Automation in Construction 142 (2022) 104519

Contents lists available at ScienceDirect

Automation in Construction

journal homepage: www.elsevier.com/locate/autcon

Automated semantic segmentation of bridge components from large-scale point clouds using a weighted superpoint graph

Xiaofei Yang [a,*], Enrique del Rey Castillo [a], Yang Zou [a], Liam Wotherspoon [a], Yi Tan [b]

[a] Department of Civil and Environmental Engineering, University of Auckland, Auckland 1023, New Zealand
[b] College of Civil Engineering, Shenzhen University, Shenzhen, China

ARTICLE INFO

Keywords:
Bridge component recognition
Deep learning
Semantic segmentation
Large-scale point clouds
Weighted Superpoint Graph

ABSTRACT

Deep learning techniques have the potential to provide versatile solutions for automated semantic segmentation of bridge point clouds, but previous studies were limited to small-scale bridge point clouds and focused on limited bridge component categories due to training sample scarcity. Additionally, no prior work considered the intrinsic data imbalance problem in the bridge dataset, with the points unequally distributed between the various components. This paper presents a weighted superpoint graph (WSPG) method, where bridge point clouds were firstly clustered into hundreds of semantically homogeneous superpoints that were then classified into different bridge components using PointNet and Graph Neural Networks. The WSPG method can recognize components directly from large-scale bridge point clouds and alleviate the data imbalance by leveraging a novel loss function that assigns weights according to the number of points contained in different bridge components. The effectiveness of the method was validated on both a real-world dataset with 5 categories of bridge components and a synthetic dataset with 8 categories of bridge components. Experiment results on the real-world dataset showed that the WSPG model achieved the best performance on all overall evaluation metrics of overall accuracy (OA: 99.43%), mean class accuracy (mAcc: 98.75%) and mean Intersection over Union (mIoU: 96.49%) compared to the existing cutting edge models such as PointNet, DGCNN and the original SPG. Additionally, the WSPG method also surpassed the cutting edge representatives in terms of mAcc and mIoU on the synthetic dataset, especially increasing the original SPG by 8.5% mAcc and 6.7% mIoU. The successful application of the proposed method will significantly improve upper-level tasks such as digital twining for existing bridges.

1. Introduction

Bridges are essential components of infrastructure systems and can suffer from excessive deterioration and corrosion during their design life. In the United States, 8.86% of 612,408 bridges are labelled as structurally deficient or functionally obsolete [1], and $12.8 billion are spent every year on maintaining deteriorating bridges [2]. Bridges are inspected on a periodic basis by well-trained inspectors through visual observation, manual measurement of defect extents and application of rating frameworks, a process that has been recognized as time-consuming, subjective and error-prone [3]. Additionally, the inspection data has typically been collated using a paper-based approach, making information exchange less efficient and imposing challenges for bridge management. Therefore, new techniques to boost bridge inspection productivity and improve the management and re-use of inspection data are in high demand.

Emerging survey technologies that can generate accurate point cloud data, such as laser scanning, photogrammetry and Light Detection and Ranging (LiDAR), have seen increasing use as part of bridge inspection and management. A point cloud is a set of three-dimensional (3D) data points with coordinates and radiometric attributes such as color and intensity information, meaning they can store damage information as well as geometric and topological properties of structures. Therefore, point clouds can be processed and analyzed for future use such as three sub-tasks of the bridge monitoring: 1) component-wise damage assessment including damage detection, localization and quantification [4]; 2) measurement of deformation and deflection at component level [5]; and 3) digital twinning of existing bridges by fitting the recognized bridge component point clouds with Industry Foundation Class (IFC) models [2,6]. However, raw bridge point clouds do not contain any semantics nor specific information with respect to bridge components [7]. Thus, the semantic segmentation of bridge point clouds is an intermediate

* Corresponding author.
E-mail address: fyan983@aucklanduni.ac.nz (X. Yang).

https://doi.org/10.1016/j.autcon.2022.104519
Received 13 May 2022; Received in revised form 25 July 2022; Accepted 29 July 2022
Available online 4 August 2022

procedure needed to enable the full automation of the above future use. This task aims to detect and classify critical components of bridges, consisting of two main steps: 1) segmenting the full-scale bridge point clouds into sub-point clusters corresponding to different bridge components, and 2) assigning semantic labels to the sub-point clusters [8].

Most previous studies apply conventional computer vision techniques to manually construct geometric feature descriptors of individual bridge components. These methods only adapt to specific types of bridges because bridge components of different types of bridges are normally distinct in geometries [9]. Recently, deep learning-based methods have attracted increasing interest within methods for the semantic segmentation of point clouds. Compared with previous computer vision techniques, deep learning-based methods can automatically extract useful features and be applied to generic bridge types as long as a sufficient number of training samples are available [10]. However, five challenges remain for the use of deep learning-based techniques: 1) Existing research is limited by a fixed number of input points such as 2048 and therefore only suitable for small-scale bridge point clouds with the assistance of a cumbersome subspace partitioning step [11]. Notably, the global features of bridges were generally lost when conducting the subspace partitioning step, imposing challenges to process full-scale bridge point clouds [12]; 2) Applying deep learning techniques requires a huge amount of training samples, but real-world bridge point clouds are difficult to acquire due to the high capture cost and the potential risks that the surveyor may be exposed to [13]. Additionally, this difficulty is amplified by the fact that different types of bridge point clouds are normally distinct in geometries. Although synthetic point clouds could be substitutes [14], there is little research related to the implementation of a practical user-friendly framework for generating synthetic point clouds; 3) Only limited bridge components have been studied in previous research due to insufficient training samples from bridge point clouds. It is not clear whether deep learning-based methods are able to conduct semantic segmentation for all major bridge components; 4) Bridge point clouds are often dominated by majority components (e.g. decks and girders) in terms of the number of points. This phenomenon causes an imbalance between different component categories, where majority component categories have a larger number of points than minority component categories such as piers and pier caps; 5) It is unclear whether deep learning-based algorithms trained on one bridge scene can be generalized to a new bridge scene with different types of components.

A new deep learning-based method called superpoint graph (SPG) algorithm [15] was initially developed to conduct automated semantic segmentation of building interior point clouds such as walls, ceilings and floors. The term superpoint refers to a set of points with geometrically isotropic and simple shapes. Leveraging superpoints as input can significantly reduce the number of input points. This method has a compelling advantage over other existing deep learning methods as it can directly process large-scale point clouds by classifying hundreds of small geometrically simple superpoints generated by a clustering algorithm instead of a large amount of individual points. Given its successful application to the recognition of building point clouds, it has the potential to be applied to bridge component recognition and overcome the aforementioned challenges. However, the imbalance between different components within bridge point cloud scenes could mean that the original SPG algorithm may not be effective when classifying minority component categories such as abutment, pier cap, diaphragm and lamp post.

This study aims to evaluate the use of SPG-based methods to directly segment large-scale bridge point clouds. The main contributions of this study are:

- Two expert-annotated datasets both for the real-world and synthetic bridge point clouds are established, where bridge point clouds contain different bridge component geometries and curve rates that can be longitudinally or transversely tilted, horizontally or vertically curved.
- A framework for the generation of synthetic bridge point clouds from bridge information models (BrIMs) is proposed.
- A WSPG model is presented, which implements automated semantic segmentation of bridge components directly from full-scale bridge point clouds and alleviates the data imbalance between different components by using a novel weighted loss function improving the segmentation ability for minority component categories.

The rest of this paper is structured as follows. Section 2 discusses the current state of research. Section 3 introduces the data preparation process and outlines the proposed method. The real-world bridge point cloud dataset is presented, followed by the process used to build the synthetic bridge point cloud dataset. The details of the WSPG method are then presented. Section 4 elaborates on the experiment implementation, evaluation and analysis. The effectiveness of the proposed WSPG model is validated on both the real-world and synthetic dataset. Five types of bridge components are recognized from the real-world dataset and eight types of bridge components are detected from the synthetic dataset. The performance of the WSPG method is compared with the original SPG, PointNet and dynamic graph-based convolutional neural network (DGCNN) algorithms. Section 5 concludes the article and discusses the future research.

2. Literature review

Extensive research has been published to fully automate bridge component recognition from point clouds, which can be divided into three groups: bottom-up segmentation, top-down partitioning, and deep learning-based methods [13]. The first two methods were based on local salient features and domain knowledge respectively, and tailored for specific types of bridges, thus limiting their generalization to generic bridges. Deep learning-based methods have significant advantages, compared with the first mentioned two approaches, and are gaining increased popularity in the bridge component recognition. Each approach is reviewed and discussed further in the following subsections.

2.1. Bottom-up segmentation

Bottom-up segmentation starts with grouping points with local homogeneous features to formulate individual objects. These local salient features consist of geometric and textural features such as geometry, curvature, continuity of boundary surfaces, intensity and color [16]. Multiple algorithms have been proposed for bridge component recognition based on these local salient features. An early example is the region growing algorithm, firstly proposed by Walsh et al. [17]. This algorithm starts by detecting the sharp features such as an edge (the line that connects two surfaces) from point clouds. This detection phase is based on searching the points that have a high normal vector variation between their neighboring points. Subsequently, a smoothness-constraint-based region growing method is employed to find smoothly connected areas in the point cloud based on normal vectors. The seed point that refers to the initial point is selected according to a strategy of segmenting planar regions first, followed by grouping neighboring points with similar features until reaching the edge of an object. This results in points that are segmented into different regions. These regions are then used for quadratic surface fitting to determine surface types. Finally, these surfaces bounded by edges are matched with the model base to conduct object detection. Another example is an improvement of the region growing algorithm which can adaptively find the seed. This improvement clusters points based on surface roughness and principle curvatures and thus can deal with curved surfaces [16]. However, the performance of this algorithm decreased significantly when processing incomplete point clouds caused by ambient occlusions. Additionally, a

recent study proposed a three-stage cell- and voxel-based region growing method for bridge component recognition and achieved high segmentation accuracy [18]. The first stage consists of coarse and fine extraction to group raw point cloud data into different clusters. The coarse extraction is implemented by decomposing bridge point clouds into two dimensional (2D) cells in the xy plane and analyzing point distribution along the vertical direction. Fine extraction is then conducted leveraging cell- and voxel-based region growing methods to cluster surfaces. Subsequently, the superstructure and substructure of bridges are identified by contextual knowledge. This method depends on suitable input parameters such as angle, distance and residual thresholds, and the parameters relevant to data quality are still empirically selected. A Random Sample Consensus (RANSAC) algorithm was used by Schnabel et al. [19] to group point clouds with basic shapes such as planes, spheres, cylinders, cones and tori. While this algorithm performed well in relatively simplified scenarios, it was not able to detect bridge components with complex geometries. An unsupervised learning algorithm called k-means [20] was also proposed for bridge component recognition, which extracted bridge decks in point clouds and compared the results with an object-based region growing algorithm. The experiment indicated that the region growing method achieved better performance. In addition, the integrated clustering technique was explored by Perry et al. [21]. They first detected decks from point clouds according to surface normal angle and then leveraged Gaussian Mixture Model (GMM) in combination with Agglomerative Clustering to extract the remaining main components from bridge point clouds. This approach achieved 99% segmentation accuracy with around 10-min processing time, outperforming the performance of using a single cluster method, such as GMM or Agglomerative Clustering. Nevertheless, this method still required the manual selection of an initial point and manual assignment of semantic labels to extracted elements. In addition, this method has only been tested on simple, small-scale beam bridges, with its applicability to other more complicated bridge types unclear.

Bottom-up methods are still considered as insufficient for bridge component recognition, despite the progress made in previous research. The main reason for this inadequacy is the reliance on local features to finish the point clustering process, whereas most bridge components exhibit planar patterns that make it difficult to determine whether local surfaces and patches belong to the same component. In addition, the bottom-up segmentation method is time-consuming and sensitive to noise and outliers [22], which is not feasible for real bridge point clouds with millions of points [6]. The intervention of contextual information and domain knowledge could be a solution to these issues because it defines the rules for spatial relationships between the components [13].

2.2. Top-down partitioning

The top-down partitioning approach adopts domain knowledge and can be classified into optimization-based methods and heuristic-based methods [23]. The principle of optimization-based methods is the use of parametric models to initially encode the domain knowledge on the geometrical and topological constraints of each component as well as spatial relationships between each individual component. Subsequently, the model is associated with the input point clouds via a loss function to measure their similarity. The component recognition problem can be converted into an optimization problem by minimizing the loss function. An example is the global energy optimization approach [24], which extracted planar objects from point clouds by optimizing model parameters such as geometric fitness, spatial coherence and model complexity. However, this method has been shown to be inefficient and computationally expensive because it has extensive search demands for bridge components with complex geometry, despite being technically possible for component recognition tasks.

