point clouds from the FEM results. The point cloud crack labels in Figure 5.7a-d correspond to the FEM results in Figure 5.6a-d. The correlated response mechanisms are given:

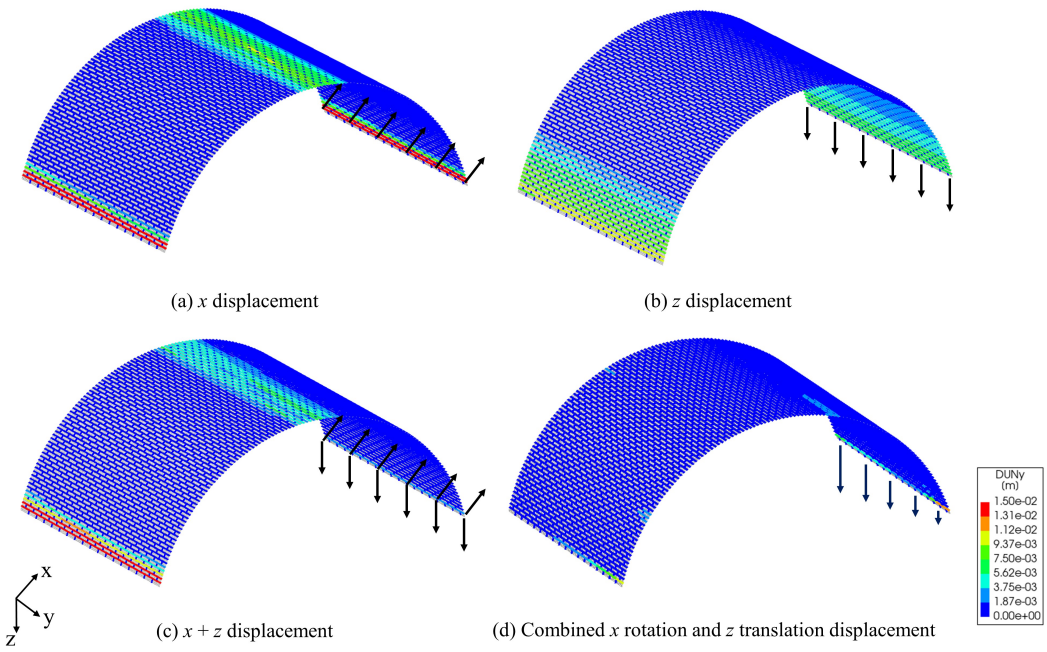


Figure 5.6: (a)-(d) visualize masonry arch FEM analysis results for four support movement cases. Colors denote interface normal direction relative displacements $DUNy$ which describe crack widths in the range of 0 to 15mm. Cracks over 15mm are coloured with red.

1. x displacement: three diffuse plastic hinges are generated, an extrados hinge on the crown and two intrados hinges on the springings;

2. z displacement: two diffuse plastic hinges are generated, an intrados hinge on the stationary springing and an extrados hinge on the settling springing;

3. $x + z$ displacement: two diffuse plastic hinges are generated: an extrados hinge on the crown and an intrados hinge on the stationary springing;

4. combined x rotation and z translation: the largest cracks are localized on the stationary springing side, forming a 3D mechanism. Cracks are less diffuse in this case.

The results are visualized using heatmaps in Figure 5.7e-h where the anomaly scale is normalized within the range [0, 1]. The blue-white-red colours indicate increasing geometric anomaly. For the cracks introduced by translational displacements in Figures 5.7e-g, the heatmaps demonstrate that *PatchCore* can accurately localize both extrados and intrados cracks. Cracks, which are introduced by x, z, and $x + z$ displacements, feature out-of-plane displacements due to mechanism formation, which generates curvature changes which are spotted by FPFH. In the combined x rotation and z translation case, *PatchCore* fails to capture some of the

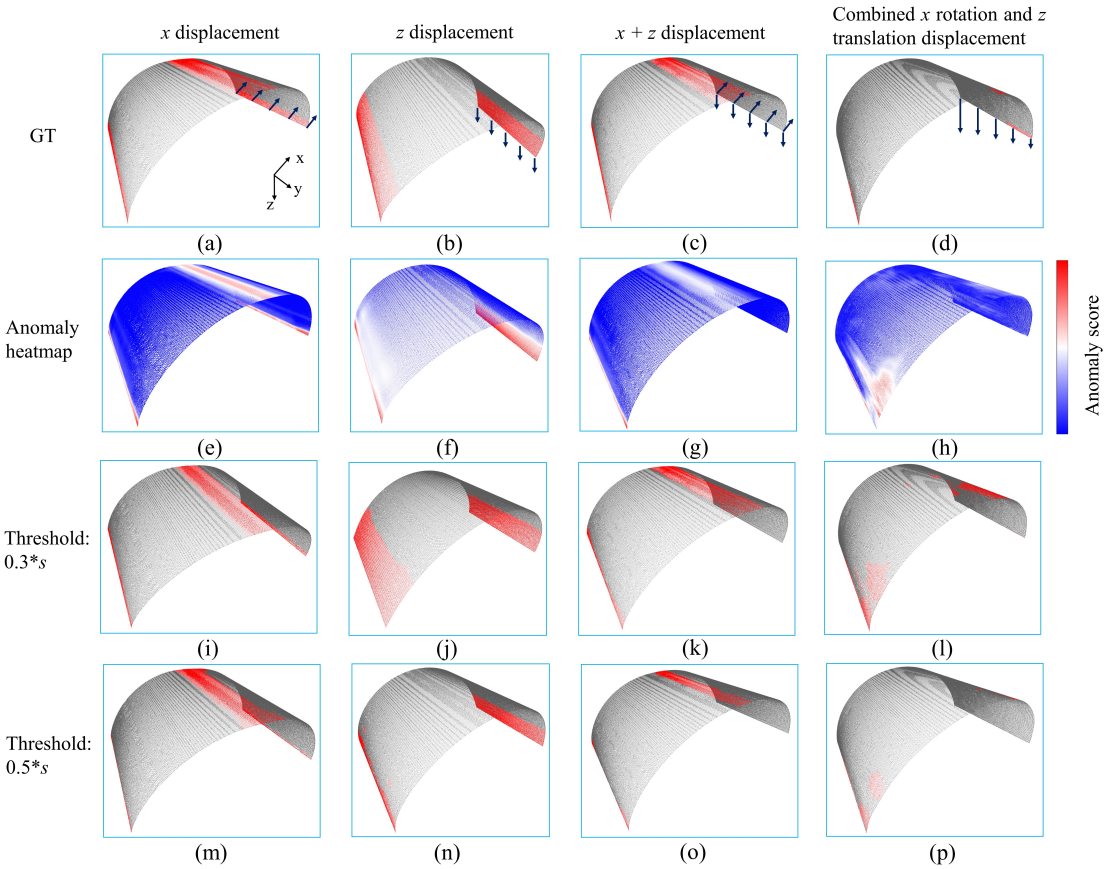


Figure 5.7: Anomaly detection results on the masonry arch for 4 different support movement cases. (a)-(d) are the ground truth; (e)-(h) are heatmaps which represent the anomaly score of each point; (i)-(l) and (m)-(p) are the binary classifications results by setting different anomaly thresholds.

cracks. The crack on the BS2 support does not involve out of plane displacements of arch segments on either side of the crack. Instead, it predominantly features in-plane stretching which leads to minor changes in surface curvature. In addition, the localized crack near the quarter span location on the side of BS1 is also not captured. This is attributed to the small length and width of this crack, which makes detection more difficult.

Anomalies are identified by setting thresholds on binary classifications, affecting detection accuracy. As shown in Figure 5.7i-l and Figure 5.7m-p, regions of crack spread more when the threshold is set at $0.3*s$ rather than $0.5*s$. *Precision* is generally better for $0.3*s$ than for $0.5*s$ while the opposite relationship can be observed on *Recall* (see Table 5.3 versus Table 5.4). The increased performance is attributed to an increase in TPs. At the same time, *Recall* is compromised with the threshold decrease due to wrong anomalous classifications.

The results in Figure 5.8 show that a unique optimal s cannot be determined

Table 5.3: *Precision*, *Recall*, and *F1-score* for the 4 different support movement cases of the masonry arch with a threshold of $0.5*s$

	x	z	$x + z$	$x\,rot + z\,trans$
Precision	0.713	0.589	0.145	0.308
Recall	0.987	0.999	0.999	0.166
F1-score	0.828	0.741	0.254	0.216

Table 5.4: *Precision*, *Recall*, and *F1-score* for the 4 different support movement cases of the masonry arch with a threshold of $0.3*s$

	x	z	$x + z$	$x\,rot + z\,trans$
Precision	0.817	0.975	0.830	0.598
Recall	0.809	0.953	0.963	0.088
F1-score	0.813	0.964	0.891	0.152

since different values of s generate the best *F1-score* for the four different support movement cases. This discrepancy is not surprising; the maximum crack width and the geometric distortion for each support movement case are different. Since s reflects the largest geometric deviation in each examined dataset, an optimal threshold might be expected to depend on the magnitude of the crack opening. Similarly, the datasets suggest that the cracking pattern (e.g. whether the crack involves shearing or not) and size (e.g. longer cracks are easier to detect due to the large FPFH radii) may also influence the anomaly score. Finally, the dependence of the anomaly score on the crack labelling also needs to be noted. Bricks are labelled as damaged if the detachment of the neighbour mortar joints is greater than 1mm as described in Section 2.2. However, labelling can be changed by setting 5mm (slight damage) or 10mm (moderate damage) as the new thresholds, which can considerably decrease the number of 'damaged' bricks. Therefore, we argue that a suitable anomaly score s should be chosen by the inspectors on a case by case basis. For these reasons, case-specific thresholds are used in this study, instead of trying to identify an optimal threshold.

Varying support movement magnitudes

Figure 5.7a explored 0.4m x support movement. Figure 5.9 investigates smaller support movements in the range of 4-16cm to understand the size of geometric distortions required for detection. The ground truth results are shown in the top row. As shown in Figure 5.9e and i, *PatchCore* does not detect springing hinges

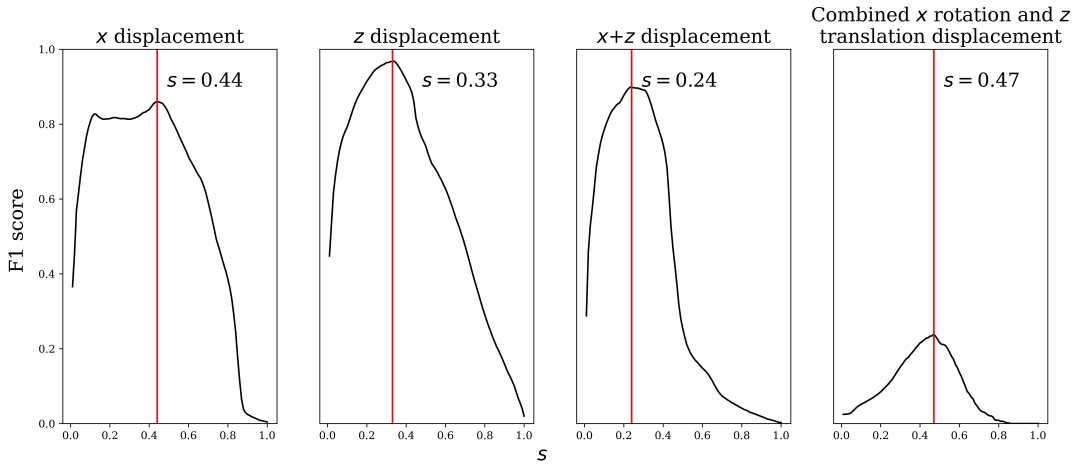


Figure 5.8: $F1$-*score* versus detection threshold where the highest $F1$-*score* are highlighted by vertical red lines. (a)-(d) represents 4 different support movement cases.

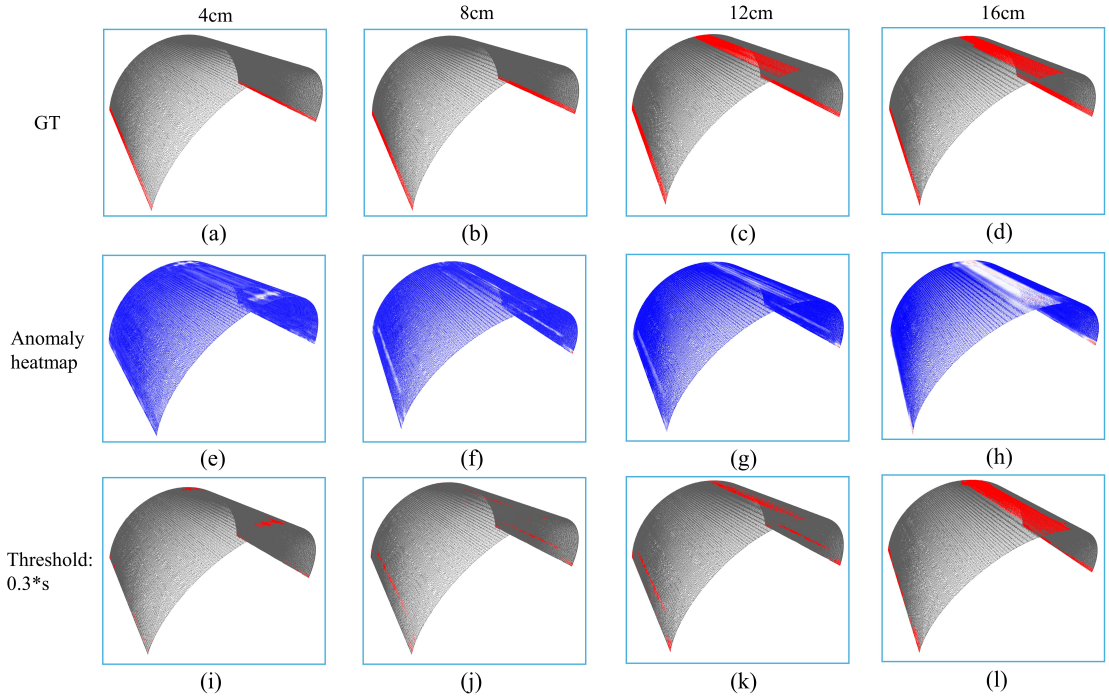
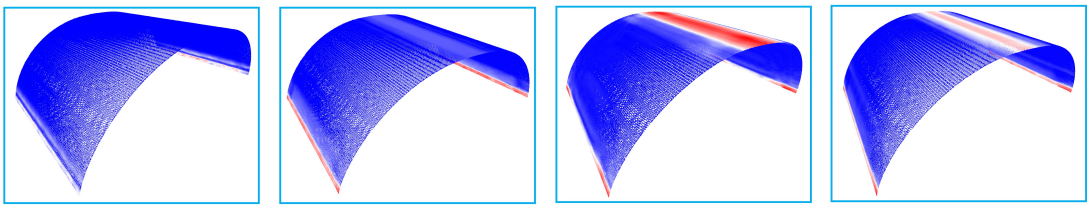


Figure 5.9: Anomaly detection results of the masonry arch subjected to increasing x support movement magnitudes. (a)-(d) are the ground truth; (e)-(h) are heatmaps which represent the anomaly score of each point; (i)-(l) are the binary classifications results for anomaly threshold 0.3^*s.

for the 4cm displacement case which produces a maximum crack width of 1.7mm. However, *PatchCore* detection improves with growing displacements, as shown in Figure 5.9f-h and Figure 5.9j-l. The dissimilarities between $\boldsymbol{l}^*_{agg}$ and $\mathcal{M}^*_{sub}$ can not be captured by the noisy FPFH features for small support movements. In this case, cracking induced distortions are not sufficiently different from elastic displacements and detection is not possible.



(a) 0.4m (b) 0.8m (c) 1.2m (d) 0.4m+0.8m+1.2m

Figure 5.10: Anomaly scores for the masonry arch subjected to x support movements. (a), (b), (c), and (d) show the scores using heatmaps for different FPFH radii.

Varying FPFH radius

The influence of FPFH radius on anomaly detection is explored next with the 0.4m x support movement case. As shown in Figure 5.6a, intrados hinges on springing lead to more localized cracking compared to the diffuse hinge on the crown, where the cracks are spread across several bed joint layers. The curvature variations are therefore more concentrated for springing hinges which leads to larger crack widths at this location (20 mm maximum crack at springing vs 10 mm at crown). To capture the sharp curvature change at the springing, a smaller FPFH radius is beneficial. In this case, the radius should be large enough to deal with measurement noise but small enough to capture the change induced by the localized crack.

Cracks which are captured by varying FPFH radius are visualized in Figure 5.10a-c. Figure 5.10a and b capture the springing hinges with the smaller radii of 0.4m and 0.8m; these ensure significant dissimilarities between $\boldsymbol{l}^*_{agg}$ and $\mathcal{M}^*_{sub}$. However, the crown hinge can only be accurately located by the larger FPFH radius of 1.2m. In this case, due to the diffuse nature of the crack, curvature variations happen over larger distances, which require larger FPFH radii for detection. The inspection of cracks can be improved by aggregating FPFH from different radii as shown in Figure 5.10d, where cracks at all hinges are captured. The performance improvement is quantified in Table 5.5, where the $F1$-*score* of the aggregated radii is the highest by a notable margin.

Initial geometric distortion

Whilst arch point clouds with perfect cylindrical geometry were considered 'normal' in the previous section, arches in most masonry bridges feature geometric distortions. These may appear during construction (e.g. after striking the timber

Table 5.5: *Precision*, *Recall*, and *F1-score* for the x support movement case of the masonry arch with a threshold of 0.3^*s and varying FPFH radii

	$0.4m$	$0.8m$	$1.2m$	$0.4m+0.8m+1.2m$
Precision	0.220	0.243	0.816	0.741
Recall	0.957	0.998	0.763	0.983
F1-score	0.358	0.392	0.789	0.848

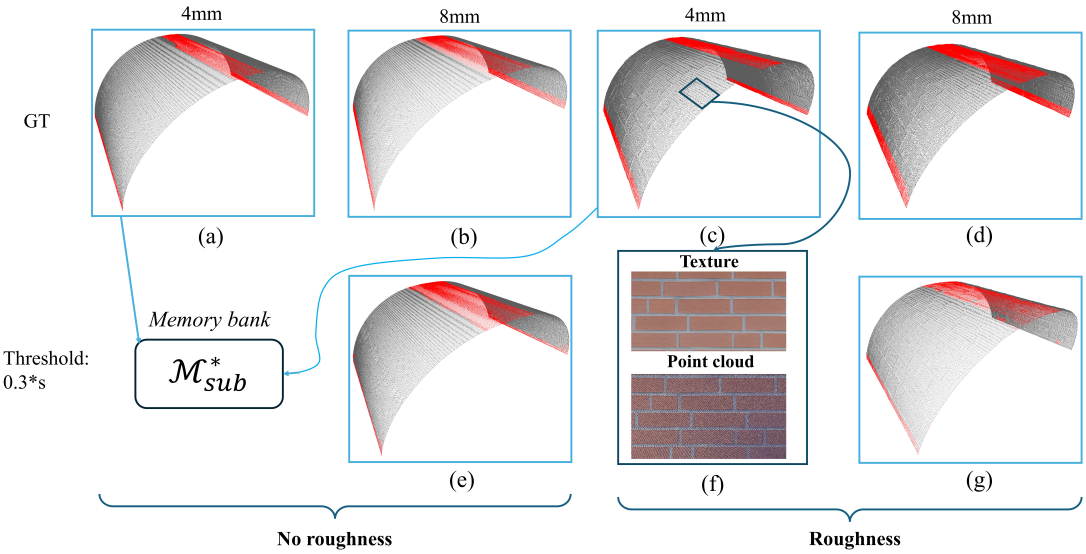


Figure 5.11: Anomaly detection for the masonry arch with initial distortions and surface roughness subjected to x support movements. The arch with the maximum crack width of 4mm is assumed to be normal data to predict anomalies in 8mm crack width case. (a)-(d) are the ground truth; (c) is the binary prediction for the point cloud without surface roughness; (f) visualizes the 3D texture and the generated surface roughness in the point cloud; (g) is the binary prediction for the point cloud with surface roughness.

scaffolding) [Brencich and Morbiducci 2007] or service life (e.g. due to live loads or ground movements) [Acikgoz et al. 2018; Acikgoz et al. 2017a]. In this section, 'normal' data is associated with an arch with modest initial geometric distortions and cracking. This investigation evaluates the feasibility of applying *PatchCore* for periodic inspection of structures with existing geometric distortions.

As shown in Figure 5.11a and b, two point clouds are extracted from FEM with increasing x displacements of 8cm and 15cm, where the maximum crack width is 4mm and 8mm correspondingly. The 4mm case is treated as the 'normal' data for the memory bank $\mathcal{M}^*_{sub}$. It is then used to perform anomaly detection on the 8mm case. The results are visualized using binary classifications with threshold 0.3^*s as shown in Figure 5.11e. The cracks are detected accurately, as demonstrated by the *F1-score* in Table 5.6 (represented in the table under the 'no roughness' case).

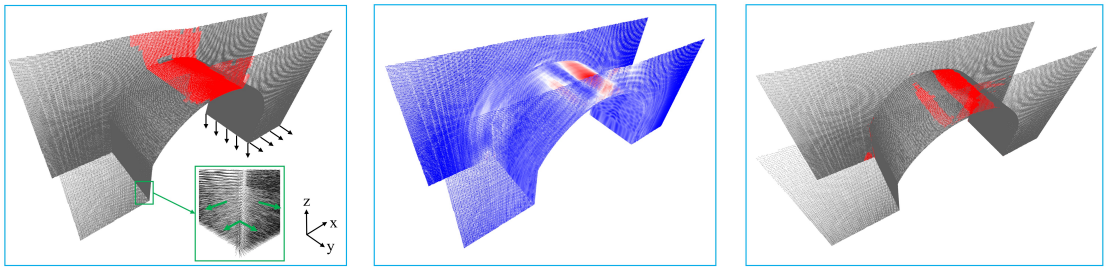
Table 5.6: *Precision*, *Recall*, and $F1$-*score* for the x support movement case of the masonry arch with initial geometric distortions for a threshold of $0.3^{*}s$. Performance parameters are compared for point clouds with and without masonry surface roughness.

	No roughness	Roughness
Precision	0.755	0.673
Recall	0.875	0.877
$F1$-*score*	0.811	0.761

Masonry surface roughness

Masonry structures feature rough surface patterns which arise due to uneven mortar and unit surfaces and their arrangement on the masonry bond. The surface roughness can influence the performance of anomaly detection techniques. To understand its influence, a masonry texture (obtained from ambientCG Docs, accessed 2024 [n.d.]) is UV mapped onto the 3D arch model in Blender (using displacement texture mapping). Each unit is first subdivided into 10×3 grids where each grid represents a mesh in the FEM model with a size of 0.02m. The mesh is further subdivided into 5mm×5mm sub-grids by using the subdivision modifier in Blender, where the displacement texture is applied in the surface normal direction. The mapping involves a rough match between the interface and mortar locations; a tight correspondence was not considered necessary since the mortar joints are not explicitly modelled in FEM and actual cracks may appear both in the units and mortar. The chosen texture involves units protruding by 12mm (on average) from pared back mortar surfaces. The texture and generated point clouds are shown in Figure 5.11f.

The point cloud with surface roughness is included in Figure 5.11c and d, where the minimum crack width of 4mm is again treated as normal data. The prediction on the 8mm case is visualized in Figure 5.11g. Crack detection performance is not significantly impacted by surface roughness. $F1$-*score* in Table 5.6 with roughness indicates only a minor reduction in crack detection performance. This is attributed to the noisier estimation of normal vectors, which impacts the FPFH features and *PatchCore* anomaly detection.



(a) GT (b) Anomaly heatmap (c) Threshold 0.5*s

Figure 5.12: Visualizations of prediction results on the masonry bridge with the dominant displacements in z and x directions. The ground truth, heatmap and binary classification are given by (a)-(c) correspondingly.

5.4.3 Anomaly detection on synthetic masonry arch bridge data

To demonstrate the application of *PatchCore* on a complete bridge model, a synthetic point cloud from the masonry bridge FEM dataset created by Y. Jing et al. [2023] is tested in this section. The bridge is dominated by prescribed displacements in the x (0.233m) and z (0.402m) directions, indicated by black arrows in Figure 5.12a. A macro-scale finite element model with a rotating crack model is used in Y. Jing et al. [2023] instead of the meso-scale model with elastoplastic interfaces introduced in Section 2. Further details regarding the model are available in Y. Jing et al. [2023] which are not repeated here.

The resultant crack labels in the arch crown and spandrel walls (both sides) are shown in Figure 5.12a. Due to the implementation of the macro-scale model, cracks have a more diffuse nature. For this reason, a larger FPFH radius of 3m is used for anomaly detection in the single scale. The anomaly heatmap, in Figure 5.12b, indicates that the algorithm captures some of the cracks around the crown of the arch and on the front spandrel wall. However, in-plane shear cracks in the flat crown area are not captured; these do not introduce geometric curvatures that can be determined by FPFH. The use of color, which was neglected in this study, may enable the detection of in-plane cracks.

Crack detection also fails on the back spandrel wall. The reason for this can be attributed to two aspects. First, the crack width and curvatures are relatively small on the back spandrel wall (compared to the front spandrel wall) due to passive soil resistance at this location. Second, *PatchCore* is highly reliant on the accurate estimation of normal vectors for FPFH computations. However, normal vector es-

timations are particularly noisy at the intersections of different surfaces, see Figure 5.12 inset. The red arrows, which indicate the normal vectors, are not consistent at the intersection of structural components. Poor normal vector estimations can also explain the reason for high anomaly scores at the intersection of the spandrel wall and the arch (see the white rings in Figure 5.12). This observation indicates that it may be beneficial to segment bridge components before carrying out anomaly detection. When the threshold of 0.5^*s is applied the *Precision*, *Recall*, and *F1-scores* of 0.461, 0.758, and 0.574 are obtained.

Table 5.7: File names and support movement magnitudes for the 4 support settlement cases in the provided dataset. 'cw4' and 'cw8' correspond to the maximum crack width of 4mm and 8mm respectively in Section 4.2.4, which are represented by '8cm' and '16cm' in the dataset.

	x				
File name	4cm	8cm (cw4)	12cm	16cm (cw8)	40cm
Support movement magnitude	0.04m	0.08m	0.12m	0.16m	0.4m
	z	$x + z$		$x\,rot + z\,trans$	
File name	disp_z	disp_xz		rot_x	
Support movement magnitude	0.4m	0.283m+0.283m		0+0.1m	

5.5 Conclusions

This paper developed a geometry-based crack detection framework for masonry arch bridges. It stems from the observation that cracks of structural interest are accompanied by significant geometric distortions, which are neglected in image-based crack detection techniques. A 3D anomaly detection algorithm called *PatchCore* is adapted to identify cracks in 3D point clouds. The method involves the creation of a representative memory bank of geometry features (in this case FPFHs) from normal data. Later, the memory bank is used to identify features (e.g. points with anomalous curvatures) when it encounters cracked masonry arch bridge point clouds. The method only requires limited datasets (e.g. point cloud of the structure before and after deformations) to identify crack locations.

Due to the lack of suitable data to validate the framework, a synthetic point cloud

generation pipeline is also developed. It first simulates the mechanical response of structures by using nonlinear FEM predictions. The FEM output is reconstructed by 3D graphical modelling tools to create physically consistent geometric distortions and crack patterns. It also enables realistic modelling of important aspects, such as surface roughness and laser scan measurement geometries and errors. The pipeline was used to parametrically generate the data required for algorithm development in an automated manner and may help resolve the scarcity of 3D point cloud data in future data driven algorithm developments.

The anomaly detection framework is evaluated for masonry arches and arch bridges subjected to support movements. Most of the investigations are conducted on synthetic data derived from a meso-scale FEM model of a masonry arch. To demonstrate the potential use of the technique in the field, *PatchCore* is also applied to a synthetic complete masonry arch bridge model. Results showed that *PatchCore* can detect cracks in the presence of significant measurement noise and masonry surface roughness. It can detect internal cracks (e.g. which occur as a result of intrados hinge formation) as well as surface cracks. The presence of initial geometric distortions does not degrade crack detection performance significantly for the examined cases.

Code and the related point cloud dataset will be available at https://github.com/Jingyixiong/patchcore_masonrybridge, which is to encourage the reproduction of our work. Details of the dataset are summarized in Table 5.7 with file names and support movement magnitudes for the 4 support settlement cases.

Limitations and future work. Our synthetic simulation framework does not model the new surfaces generated by cracking. Modelling these surfaces will result in edge effects [Stałowska et al. 2022] which may need to be integrated into analyses. However, the consideration of new crack surfaces in the synthetic simulator may enable improved anomaly detection as points obtained from these surfaces have distinctive geometric and texture characteristics. This aspect will be addressed in future developments.

Important limitations arise due to the FPFH feature descriptor used in this study. This descriptor is not sensitive enough to capture small geometric distortions and cracks. In its current state, anomaly detection can be used to identify the large

existing distortions in bridges but its accuracy is unlikely to be sufficient for monitoring. There may be benefits in combining laser intensity, color and geometry based detection in future studies. This may, for instance, enable the detection of in-plane failures, which do not cause changes in curvature but may lead to marked visual changes in the structure. Adopting learning-based features instead of handcrafted ones may also enable increased robustness against noise and lead to improved performance. However, this requires overcoming inherent issues in anomaly detection problems related to the heavy data imbalance.

Currently, *PatchCore* requires the definition of a threshold for anomaly detection, which has a significant influence on crack detection performance. The magnitude, type and extent of cracking, FPFH radii and the size of units may influence the optimal threshold. These aspects may be addressed in the future by introducing DL-based anomaly detection which does not require parametric considerations for anomaly detection.

PatchCore has not yet been tested on real masonry arch bridge point clouds. The proposed method is only demonstrated on synthetic masonry arch and bridge point clouds. Therefore, further experiments should be performed on point clouds collected from deforming bridge structures in the future to demonstrate its potential practical use.

Statement of Authorship in Chapter 5

Paper title	Anomaly detection of cracks in synthetic masonry arch bridge point clouds using fast point feature histograms and PatchCore
Authors	Yixiong Jing, Jia-Xing Zhong, Brian Sheil, Sinan Acikgoz
Publication status	Published
Publication details	Automation in Construction, 2024.

Student Confirmation

Student name	Yixiong Jing	
Contribution to the paper	First-author contribution: • conception of research ideas • design and implementation of models • writing and presentation of the paper	
Signature and Date		Jan. 15th 2025

Supervisor Confirmation

By signing the Statement of Authorship, the supervisor is certifying that the candidate made a substantial contribution to the publication, and that the description above is accurate.

Supervisor name	Prof. Sinan Acikgoz	
Supervisor comments		
Signature and Date		Jan. 15th 2025

Chapter 6

A 3D multimodal feature for infrastructure anomaly detection

The paper has been subjected to major revision on Automation in Construction, 2025.

