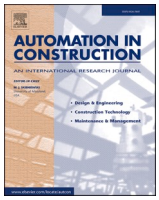
Automation in Construction 158 (2024) 105176

Contents lists available at ScienceDirect

Automation in Construction

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



Automated production of synthetic point clouds of truss bridges for semantic and instance segmentation using deep learning models

Daniel Lamas *, Andrés Justo, Mario Soilán, Belén Riveiro

CINTECX, Universidade de Vigo, GeoTECH Group, Campus Universitario de Vigo, As Lagoas, Marcosende, 36310 Vigo, Spain

ARTICLE INFO

Keywords:
Point clouds
Truss bridge
Semantic segmentation
Instance segmentation
Synthetic data
Deep learning

ABSTRACT

The cost of obtaining large volumes of bridge data with technologies like laser scanners hinders the training of deep learning models. To address this, this paper introduces a new method for creating synthetic point clouds of truss bridges and demonstrates the effectiveness of a deep learning approach for semantic and instance segmentation of these point clouds. The method generates point clouds by specifying the dimensions and components of the bridge, resulting in high variability in the generated dataset. A deep learning model is trained using the generated point clouds, which is an adapted version of JSNet. The accuracy of the results surpasses previous heuristic methods. The proposed methodology has significant implications for the development of automated inspection and monitoring systems for truss bridges. Furthermore, the success of the deep learning approach suggests its potential for semantic and instance segmentation of complex point clouds beyond truss bridges.

1. Introduction

Infrastructure systems serve as the foundation of a nation, exerting significant influence on its society and economy. Specifically, Critical Infrastructure Systems (CIS), such as water, power supply, or transportation, play a pivotal role in the nation's development. Furthermore, in the event of a disaster, CIS are integral to the recovery process and are responsible for managing and mitigating the associated risks. Thus, the resilience of these assets is of paramount importance, as they must endure the negative consequences arising from such events, minimizing their impact and costs, safeguard their users, and hasten their recovery [1,2].

Transportation infrastructure is recognized as a CIS, essential for the optimal functioning of the society. Its function of facilitating the movement of goods and individuals has a direct impact on the well-being of citizens and the economy as a whole. Additionally, it serves as an intermediary between other infrastructure systems by enabling the movement of resources, including both tangible and intangible assets between them [3].

For the specific case of bridges, these are infrastructures to overcome the physical barriers and limitations of the terrain, such as rivers and canyons, thereby supporting other transportation networks, including roads, railways, and pedestrian walkways. The advantages of bridges are multifaceted, encompassing cultural, historical, social, and economic dimensions. However, a large part of the bridge stock are currently aging and require urgent maintenance. In some cases, the deterioration of infrastructure can result in structural collapse. According to the scientific and knowledge service of the European Commission [4], there are over 1234 km of road bridges longer than 100 m in Europe, many of which were constructed during the 1950s and have surpassed their expected lifespan. Given these concerns, and the fact that visual inspection remains the most common method of assessing bridge condition, there is a growing demand for research in the area of bridge maintenance. In this regard, digitizing assets for inspection tasks can offer a substantial boost in productivity.

Laser scanning is a highly popular technology for digitalizing infrastructure and obtaining precise geometric data of existing assets. Numerous studies have demonstrated the effectiveness of laser scanners for infrastructure mapping, with several examples cited in the literature [5–8]. Laser scanners can create dense 3D point clouds, which accurately represent the environment from a dimensional point of view.