The heuristic-based method starts with breaking a full-scale bridge into distinct component assemblies, reducing the complexity of geometries for posterior processing. These component assemblies will subsequently be processed by bottom-up methods or optimization-based method to detect individual components. Lu et at. [13] presented pioneering work based on top-down methods for automated bridge component recognition from point clouds. This method started with aligning the full-scale bridge point clouds from an arbitrary position and orientation, and then sliced the entire bridge into deck assembly and substructures. Bridge engineering knowledge was then used to detect each component. Although this method achieved very high detection accuracy and time-efficiency, some drawbacks remained: 1) it requires the manual removal of background points such as ground, water, and vegetation, making the process time-consuming and labor-intensive; 2) it only detects a limited number of component categories including slab, pier, pier cap and girder in reinforced concrete (RC) bridges; 3) it only adapts to simple RC slab bridges and beam-slab bridges but is not applicable to curved bridges. Yan et al. [23] adopted a similar method to extract components from steel girder bridges. The novelty of this study was the application of a piece-wise linear skeleton model to estimate the traffic direction, followed by the semantic segmentation of bridge components from point clouds with the assistance of an incremental cross-section alignment process. This innovative method was capable of processing curved and tilted bridge point clouds. Another improvement for the work of Lu et at. work was proposed by Zhao et al. [6], and most visual components such as pier cap, pier, abutment, slab, girder, and parapet can be detected. The main workflow included: 1) denoising the points irrelevant to bridges, 2) employing geometric heuristics and a reference frame to discriminate the individual spans, 3) decomposition of a bridge into the substructure, superstructure and deck and further detection of individual components.

While previous research based on heuristic-based algorithms made great progress, this method still faces two main challenges: 1) manual denoising of background points made the whole process labor-intensive and time-consuming; and 2) these algorithms are tailored for the specific type of bridges and perform relatively poorly on diverse shapes of components, which are difficult to be generalized to all types of bridges.

2.3. Deep learning-based methods

With the rapid accumulation of digital data and advances in computing power, deep learning-based methods for semantic segmentation of point clouds have attracted significant interest in the realm of bridge component recognition. Pioneering work proposed by Kim et al. [10] explored the use of the PointNet algorithm [25]. The advantages of this approach are that it is generally applicable to all types of bridges, even for curved or tilted bridges with diverse shapes of components, and the background denoising step is eliminated. However, since the PointNet algorithm was designed for processing small-scale point cloud with a fixed number of input points of 2048, it was difficult to adopt PointNet to directly process large-scale point clouds of an entire bridge. To overcome this deficiency, Kim et al. [10] started with partitioning the full-scale bridge point clouds into subspaces with a fixed distance, followed by employing a PointNet network to segment point clouds for each subspace automatically. This research successfully demonstrated that a deep learning-based method was a promising solution for bridge component recognition, although it only classified bridge components into deck, pier and background. It is worth noting that deep learning-based methods are still underused in the realm of bridge component recognition due to the lack of sufficient training data. Here note that sufficient training samples refer to the number and size of bridge components with diverse geometries. Dividing a bridge into repetitive subspaces will not increase the number of bridge components and their sizes. Other deep learning-based methods such as PointCNN and Dynamic Graph Convolutional Neural Network (DGCNN) were explored and compared for the recognition accuracy of abutment, girder, slab, background and pier components [11]. The results showed DGCNN surpassed the other two algorithms but the detection accuracy of abutment was relatively low at 73.78%, possibly due to an imbalance

between different categories of components. In addition, a graph-based hierarchical DGCNN (HGCNN) model was presented for semantic segmentation of railway bridges having electric poles [26]. The results demonstrated that the HGCNN method improved the detection accuracy between boundaries, and it also enhanced the performance of electric poles for Intersection over Union (IoU) with 3%. However, this method was only tested on deck, pier, abutment, pole as well as background, ignoring the girder, pier cap, diaphragm and parapet. It is still limited by the number of input points. Additionally, this method also relies heavily on the subspace partition step when segmenting full-scale bridges.

To summarize, the deep learning-based method is a generic approach for automated bridge component recognition of all types of bridges instead of designing specific algorithms according to bridge types like top-down methods. However, existing deep learning-based methods still face several challenges. Firstly, the lack of sufficient bridge point cloud data with various bridge components hinders the application of deep learning-based methods. Secondly, previous research only adapts to small-scale bridges with simple component configurations. Few studies focus on detecting all the main bridge components rather than partial components directly from large-scale bridge point clouds without

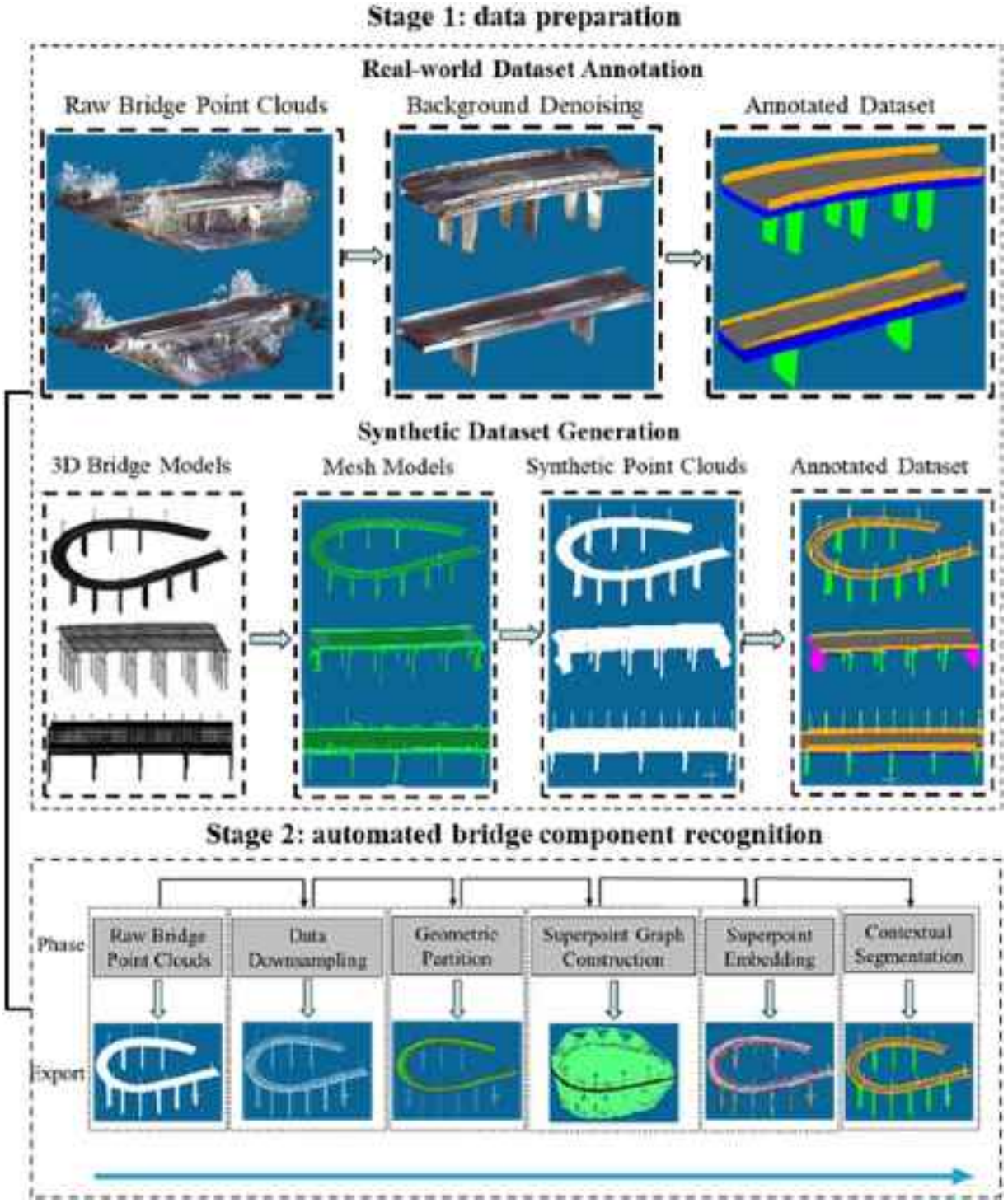

Fig. 1. Two-stage workflow for data preparation and bridge component recognition.

subspace partition operation. Next, the recognition performance of minority component classes is low due to the imbalance between different component classes. In addition, no research has demonstrated whether deep learning-based algorithms trained on one bridge scene can be applied to new bridge scenes with different types of components.

3. Methodology

3.1. Overview

The proposed solution was divided into two stages. The first stage was to establish two well-annotated datasets for both real-world and synthetic bridge point clouds. We firstly annotated an open-source real-world dataset provided by Lu et al. [13], consisting of 5 types of bridge components. There is a scarcity of real-world datasets with various types of bridge components. This is likely because it is risky and time-consuming for surveyors to collect data on in-service bridges. Hence, to include more types of bridge components within our dataset, we established a synthetic bridge dataset. It is cost-saving and time-efficient to establish a synthetic dataset that contains a good number of bridge point cloud models with diverse component categories. Most importantly, such dataset can also retain the geometric and topologic features of real-world bridges.

In the second stage, we proposed a WSPG method to directly process large scale bridge point clouds. To mitigate the imbalance between majority component categories and minority component categories, the proposed method leveraged a new loss function [27] capable of assigning weights according to the number of points contained in different components. This novel loss function can improve the segmentation performance of minority categories, thus advancing the overall segmentation performance of the method. The proposed method directly uses point cloud coordinates (e.g., XYZ) as input, and predicts a semantic label for each point in an end-to-end manner. Fig. 1 displays the two-stage framework in the proposed solution. Each stage of this framework is discussed in more detail in the following sections.

3.2. Data preparation

3.2.1. Real-world dataset

The raw real-world dataset leveraged in this study is an open-source dataset consisting of 10 RC slab and beam-slab bridges collected by a FARO Focus 3D X330 Terrestrial Laser Scanner in the United Kingdom. These two types of bridges reflect the main bridge types of the existing highway bridges and planned future bridges in England [28]. To annotate the raw real-world dataset, the background denoising step was firstly conducted to manually remove the background point clouds such as the on-site traffic, vegetation and ground. A manual approach was used as previous research suggested that it was easier for a human modeler to remove background point clouds than a computer [13]. The preprocessed dataset was then split into a training dataset and a testing dataset. Part of Bridge 1, Bridge 5, Bridge 9 were selected as the testing dataset because similar bridge types were contained in the training dataset. Finally, an expert-annotated dataset was generated which contained five types of bridge components including pier, pier cap, girder, deck and parapet. Fig. 2 presents the visualization example of the well-annotated real-world dataset.

3.2.2. Synthetic dataset

3.2.2.1. Synthetic bridge point cloud generation. Fig. 3 displays the workflow for generating synthetic bridge point clouds using existing software. In this study, a synthetic dataset was generated from as-designed bridge information models (BrIMs). The proposed framework for the synthetic dataset generation consisted of four steps: 1) 3D modelling of bridges based on 2D drawings by leveraging Autodesk Revit and conversion of the acquired BrIMs with a Revit file format (.rvt) into Industry Foundation Class (IFC) format (.ifc) [29]; 2) converting the IFC models to mesh models (.obj) through ifcConvert [30]; 3) sub-sampling these mesh models to point cloud data (.txt) with different densities via CloudCompare [31]; and 4) annotating the point cloud data to different point clusters corresponding to the component categories utilizing CloudCompare. The annotation step was a manual and tedious process, resulting in an eight-class classification (abutment, pier, pier cap, girder, diaphragm, deck, parapet, lamp post). Although care was taken during this process, it is worth noting that mislabeling may occur particularly in the boundaries of different classes due to the subjective manual process [32].

The outcome of this process was a synthetic dataset containing various shapes of bridge components. Fig. 4 shows the visualization of the synthetic dataset for three different bridge examples.