A 3D Multimodal Feature for Infrastructure Anomaly Detection

Yixiong Jing*
University of Oxford
Department of
Engineering Science,
University of Oxford,
Parks Road, Oxford, UK
yixiong.jing@wolfson.ox.ac.uk

Wei Lin
Tongji University
Department of
Geotechnical Engineering,
Tongji University,
Shanghai, China
linwei0763@tongji.edu.cn

Brian Sheil
University of Cambridge
Construction Engineering,
University of Cambridge,
Cambridge, UK
bbs24@cam.ac.uk

Sinan Acikgoz
University of Oxford
Department of
Engineering Science,
University of Oxford,
Parks Road, Oxford, UK
sinan.acikgoz@eng.ox.ac.uk

May 26, 2025

Abstract

Ageing structures require periodic inspection to identify structural defects. Previous work has used geometric distortions to locate cracks in synthetic masonry bridge point clouds but has struggled to detect small cracks. To address this limitation, this study proposes a novel 3D multimodal feature, 3DMulti-FPFHI, that combines a customized Fast Point Feature Histogram (FPFH) with an intensity feature. This feature is integrated into the *PatchCore* anomaly detection algorithm and evaluated through statistical and parametric analysis of synthetic point clouds. The method is further evaluated using point clouds of a real masonry arch bridge and a full-scale experimental model of a concrete tunnel. Results show that the 3D intensity feature enhances inspection quality by improving the detection of cracks; it also enables the identification of water ingress which introduces intensity anomalies. The 3DMulti-FPFHI outperforms FPFH and a state-of-the-art multimodal anomaly detection method. The potential of the method to address diverse infrastructure anomaly detection scenarios is highlighted by the minimal requirements for data compared to learning-based methods.

6.1 Introduction

Critical civil infrastructure requires regular inspection to ensure safety and structural integrity. Over time, defects such as cracks and water patches may develop, which need to be detected and repaired actively to prevent further deterioration [Acikgoz et al. 2018; Liao et al. 2022a]. Traditional inspection methods, which commonly rely on visual assessments, are labour-intensive, time-consuming, and often subjective, leading to variability in defect identification and classification [Daniel Brackenbury 2022b]. With the increasing complexity and scale of modern infrastructure, there is a growing need for automated, data-driven inspection techniques that provide consistent and reliable assessments to support maintenance decisions.

Advances in deep learning (DL) provide opportunities to overcome these challenges with automated structural health monitoring of infrastructure. Recent work has facilitated automated classification, localization, and quantification of structural defects from image data [Cha et al. 2017; F.-C. Chen and Jahanshahi 2017; Cha et al. 2018; Attard et al. 2019b; Zhenqing Liu et al. 2019; G. Li et al. 2020b; T. Li et al. 2024]. DL-based techniques have shown promise in detecting cracks in buildings [Perez et al. 2019; Y. Jiang et al. 2021], bridges [Dais et al. 2021; Hallee et al. 2021; Loverdos and Sarhosis 2022b], tunnels [Liao et al. 2022a; Protopapadakis et al. 2019], and roads [Z. Fan et al. 2020]. For example, recent studies reported the successful detection of cracks with widths greater than $\leq$ 1 mm [Liao et al. 2022b; Mohammadi et al. 2019]. However, 2D image data acquisition is sensitive to illumination, which can adversely affect defect detection accuracy [Gupta and Dixit 2022]. Furthermore, since 2D images lack geometric information, anomaly detection focuses only on surface anomalies. Given the strong correspondence between deformations and crack development, geometric information can help identify structurally significant cracks associated with geometric distortions. This approach may also help detect internal defects that, while not visible on the surface, cause noteworthy deformations [Acikgoz et al. 2017b].

Point clouds, acquired through laser scanning or photogrammetry, provide valuable data for detecting geometric distortions related to structural defects. Several studies have used point clouds to measure geometric deviations between structural

components and predefined primitive geometries, such as planes or cylinders, to detect distortions [Pesci et al. 2011; W. Liu et al. 2011; C. Ye et al. 2018; Dong et al. 2023; del Río-Barral et al. 2022; W. Lin et al. 2023; W. Lin et al. 2024d]. These distortions are typically identified by abrupt changes in elevation, normal vectors, and/or curvature. Alternatively, Mohammadi et al. [2019] detected damage on smooth concrete surfaces by identifying anomalies in geometric feature distributions. Other studies, such as E. T. Lee et al. [2023], have extracted defect-induced geometric features from point cloud normal vectors, while Stałowska et al. [2022] captured cracks on concrete surfaces by analysing changes in laser beam intensity. Despite these advances, these methods rely on case-specific workflows, limiting their generalizability.

In recent work, Y. Jing et al. [2024b] adopted the *PatchCore* [Roth et al. 2022] anomaly detection technique to detect cracks in masonry arch point clouds [Y. Jing et al. 2023], achieving high accuracy and robustness. That method differentiates undeformed ('normal') point clouds from those with cracks ('anomalous') by extracting geometric features with Fast Point Feature Histograms (FPFH) [Rusu et al. 2009]. Cracks were detected by measuring discrepancies between paired points in 'normal' and 'anomalous' point clouds, effectively identifying both visible and non-visible cracks (e.g., intrados and extrados cracks in masonry arches). Unlike learning-based anomaly detection methods [Schlegl et al. 2019; Akcay et al. 2019; Beggel et al. 2020; Nguyen et al. 2019; Yi and Yoon 2020], *PatchCore* does not require extensive pre-training on normal datasets, making it practicable when data is scarce.

However, the anomaly detection method proposed by Y. Jing et al. [2024b] had three significant drawbacks:

(1) It was observed that geometric features encoded in FPFH are not sufficiently sensitive to capture small cracks [Y. Jing et al. 2024b]. Consequently, FPFH features were deemed limited to identifying defects with pronounced geometric distortions.

(2) Non-crack defects, such as water patches, are important for infrastructure defect detection. Such defects may not cause geometric distortions and are likely to be undetected in the original anomaly detection framework.

(3) The method has only been validated on a synthetic arch dataset [Y. Jing et al. 2023]. While the synthetic dataset captured primary failure mechanisms, it does not explicitly simulate crack geometry. In other words, new surfaces created by the cracking were neglected.

Drawbacks (1) and (2) arise from using solely geometric features for anomaly detection. Yue Wang et al. [2023] proposed a multimodal Transformer-based model [Vaswani et al. 2017] to fuse two modalities (e.g., 3D geometry and 2D colour information) for anomaly detection. To improve memory and computational efficiency, Costanzino et al. [2024] used layer pruning to discard unnecessary layers from 2D and 3D feature extractors, while Y. Li et al. [2024] directly concatenated 2D and 3D feature maps for fusion. Despite state-of-the-art (SOTA) performance on datasets such as MVTec 3D-AD [Bergmann et al. 2021], current multimodal anomaly detection methods still rely heavily on Transformers for 3D feature extraction and fusion, which are known to be memory intensive.

To avoid the heavy memory consumption of multimodal methods, Cao et al. [2023] proposed a complementary pseudo-multimodal feature (CPMF), which uses handcrafted descriptors that concatenate geometric features (extracted by FPFH) and 2D colour features (extracted by pre-trained 2D neural networks [Cao et al. 2022; Q. Wan et al. 2021]). To obtain images for extracting 2D colour features, 3D point clouds are projected by a camera simulator with known intrinsic and extrinsic parameters in multiple views. The correspondence between pixels and points can therefore be determined which enables direct integration of 2D and 3D features. While CPMF demonstrates SOTA performance on MVTec 3D-AD, it may be constrained by the colour features from 2D neural networks, which were not optimized for crack detection.

To overcome the aforementioned shortcomings, we introduce a multimodal feature, 3DMulti-FPFHI, which combines geometry and intensity features to provide robust anomaly detection. This approach leverages intensity information (obtained either from laser intensity or visible-light colour information), which has the potential to enhance anomaly detection performance, especially with regard to subtle structural cracks. Further, 3DMulti-FPFHI may enable the detection of non-geometric defects (such as water patches) that are characterized by intensity anomalies. To demonstrate the robustness of the proposed 3DMulti-FPFHI,

CPMF is utilized as a baseline for comparisons in our synthetic data, supported by comprehensive statistical analysis.

Several studies in the literature aimed to address drawback (3) by improving synthetic simulators. A previous study by Mohammadi et al. [2019] introduced defects directly into the geometry, but neglected surface textures. Narazaki et al. [2021] mapped concrete crack textures onto mesh surfaces, capturing visual but not geometric details. Recently, J. Liu et al. [2024] created the Real3D-AD benchmark, which manually adds anomalies to scanned high-resolution point clouds, though data collection and manipulation are non-trivial. By combining all the merits of previous studies, we propose a mechanically consistent approach for synthesizing the geometry of intrados cracks in the synthetic data in a fully automated manner.

Although previous studies [Y. Jing et al. 2022; Y. Jing et al. 2024a] have increased confidence that synthetic data can substitute real data for segmentation tasks, the effectiveness of such an approach for infrastructure anomaly detection remains an open question. To address this issue, the performance of anomaly detection techniques needs to be evaluated using real datasets. To this end, this paper explores the application of the framework to both a masonry arch bridge [Acikgoz et al. 2017b] and a large-scale experimental model of a segmental concrete tunnel lining, where 'normal' and 'anomalous' data were clearly defined.

To summarize, the academic contributions of this work are as follows: (1) the introduction of 3DMulti-FPFHI as a novel multimodal feature for anomaly detection, benchmarked against a SOTA method CPMF; (2) the development of an enhanced synthetic dataset that bridges the gap between controlled experiments and real-world scenarios; and (3) the validation of the proposed framework on real-world datasets, demonstrating its generalizability for detecting defects across varied infrastructure types.

The paper is organized as follows. In Section 2, we detail the architecture of the anomaly detection framework, with a focus on the construction of the 3DMulti-FPFHI. Section 3 describes a new synthetic and two real-world datasets which we use to test the 3DMulti-FPFHI. Different from our previous work, [Y. Jing et al. 2024b], the new synthetic point clouds include intensity information and simulate new surfaces formed by cracks. Section 4 presents a comprehensive analysis of the

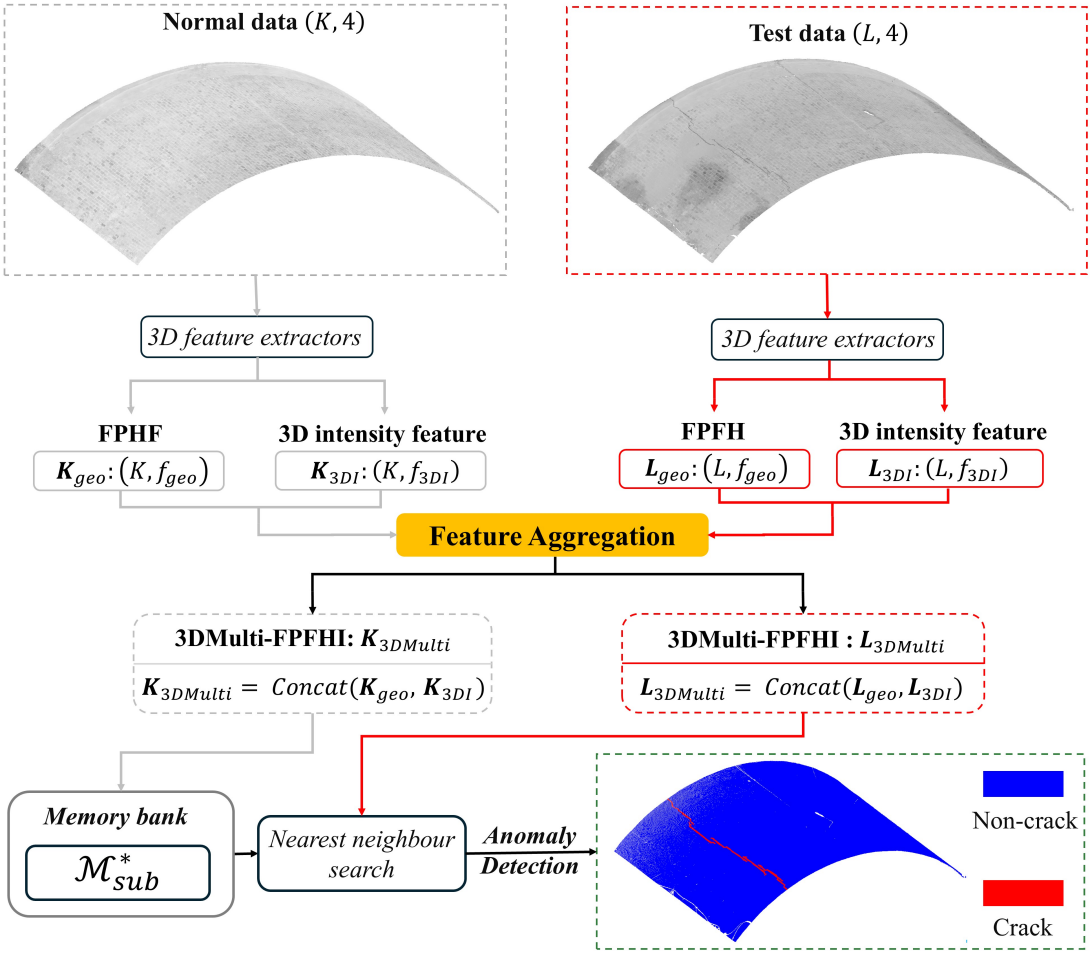


Figure 6.1: Overview of anomaly detection using *PatchCore*. The input point cloud data consist of intensity values along with x, y, and z coordinates. FPFH are extracted from both normal and test data (test data includes both normal and anomalous points) using the coordinates. The 3D intensity feature is derived from the intensity information of the point clouds. By concatenating FPFH and the 3D intensity feature, the proposed 3DMulti-FPFHI enhances anomaly detection accuracy. 3DMulti-FPFHI from the normal data are compressed into a memory bank and compared with test data features to detect anomalies, which are highlighted in red colour.

proposed anomaly detection approach. The synthetic data are used to conduct feature analysis and sensitivity evaluations while the real-world datasets are used to assess the framework's practical applicability in operational conditions. Finally, we discuss our findings and potential developments for future research in Section 5.

6.2 Methodology

To explore the benefits of incorporating intensity information for anomaly detection, local intensity features are extracted using both 2D and 3D methods for

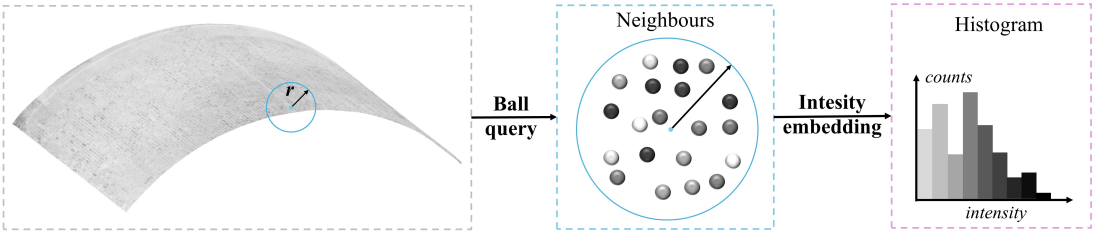


Figure 6.2: Computation of intensity histogram using FPFH. Intensity values of neighbours (defined by ball query with radius r) are incorporated into the histogram as intensity features.

comparison. The framework is shown in Figure 6.1 and comprises three components: (1) geometric (e.g. FPFH) and 3D intensity feature extraction, and their aggregation in 3DMulti-FPFHI; (2) rendering of a 3D point cloud into images, followed by the extraction of 2D intensity features to generate CPMF; and (3) the utilization of *PatchCore* for anomaly detection.

6.2.1 Geometry and intensity feature extraction and aggregation

FPFH [Rusu et al. 2009] is employed as a rotationally invariant geometric feature descriptor to capture local geometry. The computation of FPFH relies on the surface normal at each point in the point cloud. To obtain FPFH features, the ball query algorithm searches for neighbours around each point. For each neighbour-to-center point pair, a Darboux frame is defined, along with normal vectors to compute three intersection angles. These angles are embedded into three histograms, which are concatenated to describe normal, geodesic and torsional curvature. Geometric features from normal and test data are denoted as $\boldsymbol{K}_{geo}$ and $\boldsymbol{L}_{geo}$ respectively (see Figure 6.1). Implementation details can be found in Y. Jing et al. [2024b] and Rusu et al. [2009].

For simplicity, it is assumed that intensity information for each point takes a scalar value (e.g., a laser return amplitude or a greyscale colour amplitude). To extract the 3D intensity feature, intensity values are encoded into a histogram, following a procedure analogous to FPFH. The method is illustrated in Figure 6.2: (1) Ball query is first adopted to obtain neighbours for each point within a radius r, and (2) local intensity features from these neighbours are then embedded into a histogram with a predefined bin size. The 3D intensity feature for normal and

test data are represented as $\boldsymbol{K}_{3DI}$ and $\boldsymbol{L}_{3DI}$, respectively, as shown in Figure 6.1. Inspired by Logoglu et al. [2016], this embedding computes the absolute relative values between the centre point and its neighbours, reducing mismatches in the 3D intensity feature between normal and test data, which may occur due to changes in environmental conditions during data collection. This aspect is further discussed in Section 4.3.

To enhance the representation of FPFH (e.g., $\boldsymbol{K}_{geo}$ and $\boldsymbol{L}_{geo}$), they are concatenated with 3D intensity feature $\boldsymbol{K}_{3DI}$ and $\boldsymbol{L}_{3DI}$ to create 3DMulti-FPFHI (e.g., $\boldsymbol{K}_{3DMulti}$ (normal data) and $\boldsymbol{L}_{3DMulti}$ (test data) as in Figure 6.1).

The original FPFH feature dimension is determined by the total number of bins across three histograms for three intersection angles, with each histogram containing 11 bins [Rusu et al. 2009]. However, experiments show that this bin count limits FPFH's ability to detect subtle geometric distortions due to its low resolution. To enhance anomaly detection accuracy, this study increases the bin count to 30 for both FPFH and the 3D intensity feature, providing finer resolution and better sensitivity.

6.2.2 2D feature extraction and aggregation for CPMF

To leverage the strength of 2D deep learning methods, Cao et al. [2023] proposed extracting intensity features using 2D neural networks [T. Lin et al. 2017]. These 2D features are concatenated with handcrafted 3D features (e.g., FPFH in their study) for the CPMF. CPMF was tested within the anomaly detection method *PatchCore* [Roth et al. 2022], which achieved SOTA performance on the MVTec 3D-AD dataset [Bergmann et al. 2021]. Thus, CPMF is used as a baseline to evaluate the performance of various multi-modal features in this study.

The computation of CPMF is illustrated in Figure 6.3, where normal point cloud data are rendered into images using specified camera positions and rotation angles (e.g., x_{rot}, y_{rot}, and z_{rot}). These angles are sampled from a predefined set, resulting in 27 different orientations ($3 \times 3 \times 3$). The camera is centred in the arch, with its z coordinate adjusted to cover the full structure. Initially, the camera faces the positive z direction. With the specified rotation angles and position, a series of extrinsic matrices are determined for the camera. To capture the whole structure,

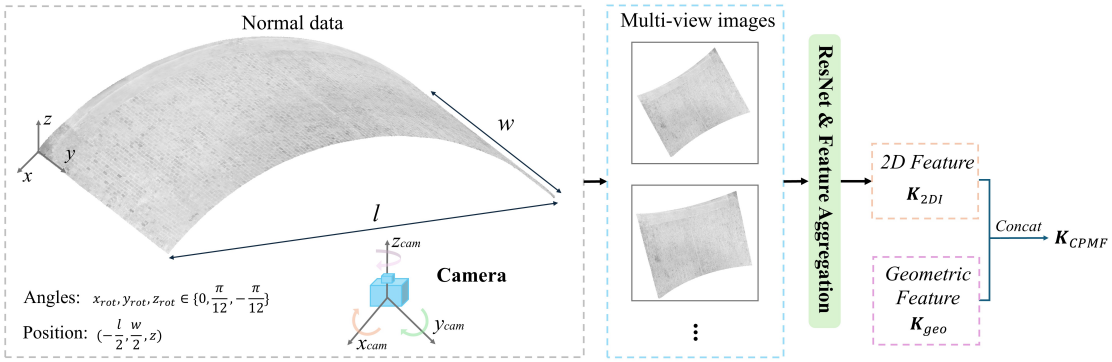


Figure 6.3: Computation of the pseudo-2D image feature on normal data. The point cloud intensity values are first projected to images using a synthetic camera (facing the positive z direction in the initial position) with multiple viewing angles. Features are then extracted using ResNet50d [K. He et al. 2016] and aggregated into the 2D feature $\boldsymbol{K}_{2DI}$.$\boldsymbol{K}_{2DI}$ is finally concatenated to K_{geo} to form the CPMF.

the focal lengths of the camera for capturing synthetic and real arch point clouds are set to 700mm and 450mm respectively. The image size is set to 512×512 pixels, ensuring that the total pixel count approximates the number of points in the point cloud. The transformations between the point clouds and pixels of projected images can be straightforwardly obtained from the extrinsic and intrinsic matrices of the camera.

This study uses the first three blocks from ResNet50d [K. He et al. 2016], with each feature map interpolated back to a 512×512 resolution and concatenated to form the 2D feature $\boldsymbol{K}_{2DI} \in \mathbb{R}^{K \times f_{2DI}}$, where f_{2DI} equals 1792 (the sum of feature channels from the output of three blocks). $\boldsymbol{K}_{2DI}$ is normalized in feature dimension to ensure the same contribution as $\boldsymbol{K}_{geo}$ in the anomaly detection. CPMF for normal data, namely $\boldsymbol{K}_{CPMF}$ is obtained by concatenating $\boldsymbol{K}_{geo}$ with $\boldsymbol{K}_{2DI}$, as shown in Figure 6.2. The same procedure is applied to compute the test data (resulting in $\boldsymbol{L}_{CPMF}$), which is omitted for simplicity. Theoretical background and implementation details are provided in Cao et al. [2023].

6.2.3 Anomaly detection method

Anomaly detection is performed using the *PatchCore* method [Roth et al. 2022], which involves two key steps. First, features extracted from normal data (e.g., $\boldsymbol{K}_{3DMulti}$ and $\boldsymbol{K}_{CPMF}$) are compressed using coreset subsampling [Sener and Savarese 2017] to create a representative subset $\mathcal{M}^*_{sub}$ that approximates the original features. Coreset subsampling helps reduce redundancy in the memory bank, where

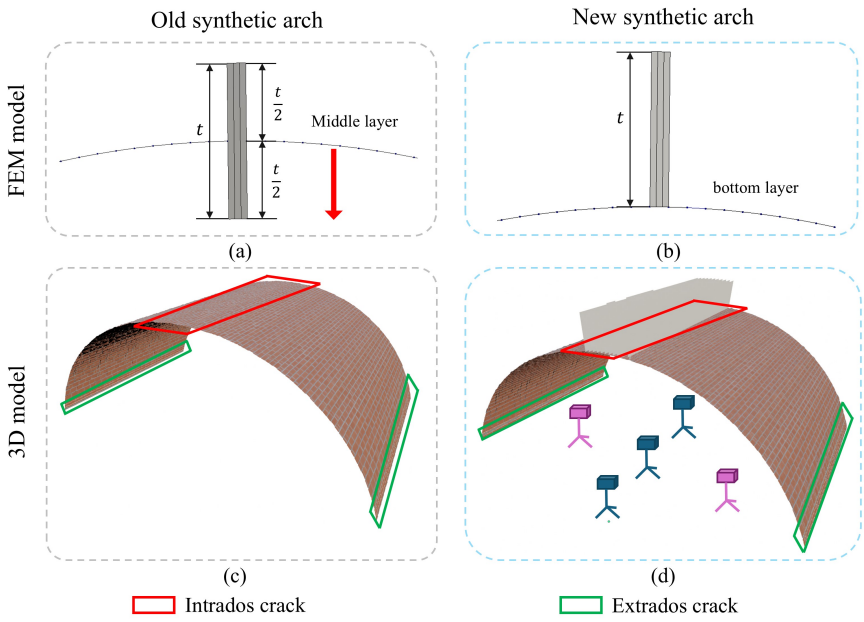


Figure 6.4: Modifications to the synthetic datasets, including updates to both the FEM model and its 3D representation in Blender. (a) and (b) show the layers where nodes are placed in the FEM model for the original and modified synthetic arches respectively. In the updated model, an eccentricity is applied by moving the node points to the bottom layer. (c) and (d) show the reconstructed 3D models in Blender for the original and modified arches, with intrados and extrados cracks highlighted in red and green, respectively. The new model now incorporates idealized intrados crack surfaces, points from which can be sampled by laser scanners.

normal features are stored (Figure 6.2). Secondly, test data features (e.g., $\boldsymbol{L}_{3DMulti}$ and $\boldsymbol{L}_{CPMF}$) are paired with features in $\mathcal{M}^*_{sub}$ using nearest neighbour search, where the Euclidean distances between feature pairs, min_dists, are calculated. The individual values of min_dists are normalised with respect to the maximum value in the min_dists set, and the values above the anomaly score s are classified as anomalies. Further details are available in Y. Jing et al. [2024b] and the provided code.

6.3 Datasets

Three datasets are employed to evaluate and validate the proposed method: (1) a synthetic masonry arch dataset for statistical feature analysis and sensitivity evaluation; (2) a real masonry arch dataset; and (3) a large-scale experimental model of a concrete segmental tunnel.

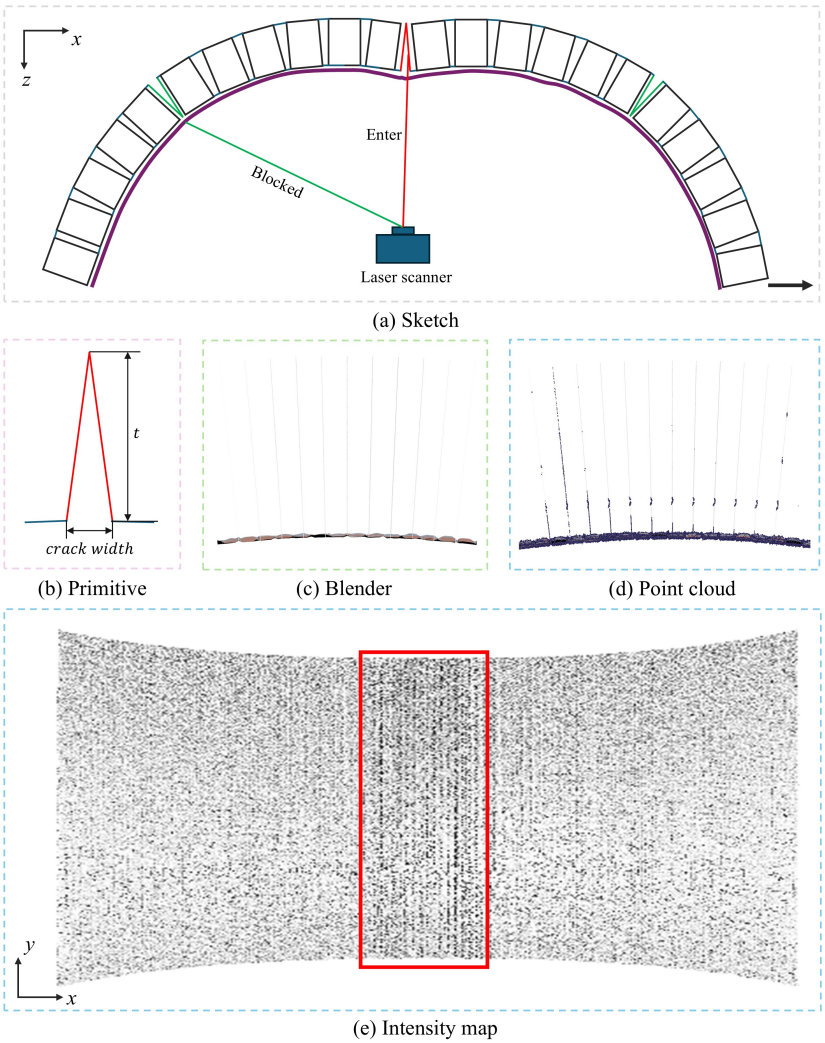


Figure 6.5: Integration of the intrados crack surfaces in the updated synthetic models. (a) shows a sketch comparing intrados (red colour) and extrados (green colour) cracks, highlighting how laser scanners may sample points from surfaces created by intrados cracks; (b) depicts the adopted idealized geometry of the intrados crack; (c) and (d) shows the reconstructed 3D model with intrados cracks in Blender and the synthetic point cloud, respectively. (e) represents the bottom view of the intensity map (greyscale) where the region of intrados cracks is shown in the red inset.

6.3.1 Synthetic masonry arch dataset

The framework documented in Y. Jing et al. [2024b] is adopted to synthesize point clouds from finite element method (FEM) models of masonry arches. Details of the FEM modelling and point cloud generation process are provided in Y. Jing et al. [2024b] and are omitted here for brevity. However, modifications have been made to both the FEM model and the reconstructed 3D model in Blender for improved realism, as illustrated in Figure 6.4a-d.

In the original synthetic arch FEM model (Figure 6.4a), nodal displacements were

extracted from integration points in the middle layer of the shell elements. However, these displacements are inconsistent with real-world scenarios since the laser scanner can only capture points on the intrados surface. To address this, the new FEM model prescribes an eccentricity of $t/2$, enabling the nodes to be located on the bottom layer of the shell, as illustrated in Figure 6.4b, where t is the thickness of the masonry units. The new model also distinguishes between intrados and extrados cracks, as shown in Figure 6.4d. Extrados cracks can only be detected through geometric features since the surfaces they create are not visible on the intrados. In contrast, intrados cracks create surfaces which may be visible.

To illustrate this aspect further, Figure 6.5a shows a deformed arch subjected to a support movement in the x direction. The purple outline represents the deformed bottom arch surface. The laser scanner can sample points from the new surfaces created by intrados cracks (marked in red) but cannot sample points for extrados cracks (marked in green). In [Y. Jing et al. 2024b], the new surfaces created by intrados cracks were not considered (see Figure 6.4c). However, in this study, a simple approach is developed to consider points from new surfaces created by intrados cracks. The idealized geometry of an intrados crack is shown in Figure 6.5b, where the front view represents an isosceles triangle of height t, and with a base width equal to the crack width on the bottom surface of the shell elements. Every time an intrados crack forms due to the separation of two shell elements, triangular crack surfaces are created. The reconstructed crack geometry and the corresponding point cloud for cracks in the crown area are shown in Figure 6.5c and d.

For simplicity, we assume uniform reflectivity across all structural surfaces (e.g., masonry, mortar, and inner crack surfaces). Consequently, variations in intensity between the inner crack surfaces and the rest regions are attributed to the geometry of the inner cracks. In the lidar simulator [Reitmann et al. 2021b], synthesized laser beams undergo multiple reflections between the inner crack surfaces before returning to the sensor, resulting in lower intensity values. This phenomenon is visualized in the red inset of Figure 6.5e. The intensity map aligns with the experimental results of [Stałowska et al. 2022], where crack regions exhibit lower intensity values compared to other surface regions.