To enhance the utility of 3D point clouds as digital representations of infrastructure, it is necessary to accelerate the access to the information contained in them. Automatic segmentation became a necessary step when processing these huge datasets. Several researchers have proposed algorithms for automatic point cloud segmentation in the context of

* Corresponding author.
E-mail addresses: daniel.lamas.novoa@uvigo.gal (D. Lamas), andres.justo.dominguez@uvigo.gal (A. Justo), msoilan@uvigo.gal (M. Soilán), belenriveiro@uvigo.gal (B. Riveiro).

https://doi.org/10.1016/j.autcon.2023.105176
Received 5 May 2023; Received in revised form 30 October 2023; Accepted 1 November 2023
Available online 17 November 2023

bridge. Considering the difficulties of these kind of analysis, various works on the use of deep learning techniques to analyse point clouds have been explored. However, having labelled point clouds for training purposes is very costly, much more so than in the case of images. Therefore, research has been carried out on the generation of synthetic point cloud. Specifically, in the field of bridges, there is research on how to generate synthetic point clouds to train deep learning models. Nonetheless, there is a knowledge gap in truss bridges, which are a more complex scenario to segment.

In the context outlined above, the main objectives of this paper are:

1. To outline a novel methodology for creating synthetic point clouds of truss bridges.
2. To demonstrate the practicality of utilizing a deep learning approach for semantic and instance segmentation of point clouds related to truss bridges.
3. To study the effect of training with different types and amounts of synthetic point clouds.

This work is structured as follows: Section 2 presents the state-of-the-art. Section 3 details the methodology, where Section 3.1 explains the implemented methodology for generating synthetic point clouds of truss bridges, Section 3.2 elaborates on the artificial neural network used, and Section 3.3 outlines the process of training and testing the neural network. Section 4 of the paper provides an in-depth analysis of the obtained results from the experimentation. This section is divided into two subsections: Section 4.1 presents the results obtained from the synthetic data, and Section 4.2 elaborates on the segmentation metrics achieved by the deep learning approach. Finally, Section 5 presents the conclusions of the paper and highlights future directions for research.

Repositories available at https://github.com/GeoTechUVigo/synthetic_truss_bridges and https://github.com/GeoTechUVigo/truss_bridge_pc_segmentation_dl.

2. Related work

This section shows the state-of-the-art related to the use, synthetic generation, and segmentation of point clouds in the field of truss bridge.

2.1. Segmentation of 3D point clouds of bridges using heuristic methods

Several authors have explored and continue to explore different techniques for point cloud segmentation using heuristic methods. For instance, Lu et al. [10] devised a slicing approach that can handle complicated topologies, reduce computation costs through data splitting, and cope with occlusions, local variability density, and frequent point cloud events. Similarly, Yan and Hajjar [11] introduced a heuristic-based method to segment steel-girder bridges based on geometric and topological constraints. In a previous study [12], we developed a process to automatically detect the components of masonry bridges by utilizing normal surface calculations. However, these segmentation methods are not applicable to truss bridges, which are more intricate than the bridge types investigated in prior research.

In the domain of truss bridges, Gyetvai et al. [13] proposed a workflow to construct a finite element model for structural evaluation using a truss bridge point cloud. Their study consisted of two main processes. First, the cross-section of each element was identified by comparing the section-based point cloud to a library of sections. Second, the primary dimensions of the bridge were estimated to generate the model. However, the segmentation process for each item was not automated. Shang et al. [14] introduced an approach for 3D reconstruction of truss bridges using images instead of point clouds. They devised a flight plan for unmanned aerial vehicles (UAVs) to capture bridge images while minimizing occlusions. These images were employed to construct the bridge meshes. Nonetheless, their model only provided information on the bridge as a whole and not on each of its elements. In previous work we [15] have developed a heuristic method that utilizes principal component analysis (PCA) and clustering to analyse each point and its neighbouring points in the point cloud. The algorithm can automatically segment the point clouds of truss bridges into its instances, extract sufficient information to create a geometric model of the structure with connected nodes for each element. However, this method requires some manual measurements as input and, due to its intended purpose, might not entirely segment the point cloud.