3.2.2.2. Synthetic dataset description. The compiled bridge dataset

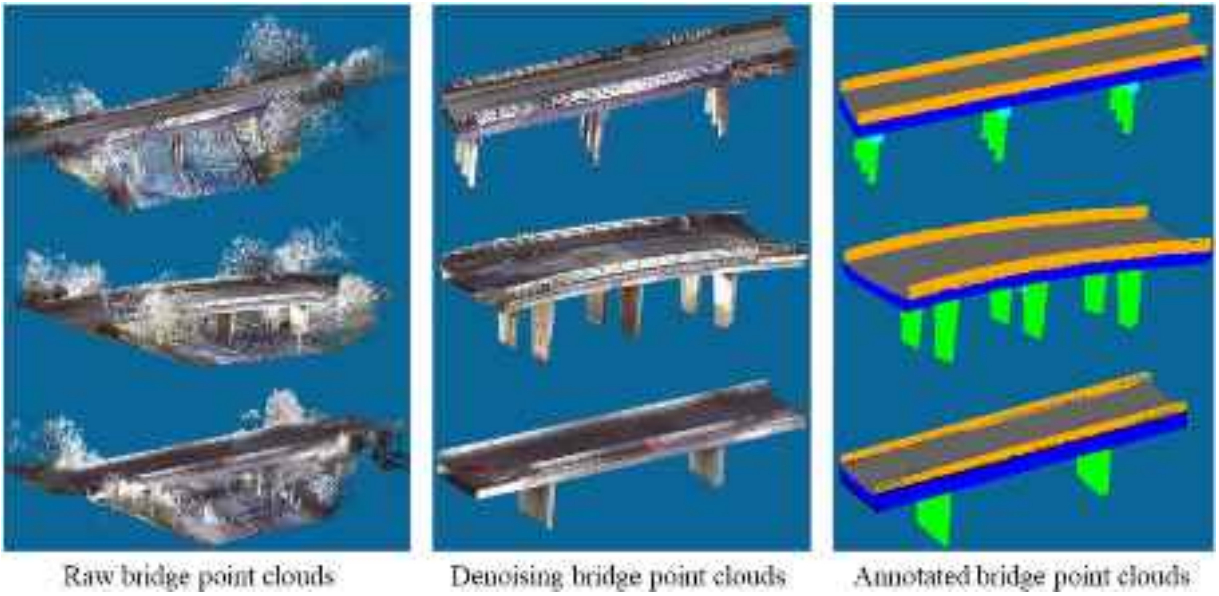

Fig. 2. Visualization examples of the annotated real-world dataset.

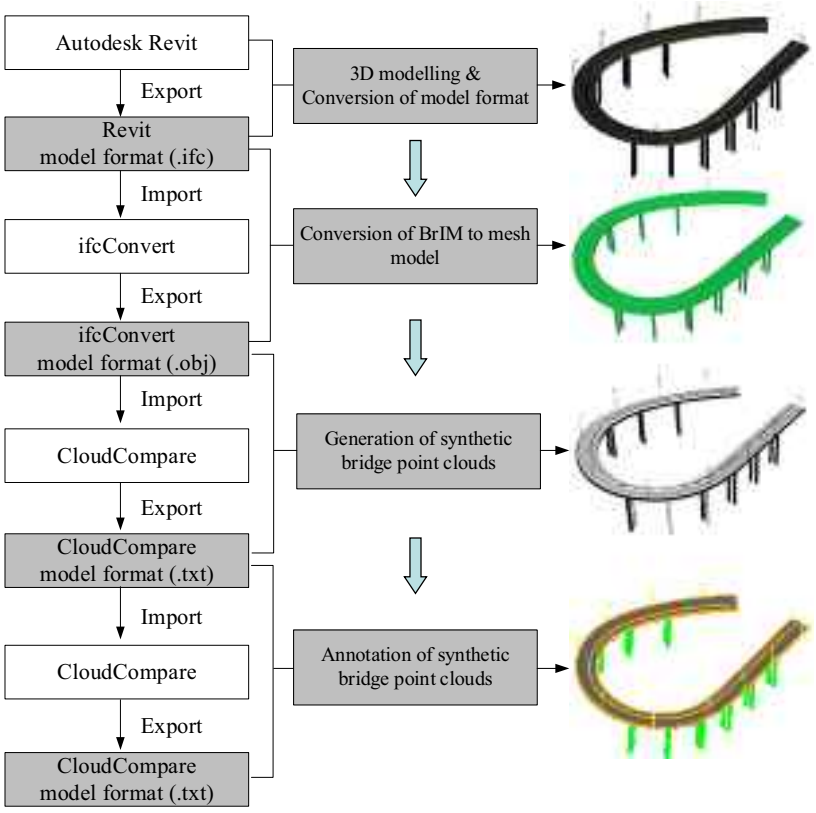

Fig. 3. Workflow for generating the synthetic bridge point cloud dataset and annotation of each component category.

consisted of 22 synthetic point cloud models from as-designed BrIMs of real bridges, most of which were reinforcement concrete (RC) beam bridges with various types of component configurations. This type of bridge was selected because the RC beam bridge accounts for the most existing bridge stock [13]. The key components of beam bridges included abutment (abut), pier, pier cap, girder, diaphragm (diap), deck, parapet (para), and lamp post. Fig. 5 provides examples of the raw bridge component point clouds for a range of components in the dataset.

The total length of each bridge in the dataset ranged from 32 m to 538 m. The size of point clouds for each bridge ranged from 70,000 to nearly a million points. In addition, we randomly deleted parts of the point clouds in some of components to simulate the influence caused by occlusions, similar to how vegetation and moving vehicles result in scattered and irregular occlusions in bridges [12], as shown in Fig. 6. The synthetic dataset was divided into 6 folders for later standard 6-fold cross-validation. A detailed summary of the synthetic bridge dataset is presented in Table 1 and Fig. 7.

3.3. Weighted superpoint graph

The WSPG method consisted of five major steps, as illustrated below:

- *Data downsampling:* Data downsampling should be implemented prior to the recognition of major bridge components to reduce the redundancy of the raw data.
- *Geometric partition*: The aim of geometric partition was to segment the full-scale bridge point clouds into superpoints.
- *Superpoint graph construction:* A structured and oriented superpoint graph was built so that spatially adjacent superpoints are linked by superedges.
- *Superpoint embedding*: Each superpoint corresponding to a graph node stands for a geometrically simple primitive carrying homogeneous semantic information. Superpoint embedding inferred a feature vector from each superpoint and leveraged this feature vector to represent superpoints.
- *Contextual segmentation:* Each superpoint was classified based on contextual relationships corresponding to its feature vector and surroundings.

The details of each step are described in the following subsections.

3.3.1. Data downsampling

The raw bridge point cloud data is typically dense and with redundant information, containing a large amount of points spanning hundreds of meters. A data downsampling process was inevitably required when processing the point cloud data through a deep neural network without losing useful point features. In this study, the Voxel Grid Filter algorithm [33] was used for down-sampling, based on a preset voxel width to subdivide the input space and replace the points in each voxel by the centroid of these points. The two most important factors to consider when specifying the desired voxel size are: 1) the computing efficiency and 2) the adequate coverage of small-scale as well as low-density objects. Otherwise, the useful features of small-scale targets will be lost if the sampling resolution is too large. Thus, a voxel size 0.03 m was set as the optimal space resolution that is sufficient to recognize major bridge components and can achieve an excellent trade-off between efficiency and effectiveness.

3.3.2. Geometric partition

The objective of this step was to break down a full-scale bridge point cloud data into geometrically simple segments. These segments should be semantically homogeneous as well, thus not including objects from different classes [15]. It should be noted that this process was an unsupervised step.

Geometric partition was based on the graph segmentation algorithm, as shown in Fig. 8, and the key idea behind this method was to construct

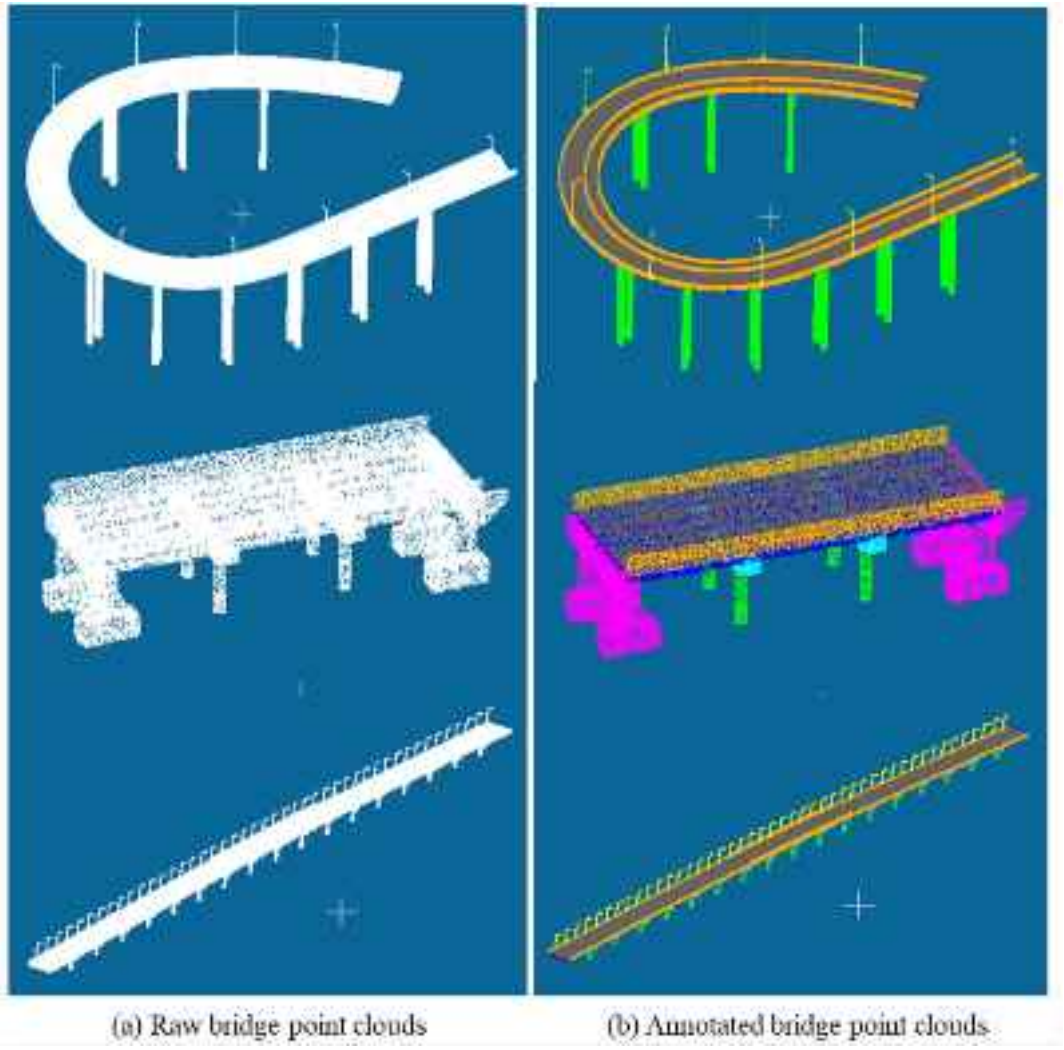

Fig. 4. Visualization of the synthetic dataset for three different bridge examples.

an undirected graph $G_{nn} = (V, E_{nn})$, where G_{nn} is a function of the nodes V, which represents points of the input point cloud, and the edges E_{nn}, which denotes the mean distance of the 10-nearest neighbors, encoding their adjacent relationship. We computed the 10-nearest neighbors' graph, as suggested in [34]. Notably, This graph defined a symmetric graph-adjacency relationship that is different from the optimal neighborhood mentioned by Weinmann et al. [35].

For each point, local geometric feature vectors $f_i \in R^{dv}$ were computed to constrain the graph. In this study, the structure of the bridges was geometrically simple in general, thus well-selected geometric features associated with respective point clusters were expected to be spatially regular. Linearity, planarity, scattering, verticality and elevation were selected as the local geometric features [36], resulting in a feature dimension d_v equal to 5. The geometric partition for superpoint generation was known as a generalized minimal partition problem and can be seen as the problem of minimizing the following Potts energy model within a continuous-space:

$$g^* = \underset{g \in R^{d_V \times |V|}}{arg\ min} \sum_{i \in V} |g_i - f_i|^2 + \rho \sum_{(i,j) \in E_{nn}} \zeta [g_i - g_j \neq 0] \quad (1)$$

where $[g_i - g_j \neq 0]$is 1 if true and 0 otherwsie.

In Eq. (1), the first part is the fidelity function to ensure g* corresponds to the homogeneous values of f, where g* is a piece-wise constant approximation of f structured by the graph G. The second part is the regularizer, which adds a penalty for each edge between two segments to ensure the smoothness of each superpoint, where ρ denotes a regularization factor determining the coarseness of the partition, and ζ is the edge weight. A set of superpoints denoted as $S = \{S_1, \ldots, S_k\}$ is the solution of Eq. (1) through using a l_0-cut pursuit algorithm.

3.3.3. Superpoint graph construction and superpoint embedding

The SPG is defined as an oriented attributed graph $G = (S, \varepsilon, F)$, consisting of a set of superpoints S, superedges ε, and superedge features $F \in R^{|\varepsilon| \times d\varepsilon}$. Each superpoint represents a small segment from the entire bridge point cloud. Additionally, Superedges were derived from a Voronoi adjacency graph $G_{vor} = (C, E_{vor})$ [37], which encodes the adjacent relationship between superpoints. Superedge features were computed by comparing the shape and size of superpoints, such as mean offset, volume ratio and point count ratio, where $|\varepsilon|$ denotes the number of adjacent superpoints and $d\varepsilon = 13$ indicates the dimension of superedge features.