The number of synthetic laser scanners has been increased from three to five in

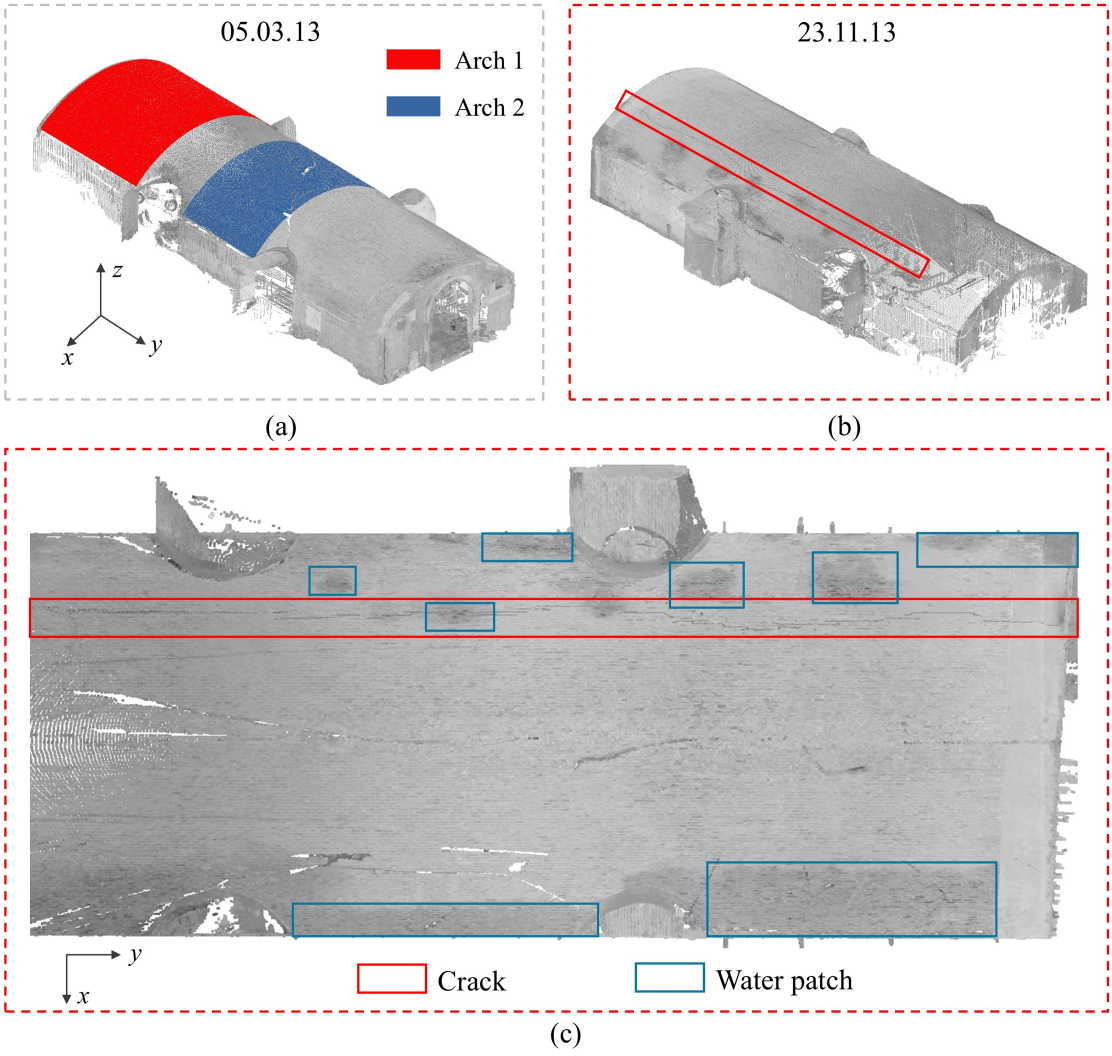


Figure 6.6: Visualization of the London Bridge Station point cloud for arch E57. (a) shows the point cloud collected on 05.03.13, highlighting segmented Arch 1 (blue) and Arch 2 (red) for anomaly detection. (b) indicates the point cloud collected on 23.11.13, showing a developed longitudinal crack in the red inset. (c) represents the plan view of the point cloud on 23.11.13, with newly appeared water patches highlighted by the blue insets.

comparison to the previous setup in Y. Jing et al. [2024b], (see Figure 6.4d). In addition to the original three scanners (dark blue), two additional scanners (purple) are positioned on either side, spaced evenly along the direction orthogonal to the generatrix. This increased scanner count enhances the point density.

Surface roughness is defined by UV mapping brick and mortar depth profiles onto the masonry arch surfaces. Unlike the previous study [Y. Jing et al. 2024b], which only considered surface roughness for the x support movement case, surface roughness is now included in all synthetic models to better reflect real masonry bridges. Surface roughness settings are consistent with those used in previous work [Y. Jing

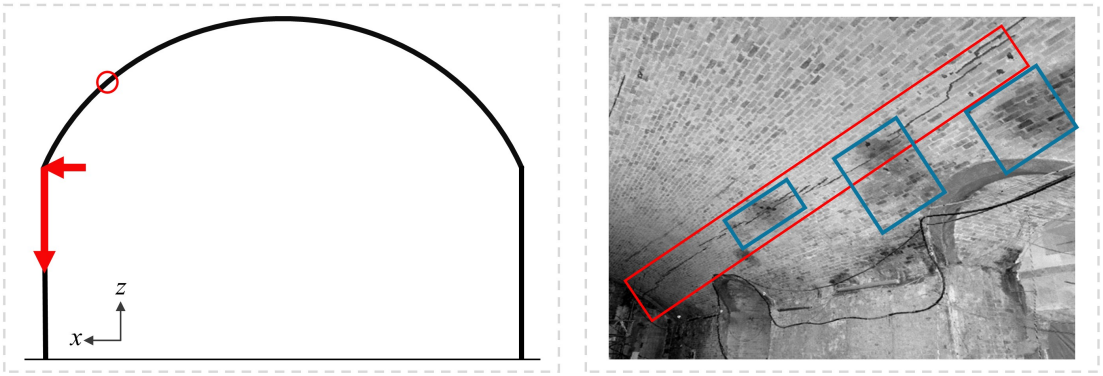


Figure 6.7: (a) Section view of Arch E57, highlighting settlement records indicated by red arrows. A crack caused by the settlement is enclosed within a red circle. (b) Visualization of the crack and water patches, marked with red and blue insets, respectively.

et al. 2024b].

6.3.2 Masonry arch dataset

London Bridge Station, which was constructed during the 19th century, underwent a redevelopment focusing on the removal and then replacement of sections of its viaducts in 2013. This redevelopment included the construction of new piles and buttress walls supported by these piles [Acikgoz et al. 2017b]. The most impactful phase, which introduced significant settlements, was the piling phase. During this phase, new piles were constructed under Arch E55. Acikgoz et al. [2017b] investigated the influence of these construction activities on the neighbouring Arch E57. Piling began on 31.01.13 and was completed by 16.08.13. The ensuing construction of buttress walls was completed by 21.11.13.

Point cloud data was collected using a Faro Focus X330 lidar scanner on three dates: 14.12.12, 05.03.13 and 23.11.13. These dates correspond to the periods when 0%, 50% and 100% of the piles were constructed, respectively. However, the first scan on 14.12.12 was noisy and incomplete. Therefore, only the data collected on 05.03.13 and 23.11.13 are utilized in this study as shown in Figure 6.6a and b. Between these dates, a span opening of approximately 5mm was recorded and subsequently filled with mortar, as highlighted in the red inset of Figure 6.6b. The intensity differences between the longitudinal crack (along the y-axis) and the neighbouring non-crack regions are attributed to the large reflectivity differences between the filling mortar and the original materials. The crack is

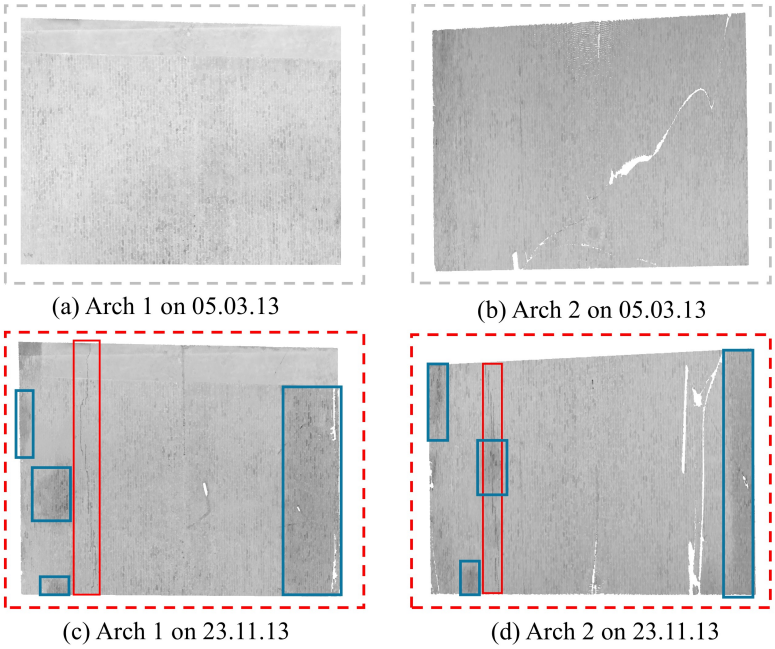


Figure 6.8: Top view of the segmented point clouds of arches 1 and 2, with intensity values mapped to greyscale. (a) and (b) represent the data collected on 05.03.13, showing no defects. (c) and (d) show the data from 23.11.13, highlighting a crack (red insets) and a water patch (blue insets).

evident in the plan view of the point cloud in Figure 6.6c, due to this intensity difference. Additionally, water patches were observed around the crack and the piers (highlighted with blue boxes in Figure 6.6c).

During the works, monitoring targets were installed to record support movements. Significant vertical and lateral displacements were observed at the pier shared between Arches 55 and 57 on 23.11.13, as indicated by the red arrows in Figure 6.7a. The crack location is highlighted by a red circle. Figure 6.7b visualizes the defects sustained by the masonry arch E57 during the piling work, including the crack and associated water patches.

Table 6.1: Point numbers and the corresponding density measured on Arch 1 and Arch 2.

	Arch 1		Arch 2	
	05.03.13	23.11.13	05.03.13	23.11.13
Point number	2,917,522	4,015,773	239,243	289,414
Density(mm)	3	3	13	11

Anomaly detection was performed on the areas labelled Arch 1 and Arch 2 (highlighted in red and blue, respectively, in Figure 6.6a). These two arches were segmented from the full point cloud and analyzed separately, as shown in Fig-

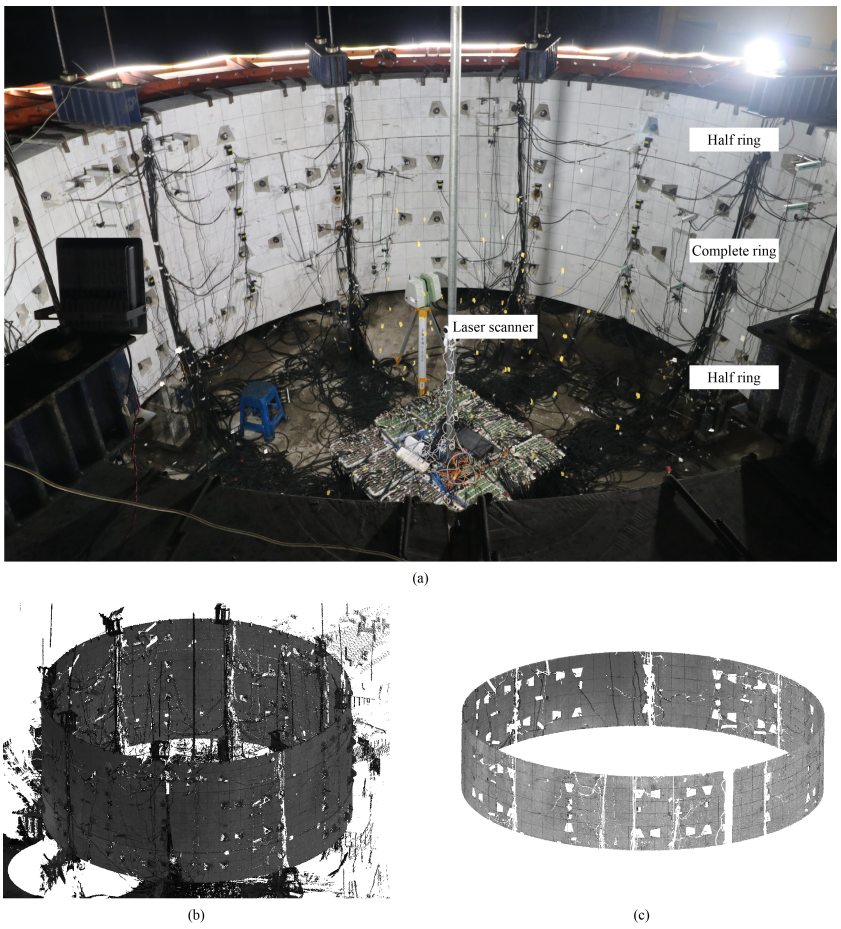


Figure 6.9: Visualization of the full-scale physical experiment and the resulting point clouds. (a) shows the experimental setup, including the laser scanner and tunnel lining rings. (b) shows the raw registered point cloud while (c) shows the processed (segmented and cleaned) point cloud of the middle complete ring.

ure 6.8a-d. The point numbers and corresponding densities are shown in Table 6.1, where density is measured by averaging the nearest neighbour distances in the point clouds. Although this study focuses primarily on crack detection, water patches, were also defined as anomalies to demonstrate the potential benefits of integrating the intensity feature. Further details on the case study of Arch E57 can be found in Acikgoz et al. [2017b].

6.3.3 Concrete segmental tunnel dataset

Point clouds were acquired during a full-scale experiment on a segmental concrete tunnel lining. The experimental structure consists of one complete (e.g. full-length in the tunnel longitudinal direction) lining ring and two half-rings (e.g. half-length in the tunnel longitudinal direction), as shown in Figure 6.9a and b. The complete ring is stagger-jointed and the half-ring on either side replicates

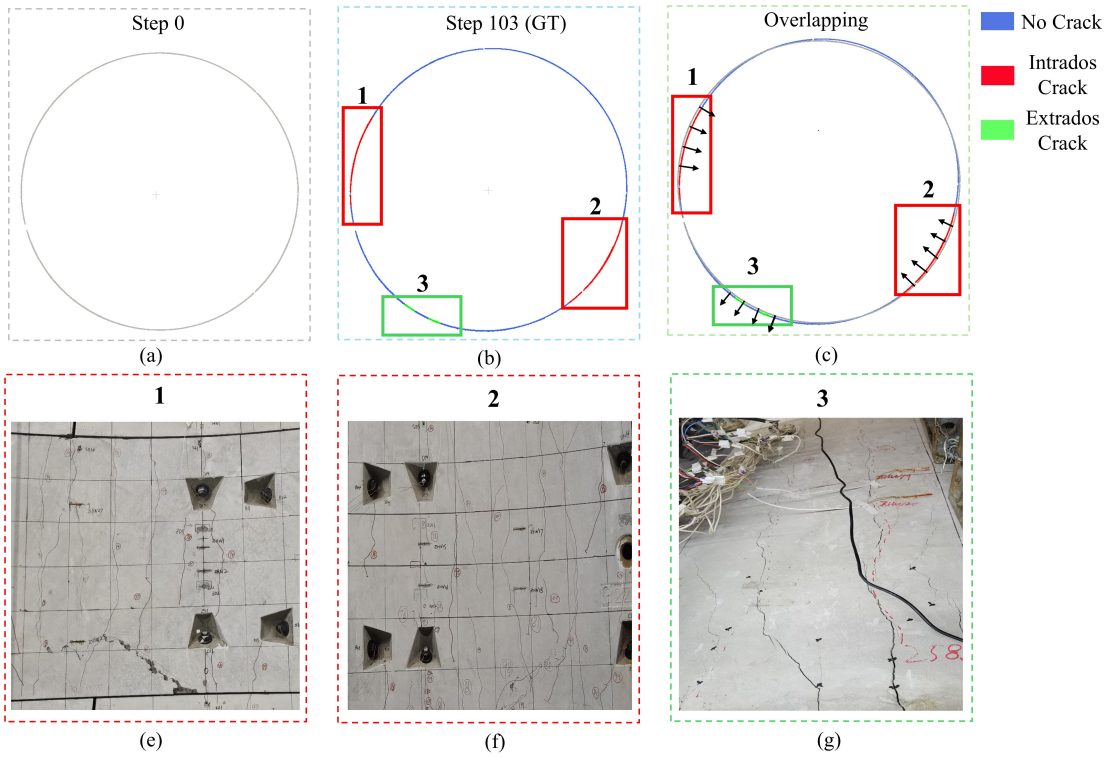


Figure 6.10: Visualization of the geometric distortion of the real tunnel in bird view and corresponding crack photos. (a) visualizes the unloaded tunnel; (b) ground truth on loading step 103 (blue, red, and green indicate the no crack, intrados crack, and extrados crack labels); (c) is retrieved by overlapping (b) and (c) where deformations are indicated by black arrows; (e), (f), and (g) show photos of cracks indexed as **1**, **2**, and **3** respectively.

the boundary conditions of a real-world lining ring. Only point clouds from the complete ring (Figure 6.9c) are used for anomaly detection. The lining consists of six concrete segments per ring, with an inner diameter of 2750 mm, a thickness (radial direction) of 350 mm, and a length (longitudinal direction) of 1200 mm.

In the experiment, external loads were applied to the lining by a set of servo-hydraulic jacks positioned concentrically around the rings to simulate earth pressure. Loading continued until structural failure, meaning the dataset captures a wide range of conditions — from intact to severely damaged. There were a total of 103 loading steps.

The point clouds were scanned using a terrestrial laser scanner (Leica ScanStation C10). Given the lack of occlusions, a single scan was used to capture the whole structure in each loading step. In addition to the geometry information, laser intensity information was also recorded. Point clouds were acquired at loading steps 0, 20, 25, 76, 89, 96 and 103.

The point clouds of the complete ring were manually segmented and non-lining points were all removed. As a result, the clean point clouds of the complete ring have ~3,000,000 points each and the average distance between neighbouring points is 2 mm. Cracks were recorded after each pre-defined loading step, allowing anomaly detection results to be validated against visual observations.

We visualize the point cloud corresponding to step 0 (e.g., the reference geometry) and step 103 with cross-sections in Figure 6.10a and b. Regions containing intrados cracks (indices 1 and 2) are highlighted in red boxes in Figure 6.10b, and were directly captured by the laser scanner. In contrast, extrados cracks were not directly captured by the laser scanner. An example extrados crack (shown with index 3) is highlighted with a green box in Figure 6.10. By overlaying Figure 6.10a and b, the resultant Figure 6.10c illustrates the deformation of tunnel rings in the crack regions under the applied force. Intrados cracks are characterised by inward bulging, while extrados cracks exhibit outward bulging, indicated by black arrows. Figure 6.10e-g provide photographs of the cracks.

To enable an evaluation of the performance of the algorithm, crack labels were needed. The surface of the concrete was marked with a 20cm square grid as shown on the concrete surface of Figure 6.9a. If a crack is observed within this grid, all points in the point cloud in this region are associated with a crack label. Then due to the 2D nature of the experimental setup, it was assumed that the cracks extended uniformly along the length of the ring. The labelling was performed for each acquired point cloud, by identifying the newly added cracks (compared to the last scanned loading step) as shown with black mark lines in Figure 6.10e-g.

6.4 Experiments and results

6.4.1 Implementation details and metrics

To improve computational efficiency and achieve uniform density, point clouds from different datasets are subsampled via voxel downsampling. The synthetic and real arch point clouds are downsampled to a voxel size of 0.02m, while the tunnel point clouds are downsampled to 0.01m to retain surface detail. Ball query with a radius of 0.12 m is used to identify neighbouring points to accurately and

robustly estimate the surface normals. A 1m radius is used for FPFH feature computation to capture the local geometry of synthetic and real masonry arch, whilst the radius is adjusted to 0.6m for the tunnel dataset. The size of $\mathcal{M}^*_{sub}$ is set to 4000 for storing normal features.

The intrinsic and extrinsic matrix settings for the 2D-to-3D projection are included in the provided code. CPMF is computed and compared across different feature types using synthetic data only, as our results and statistical analysis clearly highlight the limitations of CPMF when directly applied to infrastructure anomaly detection, which will be elaborated on in Section 4.2. Therefore, further analysis of real datasets has been omitted for simplicity.

Experiments are conducted on an AMD Ryzen 9 5900 CPU with 128GB memory (for FPFH computation) and RTX 3080 with 10GB memory (for feature map computation in images). Since the relationship between geometric anomalies and crack width is non-linear, a single optimal anomaly score s cannot be defined for binary classification across all datasets. Therefore it is difficult to use traditional performance metrics (e.g., *Precision*, *Recall*, and $F1$-*score*) to quantify the performance of the features for anomaly detection.

To investigate this aspect, Section 4.2.2 introduces statistical analysis approaches for a more comprehensive evaluation of feature types and their influence on anomaly detection results. The conclusions drawn from this analysis can be generalized to other tests. However, $F1$-*score* with varied thresholds (e.g., $0.3 * s$ and $0.5 * s$) are still provided in Sections 4.2 to 4.4 for comparing the performance of different features. Corresponding *Recall* and *Precision* values are available in Appendix A and Appendix B (e.g., 6.4.4 and 6.4.4 correspondingly).

6.4.2 Results for the synthetic arch

Anomaly detection under different support movement cases

The masonry arch FEM model is subjected to four distinct support movements: x, z, $x + z$, and a combination of x rotation and z translation, as shown in Figure 6.11a-d respectively. Although *PatchCore* does not differentiate crack types, points are categorised as 'no crack', 'intrados crack', 'extrados crack', and 'inner crack' (represented as blue, yellow, green, and purple respectively in Figure 6.11a-d).

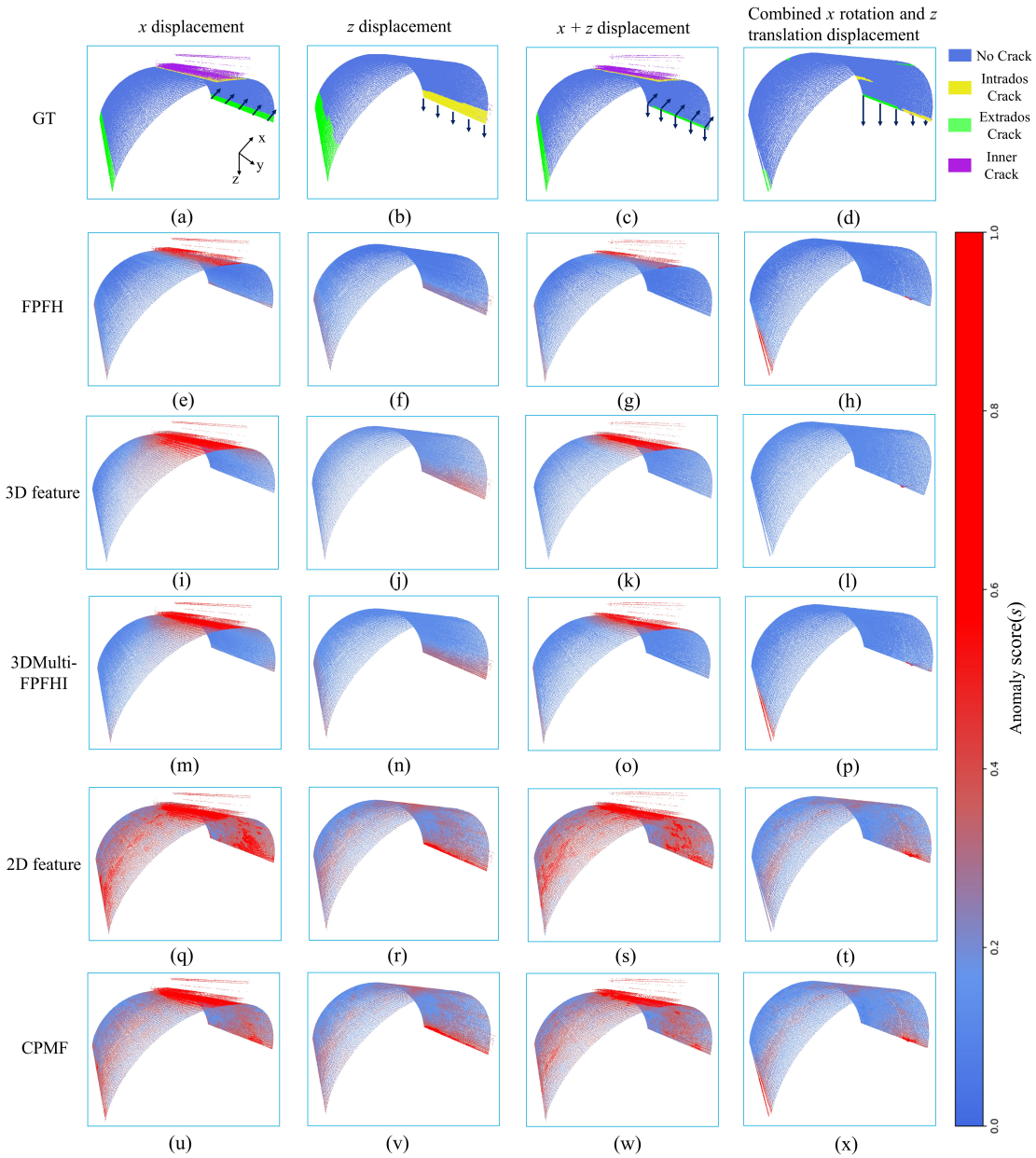


Figure 6.11: Anomaly detection results on the masonry arch for four different support movement cases. (a)-(d) show the ground truth where blue, yellow, green, and purple colours represent non defect, intrados crack, extrados crack and inner crack correspondingly; (e)-(h), (i)-(l), (m)-(p), (q)-(t), and (u)-(x) display heatmaps representing the anomaly score for each point, using different feature types.

Specifically, 'inner crack' refers to points on the intrados cracks where points were samples from the new surfaces created by cracks. To improve detailed visualization of 'inner crack' points after exposing the bridge to x rotation and z translation, supplementary views are presented in Appendix A (6.4.4). The sub-classification of crack types will later assist in analysing how different crack geometries influence anomaly detection, allowing for more in-depth statistical analysis.

Results are visualized using heatmaps in Figure 6.11e-x, where the anomaly scale is

in the range [0, 1]. Points with an anomaly score above 0.5 are shown in red, whilst the remaining points are illustrated on a blue-to-red scale, representing increasing geometric anomalies. The threshold of 0.5 was chosen as it highlights extrados cracks more clearly; these cracks are often associated with low anomaly scores as their geometric distortions are less pronounced than other cracks.

The performance of the anomaly detection method is compared across five different feature types in the synthetic dataset. Figure 6.11e-h illustrate the detection performance using FPFH, where both intrados and extrados cracks are identified using geometric information. To examine the contribution of adding the 3D intensity feature, Figure 6.11i-l provide the anomaly detection results using only the 3D intensity feature. Compared to FPFH, the anomaly scores associated with the 'intrados crack' showed greater differentiation from normal data for the 3D intensity feature. The reason for this relates to the intensity anomalies introduced by new crack surfaces around the intrados cracks. However, the 3D intensity feature failed to detect extrados cracks, as intensity values remain invariant in regions where no new crack surfaces are created.

Additionally, the 3D intensity feature introduces diffusion of anomalies for intrados cracks, where anomalies spread from the cracks to adjacent normal regions with gradually decreasing anomaly scores, as featured by Figure 6.11i and k. The diffusion is caused by the large size of the selected ball query radius (e.g. 1m in the synthetic arch dataset). The diffusion occurs for both intrados and extrados cracks, potentially affecting the precision of localizing anomalies.

Detection results using the 3DMulti-FPFHI are presented in Figure 6.11m-p. 3DMulti-FPFHI preserves the advantages of the 3D intensity feature in detecting inner cracks while effectively capturing inner/extrados cracks by integrating FPFH. As shown in Figure 6.11m and o, the diffusion effect observed with the 3D intensity feature is mitigated in 3DMulti-FPFHI. This improvement is achieved by FPFH balancing the contribution of the 3D intensity feature, suppressing the anomaly scores in diffused regions.

Figure 6.11q-t illustrate the performance when using the 2D intensity feature. While the 2D intensity feature can roughly identify the locations of intrados cracks by detecting intensity anomalies in 2D pixels, anomaly detection remains impre-

Table 6.2: $F1$-*score* for the four different support movement cases of the masonry arch with thresholds of $0.3 * s$ and $0.5 * s$. Red text indicates the best metrics whereas blue text denotes the second-best metrics.

	FPFH	3D feature	3DMulti-FPFHI	2D feature	CPMF
$x(0.3 * s)$	0.730	0.596	0.676	0.430	0.430
$x(0.5 * s)$	0.240	0.598	0.673	0.444	0.669
$z(0.3 * s)$	0.120	0.453	0.505	0.298	0.299
$z(0.5 * s)$	0.090	0.097	0.102	0.486	0.531
$x + z(0.3 * s)$	0.313	0.639	0.678	0.316	0.316
$x + z(0.5 * s)$	0.184	0.680	0.639	0.330	0.441
$x\,rot + z\,trans(0.3 * s)$	0.310	0.213	0.330	0.060	0.072
$x\,rot + z\,trans(0.5 * s)$	0.302	0.182	0.318	0.368	0.427

cise, even when combined with FPFH to form CPMF, as shown in Figure 6.11u-x. The 2D intensity feature struggles to capture the fine details of cracks. One potential reason for this relates to the ImageNet training data [J. Deng et al. 2009], which lacks images representing infrastructure anomalies (e.g., cracks).

Anomaly detection results across all five features are quantified in Table 6.2 using the $F1$-*score* metric. 3DMulti-FPFHI demonstrates superior performance compared to other feature types in most cases. Notably, the 2D intensity feature and CPMF achieve higher performance for the $x\,rot + z\,trans$ case when the threshold is $0.5s$. However, results shown in Figure 6.11q-t and u-x clearly indicate CPMF's inability to localize cracks; $F1$-*score* metric does not capture this deficiency.

Statistical analysis for the x-direction support movement case

To investigate further, the values of *min_dists* for the five different feature types are approximated using Kernel Density Estimation (KDE) [Weglarczyk 2018] and represented with continuous probability density functions. In these distributions, the horizontal axis denotes the *min_dists* values, while the vertical axis represents the estimated probability density. The KDE distributions for the 'x-displacement' case are illustrated in Figure 6.12a-t using FPFH, 3D intensity feature, 3DMulti-FPFHI, 2D intensity feature, and CPMF respectively. For an initial high-level analysis, 'intrados crack', 'extrados crack', and 'inner crack' are grouped under the 'crack' label. Further crack analyses fuse 'intrados crack' and 'inner crack' labels under 'intrados crack' (Figure 6.5d) for simplicity.