2.2. Segmentation of 3D point clouds using deep learning

Deep learning neural networks have revolutionized many fields, including image processing and computer vision. In recent years, deep learning models have been developed to address the challenging problem of instance segmentation on 3D point clouds, demonstrating their effectiveness in this field. Commonly used datasets for evaluating such models include S3DIS [16] and ScanNet [17], and web platforms like Papers With Code [18] offer rankings of papers with accompanying code on these datasets. Recently, Vu et al. [19] present SoftGroup, a novel approach for instance segmentation on 3D point clouds that performs grouping on soft semantic scores to address the problem of hard grouping on locally ambiguous objects. Sun et al. [20] introduces SPFormer, a two-stage framework that combines proposal-based and grouping-based methods. SPFormer achieves state-of-the-art results while retaining fast inference speed. Zhong et al. [21] introduces MaskGroup, a framework that uses a Hierarchical Point Grouping algorithm to progressively merge points into multi-scale groups for better instance prediction. Additionally, they also propose MaskScoreNet which effectively eliminates noisy points from the instances. Kolodiazhnyi et al. [22] present Top-Down Beats Bottom-Up (TD3D). TD3D is a fully-convolutional and top-down method that does not rely on prior assumptions about objects. It achieves state-of-the-art results on ScanNet v2, ScanNet200, and S3DIS benchmarks, while being $>1.5\times$ faster than the previous best method. Schult et al. [23] propose Mask3D, which utilizes Transformer decoders [24] to predict semantic instance masks without the need for hand-selected voting schemes or hand-crafted grouping mechanisms.

2.3. Synthetic 3D point cloud generation

Deep Learning applications typically require large amounts of data for training. While there are open access datasets available to train and test artificial networks for point clouds segmentation, datasets for specific topics may not exist. Unlike images, point cloud datasets for infrastructure require much more planning and surveying to be created. This entails the transport of both human resources and material to the location of the survey, the study of the environment to determine the best scanning spots, and the consideration of weather conditions that might interfere with the laser. Therefore, obtaining an appropriate number of point clouds of different characteristics that are fitting for training a neural network is an extremely challenging and consuming task.

Similar to the trend in synthetic images [25–28], the utilization of synthetic point clouds is gaining popularity as a substitute for those acquired from real-world environments due to certain limitations [29]. These point clouds are generated through computer graphics techniques by simulating different objects, environments, and scenarios, making it possible to create large and diverse datasets that can be customized to meet specific requirements. One of the advantages of synthetic data is that it is often easier and more cost-effective to generate than real-world data since it eliminates the need for time-consuming and expensive data collection and annotation. Generating synthetic data is a complex process that requires careful research and planning to establish an effective procedure. While the use of synthetic data in the field of images is well-established [30–32], generating synthetic 3D point clouds for specific fields is still an area of ongoing research. Many authors have developed

various methods for generating synthetic 3D point clouds to address specific needs and applications. Griffiths et al. present SynthCity [33], which is a dataset created using the Blender 3D graphics software [34], with models downloaded from an online database and duplicated with shuffling to ensure diversity. Additional building models were added to populate unoccupied spaces. The dataset contains over 130 buildings, 196 cars, and various other objects. SynthCity also presents an identical point cloud with Gaussian sampled noise for a more realistic appearance. Curnis et al. propose GTASynth [35], a synthetic dataset for outdoor environments generated called using Grand Theft Auto V (GTAV), a video game that simulates sensing accurately. The data production technique is based on DeepGTAV-PreSIL [36], which uses a simulated LiDAR and camera installed on a vehicle driven through the GTAV map to produce data. The goal of this work is to produce a large amount of data to train neural networks, and the characteristics of the sensors are chosen to accurately simulate real ones to be used with real data. Gaidon et al. [37] use computer graphics to generate fully labelled, dynamic, and photo-realistic proxy virtual worlds, and validate their approach by building a new video dataset called "Virtual KITTI" with accurate ground truth for various tasks. They provide experimental evidence that deep learning algorithms pre-trained on virtual data can improve performance, and virtual worlds enable measuring the impact of various weather and imaging conditions on recognition performance. Cabon et al. [38] present an updated version of the Virtual KITTI dataset, which consists of 5 sequence clones from the KITTI tracking benchmark. The dataset provides various modified weather conditions and camera configurations for each sequence, along with multiple sets of images containing RGB, depth, class segmentation, instance segmentation, flow, and scene flow data, as well as camera parameters and poses and vehicle locations. Deschaud et al. published Paris-CARLA-3D [39]. This dataset is a collection of dense coloured point clouds of outdoor environments created by a mobile LiDAR and camera system. It consists of two sets of data, one with synthetic data from the open-source CARLA simulator [40] and the other with real data acquired in Paris. The dataset has been manually annotated with semantic tags, allowing for the testing of transfer methods from synthetic to real data. The objective of the dataset is to provide a challenging benchmark for 3D mapping tasks such as semantic segmentation, instance segmentation, and scene completion.