Regarding the superpoint embedding, The aim of this step was to infer a feature descriptor $Q_i \in R^{du}$ for each superpoint by employing a simplified PointNet architecture, where d_u is the output dimension. PointNet architecture was proposed in 2016 [25], and it can efficiently learn superpoint features via Multi-Layer Perceptrons (MLPs) and a max-pooling layer. The workflow for the superpoint feature embedding process is shown in Fig. 9.

Each superpoint was embedded independently. The input number of points contained in each superpoint S_i can be denoted by N_{S_i}. The input

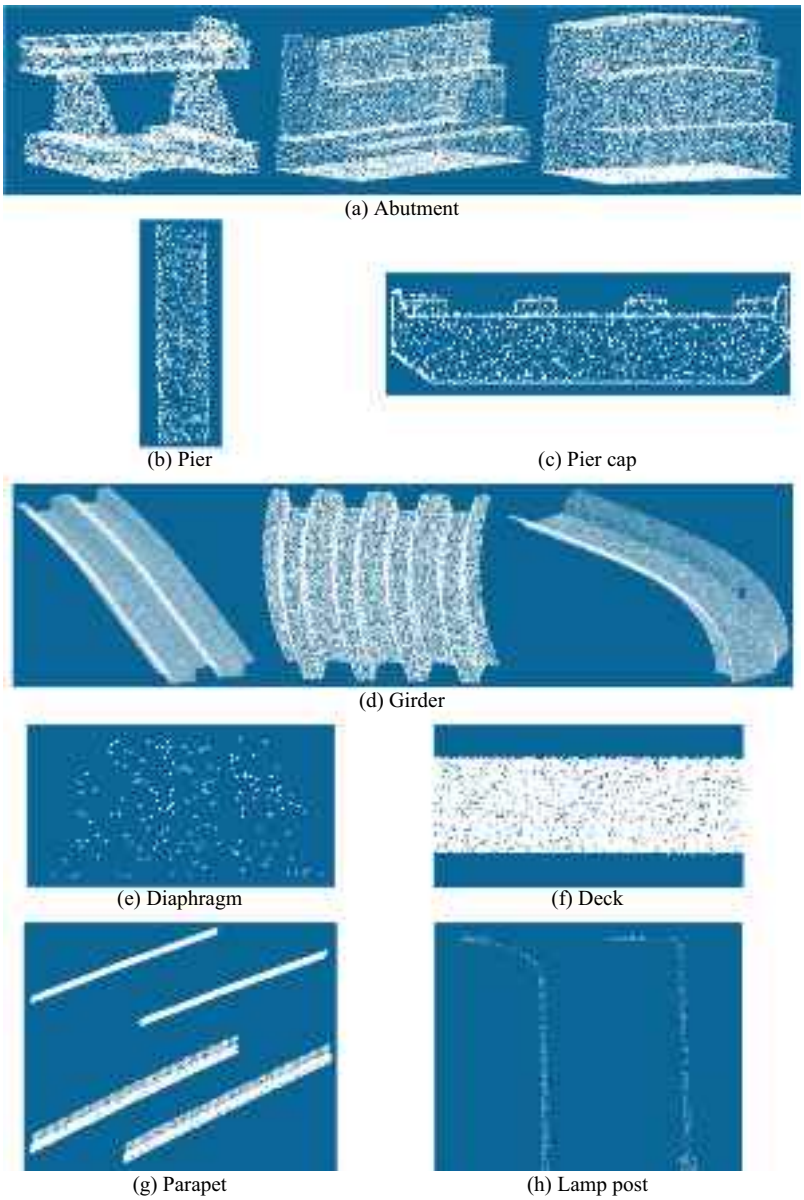

Fig. 5. Visualization examples of each component category in the dataset.

points are represented by their attributes consisting of normalized position p_i' $(\hat{x}, \hat{y}, \hat{z})$ and five geometric features f_i (linearity, planarity, scattering, verticality, elevation), such that the input points have d_f=8 dimensional features. These points were first subsampled or upsampled to N_p=128 points for computation efficiency and contribution to data augmentation. Note that superpoints with less than 25 instead of 40 points were set to zero to improve overall performance. Subsequently, a Spatial Transformer Network (STN) [38] was leveraged to align these points, followed by a sequence of MLPs (widths 64,64,128,128,256) and a max-pooling to an unary vector with 256 features that formulate the global feature. Each superpoint was rescaled to a unit sphere before embedding and the scalar metric diameter was then integrated with the global feature d_g. These aggregative features were finally processed by another sequence of MLPs (widths 256, 64, 32) to obtain d_q.

3.3.4. Contextual segmentation

The final step was to classify each superpoint according to its contextual relationships retrieved by graph convolutions. This approach builds on Gated Graph Neural Networks [39] and ECC. Specifically, each superpoint S_i retains its hidden state that builds its spatial dependencies via a Gated Recurrent Unit (GRU) [40]. The initial hidden state was defined as the embedded feature descriptor Q_i. At each iteration, a GRU module takes its hidden state $h_i^{(t)}$ and an incoming message $m_i^{(t)}$ as input and outputs a new hidden state $h_i^{(t+1)}$. The incoming message $m_i^{(t)}$ of superpoint i is defined as a weighted sum of hidden states $h_j^{(t)}$ of adjacent superpoints j. The weighting of a superedge (j, i) was computed from superedge features F_{ji} via MLPs (widths 32, 128, 64, 32) with ReLUs as the activation function, applying the idea of ECC, which is defined as:

$$m_i^{(t)} = mean_j | (j,i) \in \varepsilon \, \Theta\left(F_{ji}; W_e\right) \odot h_j^{(t)} \tag{2}$$

where $\odot$ denotes element-wise multiplication and W_e is the trainable parameter in MLPs. After 10 iterations in the GRU, hidden states in each time step were aggregated and followed by a linear layer. The final output y_i corresponding to superpoint categories was formulated by:

$$y_i = W_o\left(h_i^{(1)}, \ldots, h_i^{(T+1)}\right)^T \tag{3}$$

where T denotes the number of iterations in a GRU, herein $T = 10$ and

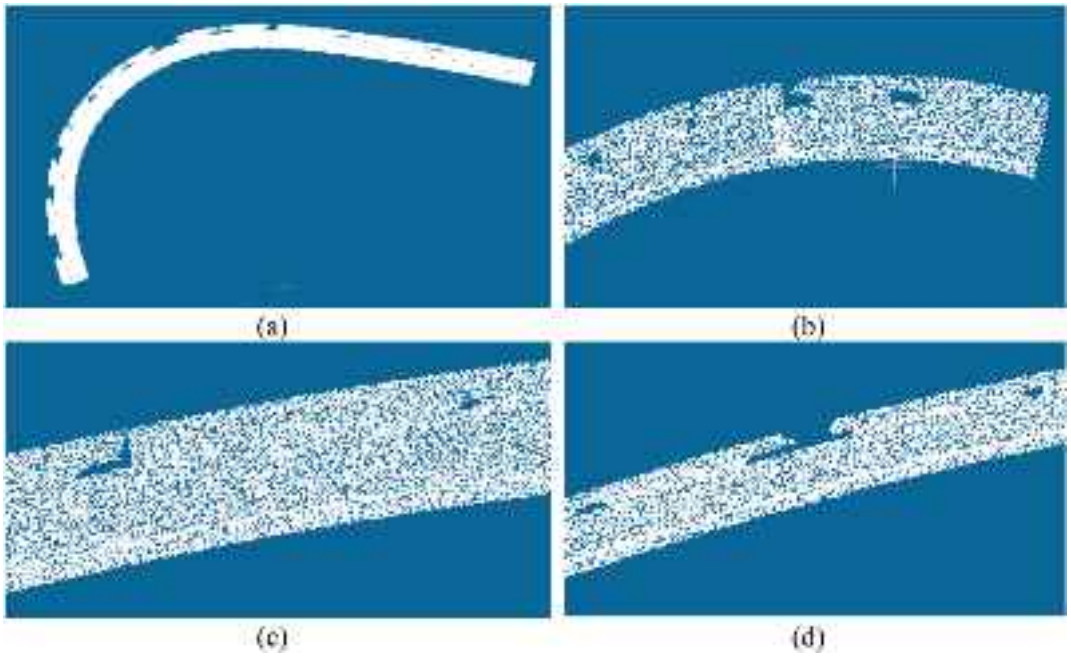

Fig. 6. Some examples of incomplete point clouds with sections removed to simulate the presence of occlusions.

Table 1
Metadata for the synthetic bridge point cloud dataset.

	Bridge number	Length (m)	Width (m)	Total number of points	Number of points per component							
					Abut	Pier	Pier cap	Girder	Diap	Deck	Para	Lamp
Folder 1	1	286	11.66	655,750	/	88,663	/	252,645	/	221,123	90,421	2898
	2	149	31.28	379,185	16,912	7116	27,572	213,458	/	90,750	23,377	/
	3	109	23.21	234,450	/	20,485	/	102,518	/	89,216	21,453	778
	4	91	15.09	102,216	15,630	4709	/	42,797	/	38,031	1049	/
Folder 2	5	202.74	8.00	181,314	/	38,178	/	59,241	/	50,221	32,143	1531
	6	148.75	31.06	471,394	21,058	7652	34,872	265,387	/	113,291	29,134	/
	7	38.17	12.35	72,763	18,922	1733	6168	23,422	1268	14,468	6782	/
	8	230.93	13.45	380,850	/	96,791	/	129,394	/	108,402	43,488	2775
Folder 3	9	98.78	13.05	180,567	24,852	10,030	13,427	65,840	2090	43,863	20,465	/
	10	245.02	7.82	220,531	/	44,823	/	73,510	/	58,927	41,394	1877
	11	109.96	23.91	239,150	/	21,144	/	104,464	/	89,158	22,362	2022
	12	217.13	12.13	306,399	8926	22,598	26,805	123,779	7453	81,381	35,457	/
Folder 4	13	99.27	13.18	178,368	25,026	9518	12,994	65,120	2079	43,466	20,165	/
	14	538.10	16.34	956,042	/	37,310	66,380	441,899	20,214	253,626	115,214	21,399
	15	186.24	7.86	171,519	/	34,220	/	58,032	/	46,206	30,871	2190
	16	236.52	23.93	422,489	25,228	31,719	9603	195,377	/	158,309	2253	/
Folder 5	17	175.70	7.73	167,822	/	41,712	/	51,332	/	43,752	29,438	1588
	18	40.05	12.21	72,471	18,866	1755	6078	23,358	1300	14,422	6692	/
	19	238.53	13.40	385,769	/	95,988	/	130,396	/	108,298	46,410	4677
Folder 6	20	538.32	16.27	947,821	/	37,238	67,872	442,031	18,825	252,545	108,245	21,065
	21	121.68	25.08	269,570	/	44,072	/	109,550	/	89,824	24,909	1215
	22	118.58	12.21	162,402	8828	6437	13,032	67,127	4078	43,855	19,045	/

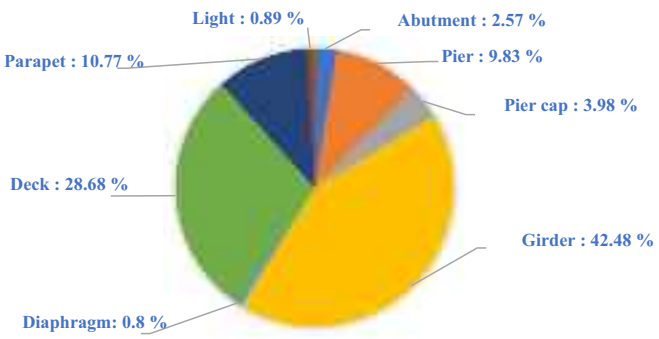

Fig. 7. Proportion of different categories in the synthetic bridge dataset.

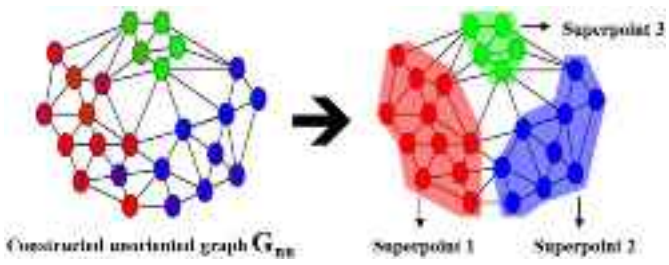

Fig. 8. Schematic representation of the workflow for superpoint generation.