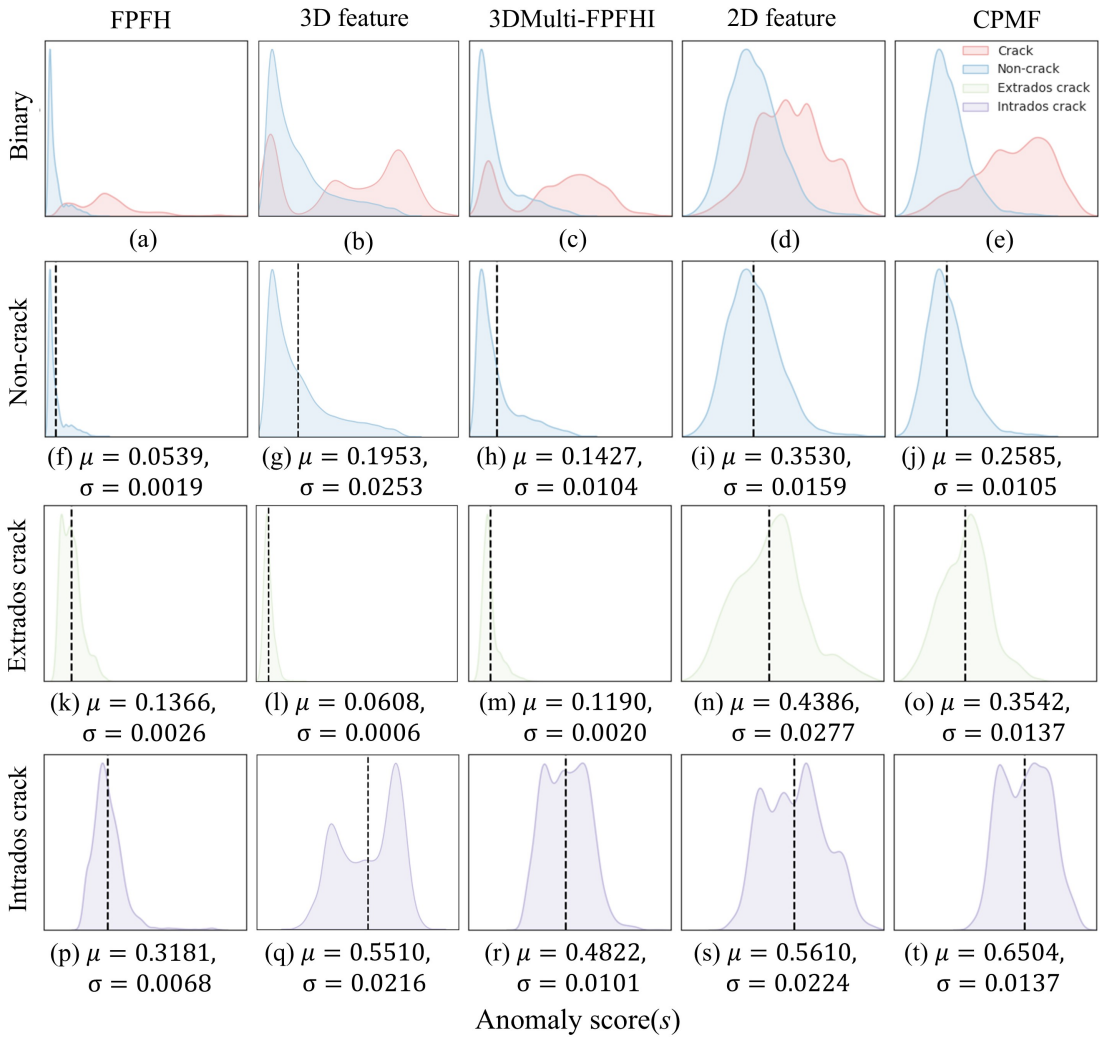


Figure 6.12: Distributions of *min_dists* for the x displacement case. (a)-(e) represents distributions of 'non-crack' (blue) and 'crack' (red) points for five different feature types. (f)-(j) show distributions of 'non-crack' points together with their corresponding mean and variance. The 'crack' label is further subdivided into 'extrados crack' (green) and 'intrados crack' (purple) shown in (k)-(o) and (p)-(t) respectively to discriminate the influence of feature type on statistical metrics of *min_dists*.

The distribution of 'non-crack' and 'crack' labels are shown in Figure 6.12a-e, where overlapping regions indicate certain detection errors. When a threshold is applied, 'non-crack' points on the right side of the threshold are misclassified as 'crack', whilst 'crack' points on the left side of the threshold are erroneously identified as 'non-crack'. These results demonstrate that feature types independent of neural networks (e.g., FPFH, 3D intensity feature, and 3DMulti-FPFHI) outperform the 2D intensity feature and CPMF by providing a clearer boundary between 'non-crack' and 'crack' distributions.

To quantitatively assess the influence of feature type on model performance, the

'non-crack' distributions are examined, where the mean value μ is indicated by a black vertical dotted line in Figure 6.12f-j. Ideally, μ would approach 0 with a small standard deviation σ, indicating precise matching of 'non-crack' points before and after support movements. However, due to non-rigid displacements and measurement errors, the anomaly scores of normal points assume non-zero values. This is best illustrated using data from regions near cracks which also experience geometric distortions; points from these regions likely end up on the right tail of the distribution in Figure 6.12f. Points in this region deviate from a normal distribution and increase the standard deviation σ. Compared to FPFH (Figure 6.12f), the 3D intensity feature and 3DMulti-FPFHI are associated with a larger value of μ; this is due to the introduction of intensity features. However, their average values are significantly less than the ones for 2D intensity feature and CPMF, explaining why the latter features misidentified areas in Figure 6.11q and u. Notably, combining intensity features with FPFH (3DMulti-FPFHI and CPMF) reduces μ and σ for 'non-crack' distributions, which indicates an important benefit of aggregating two modalities.

The 'extrados crack' and 'intrados crack' distributions are shown in Figure 6.12k-o and p-t, respectively. A comparison of the μ values for FPFH (Figure 6.12k and p) reveals that 'extrados cracks' in the examined scenario ($\mu = 0.1366$) are associated with smaller geometric distortions than 'intrados cracks' ($\mu = 0.3181$). Therefore, 'extrados cracks' are more difficult to detect with FPFH, as their μ is closer to the 'non-crack' distribution ($\mu = 0.0539$, Figure 6.12f).

For the 3D intensity feature, μ and σ are higher for the 'non-crack' distribution (Figure 6.12g) compared to FPFH, leading to increased overlap between 'crack' and 'non-crack' regions and reduced localization precision. Furthermore, the 'extrados crack' distribution shows smaller μ and σ compared to the 'non-crack' distribution (Figure 6.12g and l), making it infeasible to differentiate between them, which corresponds to the results shown in Figure 6.11i-j. In contrast, the 'intrados crack' distribution demonstrates a significant difference in μ from the 'non-crack' distribution, enabling intrados crack detection (Figure 6.12q). However, the increased σ for 'intrados crack' distributions (e.g., $\sigma = 0.068$ for FPFH and $\sigma = 0.0216$ for the 3D intensity feature corresponding to Figure 6.12p and q) highlights the diffusion effect, where anomaly scores spread from cracks to adjacent points with

lower values.

The 3DMulti-FPFHI combines the strengths of FPFH and the 3D intensity feature. While μ and σ remain higher than FPFH for the 'non-crack' distribution, the differences are minor due to FPFH mitigating the diffusion effect of the 3D intensity feature (Figure 6.12h and m). For 'intrados cracks,' μ significantly increases with 3DMulti-FPFHI ($\mu = 0.4822$, Figure 6.11r) compared to FPFH alone, indicating advancements of considering the additional intensity modality.

Figure 6.12n, o, s, and t show that the addition of 2D intensity feature and CPMF lead to large σ for both 'extrados crack' ($\sigma = 0.0277$ and $\sigma = 0.0137$ respectively) and 'intrados crack' distributions ($\sigma = 0.0224$ and $\sigma = 0.0137$ respectively). The distribution of 'extrados crack' and 'intrados' cracks overlap significantly with the 'non-crack' distribution, resulting in poor performance.

These findings highlight the strengths and limitations of each feature type. FPFH is robust for detecting geometric distortions, while the 3D intensity feature improves differentiation between 'non-crack' and 'intrados crack' regions but struggles with 'extrados crack' detection. The 3DMulti-FPFHI leverages the advantages of both, enhancing 'intrados crack' detection while mitigating intensity-induced noise, which preserves the 'extrados crack' detection accuracy. In contrast, the 2D feature and CPMF, which rely on image features extracted by the pre-trained neural network, struggle to achieve sufficient performance. Consequently, further analysis will omit 2D intensity feature and CPMF-based detection

Varying support movement magnitude

Previous work [Y. Jing et al. 2024b] explored small support movements to assess the scale of geometric distortions required for detection. The associated models lacked sufficient geometric detail, as they did not consider surface roughness and inner cracks, making them less representative of the real arch geometry. To address this limitation, the same small support movements (8 cm and 12 cm movement in the x-direction) are re-evaluated using the present models, as shown in Figure 6.13a-d and i-l respectively. The results are also compared to 8 cm and 12 cm cases without the simulation of inner cracks (labelled as '8cm no inner' and '12cm no inner'), while preserving surface roughness for fair comparisons. These results

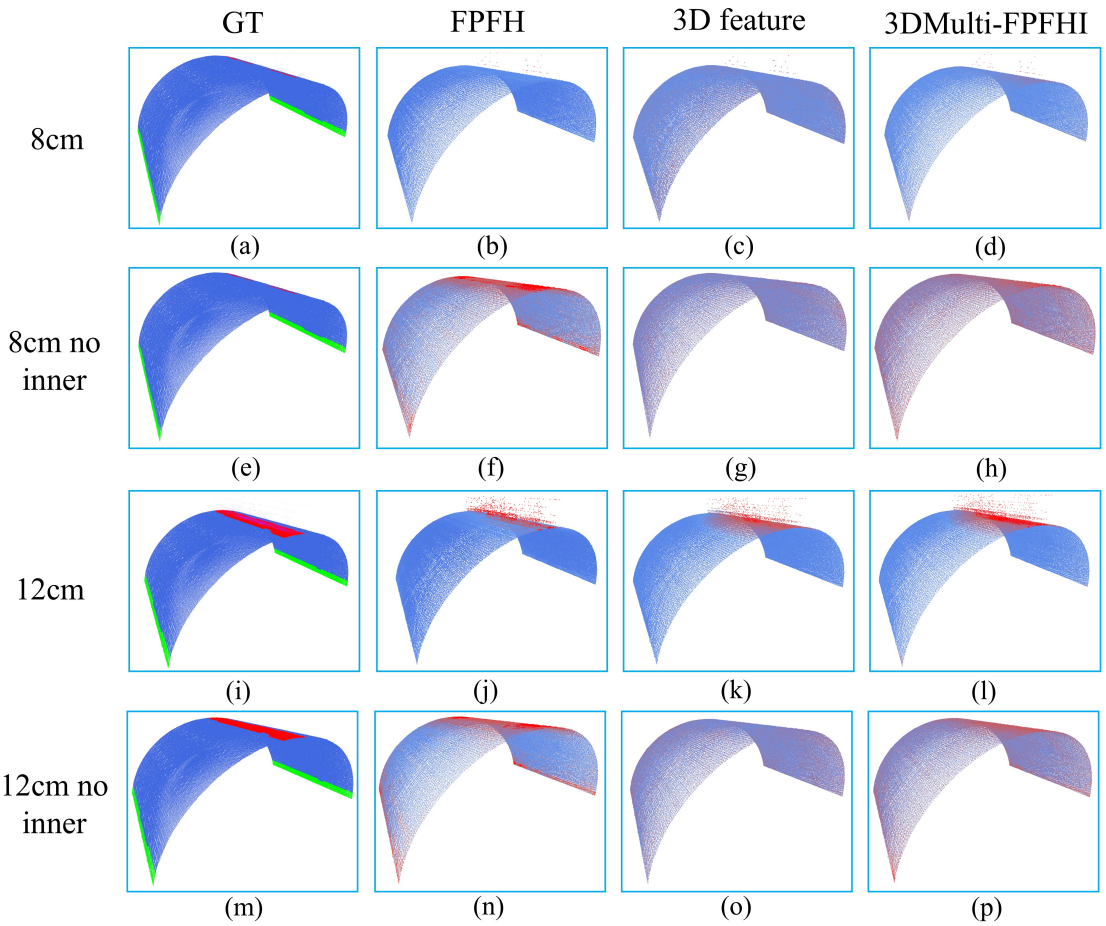


Figure 6.13: Anomaly detection results for the synthetic arch subjected to x support movement. (a)-(d) and (i)-(l) represent the ground truth (green and red colours represent extrados and intrados cracks respectively) and heatmaps for 8 cm and 12 cm support movements respectively. For investigating the influence of introducing new surfaces on intrados cracks, (e)-(h) and (m)-(p) also correspond to 8 cm and 12 cm support movements but without a definition of inner cracks.

are presented in Figure 6.13e-h and m-p respectively.

The 8 cm support movement case is first considered, where both intrados and extrados cracks begin to form (e.g., the intrados and extrados crack widths range from [0, 2 mm] and [0, 6 mm], respectively). As shown in Figure 6.13b, FPFH struggles to capture the cracks, as the geometric distortions are too small. Similar conclusions can be reached for the 3D intensity and multi-model features 6.13c-d). The poor performance is reflected in the low corresponding $F1$-*score* in Table 6.3 across all feature types. Moreover, the anomaly is less pronounced compared to the corresponding '8 cm no inner' case in Figure 6.13f-h. This can be attributed to two factors:

1. The density of 'inner crack' points is sparse in the 8 cm x support movement

Table 6.3: $F1$-*score* for varying support movement magnitudes, with and without defining inner cracks.

	FPFH	3D feature	3DMulti-FPFHI
8cm$(0.3 * s)$	0.012	0.036	0.034
8cm$(0.5 * s)$	0.012	0.005	0.013
8cm no inner$(0.3 * s)$	0.235	0.105	0.164
8cm no inner$(0.5 * s)$	0.103	0.006	0.029
12cm$(0.3 * s)$	0.257	0.511	0.536
12cm$(0.5 * s)$	0.137	0.173	0.176
12cm no inner$(0.3 * s)$	0.537	0.180	0.406
12cm no inner$(0.5 * s)$	0.592	0.007	0.037

case as crack development is in its early stages. As the inner cracks increase in the '12 cm' case (Figure 6.13j), intrados crack detection improves due to the higher density of 'inner crack' points.

2. Although sparse, 'inner cracks' exist for the 8cm x support movement case. The geometric distortions caused by these points are more significant than those caused by intrados and extrados cracks. After normalization (e.g., mapping *min_dists* to [0, 1]), the *min_dists* of the 'inner crack' points compress the distribution of other points, making them difficult to distinguish from the 'non-crack' points. This explains why the '8cm no inner' cloud indicates better detection performance than the '8cm' case.

Compared to the '12 cm no inner' case, the '12 cm' case shows more accurate localization of intrados cracks, as demonstrated in Figure 6.13j-l and n-p, respectively, and reflected in the $F1$-*score* in Table 6.3. In contrast, extrados cracks are not well captured for a similar reason to the second factor explained above for '8cm' support movement simulations.

It is also noteworthy that 3DMulti-FPFHI achieves the highest $F1$-*score* by leveraging intensity information in the '12 cm' case, as shown in Table 6.3. However, 3DMulti-FPFHI does not improve detection results when intensity anomalies are not introduced, as in the '12 cm no inner' case.

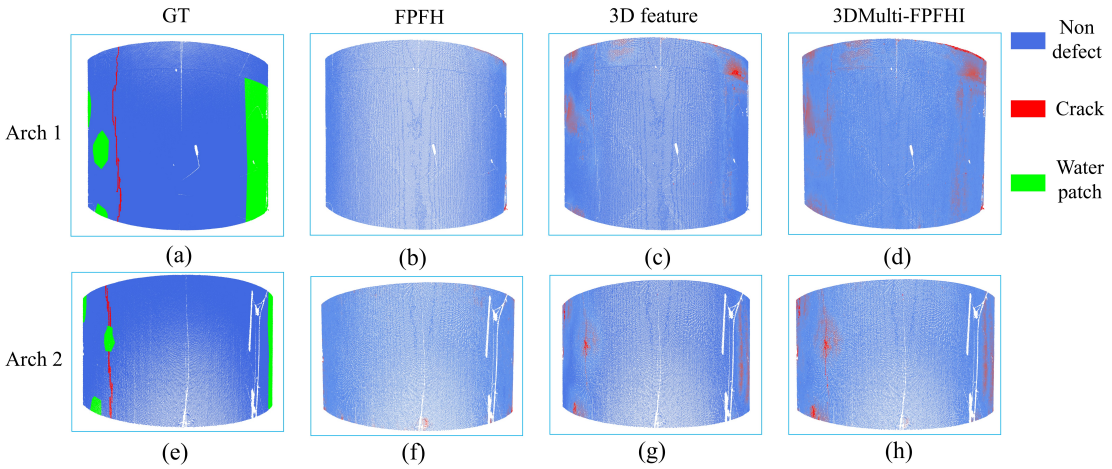


Figure 6.14: Anomaly detection results for Arch 1 and 2 of the real masonry arch point clouds. (a) and (e) represent the ground truth where blue, red and green colours represent non defect, crack and water patches correspondingly; (b)-(d) and (f)-(h) are anomaly score heatmaps obtained using different feature types, where heatmaps are visualized with the same anomaly scale as in Figure 6.11. Cracks and water patches are highlighted by red and blue insets respectively.

Table 6.4: $F1$-*score* for the real arch

	FPFH	3D feature	3DMulti-FPFHI
Arch $1(0.3 * s)$	0	0.029	0.040
Arch $1(0.5 * s)$	0	0.003	0.011
Arch $2(0.3 * s)$	0	0.176	0.289
Arch $2(0.5 * s)$	0	0.052	0.069

6.4.3 Results for the real arch

The proposed framework is applied to the real masonry bridge point cloud from London Bridge Station, as shown in Figure 6.14a-h. Figure 6.14b and f indicate that FPFH fails to capture the geometric distortion of Arch 1 and Arch 2, resulting in an $F1$-*score* of 0 (Table 6.4). Previous analysis of this bridge [Acikgoz et al. 2017b] revealed approximately 2 cm vertical support movement between 05.03.13 and 23.11.13 on one side of Arch E57; the resulting geometric distortion could not be detected by FPFH. This finding is consistent with previous work [Y. Jing et al. 2024b] and earlier results in this study which showed a failure to detect small support movements (Section 4.2.3). Moreover, the inner cracks which appeared were filled with mortar before data collection on 23.11.13, which had a negative influence on detection performance.

The cracks in Arch 1 and Arch 2 are partially captured using the 3D intensity feature, as shown in Figure 6.14c and g, respectively. The 3D intensity feature

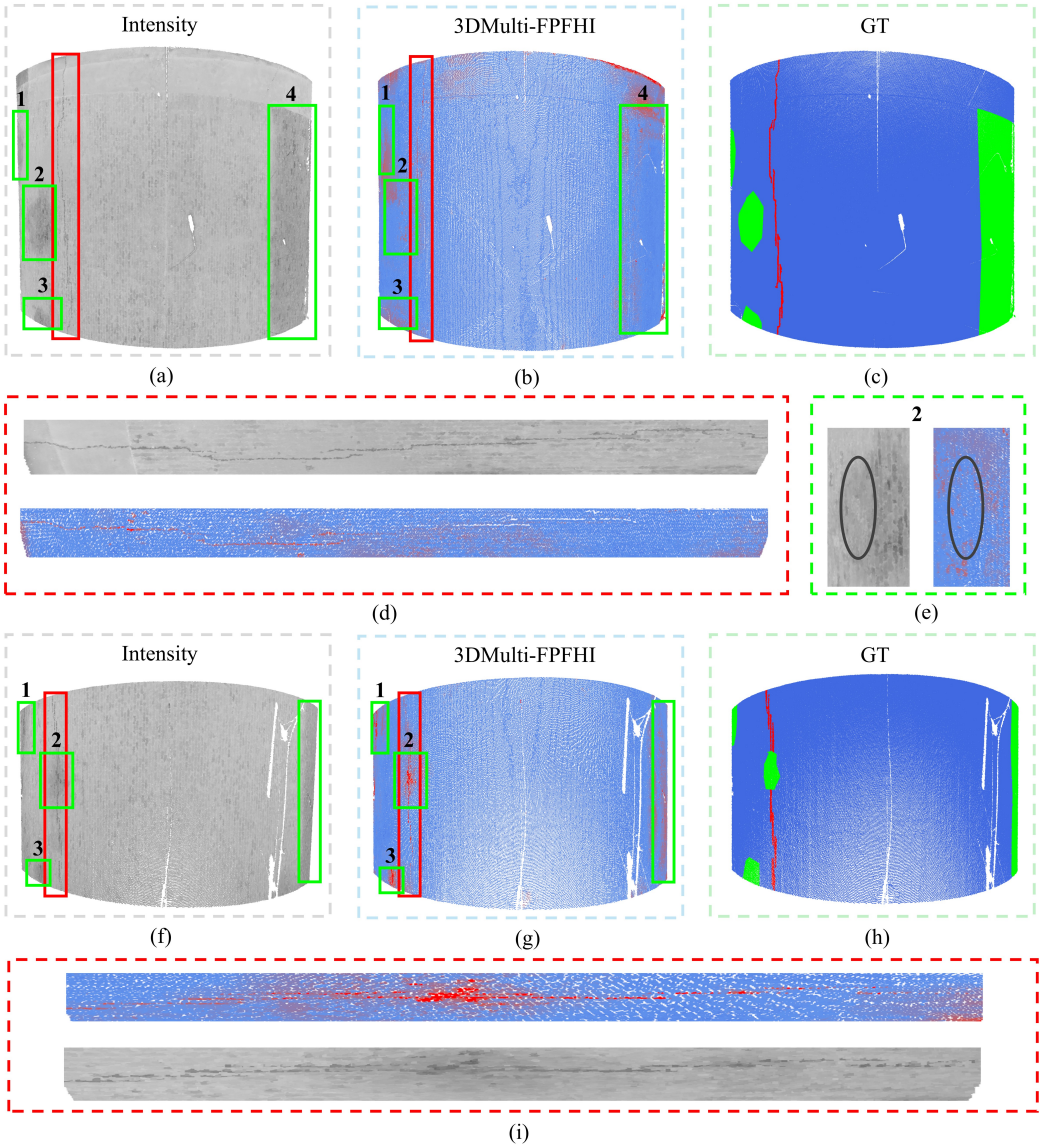
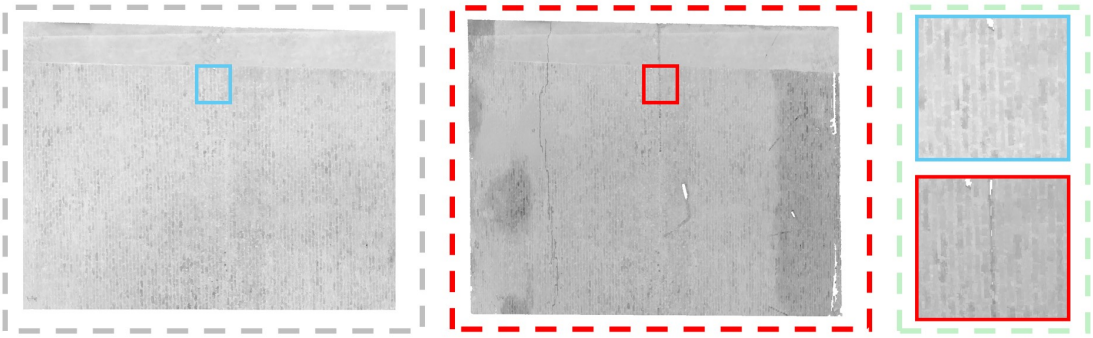


Figure 6.15: Visualization of 3DMulti-FPFHI in detecting cracks and water patches in Arch 1 and Arch 2. (a)-(c) illustrate the intensity map, 3DMulti-FPFHI detection results, and ground truth for Arch 1, where the crack is highlighted with a red inset, and water patches are labelled from **1** to **4** as shown in green insets. (d) provides a close-up view of the crack in Arch 1, shown in both the intensity map and 3DMulti-FPFHI results. (e) show a close-up view of the water patch **2** in Arch 1, where the black oval indicates the misidentified region; (f)-(h) present the corresponding results for Arch 2, with a detailed view of the crack provided in (i).

provides additional information for localizing cracks and water patches, as both defect types introduce intensity anomalies into the point clouds. The 3DMulti-FPFHI exhibits consistent detection results with the 3D intensity feature as shown in Figure 6.14d and h, even though FPFH fails to capture any geometric distortions in Arch 1 and Arch 2. Moreover, the $F1$-*score* for 3DMulti-FPFHI is higher than that of the 3D intensity feature, as shown in Table 6.4.



(a) Arch 1 on 05.03.13 (b) Arch 1 on 23.11.13 (c)

Figure 6.16: Visualization of intensity map discrepancies in Arch 1. (a) and (b) show the intensity maps of Arch 1 captured on two different dates, with blue and red insets highlighting example regions. (c) illustrates the mismatches in intensity between the two maps.

Compared to the detection results from the synthetic arch dataset, the 3DMulti-FPFHI shows a much lower $F1$-*score* in Arch 1 and Arch 2, which can be attributed to low *Precision*, as shown in Appendix B (6.4.4). However, metrics alone cannot fairly evaluate the performance of *PatchCore* in the real arch dataset due to the diffusion of crack features during detection. It is practically more important to note that the algorithm correctly indicates the locations of cracks and water patches, as further demonstrated in Figure 6.15a-i. The 3DMulti-FPFHI successfully captures cracks in Arch 1 and Arch 2, as highlighted in the red insets in Figure 6.15b and g, where zoomed views of the intensity maps and corresponding detection results accurately delineate the cracks. For water patches in Arch 1, the 3DMulti-FPFHI achieves better detection results for smaller patches (e.g., water patches **1** and **3**), while providing only vague delineations of the boundaries of larger patches, i.e., water patches **2** (the zoomed view is provided in Figure 6.15e) and **4**. A similar trend is observed in Arch 2 (Figure 6.15f-i), where smaller water patches (**1**-**3**) are better detected compared to the water patch **4**.

The limitations in detecting large water patches can be attributed to the design of the 3D intensity feature, which utilizes absolute relative values between the centre point and its neighbours. This design is to satisfy the underlying assumption of *PatchCore*, namely that $\mathcal{M}^*_{sub}$ can be paired with 3DMulti-FPFHI for 'non defect' points. This assumption holds for synthetic data, where normal and test point clouds are generated under consistent conditions (e.g., surface materials, weather, water leakage, vegetation, etc.), ensuring invariant reflectivity of 'non defect' points

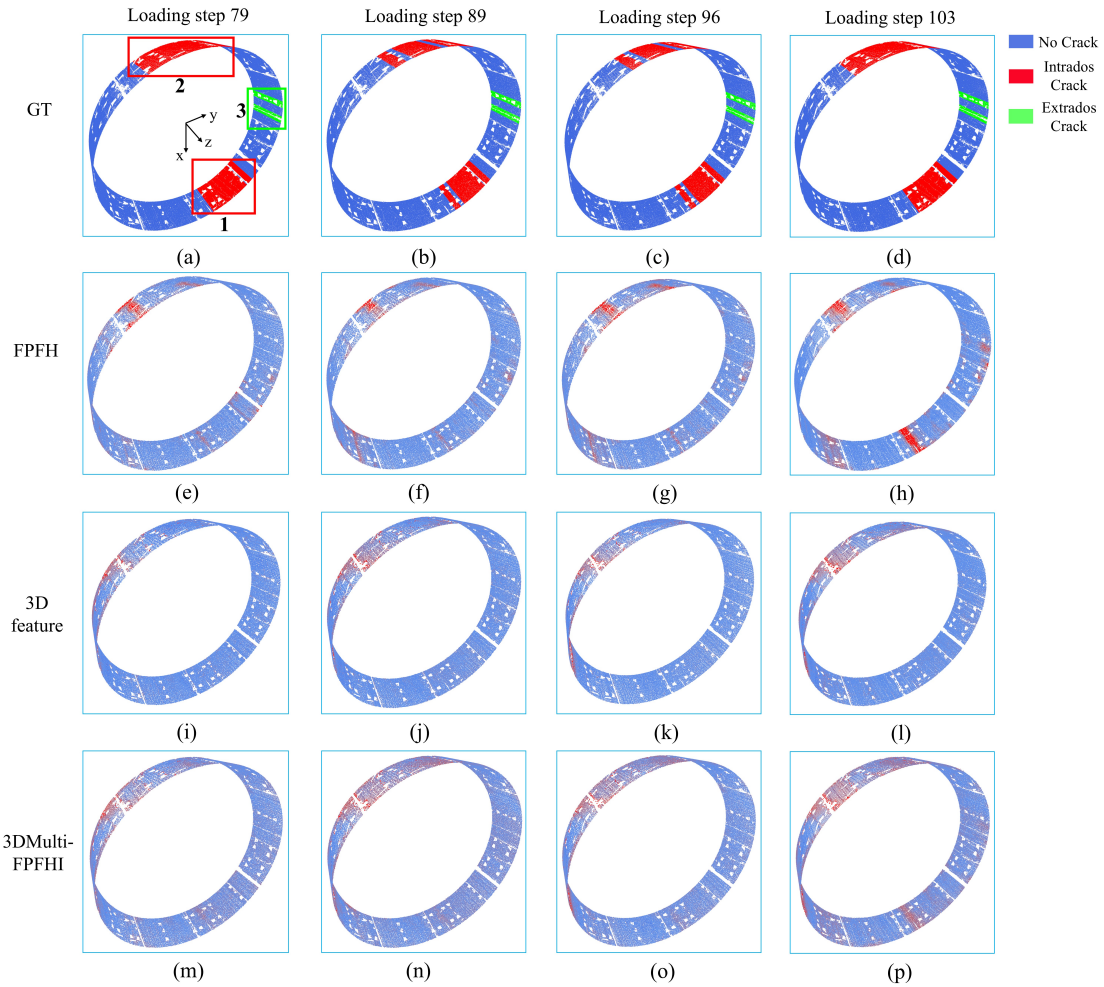


Figure 6.17: Anomaly detection results corresponding to four different loading steps on the real tunnel lining. (a)-(d) are the ground truth, where intrados and extrados cracks are represented by red and green and indexed **1-3**, respectively. (e)-(h), (i)-(l), (m)-(p) are heatmaps, where anomaly scale is consistent to Figure 6.11, representing the anomaly score of each point for the different feature types.

across normal and test data. However, in real-world scenarios, these conditions vary significantly over time, impacting the consistency of intensity values for 'non defect' regions. Figure 6.16a-c highlights the intensity mismatches in Arch 1, where the intensity values on 23.11.13 differ significantly from those on 05.03.13 for 'non-crack' points in selected regions.

Our 3D intensity feature mitigates the inconsistencies in intensity values caused by environmental effects, by utilizing absolute relative values of intensity to construct the feature. However, this approach becomes less effective when intensity values are uniformly distributed, as the absolute relative differences between the centre point and its neighbours become minimal. Consequently, the 3D intensity feature fails to differentiate between defects and 'non-crack' regions, as demonstrated in

Table 6.5: $F1$-*score* for the real tunnel

	FPFH	3D feature	3DMulti-FPFHI
Loading step $79(0.3 * s)$	0.057	0.025	0.025
Loading step $79(0.5 * s)$	0.001	0.006	0.006
Loading step $89(0.3 * s)$	0.037	0.044	0.162
Loading step $89(0.5 * s)$	0.001	0.011	0.022
Loading step $96(0.3 * s)$	0.088	0.030	0.093
Loading step $96(0.5 * s)$	0.009	0.007	0.011
Loading step $103(0.3 * s)$	0.116	0.052	0.206
Loading step $103(0.5 * s)$	0.043	0.015	0.026

the detection of water patch **2** in Arch 1, highlighted by the black oval in Figure 6.15e. This limitation explains why the 3D intensity feature primarily detects defects along the boundaries of large water patches, where the intensity gradient is highest, while failing to identify anomalies within the patch interior.