2.4. Synthetic 3D point clouds generation and segmentation of bridges

In the bridge domain, Jing et al. [9] published a work where they presented a synthetic point cloud generator and a neural network named BridgeNet. The network was trained on their synthetic data to accurately segment point clouds of masonry bridges. They describe a bottom-up methodology to automate the generation of synthetic masonry arch bridges, which aims to approximate the geometric properties of two types of real bridges. The methodology involves assembling entire bridges from two primitive shapes and deriving four subassemblies from their combinations. Different 3D masonry bridges are generated by iteratively producing those components based on bridge topologies. The resulting noiseless synthetic point cloud is then corrupted randomly to simulate geometric distortions and laser scanning errors. The synthetic dataset is used for training BridgeNet, which is tested on a dataset composed of real masonry point clouds from 7 railway bridges in the UK, demonstrating the success of the synthetic point cloud simulator in capturing the global geometric properties of real bridges. Semantic labels are provided by the neural network instead of instance labels. The clustering algorithm DBSCAN [41] is used to obtain instance labels by considering the coordinates and semantic labels of the points in the point cloud. Tian Xia et al. [42] propose a machine learning pipeline to segment semantically RC bridges. They calculated local descriptors throughout the point clouds to use it as input of a new neural network. They solve the problem of lack of data using the dataset published by Lu et al. [43]. Jun S. Lee et al. [44] presents a similar work. Instead of using an artificial neural network, they propose a graph-based hierarchical convolutional neural network DGNN (HGCNN). The types of layers of this architecture, unlike those of the previous authors, are convolutional. Besides, this architecture allows to use the neighbouring relations between points in a more efficient way than in DGNN. Despite the good results of these tow works, their method presents the following two limitations. First, their neural network is not able to segment the elements in instances, which is crucial to segment truss bridges. Second, the datasets that they use do not have truss bridges, so their trained network would not be able to segment truss bridges. Indeed, in the Tian Xia et al. they said that "since three bridges in the dataset each belongs to a different bridge type and thus cannot be used for training and testing at the same time, they are excluded of the experiment". This indicates that their model requires to be trained only with the type of bridge it would be tested on. Xiaofei Yang et al. [45] presents another deep learning strategy to segment RC bridges. To solve the problem of lack data, they apply two different data augmentation strategies for superpoint-based point cloud segmentation. The augmented data is used for training a superpoint-based algorithm from a previous work of the authors called WSPG [46]. In those previous work, they use two-real RC point clouds plus synthetic data generated from RC bridge models for training their deep learning algorithm. Their algorithm was designed to enable automated semantic segmentation of bridge components directly from extensive bridge point clouds. This last work was one of those that inspired the idea of automatically modelling truss bridges to create synthetic point clouds, which is presented as part of the contributions in this article.

3. Methodology

This section presents the methodology developed to perform the semantic and instance segmentation of truss bridge point clouds using deep learning. It covers everything from synthetic data generation (Section 3.1) to the artificial neural network architecture description (Section 3.2), and the training and testing processes (Section 3.3).