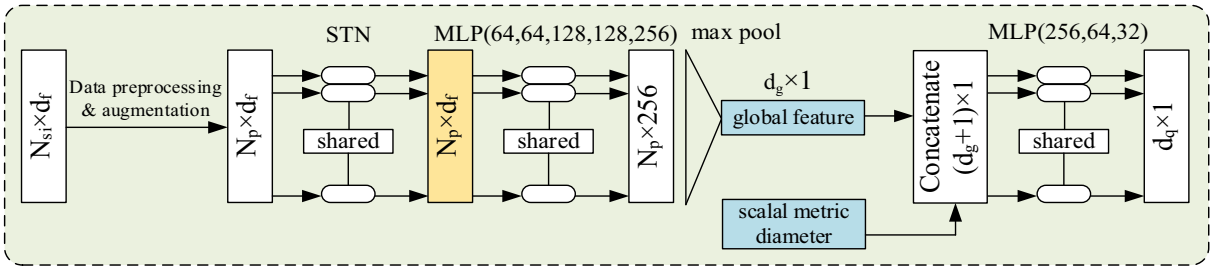

Fig. 9. Framework of the superpoint feature embedding process

W_O is the weight in the linear layer.

3.3.5. Loss function

The weighted cross entropy loss function [27] has been demonstrated to be effective for alleviating the issue of the imbalance of point cloud dataset, which can adaptively increase weights for points in minority bridge components. This novel loss function was defined as:

$$L = \sum_{c=1}^{M} \frac{\prod_{i=1, i\neq c}^{M} \sqrt{N_i}}{\sum_{i=1}^{M} \sqrt{N_i}} y_c log\left(\hat{y}_c\right) \tag{4}$$

where N_i is the total amount of points in i^{th} category and M denotes the total number of categories, while y is a label vector of ground truth and $\hat{y}$ denotes the predicted label vector.

We compared the weighted cross entropy loss function with Cross Entropy, Power Jaccard loss function, and two classical Jaccard losses to select the optimal loss function for the imbalanced datasets. The Power Jaccard loss function [41] is mainly designed for alleviating the issue of the highly imbalanced datasets by increasing the importance of smaller classes, as follows:

$$J_p\left(y_c, \hat{y}_c, p\right) = 1 - \frac{y_c \cdot \hat{y}_c + \varepsilon}{\left(y_c{}^p + \hat{y}_c{}^p - y_c \cdot \hat{y}_c\right) + \varepsilon} \tag{5}$$

where ε denotes a positive number that approaches zero, and p is a preset positive number.

Two classical Jaccard losses are Jaccard distance and Dice score, which are also commonly used loss functions for imbalanced datasets. When $p = 1$, Eq. (5) is identical to Jaccard distance [42]. Dice score [43] is defined as follows:

$$J_p\left(y_c, \hat{y}_c, p\right) = 1 - \frac{2 \cdot y_c \cdot \hat{y}_c}{\left(y_c{}^2 + \hat{y}_c{}^2\right)} \tag{6}$$

4. Experiment and results

4.1. Implementation details

The proposed WSPG method is evaluated based on the real-world and synthetic dataset. The deep learning framework Pytorch is used and implemented on Ubuntu 20.04 through an AMD EPYC 7302 CPU and a single NVIDIA GeoForce RTX 3090 GPU. We first voxelized the input point clouds with 0.03 m voxel widths as the best space resolution. For geometric partition, regularization strength is empirically set as 0.01, 0.03, 0.05 for comparison to select the best value. This step aims at striking a balance between semantic homogeneity of superpoints and the potential for their successful distinction. During training, we leverage Adam optimizer to minimize the overall loss. The initial learning rate is set to 0.01, and batch size is 4. The network is trained for 350 epochs with stepwise learning rate decay of 0.7 at epochs 275 and 320. During testing, we leverage different strategies for the real-world dataset and the synthetic dataset according to the training data sufficiency. For the real-world dataset, we conduct the evaluation experiment on the real-world testing dataset. Regarding the synthetic dataset, we adopt a standard 6-fold cross-validation [25], where the testing is recursively performed on one folder and the training on the remaining folders. The evaluation metrics are computed by averaging predictions of all test folders.

The overall accuracy (OA), mean class Accuracy (mAcc), and mean Intersection over Union (mIoU: the average value of IoUs for all categories upon the whole dataset) are used as quantitative evaluation metrics, which are defined, respectively, in Eqs. (7)–(9).

$$OA = \sum_{i=1}^{M} \frac{(TP)_i}{N_i} \tag{7}$$

$$mAcc = \frac{1}{M} \sum_{i=1}^{M} \frac{(TP)_i}{N_i} \tag{8}$$

$$mIoU = \frac{1}{M} \sum_{i=1}^{M} \frac{T_i \cap P_i}{T_i \cup P_i} = \frac{1}{M} \sum_{i=1}^{M} \frac{(TP)_i}{(TP)_i + (FP)_i + (FN)_i} \tag{9}$$

where N is the total number of points in each bridge component, TP denotes the "true positive" (e.g., for abutment, the number of points that are correctly predicted as abutment), FP means "false positive" (e.g., for abutment, the number of points that are wrongly predicted as abutment), FN represents "false negative" (e.g., for abutment, the number of points that are predicted as other components, but ground truth is abutment).

4.2. Evaluation

4.2.1. Qualitative evaluation

4.2.1.1. Real-world dataset. Fig. 10 shows the visualization of the predicted results for the proposed method on the real-world testing dataset compared to manually annotated results. As can be seen from Fig. 10, the overall performance of the proposed method is outstanding, but failure is still evident at the boundaries between adjacent bridge components.

4.2.1.2. Synthetic dataset. Fig 11 presents some visual examples of the performance of the proposed method compared to the manually annotated results. Although the synthetic bridge dataset has various components such as decks and girders with curved shapes as well as piers with different height, the proposed framework can successfully classify these complex components. The overall performance of our proposed method is satisfactory, but two situations of failure still remain. First, small-scale and low-density objects are still likely to be missed, as shown in Fig. 11(b). For example, the length of the diaphragm is relatively small at around 0.3 m, occupying less than 10 voxels for superpoint

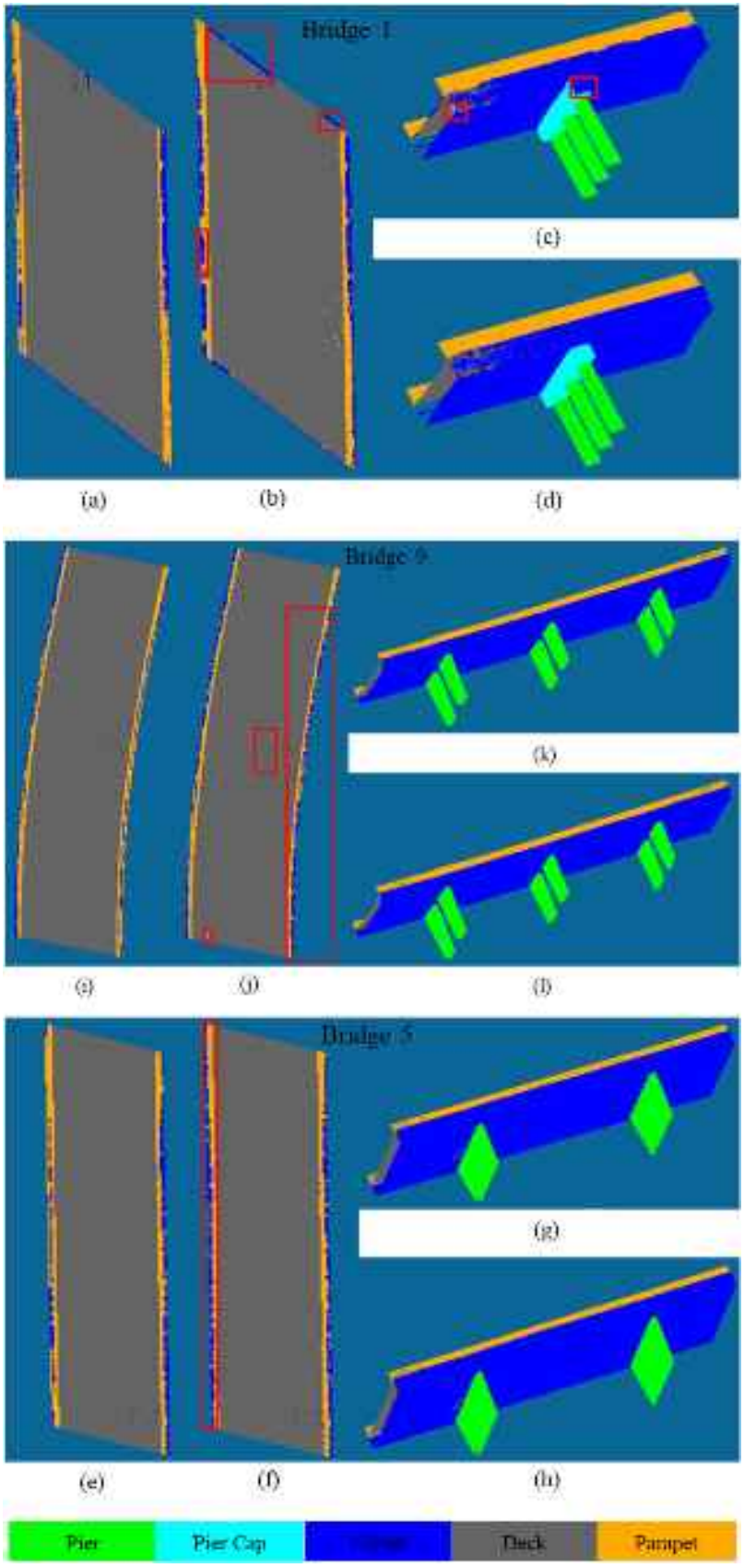

Fig. 10. Visualization of prediction results on the real-world testing dataset: (a), (c), (e), (g), (i) and (k) display manually annotated results, while (b), (d), (f), (h), (j) and (l) are the automatically predicted results. Red boxes are the misprediction areas. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

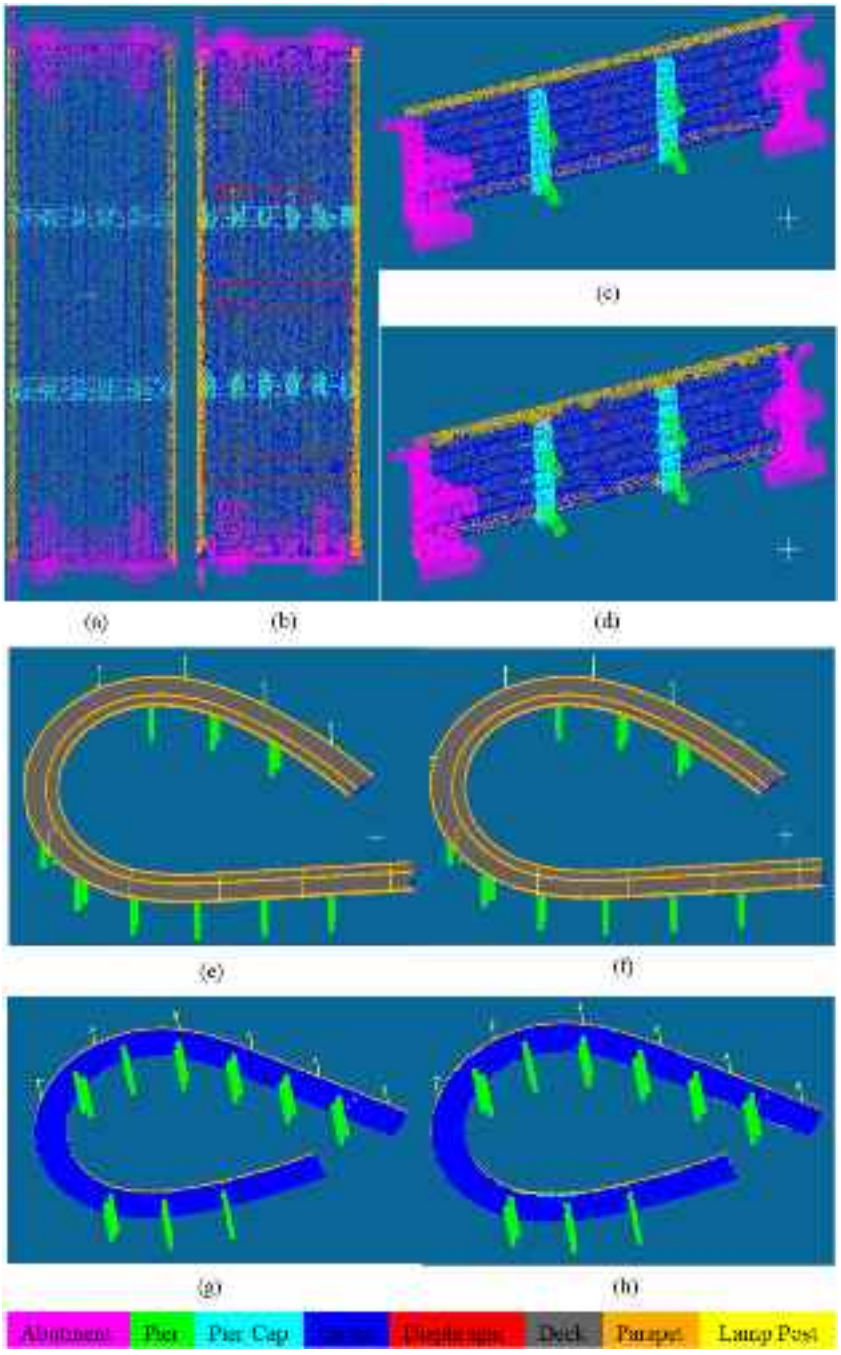

Fig. 11. Examples of prediction results on the synthetic bridge dataset: (a), (c), (e) and (g) display manually annotated results, while (b), (d), (f) and (h) are the automatically predicted results. Red boxes are the misprediction areas. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

generation and thus resulting in a lack of training samples. Second, the proposed framework cannot discriminate some boundaries correctly, such as the boundary between pier cap and girder, as shown in Fig. 11 (d), causing trouble for upper-level tasks such as defect localization and bridge information modelling. This is mainly because the boundary between two adjacent components is weakly defined due to the semantic ambiguity, meaning that a point may belong to multiple classes at the same time.