Despite this limitation, it is important to emphasize that incorporating the 3D intensity feature enables the detection of cracks and water patches—an achievement not possible using geometric features alone. Future studies could address this limitation by employing more robust 3D feature extraction techniques.

6.4.4 Results for the experimental tunnel model

The proposed framework is validated on tunnel experimental model point clouds corresponding to loading steps 79, 89, 96, and 103. Loading steps 20 and 25 are omitted due to insufficient crack development to register as anomalies. To investigate the sensitivity of our algorithm, anomaly detection is performed on these steps, with ground truth and results shown in Figure 6.17a-d and e-p, respectively. The undeformed point cloud at loading step 0 is used as the 'normal' data.

Intrados and extrados cracks are visualized and indexed as **1-3** in Figure 6.17a-d (consistent with Figure 6.10b).

Anomaly detection results using FPFH are shown in Figure 6.17e-h, where cracks are identified using only geometric distortions. FPFH shows low $F1$-*score* across all loading steps (Table 6.5). As in Section 4.2, the $F1$-*score* inadequately reflects the performance of *PatchCore* due to inconsistencies between predicted crack regions and the ground truth (caused by mesh-based label mapping).

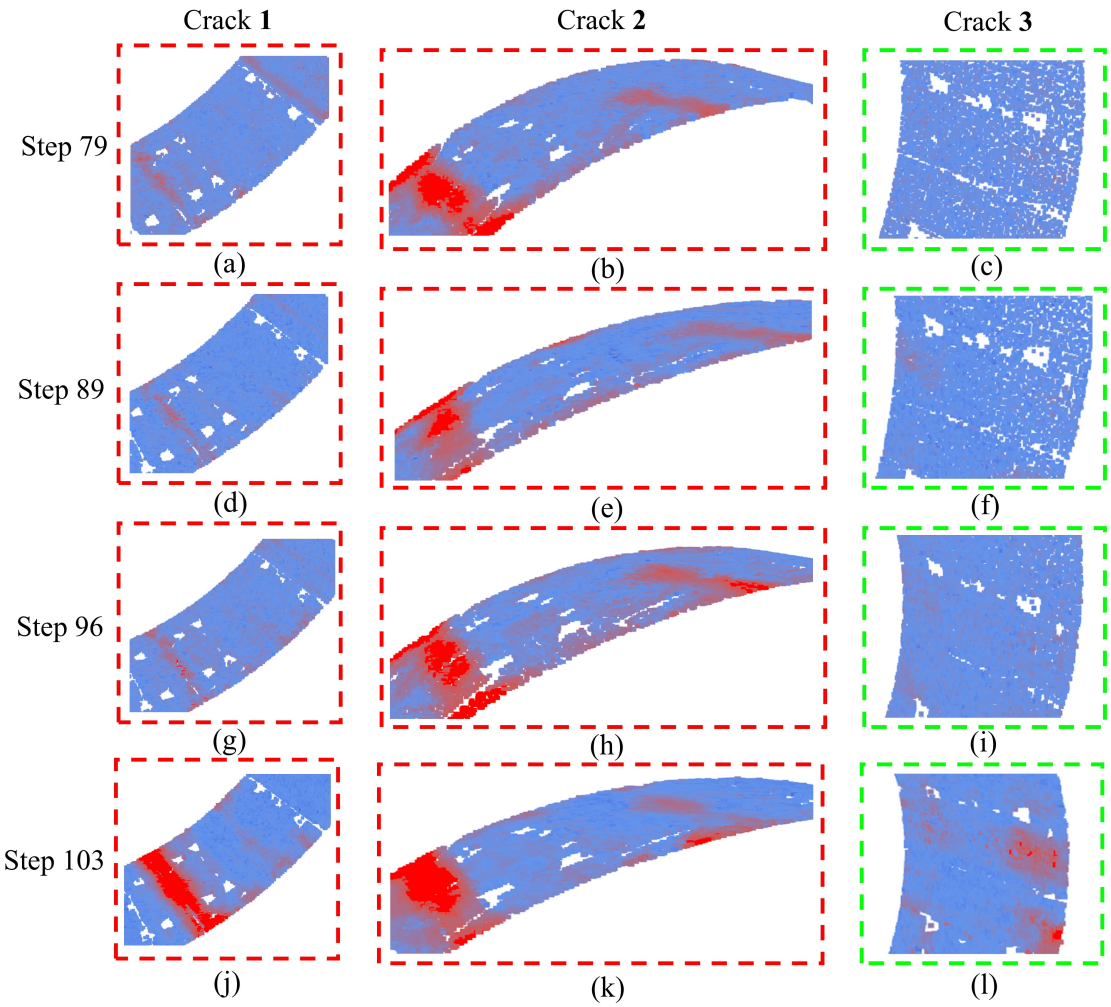


Figure 6.18: Visualization of zoomed crack detection results using FPFH. (a)-(c), (d)-(f), (g)-(i), and (j)-(l) present zoomed views of cracks **1-3** (as defined in Figure 6.17a) at loading steps 79, 89, 96, and 103, respectively.

To demonstrate the feasibility of *PatchCore*, zoomed views of cracks **1-3** are provided for FPFH detection results at different loading steps, as shown in Figure 6.18a-l. Regions corresponding to two intrados cracks **1** and **2** can be localized by leveraging only geometric information as demonstrated in Figure 6.18a, d, g, j, b, e, h, and k. The anomaly scales at loading steps 79, 89, and 96 show no significant improvements due to small variations of geometry during crack development. However, a noticeable improvement in anomaly scales is observed at loading step 103, where severe damage is introduced at the last step.

The extrados crack **3** initially shows faint signals at loading step 89 and gradually becomes more visible by step 103 (as shown in Figure 6.18c, f, i, and l). Loading step 103 highlights geometric distortions introduced by the crack **3**. Extrados tunnel cracks are often missed in defect detection tasks (e.g., using image-based

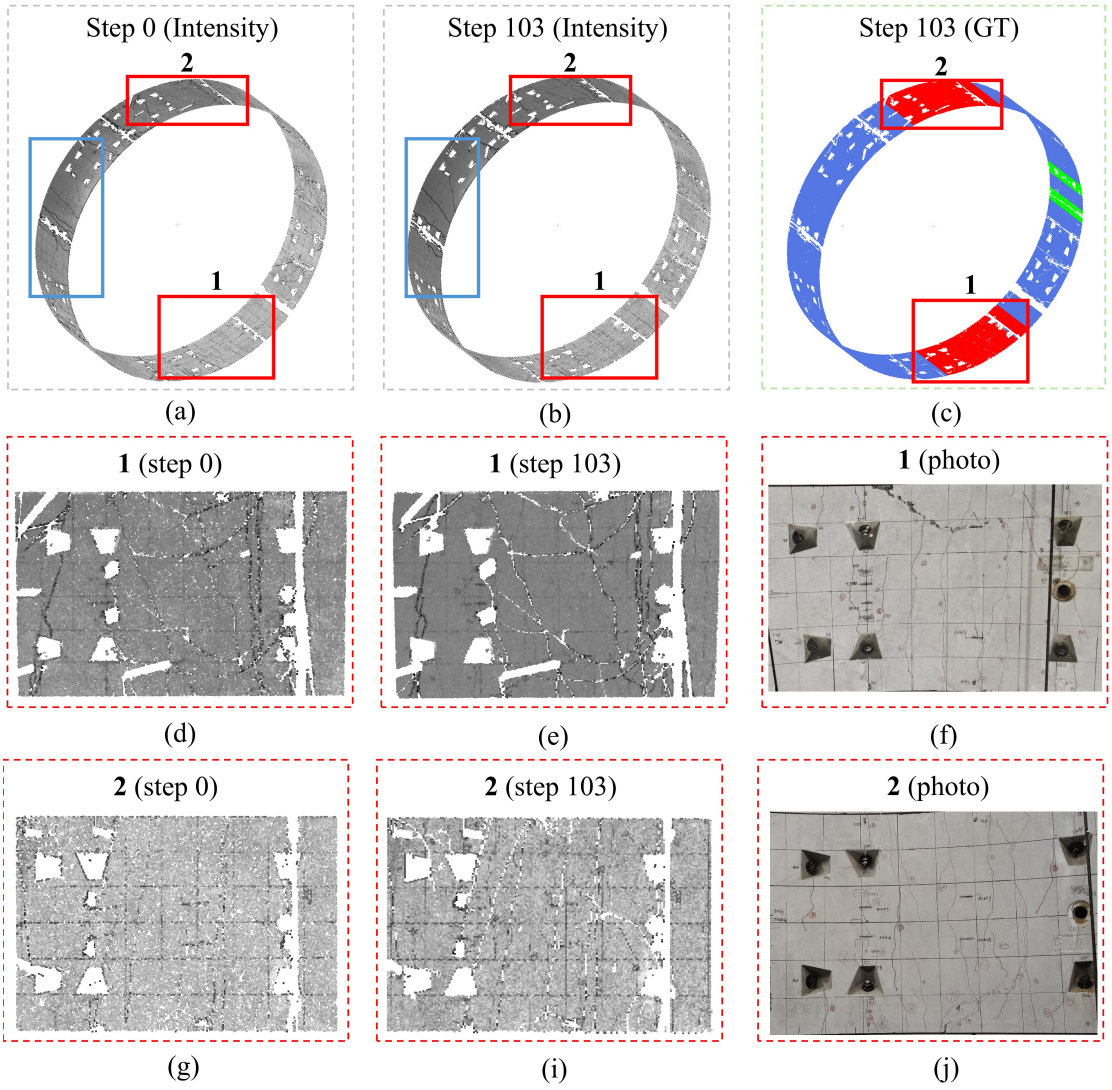


Figure 6.19: Limitations of intensity information in detecting cracks in the real tunnel point cloud. (a) and (b) represent the tunnel in loading steps 0 and 103 by transferring intensity information into the greyscale. Blue boxes indicate inconsistency of the intensity between step 0 and 103; (c) shows the ground truth of cracks **1** and **2**; (d) and (e) show insets for crack **1** for step 0 and 103; (f) shows the photo of crack **1**; and (g)-(j) shows the corresponding insets and photo for crack **2**.

algorithms) due to insufficient surface features. In contrast, FPFH provides a potential method for identifying extrados cracks by examining the geometric information indirectly.

In contrast, the 3D intensity feature fails to provide reliable detection results for the experimental tunnel dataset, resulting in the lowest $F1$-*score* in most cases as shown in Table 6.5. To investigate the cause, intensity maps (greyscale) at loading steps 0 and 103 are visualized in Figure 6.19a-b. Ground truth in Figure 6.19c includes cracks **1** and **2**, which are visible to the laser scanner. Insets in Figure 6.19d and e show zoomed views of regions corresponding to crack **1** at steps 0 and

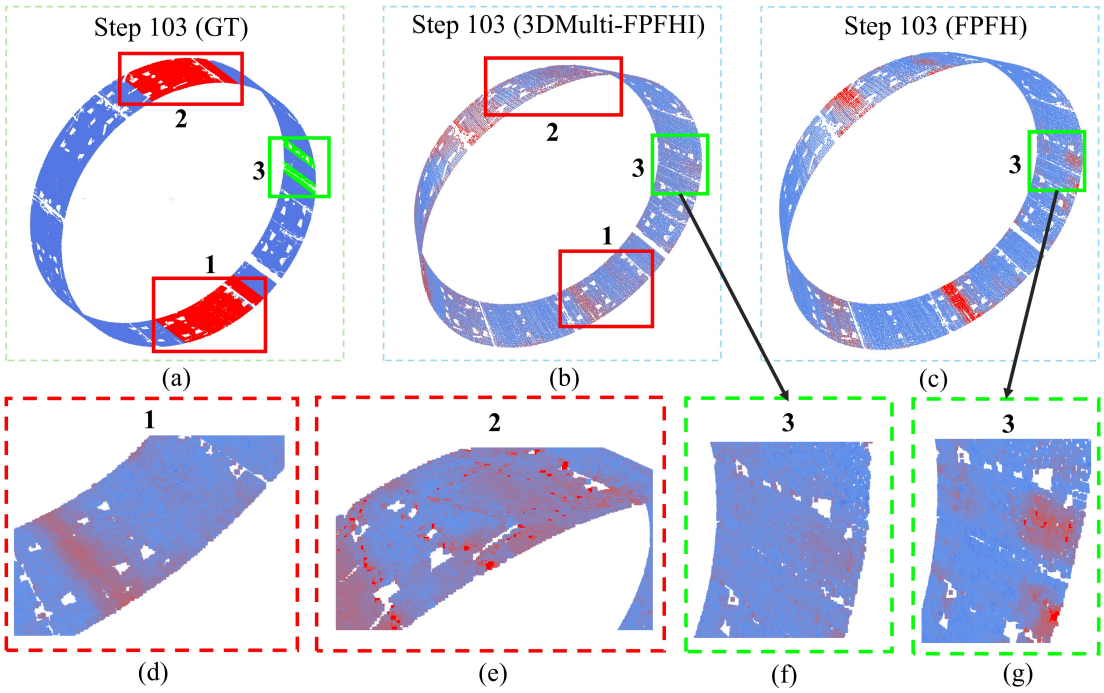


Figure 6.20: Comparison of detection results using FPFH and 3DMulti-FPFHI at loading step 103. (a), (b), and (c) depict the ground truth, FPFH detection result, and 3DMulti-FPFHI detection result, respectively. (d), (e), and (f) present zoomed views of cracks **1**-**3** for 3DMulti-FPFHI, while (g) shows the zoomed view of crack **3** for FPFH.

103. The photo of crack **1** at step 103 is provided in Figure 6.19f.

By comparing Figure 6.19d and e, no distinct intensity differentiations can be observed that indicate the presence of cracks or any different defect types seen in Figure 6.19f. The visible black lines represent cables, not cracks, suggesting that the intensity information failed to distinguish cracks from nearby normal regions, as seen in real arch dataset results. The primary reason for this limitation is that the scanning resolution is lower than the size of the cracks, resulting in edge effects [Stałowska et al. 2022]. Similar edge effects are also evident in crack **2**, as shown in Figure 6.19g-j. Meanwhile, the 3D intensity feature introduces detection noise into regions indicated by blue boxes in Figure 6.19a-b.

Results corresponding to 3DMulti-FPFHI show a similar tendency as FPFH, as 3DMulti-FPFHI can mitigate intensity inconsistencies introduced by the 3D intensity feature (Section 4.3). To demonstrate this, Figure 6.20a-g compares FPFH and 3DMulti-FPFHI detection results at loading step 103. The detection of intrados cracks **1** and **2** by the 3DMulti-FPFHI are only slightly compromised compared to FPFH. However, the fusion of the 3D intensity feature introduces noise

and diffuses defects (Figure 6.14i-l). The introduction of the 3D intensity feature compromises FPFH performance in the extrados crack **3** due to smaller geometric distortions as shown in Figure 6.20f and g, despite the $F1$-*score* (see Table 6.5) being preserved. Nonetheless, the 3DMulti-FPFHI demonstrates reasonable performance across all loading steps in capturing intrados cracks, maintaining a degree of robustness despite intensity information introducing significant noise.

In an ideal case, *min_vals* for 3D intensity feature, calculated by pairing $\boldsymbol{L}_{3DI}$ and 3D intensity feature in $\mathcal{M}^{*}_{sub}$, should be approximately zero if not detecting defects. For example, as demonstrated in Figure 6.14b and h, FPFH failed to detect any anomalies in the real masonry arch, which consequently did not degrade the performance of the 3DMulti-FPHFI.

Conclusions

This paper presents a novel 3D multimodal feature, 3DMulti-FPFHI, which fuses two modalities, i.e., geometric and intensity information from point clouds, for use with the *PatchCore* anomaly detection algorithm to identify defects from different types of infrastructure assets. To effectively leverage intensity information, we propose a 3D intensity feature that embeds intensity values by computing the absolute relative differences between the centre point and its neighbours. The developed framework is then applied to detect defects in synthetic and real point cloud datasets. The proposed method was benchmarked against standard FPFH and the SOTA multimodal anomaly detection method CPMF [Cao et al. 2023], which uses a pre-trained ResNet50d to extract 2D features from point cloud projections. The influence of the 3D intensity feature was also analyzed independently to assess the contribution of this modality to anomaly detection. A key advantage of the proposed framework over DL methods is that the method does not rely on extensive training to identify defects. Instead, it only requires two pairs of point clouds.

We first modified a previous synthetic masonry arch dataset by incorporating surface roughness and the geometry of intrados cracks to better reflect real-world conditions. The performance of the anomaly detection algorithm was re-evaluated using various feature types, demonstrating that the inclusion of 3D intensity in-

formation significantly improves the detection of intrados cracks. Performance discrepancies among different feature types were also analysed using the distribution of anomaly scores (ranging from 0 to 1, representing a low-to-high anomaly scale) and comparing their detection accuracy for intrados and extrados cracks. The analysis demonstrated how intensity-related anomalies assist detection quantitatively and explained why the 2D intensity feature and CPMF fail to provide accurate results.

The framework was further validated on real masonry bridge point cloud data from London Bridge Station, collected on two different dates (05.03.13 and 23.11.13). A longitudinal crack was identified in the 23.11.13 dataset after maintenance work, with 3DMulti-FPFHI demonstrating accurate detection results. By integrating the 3D intensity feature, 3DMulti-FPFHI successfully captured water patches that introduced intensity anomalies, expanding the ability of *PatchCore* to detect non-geometric defects. Furthermore, the 3D intensity feature effectively mitigated inconsistencies in regions without defects caused by environmental variations between the two datasets, ensuring precise localization of both cracks and water patches.

To evaluate the generalization capability of 3DMulti-FPFHI, we applied the framework to a full-scale experimental tunnel dataset subjected to increasing loading steps. Using an undeformed point cloud (loading step 0) as the reference, the algorithm successfully identified intrados cracks. FPFH also detected extrados cracks, which are often overlooked due to a lack of associated surface features. While FPFH achieved better results in this case as the 3D intensity feature failed to provide useful information, 3DMulti-FPFHI still provided satisfactory performance despite the intensity noise introduced by the experimental equipment.

Through evaluations across different datasets, we demonstrate that 3DMulti-FPFHI offers a robust solution by aggregating geometric and intensity modalities, particularly in real-world scenarios with environmental inconsistencies (e.g., lighting, vegetation, moisture, etc.) which challenge the underlying assumptions of anomaly detection methods. Additionally, its potential for generalization to various types of infrastructure has been established. The code and related point cloud dataset will be made available at https://github.com/Jingyixiong/infra_3dpseudo_inspection to encourage reproducibility and further research.

Limitations

The 3D multimodal feature failed to capture small cracks in the synthetic dataset (e.g., intrados and extrados cracks with maximum values of 2mm and 6mm respectively) and the real tunnel (e.g., external cracks in loading steps 79 and 89). From a geometric perspective, small cracks cause subtle distortions that FPFH cannot capture. Additionally, the intrinsic sparsity of point clouds limits their resolution(edge effects) compared to 2D images, making it harder to identify fine details. Therefore, the 3D intensity feature is affected not only by noise but also by insufficient resolution, reducing their ability to detect small cracks accurately.

Meanwhile, the current anomaly detection framework fails to provide an optimal boundary for distinguishing 'non-defect' and 'defect' points. Unsupervised learning methods such as DBSCAN or K-clustering require frequent hyperparameter calibration due to variations in anomaly score distributions across datasets, limiting their generalizability.

Future work could focus on developing more sensitive geometric indicators to detect subtle deformations, potentially leveraging learning-based approaches (e.g., H. Deng et al. [2018]). Additionally, incorporating methods like one-class support vector machines into *PatchCore* could help derive optimal boundaries and enhance detection performance.

Acknowledgement

The authors acknowledge Fan Zhang (Tongji University) for sharing the image and crack data for the real tunnel.

Appendix A. Supplementary images and metrics for the synthetic arch dataset

Appendix A.1 Visualization of 'inner crack' points for support movements of z and a combination of x rotation and z translation

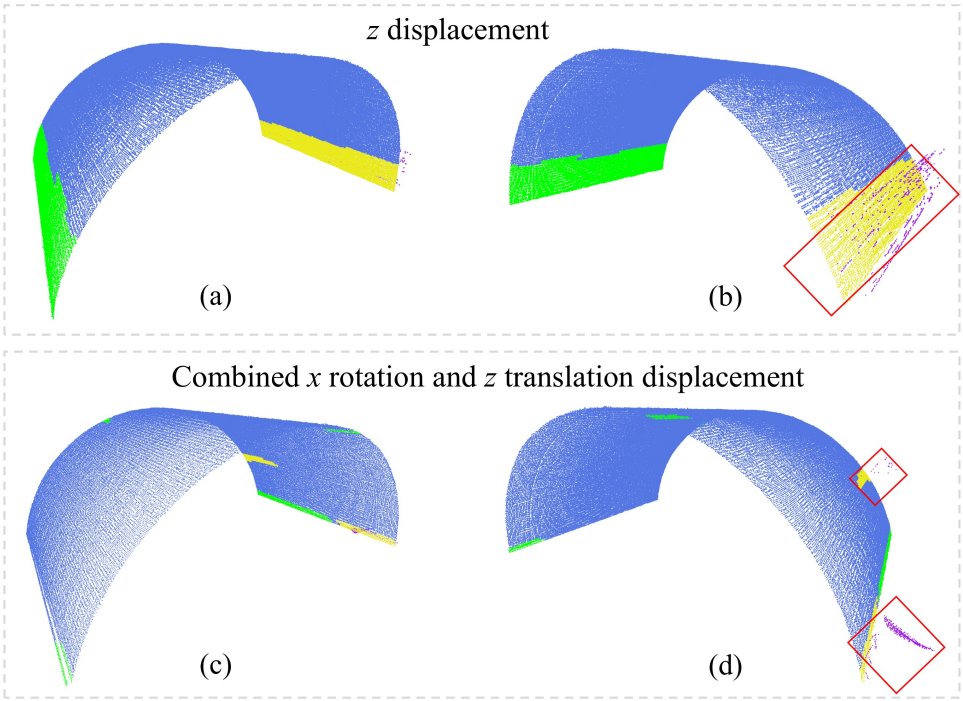


Figure 6.21: Visualization of 'inner crack' points. (a) and (b) show different views of the ground truth for the z support movement, while (c) and (d) depict the combination of x rotation and z translation support movement. 'Inner crack' points are highlighted in red insets.

Appendix A.2. Metrics for 4 different support movement cases

Table 6.6: *Recall* for the 4 different support movement cases of the masonry arch with the threshold $0.3 * s$ and $0.5 * s$.

	FPFH	3D feature	3DMulti-FPFHI	2D feature	CPMF
$x(0.3 * s)$	0.999	0.512	0.627	0.274	0.274
$x(0.5 * s)$	0.999	0.668	0.898	0.287	0.455
$z(0.3 * s)$	0.999	0.875	0.992	0.175	0.176
$z(0.5 * s)$	0.999	0.991	0.999	0.558	0.670
$x\ +\ z(0.3 * s)$	0.998	0.584	0.650	0.188	0.188
$x\ +\ z(0.5 * s)$	0.999	0.827	0.916	0.199	0.289
$x\,rot + z\,trans(0.3 * s)$	0.472	0.967	0.484	0.032	0.038
$x\,rot + z\,trans(0.5 * s)$	0.574	0.999	0.603	0.516	0.515

Table 6.7: *Precision* for the 4 different support movement cases of the masonry arch with the threshold $0.3 * s$ and $0.5 * s$.

	FPFH	3D feature	3DMulti-FPFHI	2D feature	CPMF
$x(0.3 * s)$	0.575	0.712	0.726	0.999	0.999
$x(0.5 * s)$	0.136	0.542	0.538	0.982	0.934
$z(0.3 * s)$	0.064	0.306	0.339	0.999	0.999
$z(0.5 * s)$	0.047	0.051	0.054	0.431	0.440
$x\ +\ z(0.3 * s)$	0.186	0.708	0.710	0.999	0.999
$x\ +\ z(0.5 * s)$	0.101	0.577	0.491	0.959	0.930
$x\,rot + z\,trans(0.3 * s)$	0.231	0.120	0.250	0.683	0.642
$x\,rot + z\,trans(0.5 * s)$	0.205	0.100	0.216	0.516	0.515

Appendix A.3 Metrics for varying support movement magnitude

Table 6.8: *Recall* for the varying support movement magnitude with and without defining inner cracks.

	FPFH	3D feature	3DMulti-FPFHI
8cm$(0.3 * s)$	0.999	0.103	0.686
8cm$(0.5 * s)$	0.999	0.274	0.999
8cm no inner$(0.3 * s)$	0.142	0.074	0.094
8cm no inner$(0.5 * s)$	0.084	0.109	0.457
12cm$(0.3 * s)$	0.994	0.604	0.730
12cm$(0.5 * s)$	0.999	0.999	0.998
12cm no inner$(0.3 * s)$	0.393	0.170	0.271
12cm no inner$(0.5 * s)$	0.717	0.234	0.766

Table 6.9: *Precision* for the varying support movement magnitude with and without defining inner cracks.

	FPFH	3D feature	3DMulti-FPFHI
8cm$(0.3 * s)$	0.006	0.022	0.017
8cm$(0.5 * s)$	0.006	0.002	0.006
8cm no inner$(0.3 * s)$	0.695	0.181	0.620
8cm no inner$(0.5 * s)$	0.132	0.003	0.015
12cm$(0.3 * s)$	0.148	0.443	0.423
12cm$(0.5 * s)$	0.074	0.094	0.096
12cm no inner$(0.3 * s)$	0.846	0.190	0.804
12cm no inner$(0.5 * s)$	0.504	0.004	0.019

Appendix B. Metrics for the real arch and tunnel

Appendix B.1 Real arch

Table 6.10: *Recall* for the real arch

	FPFH	3D feature	3DMulti-FPFHI
Arch $1(0.3 * s)$	0	0.173	0.221
Arch $1(0.5 * s)$	0	0.162	0.288
Arch $2(0.3 * s)$	0	0.581	0.594
Arch $2(0.5 * s)$	0	0.899	0.813

Table 6.11: *Precision* for the real arch

	FPFH	3D feature	3DMulti-FPFHI
Arch $1(0.3 * s)$	0	0.016	0.040
Arch $1(0.5 * s)$	0	0.002	0.005
Arch $2(0.3 * s)$	0	0.104	0.191
Arch $2(0.5 * s)$	0	0.027	0.036

Appendix B.2 Real tunnel

Table 6.12: *Recall* for the real tunnel

	FPFH	3D feature	3DMulti-FPFHI
Loading step $79(0.3 * s)$	0.144	0.072	0.081
Loading step $79(0.5 * s)$	0.005	0.086	0.089
Loading step $89(0.3 * s)$	0.123	0.136	0.230
Loading step $89(0.5 * s)$	0.003	0.146	0.206
Loading step $96(0.3 * s)$	0.226	0.099	0.155
Loading step $96(0.5 * s)$	0.094	0.086	0.105
Loading step $103(0.3 * s)$	0.260	0.160	0.267
Loading step $103(0.5 * s)$	0.226	0.193	0.253

Table 6.13: *Precision* for the real tunnel

	FPFH	3D feature	3DMulti-FPFHI
Loading step $79(0.3 * s)$	0.035	0.015	0.039
Loading step $79(0.5 * s)$	0.001	0.003	0.004
Loading step $89(0.3 * s)$	0.022	0.026	0.124
Loading step $89(0.5 * s)$	0.001	0.006	0.006
Loading step $96(0.3 * s)$	0.055	0.018	0.067
Loading step $96(0.5 * s)$	0.005	0.004	0.006
Loading step $103(0.3 * s)$	0.074	0.031	0.168
Loading step $103(0.5 * s)$	0.024	0.008	0.014

Statement of Authorship in Chapter 6

Paper title	A 3D Multimodal Feature for Infrastructure Anomaly Detection
Authors	Yixiong Jing, Wei Lin, Brian Sheil, Sinan Acikgoz
Publication status	Submitted
Publication details	Under review of ISPRS Journal of Photogrammetry and Remote Sensing

Student Confirmation

Student name	Yixiong Jing	
Contribution to the paper	First-author contribution: • conception of research ideas • design and implementation of models • writing and presentation of the paper	
Signature and Date		Jan. 15th 2025

Supervisor Confirmation

By signing the Statement of Authorship, the supervisor is certifying that the candidate made a substantial contribution to the publication, and that the description above is accurate.

Supervisor name	Prof. Sinan Acikgoz	
Supervisor comments		
Signature and Date		Jan. 15th 2025

Chapter 7

Conclusions, future Work, and dataset contributions

7.1 Conclusions

Periodic inspections of masonry bridges are essential for ensuring proactive maintenance and safeguarding their structural integrity. Traditional inspection methods rely on manual and visual evaluations, which are time-consuming and labour-intensive. While recent advancements in DL have introduced automation to masonry bridge inspections, these efforts have primarily focused on defect detection in 2D images.

Unlike 2D images, 3D point clouds capture the complex geometry of masonry bridges, enabling the extraction of critical geometric parameters and deformation measurements for robust and comprehensive defect inspections. Despite these advantages, the automation in defect detection on masonry bridge point clouds remains largely unexplored, which can be attributed to the scarcity of well-labelled datasets and the lack of specialized algorithms for processing large-scale point clouds.

This research aims to develop an innovative and general framework to address the urgent need for automating defect inspections of masonry bridge point clouds. By leveraging DL and computer vision technologies, the framework focuses on efficient data processing to enable accurate and scalable inspections.

The thesis begins by developing DL models for segmenting large-scale masonry bridge point clouds, essential for facilitating structural assessments and defect detection. Specifically, Chapter 3 introduced a novel DL model named BridgeNet to perform semantic segmentation on large-scale point clouds. Inspired by RandLA-Net [Hu et al. 2020] and FG-Net [Kai Liu et al. 2020], the architecture was modified to process large point cloud input efficiently.

To address data scarcity, a synthetic masonry bridge point cloud dataset was developed, and BridgeNet was trained exclusively on this synthetic data. The trained model was evaluated on seven real masonry bridge point clouds, achieving SOTA performance compared to PointNet++ [Qi et al. 2017b] and RandLA-Net. This result highlights the potential of synthetic datasets as a substitute for real data in training DL models.

Furthermore, instance-level segmentation was achieved using the unsupervised DBSCAN algorithm to differentiate individual structural components with the same semantic label. This enabled the accurate extraction of geometric parameters (e.g., radius and generatrix direction of arches), demonstrating the practical utility of segmented point clouds in providing critical geometric information.

While BridgeNet achieved accurate segmentation results for large-scale masonry bridge point clouds, its performance on imbalanced classes (e.g., rare components such as holes in bridge piers) remained suboptimal, leading to higher error rates. To address this issue, Chapter 4 proposed BridgeNetv2, which integrates two Transformer-based modules, i.e., Local Geo-Transformer and Global Geo-Transformer which are inspired by the vanilla Transformer architecture [Vaswani et al. 2017], into the original BridgeNet framework.