4.2.2. Quantitative evaluation

The quantitative evaluation of the proposed method consists of four parts. The first part experimentally derives the optimal value for the regularization strength. The second part analyzes the performance of the proposed WSPG model in detail both on the real-world and synthetic dataset. The third part assesses the performance of the proposed loss function compared with four representative loss functions. The final part compares the proposed method with three cutting edge baselines to validate the efficiency and robustness of the proposed method.

4.2.2.1. Selection of the optimal value for regularization strength. Table 2 presents the influence of different predefined values of regularization strength on the performance of the proposed method. We conducted this experiment on the synthetic dataset as it includes more types of bridge components. As can be seen from Table 2, three values of the

Table 2
Testing results (%) on the synthetic dataset for different regularization strength values.

RS	OA	mAcc	mIoU	IoU for each class							
				Abut	Pier	Pier cap	Girder	Diaph	Deck	Para	Lamp post
0.01	93.85	86.56	80.84	82.51	94.67	78.41	89.45	42.47	89.13	86.48	83.61
0.03	93.05	88.77	79.90	77.88	95.27	76.61	88.17	44.18	88.23	83.37	85.49
0.05	92.21	87.62	79.69	84.05	95.65	78.96	86.03	38.14	85.99	81.48	87.20

regularization strength have little influence on the metrics of overall accuracy and mIoU, however based on mAcc, the regularization strength value 0.03 showed the best results, exceeding the other regularization strength values by 1%–2%. Additionally, we found that the computing time increased with a decrease of the regularization strength value when training the algorithm. Thus, the regularization strength value 0.03 is chosen as the optimal value by considering the balance between the computational efficiency and the result accuracy.

4.2.2.2. Detailed performance analysis of the WSPG. We firstly validated the effectiveness of the proposed method on the real-world dataset with 5 categories of bridge components, as shown in Table 3. The overall performance of the proposed method is outstanding, achieving the overall accuracy of 99.43%,. Additionally, the proposed method achieved excellent testing results on pier (99.77%), girder (99.76%), deck (97.91%) and pier cap (95.02%) components, and performed well on parapet (90.01). The main reason for the relatively low result of parapet is the misclassification between the parapet and deck. Notably, the real-world bridge point clouds provided by Lu et al. [13] are relatively simple, and future work can focus on collecting more complicated bridge types.

Table 4 presents the computation time spent on the real-world testing dataset. As can be seen from Table 4, for a real-world 40-m long bridge with a high point density, the proposed method takes around 4.3 mins to implement automated semantic segmentation for 5 types of bridge components. It should be noted that the computation time of the proposed method significantly relies on the downsampling resolution, point density and the bridge scale. In general, the computation time is inversely proportional to the downsampling resolution and point density, and directly proportional to the scale of the bridge. The priority of this study focused on the semantic segmentation accuracy, so we chose a relatively small downsampling resolution at the expense of the speed.

For validation on the synthetic dataset, the test result on each folder and average performance of all 6 folders are reported to intuitively analyze the behavior of our proposed approach, as shown in Figs. 12 and 13.

As can be seen from Fig. 13, Folder 1 does not contain diaphragms and reflects that the model trained on one bridge scene with 8 categories can be generalized to another bridge scene with 7 categories. Next, the best testing result is obtained on Folder 5, achieving a 96.76% overall accuracy. Folder 4 indicates the lowest performance in terms of overall accuracy (89.61%). The possible reason is this folder includes complex features of bridge components that cannot be efficiently learned from the other training folders. In addition, the proposed approach achieved satisfactory results on pier (95.27%), deck (88.23%), girder (88.17%), lamp post (85.49%) as well as parapet (83.37%) and performed well on abutment (77.88%) and pier cap (76.61). Notably, the segmentation performance of diaphragm is relatively low, only achieving 44.18%.

Table 3
Evaluation results (%) on the real-world testing dataset.

OA	mAcc	mIoU	IoU for each class				
			Pier	Pier cap	Girder	Deck	Parapet
96.49	98.75	99.43	99.77	95.02	99.76	97.91	90.01

This is possibly due to the small occupancy and complex structure of these components, leading to difficulty learning effective features. Diaphragm points have some feature distributions such as the height distribution similar to girder points, imposing challenges when segmenting diaphragm points from adjacent girders. In addition, the recognition performance of abutment and pier caps are not fully satisfied. One possible reason for the deficiency of abutment segmentation is the lack of sufficient training data because a full-scale bridge only has 2 abutments compared with multiple piers, long decks and girders. Regarding the pier cap, the most likely reason for low accuracy is its spatial proximity to adjacent pier points and girder points, meaning the boundary between them can have an ambiguous definition, thus causing unavoidable misclassifications.

Table 5 summarizes the computation time over different steps of our framework for inference on the synthetic bridge dataset. It is worth noting that the bulk of the computation time was dedicated to geometric partition and SPG construction. For an approximately 160-m long bridge, the proposed solution takes around 4.5 mins on average to implement semantic segmentation for 8 types of bridge components, significantly speeding up such tasks compared to manual annotation.

4.2.2.3. Performance comparison between representative loss functions. Figs. 14 and 15 compare the results using different loss functions to demonstrate the superiority of the loss function used in our study. In general, the proposed approach significantly outperforms the competitors in terms of the metrics of overall accuracy (93.05%) and mAcc (88.77%) as well as mIoU (79.90%). These results can be credited to our proposed weighted loss function that assigns weights according to the number of points contained in different components. Notably, the performance of the proposed loss function achieved optimal results in five out of eight component categories, exhibiting the strength of our method. In addition, other loss functions do not perform well across all these components and may fail for one or two components.

4.2.2.4. Performance comparison between state-of-the-art baselines. The effectiveness and robustness of the proposed WSPG were also compared with three state-of-the-art baselines both on the real-world and synthetic dataset. Notably, PointNet and DGCNN are trained on the sub-space partitioning dataset with a subspace length of 5 m and an overlap rate of 80%. These methods use 3 NVIDIA GeoForce RTX 3090 GPUs, as they are computational expensive and time-consuming.

Fig. 16 presents a comparison of results from the proposed solution and three representative methods for the real world dataset. As indicated in Fig. 16, our proposed method achieved the best performance on all three overall evaluation metrics, especially on the metric of the mIoU, exceeding the PointNet, DGCNN and the original SPG by 6.47%, 8.36% and 4.11% respectively. The strength of the proposed method for 5 types of bridge components is shown by the results presented in Fig. 17.

The performance metrics for the synthetic dataset across different models are presented in Fig. 18. The proposed WSPG exhibits good performance based on the metrics of mAcc (88.77%) and mIoU (79.90%), surpassing the original SPG model by 8.5% and 6.7% respectively. Along with this, the WSPG method exceeds two other conventional deep learning models in terms of computational efficiency and overall performance. We also compared the performance of IoUs for

Table 4
Computation time (seconds) over different steps of the proposed method on the real-world testing dataset.

Bridge No.	Point number	Feature computation	Geometric Partition	SPG construction	Inference	Total
1	3,205,907	3.1	30.4	22.3	24.1	79.9
5	859,231	13.4	135.3	80.2	30.7	259.6
9	32,944,919	22.4	226.9	142.8	38.4	430.5

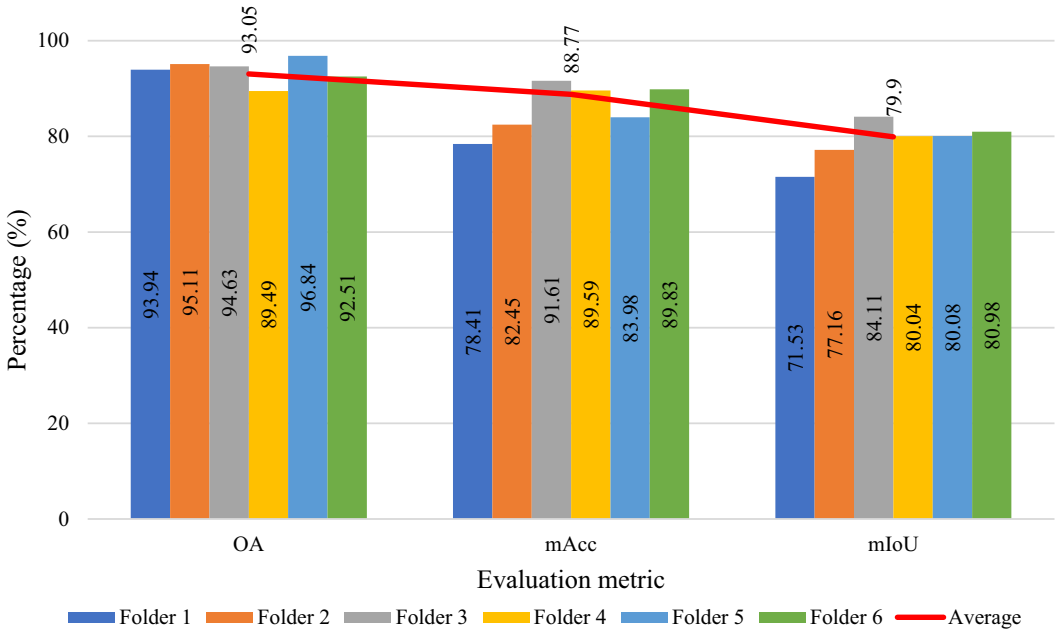

Fig. 12. Overall testing results (%) for different folders on the synthetic dataset.

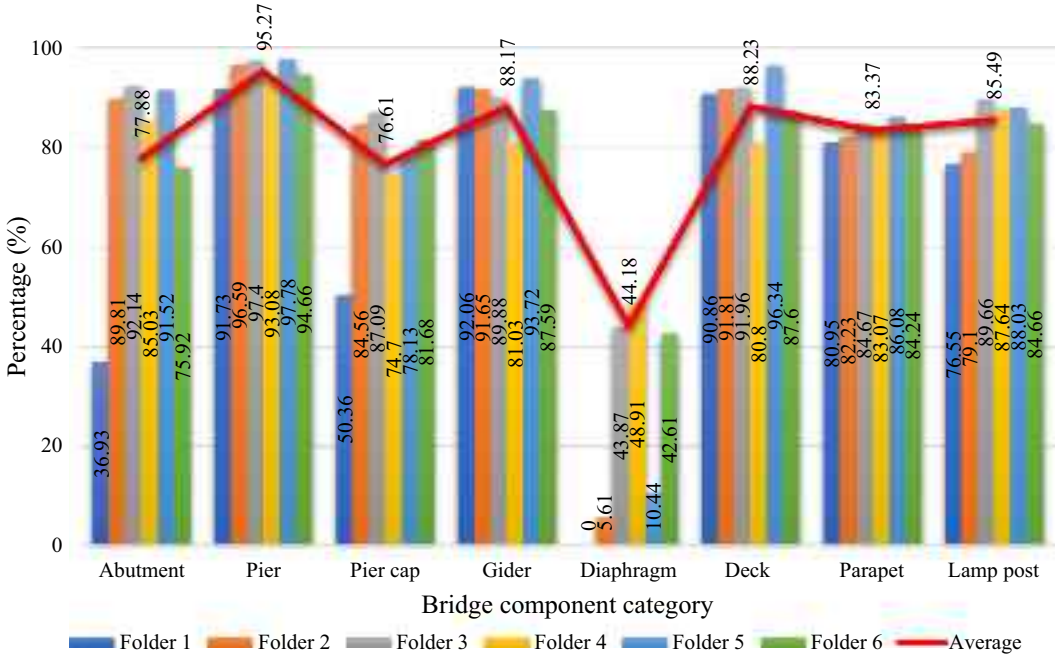

Fig. 13. IoUs for 8 categories of bridge components in different folders on the synthetic dataset.

8 categories between representative models in Fig. 19. Notably, the performance of the WSPG solution outperforms the original SPG model in all of the eight categories, and of note is the 110% improvement in the recognition of diaphragm components. The WSPG also exceeds the PointNet and DGCNN models in most categories. Although our proposed solution significantly improves the classification accuracy of diaphragm compared with the three baselines, its classification accuracy is still lower than that of other components. It should also be noted that the

Table 5
Computation time (seconds) over different steps of the proposed method on the synthetic bridge dataset.

Folder	Bridge number	Point number	Feature computation	Geometric Partition	SPG construction	Inference	Total	Average
1	4	1,371,601	45.5	448.5	513.2	61.0	1068.2	267.1
2	4	1,106,321	88.3	873.8	1031.0	78.0	2071.1	517.8
3	4	946,647	130.6	1310	1507.7	60.0	3008.3	752.1
4	4	1,728,418	210.5	1931.3	2337.6	167.1	4646.5	1161.6
5	3	626,062	193.4	1797.5	2122.7	38.4	4152.0	1384.0
6	3	1,379,793	256.5	2183.8	2713.8	309.7	5463.8	1821.3

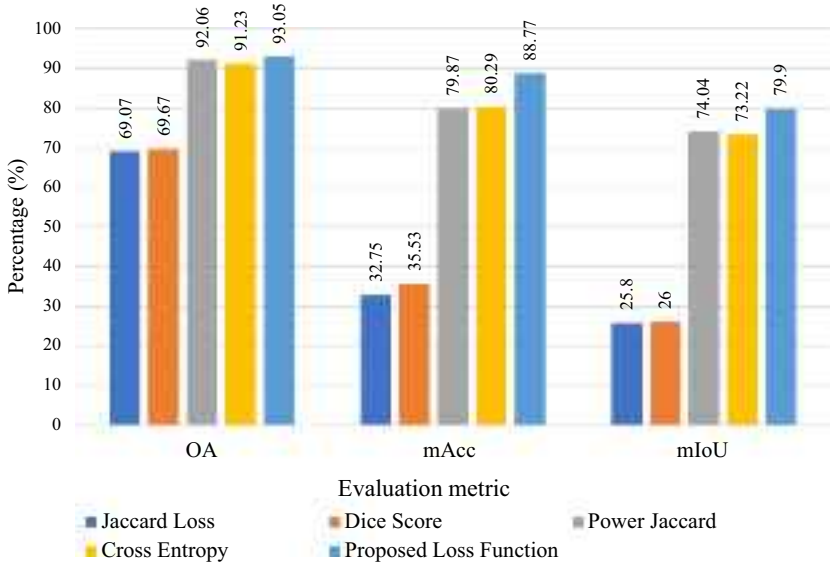

Fig. 14. Comparison results (%) using different representative loss function.

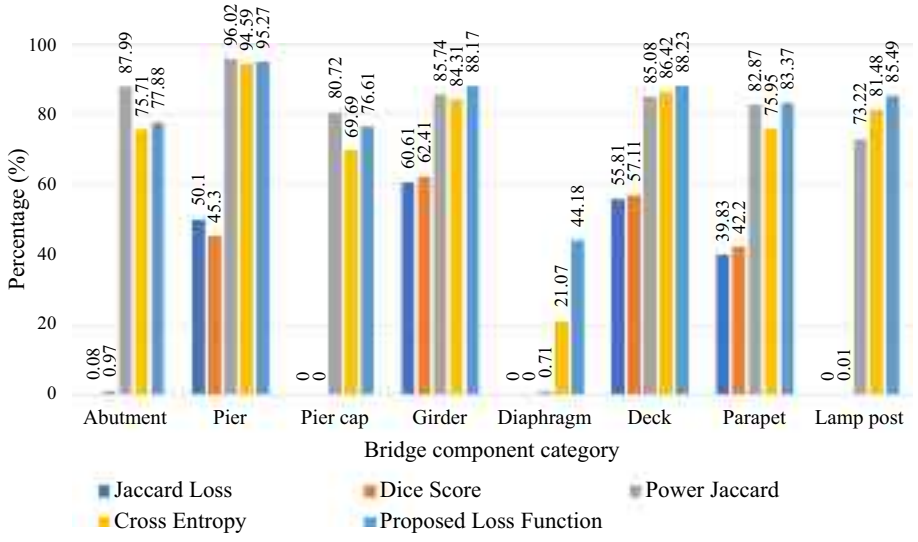

Fig. 15. Metric of IoU (%) for each component category.

PointNet algorithm does not exhibit segmentation ability for diaphragms.

5. Conclusions

Automated semantic segmentation of bridge components from large-scale point clouds remains a global challenge. In this paper, two well-annotated datasets for the real-word and synthetic bridge point clouds were established. Notably, during this step, we presented a framework for converting BrIMs into synthetic bridge point clouds by leveraging three existing software: Autodesk Revit, ifcConvert and CloudCompare. We then presented a WSPG method to implement automated semantic segmentation of bridge components directly from full-scale bridge point clouds, which is built on top of the original SPG model by leveraging a novel loss function to alleviate the imbalance between component categories and enhance the overall performance. The proposed WSPG

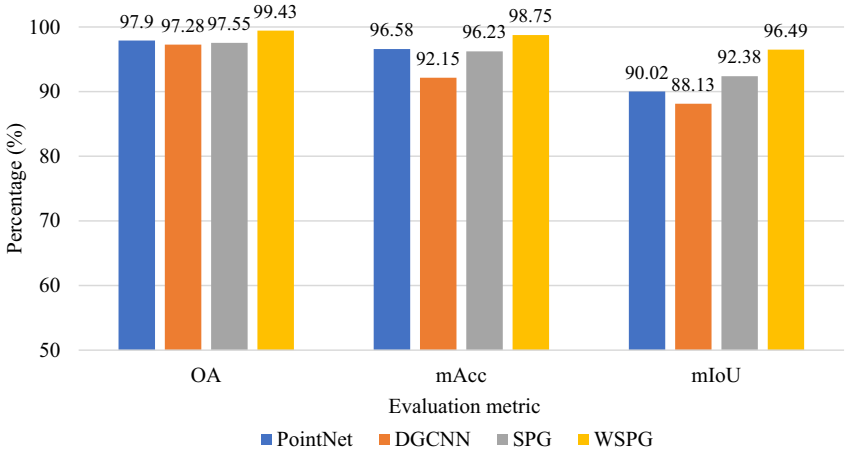

Fig. 16. Comparison of performance metrics between representative models (%) on the real-world dataset.

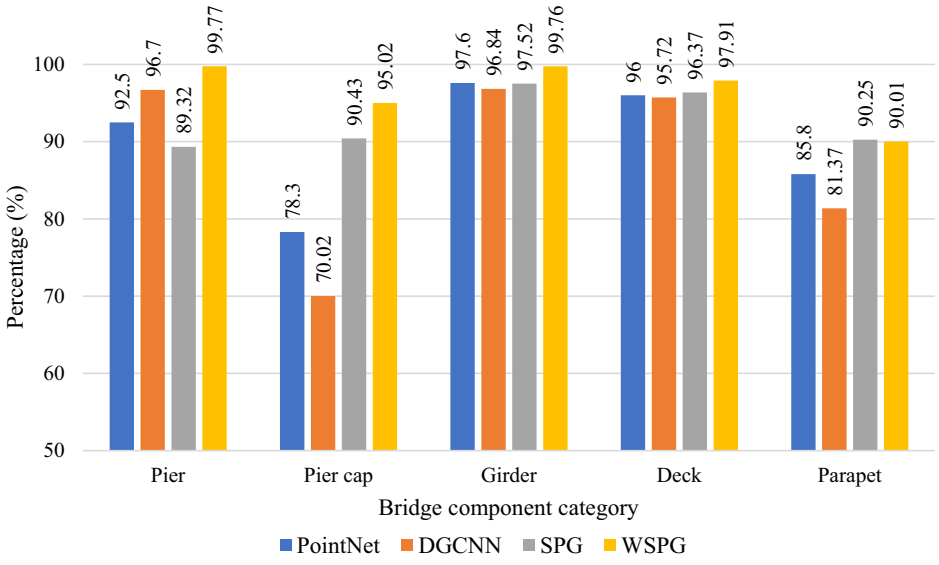

Fig. 17. Metric of IoU for each component category using representative models on the real-world dataset.

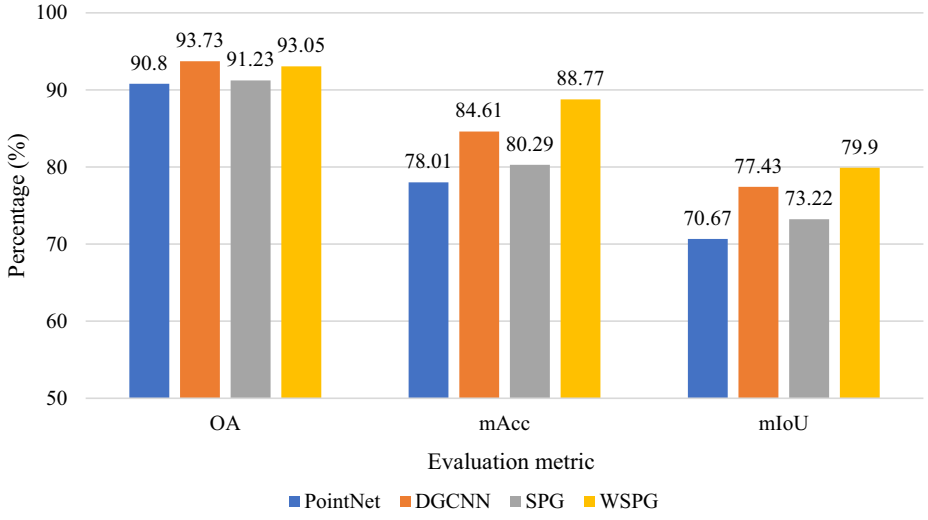

Fig. 18. Comparison of performance metrics between representative models (%) on the synthetic dataset.

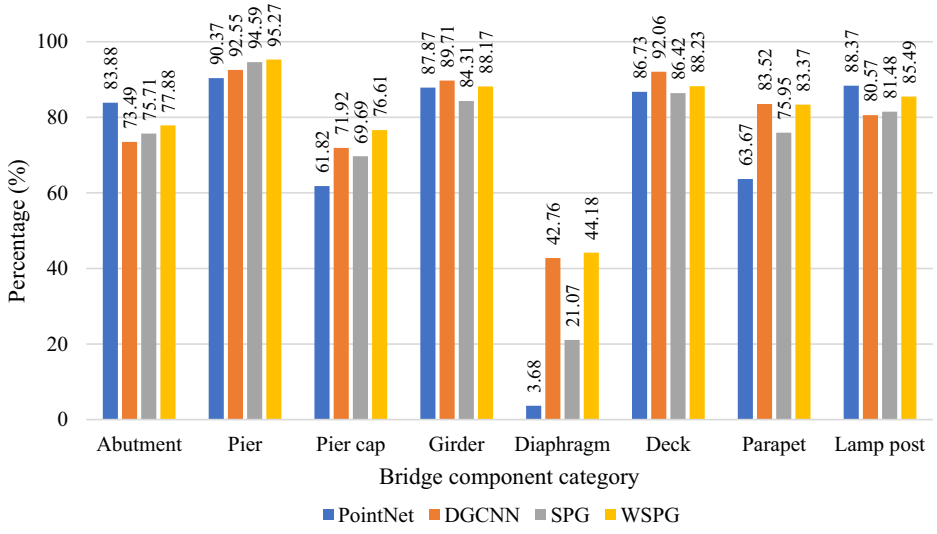

Fig. 19. Metric of IoU for each component category using representative models on the synthetic dataset.

method was validated both on the real-world dataset and the synthetic dataset. The experiment results showed that the performance of the WSPG method achieved outstanding performance on the real-world dataset, with evaluation metrics of 99.43% overall accuracy, 98.75% mAcc and 96.49% mIoU. Regarding the synthetic dataset, the proposed method is better than conventional deep learning models such as PointNet and DGCNN, with the metrics of overall accuracy equal to 93.05%, mAcc equal to 88.77%, and mIoU equal to 79.90%. Finally, the computation time was only around 4.3 mins and 4.5 mins on a 40-m long real-world bridge with a high point density and a 160-m long synthetic bridge respectively, significantly faster than the manual annotation approach. However, the proposed method does not intend to be a cure-all. Some limitations still remain: 1) deep learning-based methods require a huge amount of training samples, but little research has paid attention to leveraging synthetic point clouds to augment the real-world training dataset; 2) most existing bridges do not have BrIMs, so it is a challenge to leverage BrIMs to generate a huge amount of synthetic bridge point clouds for various bridge types; 3) this study adopts the supervised deep learning method, demanding a large amount of manual annotation work for the dataset that is tedious and time-consuming; 4) small-scale and low-density objects like diaphragms are still likely to be missed; 5) it is still a challenge for the proposed method to consistently discriminate the boundaries between two adjacent components.