The Local Geo-Transformer is inserted between the neighbourhood feature extractors of BridgeNet to alleviate information loss caused by the point cloud downsampling using an attention mechanism. To maintain the lightweight characteristic of BridgeNet, the attention computation is computed on the local k nearest neighbours rather than the entire point cloud. The Global Geo-Transformer, on the other hand, captures long-range dependencies by exploiting relationships between features.

Extensive experiments on synthetic and real masonry bridge datasets demonstrate

that BridgeNetv2 significantly improves segmentation accuracy, particularly for imbalanced classes. Furthermore, it retains the computational efficiency of BridgeNet, ensuring scalability for processing large-scale point clouds while enhancing feature representation and segmentation performance.

Building on the outputs from the segmentation, Chapter 5 adapts a 2D anomaly detection framework called *PatchCore* for detecting cracks associated with geometric distortions in structural component point clouds. *PatchCore* employs the handcrafted FPFH descriptor [Rusu et al. 2009] to extract local geometric features from point clouds. These features are stored in a memory bank constructed from normal point clouds (e.g., point clouds without cracks). During testing, *PatchCore* evaluates FPFH features of each test point by calculating the minimum Euclidean distance to FPFH features in the memory bank. Points with high deviations can indicate potential cracks, enabling efficient and localized defect detection. This approach requires only point clouds of structures before and after deformation to compute anomalies, enabling a scalable and general anomaly detection framework.

To address the lack of suitable validation data, a synthetic point cloud generation pipeline was developed. This pipeline integrates FEM simulations to model the mechanical response of masonry structures under various loading conditions. The FEM output is then reconstructed using 3D graphical modelling tools to generate physically realistic geometric distortions and crack patterns. This process also accounts for critical aspects such as surface roughness and laser scan measurement geometries and errors. By automating the parametric generation of datasets, this pipeline helps address the scarcity of 3D point cloud data, facilitating data-driven algorithm development.

By leveraging the synthetic data generation pipeline, point clouds of masonry arches and arch bridges are generated for validating the anomaly detection framework. Most evaluations were conducted on synthetic datasets derived from meso-scale FEM models of masonry arches. To explore its practical applicability, *PatchCore* was also applied to a synthetic complete masonry arch bridge model. To ensure computational scalability (particularly when processing dense point clouds), voxelization is applied on raw point clouds as a preprocessing step for downsampling and uniform density. Voxelization ensures consistent FPFH computation across datasets with varying densities. The voxel size is empirically selected to

balance detail preservation with efficiency, ensuring the detection framework remains robust across infrastructure scales and point cloud resolutions.

Results demonstrated that *PatchCore* effectively detects cracks, even in the presence of significant measurement noise and surface roughness. Importantly, it was capable of identifying both surface and internal cracks (e.g., intrados hinge cracks) while maintaining robust performance despite initial geometric distortions. This highlights the framework's potential for real-world defect detection in masonry bridges.

However, the results presented in Chapter 5 indicate that FPFH lacks the sensitivity to detect small cracks due to subtle geometric distortions. Relying solely on FPFH also prevents *PatchCore* from identifying non-geometric defects, such as water ingress or vegetation, which are unrelated to geometric distortions. Additionally, the validation of *PatchCore* was limited to synthetic data, raising concerns about its applicability in real-world scenarios. To address these limitations, Chapter 6 introduced 3DMulti-FPFHI, a novel multimodal feature for *PatchCore* that combines geometric features (e.g., FPFH) with 3D intensity information. The enhanced *PatchCore* was benchmarked against the SOTA anomaly detection method, CPMF, demonstrating its accuracy and robustness.

Building on the synthetic masonry arch dataset developed in Chapter 5, a new synthetic dataset was created by incorporating surface roughness and intrados crack geometry, providing a more realistic representation of real-world conditions. Re-evaluating the anomaly detection framework on this enhanced dataset indicated that the multimodal feature significantly improved the detection and localization of intrados cracks. Statistical analysis further demonstrated the contribution of the 3D intensity feature in enhancing detection performance. In contrast, CPMF exhibited poor accuracy, primarily due to insufficient representations of the intensity feature extracted from DL models.

The enhanced *PatchCore* was further validated on real-world datasets, including the London Bridge Station (a masonry bridge) and a full-scale experiment tunnel ring. While FPFH alone failed to detect cracks as geometric distortions being subtle, the 3DMulti-FPFHI effectively identified longitudinal cracks in the London Bridge Station point cloud. This demonstrated the effectiveness of incorporating

the 3D intensity feature. Furthermore, evaluations on the full-scale experiment tunnel ring dataset highlighted the generalizability of the framework to diverse infrastructure types, successfully detecting both intrados and extrados cracks robustly. These results indicate the potential of the enhanced *PatchCore* for practical deployment in infrastructure inspections, addressing critical challenges in defect detection using point clouds.

In summary, this thesis makes significant contributions to the field of automating masonry bridge inspection by addressing critical challenges in both point cloud segmentation and defect detection. It develops synthetic datasets and lightweight DL architectures (e.g., BridgeNet and BridgeNetv2) to tackle data scarcity and computational inefficiency in segmenting large-scale masonry bridge point clouds. Furthermore, it advances defect detection through the adaptation and enhancement of *PatchCore*, demonstrating its robustness in identifying structural defects using geometric and multimodal features. By validating these methods on synthetic and real-world datasets, the research proves the feasibility of automating masonry bridge inspections with scalable and robust techniques.

7.2 Future works

7.2.1 Towards generalized segmentation frameworks

While this thesis has made significant progress in addressing segmentation challenges for large-scale masonry bridge point clouds, several limitations remain. One limitation is related to the design of the synthetic training dataset. Specifically, the synthetic dataset was constructed with prior knowledge of the real bridge geometries later used for evaluation. Although this approach ensured that the generated data resembled realistic structural components, it implicitly introduced a feedback loop between training and testing data, which represents a soft violation of the machine learning principle of the train-test data separation. Consequently, the reported performance may reflect some degree of dataset alignment rather than true generalization capability.

Additionally, synthetic point clouds often fail to fully capture real-world features such as intricate surface textures, sensor noise, and the presence of surrounding

infrastructure. Furthermore, the segmentation tasks explored in this thesis primarily focused on a limited set of masonry bridge types, even though their geometries can vary substantially across different settings. These constraints may limit the scalability and adaptability of the proposed framework.

Future work should aim to address these limitations by:

(1) **Establishing a generalized segmentation framework:** Designing a scalable segmentation architecture by incorporating large vision models that operate across diverse infrastructure types with minimal reliance on fully labeled datasets [S. He et al. 2025]. Approaches such as transfer learning, zero-shot, and few-shot learning should be explored to reduce dependency on tailored training datasets for each new application [S. He et al. 2023; Y. Lu et al. 2023].

(2) **Mitigating training-test coupling:** To overcome the implicit coupling between synthetic training data and real-world evaluation samples, future work should avoid designing synthetic datasets based on known test geometries. Instead, adopting zero-/few-shot learning strategies would allow the segmentation model to generalize to novel structural forms, mitigating data leakage and enhancing evaluation rigor.

(3) **Developing more realistic point cloud datasets:** Leveraging advanced graphical tools to simulate highly detailed and realistic synthetic point clouds that incorporate complex surface textures, sensor noise, and contextual environments [Reitmann et al. 2021a]. Such enhancements will better reflect real-world conditions and strengthen model robustness.

7.2.2 Enhancements on anomaly detections on infrastructures

While the 3DMulti-FPFHI developed in this thesis has shown significant improvements in detecting cracks in point clouds, several limitations remain. Subtle geometric distortions caused by small cracks often fall below the sensitivity threshold of FPFH, which relies on local curvature and normal information that may not adequately capture such minor changes. Furthermore, the inherent sparsity of point clouds results in lower resolution compared to 2D images, making it difficult to

identify small cracks with a size below the point cloud resolution.

Additionally, the current synthetic dataset generation process does not incorporate material properties (such as surface reflectivity or texture) during mesh reconstruction and rendering in Blender. These properties can significantly influence the simulated intensity values in point clouds and may affect the generalizability of the multimodal feature to real-world data.

Future research directions aim to address these challenges through the following:

(1) **Developing more sensitive geometric indicators**: To enhance the detection of small cracks, advanced geometric indicators will be explored. Techniques such as higher-order geometric descriptors [Roetzer and Bernard 2024], or adaptive feature selection [Günen 2022] could improve sensitivity to small-scale distortions, boosting the algorithm's crack detection capabilities.

(2) **Integrating multimodal data fusion**: Beyond geometry and intensity, incorporating additional modalities such as calibrated thermal images, RGB images, or acoustic data could provide complementary information about defects [Cao et al. 2023; Yue Wang et al. 2023]. A multimodal approach would enhance robustness and accuracy, particularly for detecting invisible cracks or defects that are difficult to identify using geometric and intensity information alone.

(3) **Simulating realistic material properties in synthetic data**: Future work could explore integrating material shaders and reflectivity properties during mesh reconstruction in Blender to simulate more realistic surface interactions. This would improve the fidelity of synthetic intensity features and potentially enhance the domain transferability of the developed anomaly detection algorithms.

7.3 Data contribution

7.3.1 BridgePoint-Seg dataset

To support reproducible research in infrastructure segmentation, this thesis releases the **BridgePoint-Seg** dataset, i.e., a comprehensive, large-scale synthetic

dataset for semantic segmentation of masonry bridge point clouds. The dataset is publicly available on HuggingFace at https://huggingface.co/datasets/jing222/syn_masonry_bridge with a size of 8.95 GB. The corresponding code can be found at https://github.com/Jingyixiong/BridgeNetv2.

The dataset is stored in the archive `syn_data.zip`, which contains the following structure:

1. `train/` – Training set

- Contains two subfolders: `straight_bridge/` and `curved_bridge/`
- `straight_bridge/`: 2177 synthetic bridge point clouds
- `curved_bridge/`: 1500 synthetic bridge point clouds
- Each point cloud is saved in:
  - `points.npz`: 3D coordinates, accessible via the key `xyz`
  - `points_label.npz`: semantic segmentation labels, accessible via the key `sem_label`

2. `test/` – Test set

- Contains two subfolders: `straight_bridge/` and `curved_bridge/`
- `straight_bridge/`: 87 test samples
- `curved_bridge/`: 500 test samples
- Format is consistent with the training set:
  - `points.npz`: point cloud coordinates (`xyz`)
  - `points_label.npz`: semantic labels (`sem_label`)

Data Format Summary

- File format: `.npz`
- Coordinate key: `xyz` in `points.npz`
- Label key: `sem_label` in `points_label.npz`

BridgePoint-Seg provides a structured and scalable dataset to benchmark semantic segmentation methods in infrastructure point clouds. It features high-resolution synthetic data for straight and curved masonry bridges, facilitating research into model generalization across varying bridge geometries and layouts.

7.3.2 Crack3D-Defect dataset

To support reproducible research and enable further development in infrastructure anomaly detection, this thesis releases the **Crack3D-Defect** dataset, i.e., a comprehensive 3D multimodal benchmark for evaluating crack and defect detection in infrastructure point clouds. The dataset is publicly available on HuggingFace at https://huggingface.co/datasets/jing222/infra_3DALv1 with a size of 4.2 GB. The corresponding code can be found at https://github.com/Jingyixiong/3D-Multi-FPFHI.

The released archive, `infra_3DALv2.zip`, contains three main subdirectories:

1. `syn_arch_src/` – Synthetic masonry arch dataset

- Point clouds are stored in `.csv` format with the following columns:
  - `categoryID`: defect label (0 = No crack, 1 = Intrados crack, 2 = Extrados crack, 3 = Inner crack)
  - `[X, Y, Z]`: clean coordinates
  - `[X_noise, Y_noise, Z_noise]`: coordinates with synthetic LiDAR noise
  - `intensity`: simulated LiDAR intensity value
- Subfolders:
  - `memory_bank/min_cw0`: undeformed reference arch point clouds
  - `memory_bank/min_cw0.4/`: x displacement case with minimum crack width 4 mm
  - `disp_x/`, `disp_z/`, `disp_xz/`, `rot_x/`: 4 supporting displacement cases
  - `disp_x/diff_cw/`: minimum crack width for comparison (4 mm and 8 mm)

- `disp_x/diff_disps/`: varied x displacement (8 cm, 12 cm, 16 cm, 40 cm)

2. `real_arch_src/` – London Bridge Station arches

- Real scans of two masonry arches (Arch 1 and Arch 2) from London Bridge Station, collected at two timepoints:
  - `130305`: March 5, 2013 (initial state)
  - `131123`: November 23, 2013 (with deformation)
- Files are in `.asc` format, with the following columns: `[x, y, z, r, g, b, intensity]`
- Subfolders:
  - `memory_bank/`: contains point clouds of arches (`Narch_130305`, `Sarch_130305`)
  - `Narch_131123/`, `Sarch_131123/`: arches after settlement induced cracking

3. `tunnel/` – Full-scale tunnel section experiment

- Point clouds from a laboratory test of a segmental tunnel ring, sampled at multiple loading steps.
- Files are stored in `.txt` format with columns: `[x, y, z, intensity]`
- Subfolders:
  - `memory_bank/`: initial undeformed ring (step 0)
  - `76`, `89`, `96`, `103`: deformed tunnel ring scans at increasing load steps

This dataset enables benchmarking of anomaly detection methods across both synthetic and real-world conditions. It captures diverse deformation scenarios, simulates sensor noise, and provides multimodal attributes such as geometric coordinates and intensity. `Crack3D-Defect` is intended to foster future research in scalable and generalizable 3D anomaly detection for structural health monitoring applications.

References

Sajjad Abbasi, Mohsen Hajabdollahi, Nader Karimi, and Shadrokh Samavi (2020). "Modeling teacher-student techniques in deep neural networks for knowledge distillation". In: *2020 International Conference on Machine Vision and Image Processing (MVIP)*. IEEE. URL: 10.1109/MVIP49855.2020.9116923.

Ikhlas Abdel-Qader, Osama Abudayyeh, and Michael E Kelly (2003). "Analysis of edge-detection techniques for crack identification in bridges". In: *Journal of computing in civil engineering*. URL: https://doi.org/10.1061/(ASCE)0887-3801(2003)17:4(255).

Sinan Acikgoz, Matthew J DeJong, and Kenichi Soga (2018). "Sensing dynamic displacements in masonry rail bridges using 2D digital image correlation". In: *Structural Control and Health Monitoring*. URL: https://doi.org/10.1002/stc.2187.

Sinan Acikgoz, Antonio Luciano, Mark Dewhirst, Matthew J Dejong, and Robert Mair (2022). "Innovative monitoring of the response of a heritage masonry building to nearby tunnelling in London Clay". In: *Géotechnique*. URL: https://doi.org/10.1680/jgeot.19.P.243.

Sinan Acikgoz, Loizos Pelecanos, Giorgia Giardina, James Aitken, and Kenichi Soga (2017a). "Distributed sensing of a masonry vault during nearby piling". In: *Structural Control and Health Monitoring*. URL: https://onlinelibrary.wiley.com/doi/full/10.1002/stc.1872.

Sinan Acikgoz, Kenichi Soga, and Jim Woodhams (2017b). "Evaluation of the response of a vaulted masonry structure to differential settlements using point cloud data and limit analyses". In: *Construction and Building Materials*. URL: https://doi.org/10.1016/j.conbuildmat.2017.05.075.

Pankaj K Agarwal, Sariel Har-Peled, Kasturi R Varadarajan, et al. (2005). "Geometric approximation via coresets". In: *Combinatorial and computational geometry*. URL: https://courses.cs.duke.edu//fall11/cps234/reading/coreset-survey.pdf.

Samet Akcay, Amir Atapour-Abarghouei, and Toby P Breckon (2019). "Ganomaly: Semi-supervised anomaly detection via adversarial training". In: *Computer Vision–ACCV 2018: 14th Asian Conference on Computer Vision, Perth, Australia, December 2–6, 2018, Revised Selected Papers, Part III 14*. Springer. URL: https://doi.org/10.1016/j.media.2019.01.010.

Claudio Alessandri, Massimo Garutti, Vincenzo Mallardo, and Gabriele Milani (2015). "Crack patterns induced by foundation settlements: integrated analysis on a renaissance masonry palace in Italy". In: *International Journal of Architectural Heritage*. URL: https://doi.org/10.1080/15583058.2014.951795.

ambientCG Docs, accessed 2024 (n.d.). *ambientCG Docs Bricks 051*. https://ambientcg.com/view?id=Bricks051.

Jerone Andrews, Thomas Tanay, Edward J Morton, and Lewis D Griffin (2016). "Transfer representation-learning for anomaly detection". In: JMLR. URL: http://imageanalysis.cs.ucl.ac.uk/documents/JAndrews-ADWorkshop-ICML.pdf.

J Armesto, Javier Roca-Pardiñas, H Lorenzo, and P Arias (2010). "Modelling masonry arches shape using terrestrial laser scanning data and nonparametric methods". In: *Engineering Structures*. URL: https://doi.org/10.1016/j.engstruct.2009.11.007.

Julia Armesto-González, Belén Riveiro-Rodríguez, Diego González-Aguilera, and M Teresa Rivas-Brea (2010). "Terrestrial laser scanning intensity data applied to damage detection for historical buildings". In: *Journal of Archaeological Science*. URL: https://doi.org/10.1016/j.jas.2010.06.031.

Deeksha Arya, Hiroya Maeda, Sanjay Kumar Ghosh, Durga Toshniwal, Alexander Mraz, Takehiro Kashiyama, and Yoshihide Sekimoto (2021). "Deep learning-based road damage detection and classification for multiple countries". In: *Automation in Construction*. URL: https://doi.org/10.1016/j.autcon.2021.103935.

Leanne Attard, Carl James Debono, Gianluca Valentino, Mario Di Castro, Alessandro Masi, and Luigi Scibile (2019a). "Automatic Crack Detection using Mask R-CNN". In: *2019 11th International Symposium on Image and Signal Processing and Analysis (ISPA)*.

Leanne Attard, Carl James Debono, Gianluca Valentino, Mario Di Castro, Alessandro Masi, and Luigi Scibile (2019b). "Automatic crack detection using mask R-CNN". In: *2019 11th international symposium on image and signal processing and analysis (ISPA)*. IEEE. URL: 10.1109/ISPA.2019.8868619.

Rytis Augustaukas and Arūnas Lipnickas (2019). “Pixel-wise Road Pavement Defects Detection Using U-Net Deep Neural Network”. In: *2019 10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*.

Fardin Bahreini and Amin Hammad (2024). “Dynamic graph CNN based semantic segmentation of concrete defects and as-inspected modeling”. In: *Automation in Construction*. URL: https://doi.org/10.1016/j.autcon.2024.105282.

Laura Beggel, Michael Pfeiffer, and Bernd Bischl (2020). “Robust anomaly detection in images using adversarial autoencoders”. In: *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2019, Würzburg, Germany, September 16–20, 2019, Proceedings, Part I*. Springer. URL: https://doi.org/10.1007/978-3-030-46150-8_13.

Marcel Bengs, Finn Behrendt, Julia Krüger, Roland Opfer, and Alexander Schlaefer (2021). “Three-dimensional deep learning with spatial erasing for unsupervised anomaly segmentation in brain MRI”. In: *International journal of computer assisted radiology and surgery*. URL: arXiv:2109.06540v1.

Paul Bergmann, Michael Fauser, David Sattlegger, and Carsten Steger (2019). “MVTec AD–A comprehensive real-world dataset for unsupervised anomaly detection”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. URL: 10.1109/CVPR.2019.00982.

Paul Bergmann, Xin Jin, David Sattlegger, and Carsten Steger (2021). “The mvtec 3d-ad dataset for unsupervised 3d anomaly detection and localization”. In: *arXiv preprint arXiv:2112.09045*. URL: https://doi.org/10.5220/0010865000003124.

Paul Bergmann, Sindy Löwe, Michael Fauser, David Sattlegger, and Carsten Steger (2018). “Improving unsupervised defect segmentation by applying structural similarity to autoencoders”. In: *arXiv preprint arXiv:1807.02011*. URL: https://doi.org/10.48550/arXiv.1807.02011.

Paul Bergmann and David Sattlegger (2023). “Anomaly detection in 3d point clouds using deep geometric descriptors”. In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. URL: 10.1109/WACV56688.2023.00264.

R Blomley, M Weinmann, J Leitloff, and B Jutzi (2014). “Shape distribution features for point cloud analysis–A geometric histogram approach on multiple scales”. In: *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*. URL: https://doi.org/10.5194/isprsannals-II-3-9-2014.

Neshat Bolourian, Majid Nasrollahi, Fardin Bahreini, and Amin Hammad (2023). "Point Cloud–Based Concrete Surface Defect Semantic Segmentation". In: *Journal of Computing in Civil Engineering*. URL: https://ascelibrary.org/doi/abs/10.1061/JCCEE5.CPENG-5009.

A. Boulch, B. Le Saux, and N. Audebert (2017). "Unstructured point cloud semantic labeling using deep segmentation networks". In: *Proceedings of the Workshop on 3D Object Retrieval*. 3Dor '17. Lyon, France: Eurographics Association. URL: https://doi.org/10.2312/3dor.20171047.

Baran Bozyigit and Sinan Acikgoz (2022). "Dynamic amplification in masonry arch railway bridges". In: *Structures*. Elsevier. URL: https://doi.org/10.1016/j.istruc.2022.09.100.

D Brackenbury, I Brilakis, and M DeJong (2019). "Automated defect detection for masonry arch bridges". In: *International Conference on Smart Infrastructure and Construction 2019 (ICSIC) Driving data-informed decision-making*. ICE Publishing. URL: https://www.icevirtuallibrary.com/doi/full/10.1680/icsic.64669.003.

Daniel Brackenbury (2022a). "Automated Image-Based Inspection of Masonry Arch Bridges". PhD thesis. Apollo - University of Cambridge Repository. URL: https://www.repository.cam.ac.uk/handle/1810/338321.

Daniel Brackenbury (2022b). "Automated Image-Based Inspection of Masonry Arch Bridges". PhD thesis. Apollo - University of Cambridge Repository. URL: 10.17863/CAM.85731.

Leo Breiman (2001). "Random forests". In: *Machine learning*.

Antonio Brencich and Renata Morbiducci (2007). "Masonry arches: Historical rules and modern mechanics". In: *International Journal of Architectural Heritage*. URL: https://doi.org/10.1080/15583050701312926.

John B Burland, Bengt Baltzar Broms, and Victor FB De Mello (1978). "Behaviour of foundations and structures". In: URL: https://www.issmge.org/publications/publication/behaviour-of-foundations-and-structures-6189.

Juan José Cabrera, Antonio Santo, Arturo Gil, Carlos Viegas, and Luis Payá (2024). "MinkUNeXt: Point Cloud-based Large-scale Place Recognition using 3D Sparse Convolutions". In: *arXiv preprint arXiv:2403.07593*. URL: https://doi.org/10.48550/arXiv.2403.07593.

John Canny (1986). “A computational approach to edge detection”. In: *IEEE Transactions on pattern analysis and machine intelligence*. URL: 10.1109/TPAMI.1986.4767851.

Yunkang Cao, Qian Wan, Weiming Shen, and Liang Gao (2022). “Informative knowledge distillation for image anomaly segmentation”. In: *Knowledge-Based Systems*. URL: https://doi.org/10.1016/j.knosys.2022.108846.

Yunkang Cao, Xiaohao Xu, and Weiming Shen (2023). “Complementary pseudo multimodal feature for point cloud anomaly detection”. In: *arXiv preprint arXiv:2303.13194*. URL: https://doi.org/10.48550/arXiv.2303.13194.

Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko (2020). “End-to-end object detection with transformers”. In: *European conference on computer vision*. Springer.

Young-Jin Cha, Wooram Choi, and Oral Büyüköztürk (2017). “Deep learning-based crack damage detection using convolutional neural networks”. In: *Computer-Aided Civil and Infrastructure Engineering*. URL: https://doi.org/10.1111/mice.12263.

Young-Jin Cha, Wooram Choi, Gahyun Suh, Sadegh Mahmoudkhani, and Oral Büyüköztürk (2018). “Autonomous structural visual inspection using region-based deep learning for detecting multiple damage types”. In: *Computer-Aided Civil and Infrastructure Engineering*. URL: https://doi.org/10.1111/mice.12334.

Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu (2015). “ShapeNet: An information-rich 3D model repository”. In: *arXiv preprint arXiv:1512.03012*. URL: https://doi.org/10.48550/arXiv.1512.03012.

K Chatfield (2014). “Return of the devil in the details: Delving deep into convolutional nets”. In: *arXiv preprint arXiv:1405.3531*. URL: https://doi.org/10.48550/arXiv.1405.3531.

Nesrine Chehata, Li Guo, and Clément Mallet (2009). “Airborne lidar feature selection for urban classification using random forests”. In: *Laserscanning*. URL: https://hal.science/hal-02384719/.

Fu-Chen Chen and Mohammad R Jahanshahi (2017). “NB-CNN: Deep learning-based crack detection using convolutional neural network and Naıve Bayes data fusion”. In: *IEEE Transactions on Industrial Electronics*. URL: 10.1109/TIE.2017.2764844.

Jieneng Chen, Yongyi Lu, Qihang Yu, Xiangde Luo, Ehsan Adeli, Yan Wang, Le Lu, Alan L Yuille, and Yuyin Zhou (2021). "Transunet: Transformers make strong encoders for medical image segmentation". In: *arXiv preprint arXiv:2102.04306*. URL: https://doi.org/10.48550/arXiv.2102.04306.

Shaoyu Chen, Jiemin Fang, Qian Zhang, Wenyu Liu, and Xinggang Wang (2021). "Hierarchical aggregation for 3d instance segmentation". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. URL: 10.1109/ICCV48922.2021.01518.

Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia (2017). "Multi-view 3d object detection network for autonomous driving Vehicle detection from 3d lidar using fully convolutional network". In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. URL: https://doi.org/10.48550/arXiv.1611.07759.

Yukang Chen, Yanwei Li, Xiangyu Zhang, Jian Sun, and Jiaya Jia (2022). "Focal sparse convolutional networks for 3d object detection". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. URL: 10.1109/CVPR52688.2022.00535.

Ran Cheng, Ryan Razani, Ehsan Taghavi, Enxu Li, and Bingbing Liu (2021). "(AF)2-S3Net: Attentive Feature Fusion with Adaptive Feature Selection for Sparse Semantic Segmentation Network". In: *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. URL: 10.1109/CVPR46437.2021.01236.

Renjie Cheng, Reza Razani, Ehsan Taghavi, Ehsan Li, and Bo Liu (2021). "(AF)2-S3Net: Attentive feature fusion with adaptive feature selection for sparse semantic segmentation network". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. URL: https://doi.org/10.1109/CVPR46437.2021.01236.

Jun Kang Chow, Zhaoyu Su, Jimmy Wu, Pin Siang Tan, Xin Mao, and Yu-Hsing Wang (2020). "Anomaly detection of defects on concrete structures with the convolutional autoencoder". In: *Advanced Engineering Informatics*. URL: https://doi.org/10.1016/j.aei.2020.101105.

Christopher Choy, JunYoung Gwak, and Silvio Savarese (2019a). "4D spatio-temporal convnets: Minkowski convolutional neural networks". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. URL: https://doi.org/10.1109/CVPR.2019.00319.

Christopher Choy, JunYoung Gwak, and Silvio Savarese (2019b). "4d spatio-temporal convnets: Minkowski convolutional neural networks". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*.

Yu-Min Chu, Chieh Liu, Ting-I Hsieh, Hwann-Tzong Chen, and Tyng-Luh Liu (2023). "Shape-Guided Dual-Memory Learning for 3D Anomaly Detection". In: *Proceedings of the 40th International Conference on Machine Learning*. Ed. by Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett. Proceedings of Machine Learning Research. PMLR. URL: https://proceedings.mlr.press/v202/chu23b.html.

Özgün Çiçek, Ahmed Abdulkadir, Soeren S Lienkamp, Thomas Brox, and Olaf Ronneberger (2016). "3D U-Net: learning dense volumetric segmentation from sparse annotation". In: *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2016: 19th International Conference, Athens, Greece, October 17-21, 2016, Proceedings, Part II 19*. Springer. URL: https://doi.org/10.1007/978-3-319-46723-8_49.

Niv Cohen and Yedid Hoshen (2020). "Sub-image anomaly detection with deep pyramid correspondences". In: *arXiv preprint arXiv:2005.02357*. URL: https://doi.org/10.48550/arXiv.2005.02357.

D. Comaniciu and P. Meer (2002). "Mean shift: a robust approach toward feature space analysis". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*. URL: 10.1109/34.1000236.

High Speed Rail Consultation (2011). *Future Demand for Long Distance Travel*. Tech. rep. Department of Transportation. URL: https://hs2ltd.wordpress.com/wp-content/uploads/2011/05/future-demand-long-distance-travel.pdf.

Alex Costanzino, Pierluigi Zama Ramirez, Giuseppe Lisanti, and Luigi Di Stefano (2024). "Multimodal industrial anomaly detection by crossmodal feature mapping". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. URL: https://doi.org/10.48550/arXiv.2312.04521.

Fei Dai and Ming Lu (2013). "Three-dimensional modeling of site elements by analytically processing image data contained in site photos". In: *Journal of construction engineering and management*. URL: https://doi.org/10.1061/(ASCE)CO.1943-7862.0000655.

Dimitris Dais, Ihsan Engin Bal, Eleni Smyrou, and Vasilis Sarhosis (2021). "Automatic crack classification and segmentation on masonry surfaces using convolutional

neural networks and transfer learning". In: *Automation in Construction*. URL: https://doi.org/10.1016/j.autcon.2021.103606.

L Minh Dang, Hanxiang Wang, Yanfen Li, Le Quan Nguyen, Tan N Nguyen, Hyoung-Kyu Song, and Hyeonjoon Moon (2022). "Deep learning-based masonry crack segmentation and real-life crack length measurement". In: *Construction and Building Materials*. URL: https://doi.org/10.1016/j.conbuildmat.2022.129438.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré (2022). "Flashattention: Fast and memory-efficient exact attention with io-awareness". In: *Advances in Neural Information Processing Systems*.

Amrita Das, Sattar Dorafshan, and Naima Kaabouch (2024). "Autonomous Image-Based Corrosion Detection in Steel Structures Using Deep Learning". In: *Sensors*. URL: https://www.mdpi.com/1424-8220/24/11/3630.

Pablo del Río-Barral, Mario Soilán, Silvia María González-Collazo, and Pedro Arias (2022). "Pavement crack detection and clustering via region-growing algorithm from 3D MLS point clouds". In: *Remote Sensing*. URL: https://doi.org/10.3390/rs14225866.

Haowen Deng, Tolga Birdal, and Slobodan Ilic (2018). "Ppfnet: Global context aware local features for robust 3d point matching". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. URL: https://doi.org/10.48550/arXiv.1802.02669.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei (2009). "ImageNet: A large-scale hierarchical image database". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018). "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *arXiv preprint arXiv:1810.04805*.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio (2016). "Density estimation using real nvp". In: *arXiv preprint arXiv:1605.08803*. URL: https://doi.org/10.48550/arXiv.1605.08803.