To address or at least mitigate the aforementioned limitations, future work should 1) investigate the feasibility of using synthetic bridge point clouds to augment the real-world training samples; 2) explore a user-friendly framework for generating the synthetic dataset from bridge finite element models (FEM) as they are much easier to obtain; 3) develop an automated semantic annotation method for generating well-annotated synthetic point cloud datasets from BrIMs or FEMs, reducing annotation workload; 4) develop a high-quality superpoint generation method to alleviate erroneous component boundary segmentation and thus improve the segmentation performance for minority bridge components. Additionally, future work may also focus on upper-level tasks such as (a) identification of 3D spatial locations of defects as well as damage quantification by integrating the image-based defects with bridge component point clouds and (b) automated generation of digital twins for existing bridges through fitting recognized bridge component point clouds with IFC objects to efficiently support bridge documentation and management.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The authors do not have permission to share data.

Acknowledgements

The authors would like to acknowledge the support by University of Auckland FRDF Grant (Project No. 3716476).

References

[1] R. Foundation, Policy Study: Structurally Deficient Bridges. https://reason.org/policy-study/24th-annual-highway-report%20/structurally-deficient-bridges/, 27 June, 2022.
[2] R. Lu, I. Brilakis, Digital twinning of existing reinforced concrete bridges from labelled point clusters, Autom. Constr. 105 (2019), 102837, https://doi.org/10.1016/j.autcon.2019.102837.
[3] N. Galvão, J.C. Matos, D.V. Oliveira, R. Hajdin, Human error impact in structural safety of a reinforced concrete bridge, Struct. Infrastruct. Eng. (2021) 1–15, https://doi.org/10.1080/15732479.2021.1876105.
[4] Y. Yan, Z. Mao, J. Wu, T. Padir, J.F. Hajjar, Towards automated detection and quantification of concrete cracks using integrated images and lidar data from unmanned aerial vehicles, Struct. Control. Health Monit. 28 (8) (2021), e2757, https://doi.org/10.1002/stc.2757.
[5] B. Adriano, J. Xia, G. Baier, N. Yokoya, S. Koshimura, Multi-source data fusion based on ensemble learning for rapid building damage mapping during the 2018 Sulawesi earthquake and tsunami in Palu, Indonesia, Remote Sens. 11 (7) (2019) 886, https://doi.org/10.3390/rs11070886.
[6] Y. Zhao, H. Wu, P.A. Vela, Top-Down Partitioning of Reinforced Concrete Bridge Components, Computing in Civil Engineering 2019: Smart Cities, Sustainability, and Resilience, American Society of Civil Engineers Reston, VA, 2019, pp. 275–283, https://doi.org/10.1061/9780784482445.035.
[7] Q. Wang, M.-K. Kim, Applications of 3D point cloud data in the construction industry: a fifteen-year review from 2004 to 2018, Adv. Eng. Inform. 39 (2019) 306–319, https://doi.org/10.1016/j.aei.2019.02.007.
[8] R. Lu, I. Brilakis, Recursive segmentation for as-is bridge information modelling, in: F. Bosche, I. Brilakis, R. Sacks (Eds.), Proceedings of the Joint Conference on Computing in Construction Vol. 1, Heraklion, Crete, Greece, 2017, pp. 209–217, https://doi.org/10.24928/jc3-2017/0020.
[9] F. Hu, J. Zhao, Y. Huang, H. Li, Structure-aware 3D reconstruction for cable-stayed bridges: a learning-based method, Comp. Aid. Civ. Infrastruct. Eng. 36 (1) (2021) 89–108, https://doi.org/10.1111/mice.12568.
[10] H. Kim, J. Yoon, S.H. Sim, Automated bridge component recognition from point clouds using deep learning, Struct. Control. Health Monit. 27 (9) (2020), e2591, https://doi.org/10.1002/stc.2591.

[11] H. Kim, C. Kim, Deep-learning-based classification of point clouds for bridge inspection, Remote Sens. 12 (22) (2020) 3757, https://doi.org/10.3390/rs12223757.
[12] T. Xia, J. Yang, L. Chen, Automated semantic segmentation of bridge point cloud based on local descriptor and machine learning, Autom. Constr. 133 (2022), 103992, https://doi.org/10.1016/j.autcon.2021.103992.
[13] R. Lu, I. Brilakis, C.R. Middleton, Detection of structural components in point clouds of existing RC bridges, Comp. Aid. Civ. Infrastruct. Eng. 34 (3) (2019) 191–212, https://doi.org/10.1111/mice.12407.
[14] J.W. Ma, T. Czerniawski, F. Leite, Semantic segmentation of point clouds of building interiors with deep learning: augmenting training datasets with synthetic BIM-based point clouds, Autom. Constr. 113 (2020), 103144, https://doi.org/10.1016/j.autcon.2020.103144.
[15] L. Landrieu, M. Simonovsky, Large-scale point cloud semantic segmentation with superpoint graphs, Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (2018) 4558–4567, https://doi.org/10.1109/CVPR.2018.00479.
[16] A. Dimitrov, M. Golparvar-Fard, Segmentation of building point cloud models including detailed architectural/structural features and MEP systems, Autom. Constr. 51 (2015) 32–45, https://doi.org/10.1016/j.autcon.2014.12.015.
[17] S.B. Walsh, D.J. Borello, B. Guldur, J.F. Hajjar, Data processing of point clouds for object detection for structural engineering applications, Comp. Aid. Civ. Infrastruct. Eng. 28 (7) (2013) 495–508, https://doi.org/10.1111/mice.12016.
[18] L. Truong-Hong, R. Lindenbergh, Automatically extracting surfaces of reinforced concrete bridges from terrestrial laser scanning point clouds, Autom. Constr. 135 (2022), 104127, https://doi.org/10.1016/j.autcon.2021.104127.
[19] R. Schnabel, R. Wahl, R. Klein, Efficient RANSAC for Point-Cloud Shape Detection, Computer Graphics Forum Vol. 26, Wiley Online Library, 2007, pp. 214–226, https://doi.org/10.1111/j.1467-8659.2007.01016.x.
[20] S. Chen, L. Truong-Hong, D. Laefer, E. Mangina, Automated Bridge Deck Evaluation Through UAV Derived Point Cloud, CERI-ITRN2018, Dublin, Ireland, 2018, pp. 735–740. http://hdl.handle.net/2451/43478.
[21] B.J. Perry, Y. Guo, R. Atadero, J.W. van de Lindt, Streamlined bridge inspection system utilizing unmanned aerial vehicles (UAVs) and machine learning, Measurement. 164 (2020), 108048, https://doi.org/10.1016/j.measurement.2020.108048.
[22] J. Yan, J. Shan, W. Jiang, A global optimization approach to roof segmentation from airborne lidar point clouds, ISPRS J. Photogramm. Remote Sens. 94 (2014) 183–193, https://doi.org/10.1016/j.isprsjprs.2014.04.022.
[23] Y. Yan, J.F. Hajjar, Automated extraction of structural elements in steel girder bridges from laser point clouds, Autom. Constr. 125 (2021), 103582, https://doi.org/10.1016/j.autcon.2021.103582.
[24] Z. Dong, B. Yang, P. Hu, S. Scherer, An efficient global energy optimization approach for robust 3D plane segmentation of point clouds, ISPRS J. Photogramm. Remote Sens. 137 (2018) 112–133, https://doi.org/10.1016/j.isprsjprs.2018.01.013.
[25] C.R. Qi, H. Su, K. Mo, L.J. Guibas, Pointnet: deep learning on point sets for 3d classification and segmentation, Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (2017) 652–660, https://doi.org/10.1109/CVPR.2017.16.
[26] J.S. Lee, J. Park, Y.-M. Ryu, Semantic segmentation of bridge components based on hierarchical point cloud model, Autom. Constr. 130 (2021), 103847, https://doi.org/10.1016/j.autcon.2021.103847.
[27] X. Han, Z. Dong, B. Yang, A point-based deep learning network for semantic segmentation of MLS point clouds, ISPRS J. Photogramm. Remote Sens. 175 (2021) 199–214, https://doi.org/10.1016/j.isprsjprs.2021.03.001.
[28] M.-K. Kim, S. McGovern, M. Belsky, C. Middleton, I. Brilakis, A suitability analysis of precast components for standardized bridge construction in the United Kingdom, Procedia engineering. 164 (2016) 188–195, https://doi.org/10.1016/j.proeng.2016.11.609.
[29] Buildingsmart, Ifc and Related Data Model Standards Documentation. https://technical.buildingsmart.org/standards/ifc/ifc-formats/, 27 June, 2022.
[30] IfcOpenShell, IfcConvert: An Application for Converting ifc Geometry into Several File Formats. http://ifcopenshell.org/ifcconvert, 27 June, 2022.
[31] CloudCompare, CloudCompare: 3D Point Cloud and Mesh Processing Software. https://www.danielgm.net/cc/, 27 June, 2022.
[32] M. Soilán Rodríguez, A. Novoa Martínez, A. Sánchez Rodríguez, B. Riveiro Rodríguez, P. Arias Sánchez, Semantic segmentation of point clouds with PointNet and KPConv architectures applied to railway tunnels, ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci. 5 (2) (2020) 281–288, https://doi.org/10.5194/isprs-annals-V-2-2020-281-2020.
[33] R. Martin, I. Stroud, A. Marshall, Data Reduction for Reverse Engineering, RECCAD, Deliverable Document 1 COPERNICUS Project, No 1068, 1997, p. 111, doi:10.1.1.61.3620.
[34] J. Niemeyer, J.D. Wegner, C. Mallet, F. Rottensteiner, U. Soergel, Conditional random fields for urban scene classification with full waveform LiDAR data, in: ISPRS Conference on Photogrammetric Image Analysis, Springer, 2011, pp. 233–244, https://doi.org/10.1007/978-3-642-24393-6_20.
[35] M. Weinmann, B. Jutzi, C. Mallet, Semantic 3D scene interpretation: a framework combining optimal neighborhood size selection with relevant features, ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci. 2 (3) (2014) 181, https://doi.org/10.5194/isprsannals-ll-3-181-2014.
[36] S. Guinard, L. Landrieu, Weakly supervised segmentation-aided classification of urban scenes from 3D LiDAR point clouds, international archives of the photogrammetry, remote sensing and spatial, Inf. Sci. 42 (1/W1) (2017), https://doi.org/10.5194/isprs-archives-XLII-1-W1-151-2017.
[37] J.W. Jaromczyk, G.T. Toussaint, Relative neighborhood graphs and their relatives, Proc. IEEE 80 (9) (1992) 1502–1517, https://doi.org/10.1109/5.163414.
[38] M. Jaderberg, K. Simonyan, A. Zisserman, Spatial transformer networks, Adv. Neural Inf. Proces. Syst. 28 (2015) 2017–2025, doi:1506.02025.
[39] Y. Li, D. Tarlow, M. Brockschmidt, R. Zemel, Gated graph sequence neural networks, arXiv Prepr. (2015) arXiv:1511.05493. doi:1511.05493.
[40] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, Learning phrase representations using RNN encoder-decoder for statistical machine translation, arXiv Prepr. (2014) arXiv:1406.1078. doi:1406.1078.
[41] D. Duque-Arias, S. Velasco-Forero, J.-E. Deschaud, F. Goulette, A. Serna, E. Decencière, B. Marcotegui, On power Jaccard Losses for Semantic Segmentation, in: VISAPP 2021: 16th International Conference on Computer Vision Theory and Applications Vol. 5, 2021, pp. 561–568, https://doi.org/10.5220/0010304005610568.
[42] I. Martire, P. Da Silva, A. Plastino, F. Fabris, A.A. Freitas, A novel probabilistic Jaccard distance measure for classification of sparse and uncertain data, in: E.a.M. Rebeiro de Faria Paiva, Luiz, Ricardo Cerri (Eds.), Proceedings of the 5th Symposium on Knowledge Discovery, Mining and Learning (KDMiLe), 2017, pp. 81–88. https://kar.kent.ac.uk/67128/.
[43] F.I. Diakogiannis, F. Waldner, P. Caccetta, C. Wu, ResUNet-a: a deep learning framework for semantic segmentation of remotely sensed data, ISPRS J. Photogramm. Remote Sens. 162 (2020) 94–114, https://doi.org/10.1016/j.isprsjprs.2020.01.013.