Qiao Dong, Sike Wang, Xueqin Chen, Wanqi Jiang, Ruiqi Li, and Xingyu Gu (2023). "Pavement crack detection based on point cloud data and data fusion". In: *Philosophical Transactions of the Royal Society A*. URL: https://doi.org/10.1098/rsta.2022.0165.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer,

Georg Heigold, Sylvain Gelly, et al. (2020). “An image is worth 16x16 words: Transformers for image recognition at scale”. In: *arXiv preprint arXiv:2010.11929*.

Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun (2017). “CARLA: An Open Urban Driving Simulator”. In: *Proceedings of the 1st Annual Conference on Robot Learning*.

Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu (1996). “A density-based algorithm for discovering clusters in large spatial databases with noise”. In: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD)*. URL: https://dl.acm.org/doi/10.5555/3001460.3001507.

Haoqi Fan, Bo Xiong, Karttikeya Mangalam, Yanghao Li, Zhicheng Yan, Jitendra Malik, and Christoph Feichtenhofer (2021). “Multiscale vision transformers”. In: *Proceedings of the IEEE/CVF international conference on computer vision*.

Jiakai Fan, Weidong Song, Jinhe Zhang, Shangyu Sun, Guohui Jia, and Guang Jin (2024). “PAN: Improved PointNet++ for Pavement Crack Information Extraction”. In: *Electronics*. URL: https://doi.org/10.3390/electronics13163340.

Zhun Fan, Chong Li, Ying Chen, Jiahong Wei, Giuseppe Loprencipe, Xiaopeng Chen, and Paola Di Mascio (2020). “Automatic crack detection on road pavements using encoder-decoder architecture”. In: *Materials*. URL: https://doi.org/10.3390/ma13132960.

Jin Fang, Dingfu Zhou, Fangcheng Yan, Tianyang Zhao, Fei Zhang, Yanyan Ma, and Ruigang Yang (2020). “Augmented lidar simulator for autonomous driving”. In: *IEEE Robotics and Automation Letters*. URL: https://doi.org/10.1109/LRA.2020.2969927.

Reza Zanjirani Farahani and Masoud Hekmatfar (2009). *Facility location: concepts, models, algorithms and case studies*. Springer Science & Business Media. URL: https://doi.org/10.1007/978-3-7908-2151-2.

Rui Pedro Lopes Fernandes (2013). “Advantages and disadvantages of BIM platforms on construction site”. MA thesis. Universidade do Porto (Portugal). URL: https://www.proquest.com/docview/2689286905?pq-origsite=gscholar&fromopenview=true.

Jingyun Fu, Zhiyu Xiang, Chengyu Qiao, and Tingming Bai (2023). “PT-FlowNet: Scene Flow Estimation on Point Clouds With Point Transformer”. In: *IEEE Robotics and Automation Letters*.

Iván García-Aguilar, Jorge García-González, Rafael M Luque-Baena, Ezequiel López-Rubio, and Enrique Domínguez (2023). "Optimized instance segmentation by super-resolution and maximal clique generation". In: *Integrated Computer-Aided Engineering*.

Andreas Geiger, Philipp Lenz, and Raquel Urtasun (2012). "Are we ready for autonomous driving? the kitti vision benchmark suite". In: *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. URL: https://doi.org/10.1109/CVPR.2012.6248074.

Tarutal Ghosh Mondal, Mohammad R Jahanshahi, Rih-Teng Wu, and Zheng Yi Wu (2020). "Deep learning-based multi-class damage detection for autonomous post-disaster reconnaissance". In: *Structural Control and Health Monitoring*.

Benjamin Graham, Martin Engelcke, and Laurens Van Der Maaten (2018). "3d semantic segmentation with submanifold sparse convolutional networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. URL: 10.1109/CVPR.2018.00961.

David Griffiths and Jan Boehm (2019). "SynthCity: A large scale synthetic point cloud". In: *arXiv preprint arXiv:1907.04758*. URL: https://doi.org/10.48550/arXiv.1907.04758.

Eleonora Grilli, Elisa Mariarosaria Farella, Alessandro Torresani, and Fabio Remondino (2019). "Geometric features analysis for the classification of cultural heritage point clouds". In: *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*. URL: 10.5194/isprs-archives-XLII-2-W15-541-2019.

Eleonora Grilli and Fabio Remondino (2020). "Machine learning generalisation across different 3D architectural heritage". In: *ISPRS International Journal of Geo-Information*. URL: https://doi.org/10.3390/ijgi9060379.

Michael Gschwandtner, Roland Kwitt, Andreas Uhl, and Wolfgang Pree (2011). "BlenSor: Blender sensor simulation toolbox". In: *Advances in Visual Computing: 7th International Symposium, ISVC 2011, Las Vegas, NV, USA, September 26-28, 2011. Proceedings, Part II 7*. Springer. URL: 10.1007/978-3-642-24031-7_20.

Mehmet Akif Günen (2022). "Adaptive neighborhood size and effective geometric features selection for 3D scattered point cloud classification". In: *Applied Soft Computing*. URL: https://doi.org/10.1016/j.asoc.2021.108196.

Feng Guo, Yu Qian, Dimitris Rizos, Zhi Suo, and Xiaobin Chen (2021). "Automatic rail surface defects inspection based on Mask R-CNN". In: *Transportation research record*. URL: https://doi.org/10.1177/03611981211019034.

Priyanka Gupta and Manish Dixit (2022). “Image-based crack detection approaches: a comprehensive survey”. In: *Multimedia Tools and Applications*. URL: https://doi.org/10.1007/s11042-022-13152-z.

Mitchell J Hallee, Rebecca K Napolitano, Wesley F Reinhart, and Branko Glisic (2021). “Crack detection in images of masonry using cnns”. In: *Sensors*. URL: https://doi.org/10.3390/s21144929.

Mahmudul Hasan, Jonghyun Choi, Jan Neumann, Amit K. Roy-Chowdhury, and Larry S. Davis (June 2016). “ Learning Temporal Regularity in Video Sequences ”. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Los Alamitos, CA, USA: IEEE Computer Society. URL: https://doi.ieeecomputersociety.org/10.1109/CVPR.2016.86.

Sahar Hassani and Ulrike Dackermann (2023). “A systematic review of advanced sensor technologies for non-destructive testing and structural health monitoring”. In: *Sensors*. URL: https://doi.org/10.3390/s23042204.

Kosaburo Hattori, Tomonori Izumi, and Lin Meng (2023). “Defect Detection of Apples using PatchCore”. In: *2023 International Conference on Advanced Mechatronic Systems (ICAMechS)*. IEEE. URL: https://doi.org/10.3390/foods12061352.

Bulent Haznedar, Rabia Bayraktar, Ali Emre Ozturk, and Yusuf Arayici (2023). “Implementing PointNet for point cloud segmentation in the heritage context”. In: *Heritage Science*. URL: https://doi.org/10.1186/s40494-022-00844-w.

Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick (Oct. 2017). “Mask R-CNN”. In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. URL: 10.1109/ICCV.2017.322.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (2016). “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. URL: https://doi.org/10.48550/arXiv.1512.03385.

Shuting He, Henghui Ding, Xudong Jiang, and Bihan Wen (2025). “Segpoint: Segment any point cloud via large language model”. In: *European Conference on Computer Vision*. Springer. URL: https://doi.org/10.48550/arXiv.2407.13761.

Shuting He, Xudong Jiang, Wei Jiang, and Henghui Ding (2023). “Prototype adaption and projection for few-and zero-shot 3d point cloud semantic segmentation”. In: *IEEE Transactions on Image Processing*. URL: 10.1109/TIP.2023.3279660.

Yang He, Zihan Jin, Jiqiao Zhang, Shuai Teng, Gongfa Chen, Xiaoli Sun, and Fangsen Cui (2022). “Pavement surface defect detection using mask region-based

convolutional neural networks and transfer learning". In: *Applied Sciences*. URL: https://doi.org/10.3390/app12157364.

M.A. Hearst, S.T. Dumais, E. Osuna, J. Platt, and B. Scholkopf (1998). "Support vector machines". In: *IEEE Intelligent Systems and their Applications*. URL: 10.1109/5254.708428.

Rosemarie Helmerich, Ernst Niederleithinger, Christiane Trela, J Bień, T Kamiński, and G Bernardini (2012). "Multi-tool inspection and numerical analysis of an old masonry arch bridge". In: *Structure and Infrastructure Engineering*. URL: https://doi.org/10.1080/15732471003645666.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean (2015). "Distilling the knowledge in a neural network". In: *arXiv preprint arXiv:1503.02531*. URL: https://doi.org/10.48550/arXiv.1503.02531.

Eliahu Horwitz and Yedid Hoshen (2023). "Back to the feature: classical 3d features are (almost) all you need for 3d anomaly detection". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. URL: 10.1109/CVPRW59228.2023.00298.

Qingyong Hu, Bo Yang, Linhai Xie, Stefano Rosa, Yulan Guo, Zhihua Wang, Niki Trigoni, and Andrew Markham (2020). "RandLA-Net: Efficient semantic segmentation of large-scale point clouds". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. URL: https://doi.org/10.1109/CVPR42600.2020.01112.

Caiping Huang, Yongkang Zhou, and Xin Xie (2024). "Intelligent Diagnosis of Concrete Defects Based on Improved Mask R-CNN". In: *Applied Sciences*. URL: https://doi.org/10.3390/app14104148.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger (2017). "Densely connected convolutional networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. URL: https://doi.org/10.48550/arXiv.1608.06993.

Philipp Huthwohl, Ruodan Lu, and Ioannis Brilakis (Jan. 2016). "Challenges of bridge maintenance inspection". In: URL: https://repository.lboro.ac.uk/articles/conference_contribution/Challenges_of_bridge_maintenance_inspection/9425816.

Ju Huyan, Wei Li, Susan Tighe, Zhengchao Xu, and Junzhi Zhai (2020). "CrackU-net: A novel deep convolutional neural network for pixelwise pavement crack detection". In: *Structural Control and Health Monitoring*. URL: https://doi.org/10.1002/stc.2551.

Eldor Ibragimov, Hyun-Jong Lee, Jong-Jae Lee, and Namgyu Kim (2022). "Automated pavement distress detection using region based convolutional neural networks". In: *International Journal of Pavement Engineering*. URL: https://doi.org/10.1080/10298436.2020.1833204.

Anas Bin Iftikhar, Muhammad Zakir Shaikh, Sanaullah Mehran, Tanweer Hussain, Imtiaz Hussain Kalwar, and Bhawani Shankar Chowdhry (2022). "Unsupervised detection of rail surface defects and rail-head anomalies using Auto-encoder". In: *2022 Global Conference on Wireless and Optical Technologies (GCWOT)*.

Jimin Jeong, Hamin Song, Jaehyun Park, Paulo Resende, Benazouz Bradaï, and Kichun Jo (2022). "Fast and Lite Point Cloud Semantic Segmentation for Autonomous Driving Utilizing LiDAR Synthetic Training Data". In: *IEEE Access*.

Ankang Ji, Alvin Wei Ze Chew, Xiaolong Xue, and Limao Zhang (2022). "An encoder-decoder deep learning method for multi-class object segmentation from 3D tunnel point clouds". In: *Automation in Construction*. URL: https://doi.org/10.1016/j.autcon.2022.104187.

Haiyong Jiang, Feilong Yan, Jianfei Cai, Jianmin Zheng, and Jun Xiao (2020). "End-to-end 3D point cloud instance segmentation without detection". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. URL: 10.1109/CVPR42600.2020.01281.

Li Jiang, Hengshuang Zhao, Shaoshuai Shi, Shu Liu, Chi-Wing Fu, and Jiaya Jia (2020). "Pointgroup: Dual-set point grouping for 3d instance segmentation". In: *Proceedings of the IEEE/CVF conference on computer vision and Pattern recognition*. URL: 10.1109/CVPR42600.2020.00492.

Mingqiang Jiang, Yulan Wu, Tong Zhao, Zhiqiang Zhao, and Chunming Lu (2018). "PointSIFT: A SIFT-like network module for 3D point cloud semantic segmentation". In: *arXiv preprint arXiv:1807.00652*. URL: https://doi.org/10.48550/arXiv.1807.00652.

Yuhan Jiang, Sisi Han, and Yong Bai (2021). "Building and infrastructure defect detection and visualization using drone and deep learning technologies". In: *Journal of Performance of Constructed Facilities*. URL: https://doi.org/10.1061/(ASCE)CF.1943-5509.0001652.

Xiaoyun Jing (2023). "Registration Based Feature Core-Set Model for Defect Detection". In: *2023 China Automation Congress (CAC)*. IEEE. URL: Registration%20Based%20Feature%20Core-Set%20Model%20for%20Defect%20Detection.

Yixiong Jing, Brian Sheil, and Sinan Acikgoz (2022). "Segmentation of large-scale masonry arch bridge point clouds with a synthetic simulator and the BridgeNet neural network". In: *Automation in Construction*. URL: https://doi.org/10.1016/j.autcon.2022.104459.

Yixiong Jing, Brian Sheil, and Sinan Acikgoz (2023). "A Method to Generate Realistic Synthetic Point Clouds of Damaged Single-Span Masonry Arch Bridges". In: *International Conference on Structural Analysis of Historical Constructions*. Springer. URL: https://doi.org/10.1007/978-3-031-39603-8_36.

Yixiong Jing, Brian Sheil, and Sinan Acikgoz (2024a). "A lightweight Transformer-based neural network for large-scale masonry arch bridge point cloud segmentation". In: *Computer-Aided Civil and Infrastructure Engineering*. URL: https://doi.org/10.1111/mice.13201.

Yixiong Jing, Jia-Xing Zhong, Brian Sheil, and Sinan Acikgoz (2024b). "Anomaly detection of cracks in synthetic masonry arch bridge point clouds using fast point feature histograms and PatchCore". In: *Automation in Construction*.

Zhizhong Kang, Liqiang Zhang, Lei Tuo, Baoqian Wang, and Jinlei Chen (2014). "Continuous extraction of subway tunnel cross sections based on terrestrial point clouds". In: *Remote Sensing*.

Nick Kanopoulos, Nagesh Vasanthavada, and Robert L Baker (1988). "Design of an image edge detection filter using the Sobel operator". In: *IEEE Journal of solid-state circuits*.

Varun Kasireddy and Burcu Akinci (2023). "Encoding 3d point contexts for self-supervised spall classification using 3d bridge point clouds". In: *Journal of Computing in Civil Engineering*. URL: https://doi.org/10.1061/JCCEE5.CPENG-5041.

Stamos Katsigiannis, Saleh Seyedzadeh, Andrew Agapiou, and Naeem Ramzan (2023). "Deep learning for crack detection on masonry façades using limited data and transfer learning". In: *Journal of Building Engineering*. URL: https://doi.org/10.1016/j.jobe.2023.107105.

Ayoub Keshmiry, Sahar Hassani, Ulrike Dackermann, and Jianchun Li (2024). "Assessment, repair, and retrofitting of masonry structures: A comprehensive review". In: *Construction and Building Materials*. URL: https://doi.org/10.1016/j.conbuildmat.2024.137380.

In-Ho Kim, Haemin Jeon, Seung-Chan Baek, Won-Hwa Hong, and Hyung-Jo Jung (2018). "Application of crack identification techniques for an aging concrete bridge

inspection using an unmanned aerial vehicle". In: *Sensors*. URL: https://doi.org/10.3390/s18061881.

Hyunjun Kim, Jinyoung Yoon, and Sung-Han Sim (2020). "Automated bridge component recognition from point clouds using deep learning". In: *Structural Control and Health Monitoring*. URL: https://doi.org/10.1002/stc.2591.

Durk P Kingma and Prafulla Dhariwal (2018). "Glow: Generative flow with invertible 1x1 convolutions". In: *Advances in neural information processing systems*. URL: https://doi.org/10.48550/arXiv.1807.03039.

Maksim Kolodiazhnyi, Anna Vorontsova, Anton Konushin, and Danila Rukhovich (2024). "Top-down beats bottom-up in 3d instance segmentation". In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. URL: https://doi.org/10.48550/arXiv.2302.02871.

Jacek Komorowski (2021). "Minkloc3d: Point cloud based large-scale place recognition". In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. URL: https://doi.org/10.48550/arXiv.2011.04530.

Kamil Korus, Thomas Czerniawski, and Marek Salamak (2023). "Visual programming simulator for producing realistic labeled point clouds from digital infrastructure models". In: *Automation in Construction*.

Bakhytzhan Kulambayev, Magzat Nurlybek, Gulnar Astaubayeva, Gulnara Tleuberdiyeva, Serik Zholdasbayev, and Abdimukhan Tolep (2023). "Real-time road surface damage detection framework based on mask r-cnn model". In: *International Journal of Advanced Computer Science and Applications*. URL: 10.14569/IJACSA.2023.0140979.

Abhijit Kundu, Xiaoqi Yin, Alireza Fathi, David Ross, Brian Brewington, Thomas Funkhouser, and Caroline Pantofaru (2020). "Virtual multi-view fusion for 3d semantic segmentation". In: *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIV 16*. Springer. URL: https://doi.org/10.1007/978-3-030-58586-0_31.

Debra F Laefer, Linh Truong-Hong, Hamish Carr, and Manmeet Singh (2014). "Crack detection limits in unit based masonry with terrestrial laser scanning". In: *Ndt & E International*. URL: https://doi.org/10.1016/j.ndteint.2013.11.001.

Jean Lahoud, Bernard Ghanem, Marc Pollefeys, and Martin R Oswald (2019). "3d instance segmentation via multi-task metric learning". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. URL: 10.1109/ICCV.2019.00935.

Xin Lai, Jianhui Liu, Li Jiang, Liwei Wang, Hengshuang Zhao, Shu Liu, Xiaojuan Qi, and Jiaya Jia (June 2022a). "Stratified Transformer for 3D Point Cloud Segmentation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. URL: https://doi.org/10.48550/arXiv.2203.14508.

Xin Lai, Jianhui Liu, Li Jiang, Liwei Wang, Hengshuang Zhao, Shu Liu, Xiaojuan Qi, and Jiaya Jia (2022b). "Stratified transformer for 3d point cloud segmentation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Daniel Lamas, Andrés Justo, Mario Soilán, and Belén Riveiro (2024). "Automated production of synthetic point clouds of truss bridges for semantic and instance segmentation using deep learning models". In: *Automation in Construction*.

Loic Landrieu and Mounir Boussaha (2019). "Point cloud oversegmentation with graph-structured deep metric learning". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. URL: https://doi.org/10.1109/CVPR.2019.00762.

Loic Landrieu and Martin Simonovsky (2018). "Large-scale point cloud semantic segmentation with superpoint graphs". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. URL: https://doi.org/10.1109/CVPR.2018.00479.

Reza Latifi, Marijana Hadzima-Nyarko, Dorin Radu, and Rahimeh Rouhi (2023). "A brief overview on crack patterns, repair and strengthening of historical masonry structures". In: *Materials*. URL: https://doi.org/10.3390/ma16051882.

Felix Järemo Lawin, Martin Danelljan, Patrik Tosteberg, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg (2017). "Deep projective 3D semantic segmentation". In: *Computer Analysis of Images and Patterns: 17th International Conference, CAIP 2017, Ystad, Sweden, August 22-24, 2017, Proceedings, Part I 17*. Springer. URL: https://doi.org/10.1007/978-3-319-64689-3_8.

Eddie Taewan Lee, Zhaoyan Fan, and Burak Sencer (2023). "A new approach to detect surface defects from 3D point cloud data with surface normal Gabor filter (SNGF)". In: *Journal of Manufacturing Processes*. URL: https://doi.org/10.1016/j.jmapro.2023.02.047.

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh (2019). "Set transformer: A framework for attention-based permutation-invariant neural networks". In: *International conference on machine learning*. PMLR.

Kisu Lee, Goopyo Hong, Lee Sael, Sanghyo Lee, and Ha Young Kim (2020). "MultiDefectNet: Multi-class defect detection of building façade based on deep convolutional neural network". In: *Sustainability*. URL: https://doi.org/10.3390/su12229785.

Bo Li (2017a). "3d fully convolutional network for vehicle detection in point cloud". In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. URL: https://doi.org/10.1109/IROS.2017.8205955.

Bo Li (2017b). "3d fully convolutional network for vehicle detection in point cloud". In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE.

Bo Li, Tianlei Zhang, and Tian Xia (2016). "Vehicle detection from 3d lidar using fully convolutional network". In: *arXiv preprint arXiv:1608.07916*. URL: https://doi.org/10.48550/arXiv.1608.07916.

Gang Li, Biao Ma, Shuanhai He, Xueli Ren, and Qiangwei Liu (2020a). "Automatic Tunnel Crack Detection Based on U-Net and a Convolutional Neural Network with Alternately Updated Clique". In: *Sensors*. URL: https://www.mdpi.com/1424-8220/20/3/717.

Gang Li, Biao Ma, Shuanhai He, Xueli Ren, and Qiangwei Liu (2020b). "Automatic tunnel crack detection based on u-net and a convolutional neural network with alternately updated clique". In: *Sensors*. URL: https://doi.org/10.3390/s20030717.

Tao Li, Yi Rui, Hehua Zhu, Linhai Lu, and Xiaojun Li (2024). "Comprehensive digital twin for infrastructure: A novel ontology and graph-based modelling paradigm". In: *Advanced Engineering Informatics*. URL: https://doi.org/10.1016/j.aei.2024.102747.

Wenqiao Li, Xiaohao Xu, Yao Gu, Bozhong Zheng, Shenghua Gao, and Yingna Wu (2024). "Towards Scalable 3D Anomaly Detection and Localization: A Benchmark via 3D Anomaly Synthesis and A Self-Supervised Learning Network". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. URL: 10.1109/CVPR52733.2024.02096.

Yuanhang Li, Junfeng Yao, Kai Chen, Han Zhang, Xiaodong Sun, Quan Qian, and Xing Wu (2024). "A Collaborative Anomaly Localization Method Based on Multi-Modal Images". In: *2024 27th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*. IEEE. URL: 10.1109/CSCWD61410.2024.10580587.

Zhihao Liang, Zhihao Li, Songcen Xu, Mingkui Tan, and Kui Jia (2021). "Instance Segmentation in 3D Scenes using Semantic Superpoint Tree Networks". In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*.

Jianghai Liao, Yuanhao Yue, Dejin Zhang, Wei Tu, Rui Cao, Qin Zou, and Qingquan Li (2022a). "Automatic Tunnel Crack Inspection Using an Efficient Mobile Imaging Module and a Lightweight CNN". In: *IEEE Transactions on Intelligent Transportation Systems*. URL: 10.1109/TITS.2021.3138428.

Jianghai Liao, Yuanhao Yue, Dejin Zhang, Wei Tu, Rui Cao, Qin Zou, and Qingquan Li (2022b). "Automatic tunnel crack inspection using an efficient mobile imaging module and a lightweight CNN". In: *IEEE Transactions on Intelligent Transportation Systems*.

Jinhua Lin, Lin Ma, and Yu Yao (2022). "A spectrum-domain instance segmentation model for casting defects". In: *Integrated Computer-Aided Engineering*.

T. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar (2017). "Focal Loss for Dense Object Detection". In: *2017 IEEE International Conference on Computer Vision (ICCV)*. IEEE Computer Society. URL: https://doi.ieeecomputersociety.org/10.1109/ICCV.2017.324.

Wei Lin, Pan Li, Xiongyao Xie, Yuyang Cao, and Yangbin Zhang (2023). "A novel back-analysis approach for the external loads on shield tunnel lining in service based on monitored deformation". In: *Structural Control and Health Monitoring*.

Wei Lin, Brian Sheil, Xiongyao Xie, Yangbin Zhang, and Yuyang Cao (2024a). "Semantic segmentation of large-scale segmental lining point clouds using 3D deep learning". In: *IOP Conference Series: Earth and Environmental Science*. IOP Publishing. URL: https://doi.org/10.1088/1755-1315/1337/1/012026.

Wei Lin, Brian Sheil, Pin Zhang, Biao Zhou, Cheng Wang, and Xiongyao Xie (2024b). "Seg2Tunnel: A hierarchical point cloud dataset and benchmarks for segmentation of segmental tunnel linings". In: *Tunnelling and Underground Space Technology*. URL: https://doi.org/10.1016/j.tust.2024.105735.

Wei Lin, Brian Sheil, Pin Zhang, Biao Zhou, Cheng Wang, and Xiongyao Xie (2024c). "Seg2Tunnel: A hierarchical point cloud dataset and benchmarks for segmentation of segmental tunnel linings". In: *Tunnelling and Underground Space Technology*. URL: https://www.sciencedirect.com/science/article/pii/S0886779824001536.

Wei Lin, Brian Sheil, Pin Zhang, Biao Zhou, Cheng Wang, and Xiongyao Xie (2024d). "Seg2Tunnel: A hierarchical point cloud dataset and benchmarks for segmentation of segmental tunnel linings". In: *Tunnelling and Underground Space Technology*.

Cheng-Yuan Liou, Wei-Chen Cheng, Jiun-Wei Liou, and Daw-Ran Liou (2014). "Autoencoder for words". In: *Neurocomputing*. URL: https://www.sciencedirect.com/science/article/pii/S0925231214003658.

Jiaqi Liu, Guoyang Xie, Ruitao Chen, Xinpeng Li, Jinbao Wang, Yong Liu, Chengjie Wang, and Feng Zheng (2024). "Real3d-ad: A dataset of point cloud anomaly detection". In: *Advances in Neural Information Processing Systems*. URL: https://proceedings.neurips.cc/paper_files/paper/2023/file/611b896d447df43c898062358df4c114-Paper-Datasets_and_Benchmarks.pdf.

Kai Liu, Zhixiang Gao, Fengqiang Lin, and Ben M. Chen (2020). "FG-Net: Fast large-scale LiDAR point clouds understanding network leveraging correlated feature mining and geometric-aware modelling". In: *arXiv preprint arXiv:2012.09439*. URL: https://doi.org/10.48550/arXiv.2012.09439.

Kangcheng Liu, Zhi Gao, Feng Lin, and Ben M Chen (2020). "Fg-net: Fast large-scale lidar point clouds understanding network leveraging correlated feature mining and geometric-aware modelling". In: *arXiv preprint arXiv:2012.09439*.

W Liu, S Chen, and E Hauser (2011). "LiDAR-based bridge structure defect detection". In: *Experimental Techniques*. URL: https://doi.org/10.1111/j.1747-1567.2010.00644.x.

Xinhai Liu, Zhizhong Han, Yu-Shen Liu, and Matthias Zwicker (2019). "Point2sequence: Learning the shape representation of 3d point clouds with an attention-based sequence to sequence network". In: *Proceedings of the AAAI conference on artificial intelligence*.

Yahui Liu, Bin Tian, Yisheng Lv, Lingxi Li, and Fei-Yue Wang (2023). "Point cloud classification using content-based transformer via clustering in feature space". In: *IEEE/CAA Journal of Automatica Sinica*.

Yiheng Liu, Tianle Han, Siyuan Ma, Jiayue Zhang, Yuanyuan Yang, Jiaming Tian, Hao He, Antong Li, Mengshen He, Zhengliang Liu, et al. (2023). "Summary of chatgpt/gpt-4 research and perspective towards the future of large language models". In: *arXiv preprint arXiv:2304.01852*.

Yiyan Liu, Harvey John Burd, Derya Burcu Gulen, Korhan Deniz Dalgic, Ben Gilson, Alper Ilki, and Sinan Acikgoz (2024). "Estimation of settlement-induced damage in masonry buildings from displacement measurements". In: *Available at SSRN 4765699*. URL: http://dx.doi.org/10.2139/ssrn.4765699.

Yuelin Liu, Bin Fan, Gaofeng Meng, Junwei Lu, Shiming Xiang, and Chunhong Pan (2019a). "DensePoint: Learning densely contextual representation for efficient point

cloud processing". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. URL: https://doi.org/10.1109/ICCV.2019.00534.

Yuelin Liu, Bin Fan, Shiming Xiang, and Chunhong Pan (2019b). "Relation-shape convolutional neural network for point cloud analysis". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. URL: https://doi.org/10.1109/CVPR.2019.00910.

Zhenqing Liu, Yiwen Cao, Yize Wang, and Wei Wang (2019). "Computer vision-based concrete crack detection using U-net fully convolutional networks". In: *Automation in Construction*. URL: https://doi.org/10.1016/j.autcon.2019.04.005.

Zhijian Liu, Haotian Tang, Yujun Lin, and Song Han (2019a). "Point-Voxel CNN for Efficient 3D Deep Learning". In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper_files/paper/2019/file/5737034557ef5b8c02c0e46513b98f90-Paper.pdf.

Zhijian Liu, Haotian Tang, Yujun Lin, and Song Han (2019b). "Point-voxel CNN for efficient 3D deep learning". In: *Advances in Neural Information Processing Systems*. URL: https://doi.org/10.48550/arXiv.1907.03739.

K Berker Logoglu, Sinan Kalkan, and Alptekin Temizel (2016). "Cospair: colored histograms of spatial concentric surflet-pairs for 3d object recognition". In: *Robotics and Autonomous Systems*. URL: https://doi.org/10.1016/j.robot.2015.09.027.

Paulo B Lourenço and Angelo Gaetani (2022). *Finite element analysis for building assessment: Advanced use and practical recommendations*. Routledge. URL: 10.1201/9780429341564.

Paulo B Lourenço and Jan G Rots (1997). "Multisurface interface model for analysis of masonry structures". In: *Journal of engineering mechanics*. URL: https://doi.org/10.1061/(ASCE)0733-9399(1997)123:7(660).

Dimitrios Loverdos and Vasilis Sarhosis (2022a). "Automatic image-based brick segmentation and crack detection of masonry walls using machine learning". In: *Automation in Construction*. URL: https://doi.org/10.1016/j.autcon.2022.104389.

Dimitrios Loverdos and Vasilis Sarhosis (2022b). "Automatic image-based brick segmentation and crack detection of masonry walls using machine learning". In: *Automation in Construction*. URL: https://doi.org/10.1016/j.autcon.2022.104389.

D.G. Lowe (1999). “Object recognition from local scale-invariant features”. In: *Proceedings of the Seventh IEEE International Conference on Computer Vision*. URL: 10.1109/ICCV.1999.790410.

Ruodan Lu, Ioannis Brilakis, and Campbell R Middleton (2019a). “Detection of structural components in point clouds of existing RC bridges”. In: *Computer-Aided Civil and Infrastructure Engineering*. URL: https://doi.org/10.1111/mice.12407.

Ruodan Lu, Ioannis Brilakis, and Campbell R Middleton (2019b). “Detection of structural components in point clouds of existing RC bridges”. In: *Computer-Aided Civil and Infrastructure Engineering*.

Yuhang Lu, Qi Jiang, Runnan Chen, Yuenan Hou, Xinge Zhu, and Yuexin Ma (2023). “See more and know more: Zero-shot point cloud segmentation via multi-modal visual data”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. URL: https://openaccess.thecvf.com/content/ICCV2023/html/Lu_See_More_and_Know_More_Zero-shot_Point_Cloud_Segmentation_via_ICCV_2023_paper.html.

Izabela Lubowiecka, Julia Armesto, Pedro Arias, and Henrique Lorenzo (2009). “Historic bridge modelling using laser scanning, ground penetrating radar and finite element methods in the context of structural dynamics”. In: *Engineering Structures*. URL: https://doi.org/10.1016/j.engstruct.2009.06.018.

Thomas Luhmann, Stuart Robson, Stephen Kyle, and Jan Boehm (2020a). *Close-Range Photogrammetry and 3D Imaging*. Berlin, Boston: De Gruyter. URL: https://doi.org/10.1515/9783110607253.

Thomas Luhmann, Stuart Robson, Stephen Kyle, and Jan Boehm (2020b). *Close-Range Photogrammetry and 3D Imaging*. Berlin, Boston: De Gruyter. URL: https://doi.org/10.1515/9783110607253.

Jianwen Ma, Tomasz Czerniawski, and Fernanda Leite (2020). “Semantic segmentation of point clouds of building interiors with deep learning: Augmenting training datasets with synthetic BIM-based point clouds”. In: *Automation in Construction*. URL: https://doi.org/10.1016/j.autcon.2020.103144.

Prasanta Chandra Mahalanobis (2018). “On the generalized distance in statistics”. In: *Sankhyā: The Indian Journal of Statistics, Series A (2008-)*. URL: https://doi.org/10.1007/s13171-019-00164-5.

M. Mandirola, C. Casarotti, S. Peloso, I. Lanese, E. Brunesi, and I. Senaldi (2022). “Use of UAS for damage inspection and assessment of bridge infrastructures”. In:

International Journal of Disaster Risk Reduction. URL: https://doi.org/10.1016/j.ijdrr.2022.102824.

Francesca Matrone, Eleonora Grilli, Massimo Martini, Marina Paolanti, Roberto Pierdicca, and Fabio Remondino (2020). "Comparing machine and deep learning methods for large 3D heritage semantic segmentation". In: *ISPRS International Journal of Geo-Information*. URL: https://doi.org/10.3390/ijgi9090535.

Albert Matveev, Alexey Artemov, Denis Zorin, and Evgeny Burnaev (2020). "Geometric attention for prediction of differential properties in 3d point clouds". In: *Artificial Neural Networks in Pattern Recognition: 9th IAPR TC3 Workshop, ANNPR 2020, Winterthur, Switzerland, September 2–4, 2020, Proceedings 9*. Springer.

Chris Melbourne, Liam D. McKibbins, Nasir Sawar, and Christine S. Gaillard (2006). *Masonry arch bridges: Condition appraisal and remedial treatment*. London: CIRIA.

Moritz Menze and Andreas Geiger (2015). "Object scene flow for autonomous vehicles". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Mohammad Ebrahim Mohammadi, Richard L Wood, and Christine E Wittich (2019). "Non-temporal point cloud analysis for surface damage in civil structures". In: *ISPRS International Journal of Geo-Information*. URL: https://doi.org/10.3390/ijgi8120527.

Mark Moore, Brent M Phares, Benjamin Graybeal, Dennis Rolander, Glenn Washer, Janney Wiss, et al. (2001). *Reliability of visual inspection for highway bridges, volume I*. Tech. rep. Turner-Fairbank Highway Research Center.

Chiara Morbidoni, Roberto Pierdicca, Matteo Paolanti, Raffaella Quattrini, and Roberto Mammoli (2020). "Learning from synthetic point cloud data for historical buildings semantic segmentation". In: *Journal on Computing and Cultural Heritage (JOCCH)*. URL: https://doi.org/10.1145/3409262.

Hafiz Suliman Munawar, Ahmed WA Hammad, S Travis Waller, and Md Rafiqul Islam (2022). "Modern crack detection for bridge infrastructure maintenance using machine learning". In: *Human-Centric Intelligent Systems*. URL: https://doi.org/10.1007/s44230-022-00009-9.

Yasutaka Narazaki, Vedhus Hoskere, Koji Yoshida, Billie F. Spencer, and Yozo Fujino (2021). "Synthetic environments for vision-based structural condition assessment of Japanese high-speed railway viaducts". In: *Mechanical Systems and Signal Processing*. URL: https://doi.org/10.1016/j.ymssp.2021.107850.

Majid Nasrollahi, Neshat Bolourian, and Amin Hammad (2019). "Concrete surface defect detection using deep neural network based on lidar scanning". In: *Proceedings of the CSCE Annual Conference, Laval, Greater Montreal, QC, Canada*.

Duc Tam Nguyen, Zhongyu Lou, Michael Klar, and Thomas Brox (2019). "Anomaly detection with multiple-hypotheses predictions". In: *International Conference on Machine Learning*. PMLR. URL: https://doi.org/10.48550/arXiv.1810.13292.

Takafumi Nishikawa, Junji Yoshida, Toshiyuki Sugiyama, and Yozo Fujino (2012). "Concrete crack detection by multiple sequential image filtering". In: *Computer-Aided Civil and Infrastructure Engineering*. URL: https://doi.org/10.1111/j.1467-8667.2011.00716.x.

Sang-jin Oh, Min-jae Jung, Chaeog Lim, and Sung-chul Shin (2020). "Automatic Detection of Welding Defects Using Faster R-CNN". In: *Applied Sciences*. URL: https://www.mdpi.com/2076-3417/10/23/8629.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals (2018). "Representation learning with contrastive predictive coding". In: *arXiv preprint arXiv:1807.03748*. URL: https://doi.org/10.48550/arXiv.1807.03748.

Zoltán Orbán (2004). "Assessment, reliability and maintenance of masonry arch railway bridges in Europe". In: *Arch Bridges IV–Advances in Assessment, Structural Design and Construction. Eds: P. Roca and C. Molins, Barcelona*. URL: https://api.semanticscholar.org/CorpusID:8359025.

Zoltán Orbán and Marc Gutermann (2009). "Assessment of masonry arch railway bridges using non-destructive in-situ testing methods". In: *Engineering Structures*. URL: https://www.sciencedirect.com/science/article/pii/S0141029609001606.

John R. Page (1996). *A GUIDE TO REPAIR AND STRENGTHENING OF MASONRY ARCH BRIDGES*. Tech. rep. THE FUTURE OF TRANSPORT(TRL). URL: https://api.semanticscholar.org/CorpusID:106570745.

Yatian Pang, Wenxiao Wang, Francis EH Tay, Wei Liu, Yonghong Tian, and Li Yuan (2022). "Masked autoencoders for point cloud self-supervised learning". In: *European conference on computer vision*. Springer. URL: https://doi.org/10.1007/978-3-031-20086-1_35.

Chunghyun Park, Yoonwoo Jeong, Minsu Cho, and Jaesik Park (2022). "Fast point transformer". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Wei Yan Peh, Prasanth Thangavel, Yuanyuan Yao, John Thomas, Yee Leng Tan, and Justin Dauwels (2022). "Multi-center Assessment of CNN-Transformer with Belief Matching Loss for Patient-independent Seizure Detection in Scalp and Intracranial EEG". In: *arXiv preprint arXiv:2208.00025*.

Husein Perez, Joseph HM Tah, and Amir Mosavi (2019). "Deep learning for detecting building defects using convolutional neural networks". In: *Sensors*. URL: https://doi.org/10.3390/s19163556.

Arianna Pesci, Giuseppe Casula, and Enzo Boschi (2011). "Laser scanning the Garisenda and Asinelli towers in Bologna (Italy): Detailed deformation patterns of two ancient leaning buildings". In: *Journal of cultural heritage*. URL: https://doi.org/10.1016/j.culher.2011.01.002.

Brent M. Phares, Glenn A. Washer, Dennis D. Rolander, Benjamin A. Graybeal, and Mark Moore (2004). "Routine Highway Bridge Inspection Condition Documentation Accuracy and Reliability". In: *Journal of Bridge Engineering*. URL: 10.1061/(ASCE)1084-0702(2004)9:4(403).

Roberto Pierdicca, Marco Mameli, Eva Savina Malinverni, Marina Paolanti, and Emanuele Frontoni (2019). "Automatic Generation of Point Cloud Synthetic Dataset for Historical Building Representation". In: *Augmented Reality, Virtual Reality, and Computer Graphics*. Ed. by Lucio Tommaso De Paolis and Patrick Bourdot. Cham: Springer International Publishing. URL: 10.1007/978-3-030-25965-5_16.

Roberto Pierdicca, Marina Paolanti, Francesca Matrone, Massimo Martini, Christian Morbidoni, Eva Savina Malinverni, Emanuele Frontoni, and Andrea Maria Lingua (2020). "Point cloud semantic segmentation using a deep learning framework for cultural heritage". In: *Remote Sensing*. URL: https://doi.org/10.3390/rs12061005.

Eftychios Protopapadakis, Athanasios Voulodimos, Anastasios Doulamis, Nikolaos Doulamis, and Tania Stathaki (2019). "Automatic crack detection for tunnel inspection using deep learning and heuristic image post-processing". In: *Applied intelligence*. URL: https://doi.org/10.1007/s10489-018-01396-y.

Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas (2017a). "PointNet: Deep learning on point sets for 3D classification and segmentation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. URL: https://doi.org/10.1109/CVPR.2017.16.

Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas (2017b). "PointNet++: Deep hierarchical feature learning on point sets in a metric space". In: *Advances in*

Neural Information Processing Systems. URL: https://dl.acm.org/doi/10.5555/3295222.3295263.

Zheng Qin, Hao Yu, Changjian Wang, Yulan Guo, Yuxing Peng, and Kai Xu (2022). "Geometric transformer for fast and robust point cloud registration". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*.

Markus N Rabe and Charles Staats (2021). "Self-attention Does Not Need $O(n^2)$ Memory". In: *arXiv preprint arXiv:2112.05682*.

RAC Foundation (2021). *The Condition of Road Bridges in Great Britain: 2021 Annual Report*. Tech. rep. RAC Foundation. URL: https://www.racfoundation.org/research/economy/substandard-road-bridges-in-great-britain.

Aravinda S Rao, Tuan Nguyen, Marimuthu Palaniswami, and Tuan Ngo (2021). "Vision-based automated crack detection using convolutional neural networks for condition assessment of infrastructure". In: *Structural Health Monitoring*. URL: https://doi.org/10.1177/1475921720965445.

Stefan Reitmann, Lorenzo Neumann, and Bernhard Jung (2021a). "BLAINDER—A Blender AI Add-On for Generation of Semantically Labeled Depth-Sensing Data". In: *Sensors*. URL: 10.3390/s21062144.

Stefan Reitmann, Lorenzo Neumann, and Bernhard Jung (2021b). "Blainder—a blender ai add-on for generation of semantically labeled depth-sensing data". In: *Sensors*. URL: https://doi.org/10.3390/s21062144.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun (2016). "Faster R-CNN: Towards real-time object detection with region proposal networks". In: *IEEE transactions on pattern analysis and machine intelligence*. URL: 10.1109/TPAMI.2016.2577031.

Gernot Riegler, A. Osman Ulusoy, and Andreas Geiger (2017). "Octnet: Learning deep 3D representations at high resolutions". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. URL: https://doi.org/10.1109/CVPR.2017.701.

Nicolae-Cătălin Ristea, Neelu Madan, Radu Tudor Ionescu, Kamal Nasrollahi, Fahad Shahbaz Khan, Thomas B Moeslund, and Mubarak Shah (2022). "Self-supervised predictive convolutional attentive block for anomaly detection". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. URL: 10.1109/CVPR52688.2022.01321.

B Riveiro, MJ DeJong, and B Conde (2016). "Automated processing of large point clouds for structural health monitoring of masonry arch bridges". In: *Automation in Construction*. URL: https://doi.org/10.1016/j.autcon.2016.02.009.

B Riveiro, P Morer, P Arias, and I De Arteaga (2011). "Terrestrial laser scanning and limit analysis of masonry arch bridges". In: *Construction and building materials*. URL: https://doi.org/10.1016/j.conbuildmat.2010.11.094.

Paul Roetzer and Florian Bernard (2024). "SpiderMatch: 3D Shape Matching with Global Optimality and Geometric Consistency". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. URL: 10.1109/CVPR52733.2024.01378.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox (2015). "U-net: Convolutional networks for biomedical image segmentation". In: *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*. Springer.

Karsten Roth, Latha Pemula, Joaquin Zepeda, Bernhard Schölkopf, Thomas Brox, and Peter Gehler (2022). "Towards total recall in industrial anomaly detection". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. URL: 10.1109/CVPR52688.2022.01392.

Marco Rudolph, Tom Wehrbein, Bodo Rosenhahn, and Bastian Wandt (2023). "Asymmetric student-teacher networks for industrial anomaly detection". In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. URL: 10.1109/TIM.2023.3338681.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. (2015). "Imagenet large scale visual recognition challenge". In: *International journal of computer vision*. URL: https://link.springer.com/article/10.1007/s11263-015-0816-y.

Radu Bogdan Rusu, Nico Blodow, and Michael Beetz (2009). "Fast point feature histograms (FPFH) for 3D registration". In: *2009 IEEE international conference on robotics and automation*. IEEE. URL: 10.1109/ROBOT.2009.5152473.

Mayu Sakurada and Takehisa Yairi (2014). "Anomaly detection using autoencoders with nonlinear dimensionality reduction". In: *Proceedings of the MLSDA 2014 2nd workshop on machine learning for sensory data analysis*. URL: https://doi.org/10.1145/2689746.268974.

Thomas Schlegl, Philipp Seeböck, Sebastian M Waldstein, Georg Langs, and Ursula Schmidt-Erfurth (2019). "f-AnoGAN: Fast unsupervised anomaly detection

with generative adversarial networks". In: *Medical image analysis*. URL: https://doi.org/10.1016/j.media.2019.01.010.

Ruwen Schnabel, Roland Wahl, and Reinhard Klein (2007). "Efficient RANSAC for point-cloud shape detection". In: *Computer graphics forum*. Wiley Online Library. URL: https://doi.org/10.1111/j.1467-8659.2007.01016.x.

Jonas Schult, Francis Engelmann, Alexander Hermans, Or Litany, Siyu Tang, and Bastian Leibe (2023). "Mask3D: Mask Transformer for 3D Semantic Instance Segmentation". In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*.

Ozan Sener and Silvio Savarese (2017). "Active learning for convolutional neural networks: A core-set approach". In: *arXiv preprint arXiv:1708.00489*. URL: https://doi.org/10.48550/arXiv.1708.00489.

Elyas Asadi Shamsabadi, Chang Xu, and Daniel Dias-da-Costa (2022). "Robust crack detection in masonry structures with Transformers". In: *Measurement*. URL: https://doi.org/10.1016/j.measurement.2022.111590.

Mohsin M Shanoer and Fanar M Abed (2018). "Evaluate 3D laser point clouds registration for cultural heritage documentation". In: *The Egyptian Journal of Remote Sensing and Space Science*. URL: https://doi.org/10.1016/j.ejrs.2017.11.007.

Shashidhar V. Sheshappanavar and Chandra Kambhamettu (2020). "A novel local geometry capture in PointNet++ for 3D classification". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. URL: https://doi.org/10.1109/CVPRW50498.2020.00139.

Martin Simonovsky and Nikos Komodakis (2017). "Dynamic edge-conditioned filters in convolutional neural networks on graphs". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. URL: https://doi.org/10.1109/CVPR.2017.11.

M. Soilán, A. Nóvoa, A. Sánchez-Rodríguez, B. Riveiro, and P. Arias (2020). "SEMANTIC SEGMENTATION OF POINT CLOUDS WITH POINTNET AND KPCONV ARCHITECTURES APPLIED TO RAILWAY TUNNELS". In: *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*. URL: https://isprs-annals.copernicus.org/articles/V-2-2020/281/2020/.

Mercedes Solla, Susana Lagüela, Belén Riveiro, and Henrique Lorenzo (2013). "Non-destructive testing for the analysis of moisture in the masonry arch bridge of Lubians (Spain)". In: *Structural Control and Health Monitoring*. URL: https://doi.org/10.1002/stc.1545.

Mercedes Solla, H Lorenzo, FI Rial, and A Novo (2012). “Ground-penetrating radar for the structural evaluation of masonry bridges: Results and interpretational tools”. In: *Construction and Building Materials*.

Paulina Stałowska, Czesław Suchocki, and Miłosława Rutkowska (2022). “Crack detection in building walls based on geometric and radiometric point cloud information”. In: *Automation in Construction*. URL: https://doi.org/10.1016/j.autcon.2021.104065.

Haotian Tang, Zhijian Liu, Shengyu Zhao, Yujun Lin, Ji Lin, Hanrui Wang, and Song Han (2020). “Searching efficient 3d architectures with sparse point-voxel convolution”. In: *European conference on computer vision*. Springer.

Lyne Tchapmi, Christopher Choy, Iro Armeni, JunYoung Gwak, and Silvio Savarese (Oct. 2017). “ SEGCloud: Semantic Segmentation of 3D Point Clouds ”. In: *2017 International Conference on 3D Vision (3DV)*. Los Alamitos, CA, USA: IEEE Computer Society. URL: https://doi.ieeecomputersociety.org/10.1109/3DV.2017.00067.

Simone Teruggi, Eleonora Grilli, Michele Russo, Francesco Fassi, and Fabio Remondino (2020). “A hierarchical machine learning approach for multi-level and multi-resolution 3D point cloud classification”. In: *Remote Sensing*. URL: https://doi.org/10.3390/rs12162598.

Hugues Thomas, Charles R. Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J. Guibas (2019). “KPConv: Flexible and deformable convolution for point clouds”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. URL: https://doi.org/10.1109/ICCV.2019.00651.

M Tran, J Hall, A Hickford, R Nicholls, D Alderson, and S Barr (2014). *National infrastructure assessment: analysis of options for infrastructure provision in Great Britain*. Tech. rep. University of Sussex. URL: https://hdl.handle.net/10779/uos.23415389.v1.

Enrico Tubaldi, Lorenzo Macorini, and Bassam A Izzuddin (2018). “Three-dimensional mesoscale modelling of multi-span masonry arch bridges subjected to scour”. In: *Engineering Structures*. URL: 10.1016/j.engstruct.2018.03.031.

Radu Tudor Ionescu, Sorina Smeureanu, Bogdan Alexe, and Marius Popescu (2017). “Unmasking the abnormal events in video”. In: *Proceedings of the IEEE international conference on computer vision*. URL: 10.1109/ICCV.2017.315.

Enrique Valero, Frédéric Bosché, and Alan Forster (2018). “Automatic segmentation of 3D point clouds of rubble masonry walls, and its application to building surveying,

repair and maintenance". In: *Automation in Construction*. URL: https://doi.org/10.1016/j.autcon.2018.08.018.

Enrique Valero, Alan Forster, Frédéric Bosché, Ewan Hyslop, Lyn Wilson, and Aurélie Turmel (2019). "Automated defect detection and classification in ashlar masonry walls using machine learning". In: *Automation in construction*. URL: https://doi.org/10.1016/j.autcon.2019.102846.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin (2017). "Attention is all you need". In: *Advances in neural information processing systems*. URL: 10.5555/3295222.3295349.

Sara B Walsh, Daniel J Borello, Burcu Guldur, and Jerome F Hajjar (2013). "Data processing of point clouds for object detection for structural engineering applications". In: *Computer-Aided Civil and Infrastructure Engineering*.

Haifeng Wan, Lei Gao, Zhaodi Yuan, Hui Qu, Qirun Sun, Hao Cheng, and Ruibao Wang (2023). "A novel transformer model for surface damage detection and cognition of concrete bridges". In: *Expert Systems with Applications*. URL: https://doi.org/10.1016/j.eswa.2022.119019.

Qian Wan, Liang Gao, Xinyu Li, and Long Wen (2021). "Industrial image anomaly localization based on Gaussian clustering of pretrained feature". In: *IEEE Transactions on Industrial Electronics*. URL: 10.1109/TIE.2021.3094452.

Chao Wang and Yong K Cho (2015). "Smart scanning and near real-time 3D surface modeling of dynamic construction equipment from a point cloud". In: *Automation in Construction*. URL: https://doi.org/10.1016/j.autcon.2014.06.003.

Lei Wang, Yuchun Huang, Yulan Hou, Shenhua Zhang, and Jie Shan (2019). "Graph attention convolution for point cloud semantic segmentation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. URL: https://doi.org/10.1109/CVPR.2019.01054.

Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma (2020). "Linformer: Self-attention with linear complexity". In: *arXiv preprint arXiv:2006.04768*.

Weiyue Wang, Ronald Yu, Qixing Huang, and Ulrich Neumann (2018). "SGPN: Similarity group proposal network for 3D point cloud instance segmentation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. URL: https://doi.org/10.48550/arXiv.1711.08588.

Xinlong Wang, Shu Liu, Xiaoyong Shen, Chunhua Shen, and Jiaya Jia (2019). "Associatively segmenting instances and semantics in point clouds". In: *Proceedings*

of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. URL: 10.1109/ICME51207.2021.9428356.

Yanyan Wang, Kechen Song, Menghui Niu, Yanqi Bao, Hongwen Dong, and Yunhui Yan (2022). "Unsupervised defect detection with patch-aware mutual reasoning network in image data". In: *Automation in Construction*. URL: https://doi.org/10.1016/j.autcon.2022.104472.

Yue Wang, Jinlong Peng, Jiangning Zhang, Ran Yi, Yabiao Wang, and Chengjie Wang (2023). "Multimodal Industrial Anomaly Detection via Hybrid Fusion". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. URL: 10.1109/CVPR52729.2023.00776.

Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon (2019). "Dynamic graph CNN for learning on point clouds". In: *ACM Transactions On Graphics (TOG)*. URL: https://doi.org/10.1145/3326362.

Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli (2004). "Image quality assessment: from error visibility to structural similarity". In: *IEEE transactions on image processing*. URL: 10.1109/TIP.2003.819861.

Stanislaw Weglarczyk (2018). "Kernel density estimation and its application". In: *ITM web of conferences*. EDP Sciences.

Rubo Wei, Yonghong Song, and Yuanlin Zhang (2020). "Enhanced faster region convolutional neural networks for steel surface defect detection". In: *ISIJ international*. URL: https://doi.org/10.2355/isijinternational.ISIJINT-2019-335.

Laurence A Wolsey and George L Nemhauser (1999). *Integer and combinatorial optimization*. John Wiley & Sons. URL: 10.1002/9781118627372.

Laurence A Wolsey and George L Nemhauser (2014). *Integer and combinatorial optimization*. John Wiley & Sons. URL: https://books.google.co.uk/books?hl=en&lr=&id=MvBjBAAAQBAJ&oi=fnd&pg=PA14&dq=Integer+and+Combinatorial+Optimization+citation&ots=1fAxCCP7qD&sig=xAY9RvVhO_5hIHsaI9GRAZ-hI6c.

Bichen Wu, Alvin Wan, Xiaolong Yue, and Kurt Keutzer (2018). "Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. URL: https://doi.org/10.1109/ICRA.2018.8462926.

Bichen Wu, Xiao Zhou, Shuang Zhao, Xiaolong Yue, and Kurt Keutzer (2019). "Squeezesegv2: Improved model structure and unsupervised domain adaptation for

road-object segmentation from a lidar point cloud". In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. URL: https://doi.org/10.1109/ICRA.2019.8793495.

Wentao Wu, Zhuang Qi, and Li Fuxin (2019). "PointConv: Deep convolutional networks on 3D point clouds". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. URL: https://doi.org/10.1109/CVPR.2019.00985.

Xiaoyang Wu, Yixing Lao, Li Jiang, Xihui Liu, and Hengshuang Zhao (2022). "Point transformer v2: Grouped vector attention and partition-based pooling". In: *Advances in Neural Information Processing Systems*. URL: 10.5555/3600270.3602685.

ZY Wu, R Kalfarisi, F Kouyoumdjian, and C Taelman (2020). "Applying deep convolutional neural network with 3D reality mesh model for water tank crack detection and evaluation". In: *Urban Water Journal*. URL: https://doi.org/10.1080/1573062X.2020.1758166.

Tian Xia, Jian Yang, and Long Chen (2022). "Automated semantic segmentation of bridge point cloud based on local descriptor and machine learning". In: *Automation in Construction*. URL: https://doi.org/10.1016/j.autcon.2021.103992.

Saining Xie, Sainan Liu, Zeyu Chen, and Zhuowen Tu (2018). "Attentional shapecontextnet for point cloud recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Dan Xu, Elisa Ricci, Yan Yan, Jingkuan Song, and Nicu Sebe (2015). "Learning deep representations of appearance and motion for anomalous event detection". In: *arXiv preprint arXiv:1510.01553*. URL: https://doi.org/10.48550/arXiv.1510.01553.

Mutian Xu, Runyu Ding, Hengshuang Zhao, and Xiaojuan Qi (2021). "PAConv: Position Adaptive Convolution with Dynamic Kernel Assembling on Point Clouds". In: *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. URL: 10.1109/CVPR46437.2021.00319.

Tomoyuki Yamaguchi, Shingo Nakamura, Ryo Saegusa, and Shuji Hashimoto (2008). "Image-based crack detection for real concrete surfaces". In: *IEEJ Transactions on Electrical and Electronic Engineering*. URL: https://doi.org/10.1002/tee.20244.

Cheng-Kun Yang, Ji-Jia Wu, Kai-Syun Chen, Yung-Yu Chuang, and Yen-Yu Lin (2022). "An mil-derived transformer for weakly supervised point cloud segmentation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Jiancheng Yang, Qiang Zhang, Bingbing Ni, Linguo Li, Jinxian Liu, Mengdie Zhou, and Qi Tian (2019). "Modeling point clouds with self-attention and gumbel subset sampling". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*.

Liu Yang, Yi-Chun Lin, Hubo Cai, and Ayman Habib (2024). "From Scans to Parametric BIM: An Enhanced Framework Using Synthetic Data Augmentation and Parametric Modeling for Highway Bridges". In: *Journal of Computing in Civil Engineering*. URL: https://doi.org/10.1061/JCCEE5.CPENG-5640.

Xiaofei Yang, Enrique del Rey Castillo, Yang Zou, Liam Wotherspoon, and Yi Tan (2022). "Automated semantic segmentation of bridge components from large-scale point clouds using a weighted superpoint graph". In: *Automation in Construction*. URL: https://doi.org/10.1016/j.autcon.2022.104519.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le (2019). "Xlnet: Generalized autoregressive pretraining for language understanding". In: *Advances in neural information processing systems*.

Cong Ye, S Acikgoz, Stephen Pendrigh, Ewan Riley, and MJ DeJong (2018). "Mapping deformations and inferring movements of masonry arch bridges using point cloud data". In: *Engineering Structures*. URL: https://doi.org/10.1016/j.engstruct.2018.06.094.

Maosheng Ye, Rui Wan, Shuangjie Xu, Tongyi Cao, and Qifeng Chen (2021). "Drinet++: Efficient voxel-as-point point cloud segmentation". In: *arXiv preprint arXiv:2111.08318*.

Jihun Yi and Sungroh Yoon (2020). "Patch svdd: Patch-level svdd for anomaly detection and segmentation". In: *Proceedings of the Asian conference on computer vision*. URL: https://doi.org/10.1007/978-3-030-69544-6_23.

Feng Yu, Kaichun Liu, Yikai Zhang, Cheng Zhu, and Kai Xu (2019). "PartNet: A recursive part decomposition network for fine-grained and hierarchical shape segmentation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. URL: https://doi.org/10.1109/CVPR.2019.00972.

Xumin Yu, Yongming Rao, Ziyi Wang, Jiwen Lu, and Jie Zhou (Dec. 2023a). " AdaPoinTr: Diverse Point Cloud Completion With Adaptive Geometry-Aware Transformers ". In: *IEEE Transactions on Pattern Analysis & Machine Intelligence*. URL: https://doi.ieeecomputersociety.org/10.1109/TPAMI.2023.3309253.

Xumin Yu, Yongming Rao, Ziyi Wang, Jiwen Lu, and Jie Zhou (2023b). "AdaPoinTr: Diverse Point Cloud Completion With Adaptive Geometry-Aware Transformers". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Bowen Zhang and Peter Wonka (2021). “Point cloud instance segmentation using probabilistic embeddings”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. URL: https://doi.org/10.1109/CVPR.2021.00879.

Renrui Zhang, Ziyu Guo, Peng Gao, Rongyao Fang, Bin Zhao, Dong Wang, Yu Qiao, and Hongsheng Li (2022). “Point-m2ae: multi-scale masked autoencoders for hierarchical point cloud pre-training”. In: *Advances in neural information processing systems*. URL: https://doi.org/10.48550/arXiv.2205.14401.

Renrui Zhang, Liuhui Wang, Yali Wang, Peng Gao, Hongsheng Li, and Jianbo Shi (2023). “Parameter is not all you need: Starting from non-parametric networks for 3d point cloud analysis”. In: *arXiv preprint arXiv:2303.08134*.

Youqi Zhang, Baiqiang Xia, and Su Taylor (2024). “High-resolution 3-D geometry updating of digital functional models using point cloud processing and surface cut”. In: *Computer-Aided Civil and Infrastructure Engineering*.

Zhirong Zhang, Bin Sheng Hua, and Sai-Kit Yeung (2019). “ShellNet: Efficient point cloud convolutional neural networks using concentric shells statistics”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. URL: https://doi.org/10.1109/ICCV.2019.00169.

Zhiyuan Zhang, Binh-Son Hua, and Sai-Kit Yeung (Nov. 2019). “ ShellNet: Efficient Point Cloud Convolutional Neural Networks Using Concentric Shells Statistics ”. In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. Los Alamitos, CA, USA: IEEE Computer Society. URL: https://doi.ieeecomputersociety.org/10.1109/ICCV.2019.00169.

Baozhu Zhao, Qiwei Xiong, Xiaohan Zhang, Jingfeng Guo, Qi Liu, Xiaofen Xing, and Xiangmin Xu (2024). “PointCore: Efficient Unsupervised Point Cloud Anomaly Detector Using Local-Global Features”. In: *arXiv preprint arXiv:2403.01804*. URL: https://doi.org/10.48550/arXiv.2403.01804.

Benyun Zhao, Xunkuai Zhou, Guidong Yang, Junjie Wen, Jihan Zhang, Jia Dou, Guang Li, Xi Chen, and Ben M. Chen (2024). “High-resolution infrastructure defect detection dataset sourced by unmanned systems and validated with deep learning”. In: *Automation in Construction*.

Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip H.S. Torr, and Vladlen Koltun (Oct. 2021a). “Point Transformer”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. URL: https://doi.org/10.48550/arXiv.2012.09164.

Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun (2021b). "Point transformer". In: *Proceedings of the IEEE/CVF international conference on computer vision*.

Liangliang Zhao and Weidong Tao (2020). "JSNet: Joint instance and semantic segmentation of 3D point clouds". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. URL: https://doi.org/10.1609/aaai.v34i07.6994.

Lin Zhao and Wenbing Tao (Apr. 2020). "JSNet: Joint Instance and Semantic Segmentation of 3D Point Clouds". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. URL: https://ojs.aaai.org/index.php/AAAI/article/view/6994.

Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun (2018). "Open3D: A modern library for 3D data processing". In: *arXiv preprint arXiv:1801.09847*. URL: https://doi.org/10.48550/arXiv.1801.09847.

Yinlong Zhou and Oncel Tuzel (2018). "Voxelnet: End-to-end learning for point cloud based 3d object detection". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. URL: https://doi.org/10.1109/CVPR.2018.00472.

Yuhui Zhou, Shaohua Wang, Xi Mei, Wangling Yin, Chunfeng Lin, Qingwu Hu, and Qingzhou Mao (2017). "Railway tunnel clearance inspection method based on 3D point cloud from mobile laser scanning". In: *Sensors*.