3.1. Synthetic truss bridges

A truss bridge is a bridge whose superstructure is composed of a truss. A truss is a rigid structure of members connected by nodes. There are different types of truss bridges depending on the assembly of their composing members, as shown by the examples of Fig. 1.

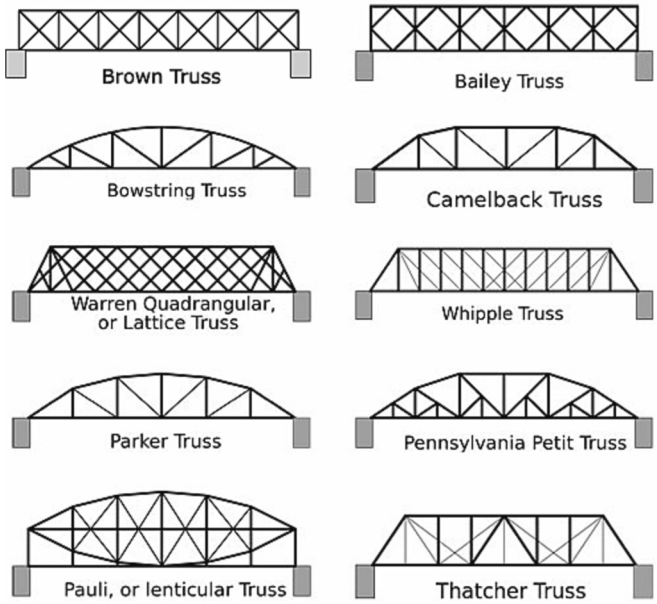


Fig. 1. Example of different types of truss bridges.

Due to the lack of real data of point cloud truss bridges, synthetic data is generated to train and test the methodology. However, this generation method is neither limited to obtain data for training algorithms, nor limited to use for generating point clouds. This tool can automatically generate a virtual structure containing its information in an accessible manner, such as the truss node positions, member profile or members orientation, that can be modified to redefine the bridge. Furthermore, in addition to saving a labelled point cloud, it also generates a mesh of the structure. These other functionalities are not used in the segmentation processes but could be useful for other applications and future work.

For a given typology, the whole truss can be defined by setting a certain number of parameters. It can be characterized by: i) the position of the nodes, ii) the nodes to which each member is connected, iii) the orientation of each member and, iv) the member profiles.

Considering this, we propose a methodology according to the workflow depicted in Fig. 2a to generate the synthetic data. In this workflow, the first process depends on the truss typology. The system is being designed so that any typology can be designed. However, as a starting point, the truss typologies studied in this work are Bailey and Brown. These typologies were selected due to their simplicity and similarity between them, which facilitated the implementation of this first version presented in this work. Another reason why these typologies were chosen is that the real data available for this work are truss bridges of one of these or similar typologies. These typologies are formed by panels, which are patterns of the structure that are repeated along the length of the bridge. Knowing the truss typology, the number of panels and their dimensions, the location of the nodes and the pair of nodes connected by each member are defined automatically. A schematic representation of these relations is shown in Fig. 3. In addition, the methodology was designed to be versatile and adaptable. Even inside a given typology, it is possible to alter the defining parameters of the truss, such as member profiles, or remove/change members entirely. For instance, it is possible to generate a Bailey Truss bridge without vertical member, or a Brown Truss with Saint Andrew's crosses.

The node coordinates, and their relationship with the members, depend on the typology of the truss. However, once these properties have been calculated, the mesh and point cloud generation are not typology dependent. To add a level of abstraction and generalization, these processes have been designed to be separated from the truss type. As such, the following steps are performed by the generalized class TrussBridge, regardless of the typology. The workflow of this process is shown in Fig. 2b.

The first step in the construction of a truss, guided by the *TrussBridge*

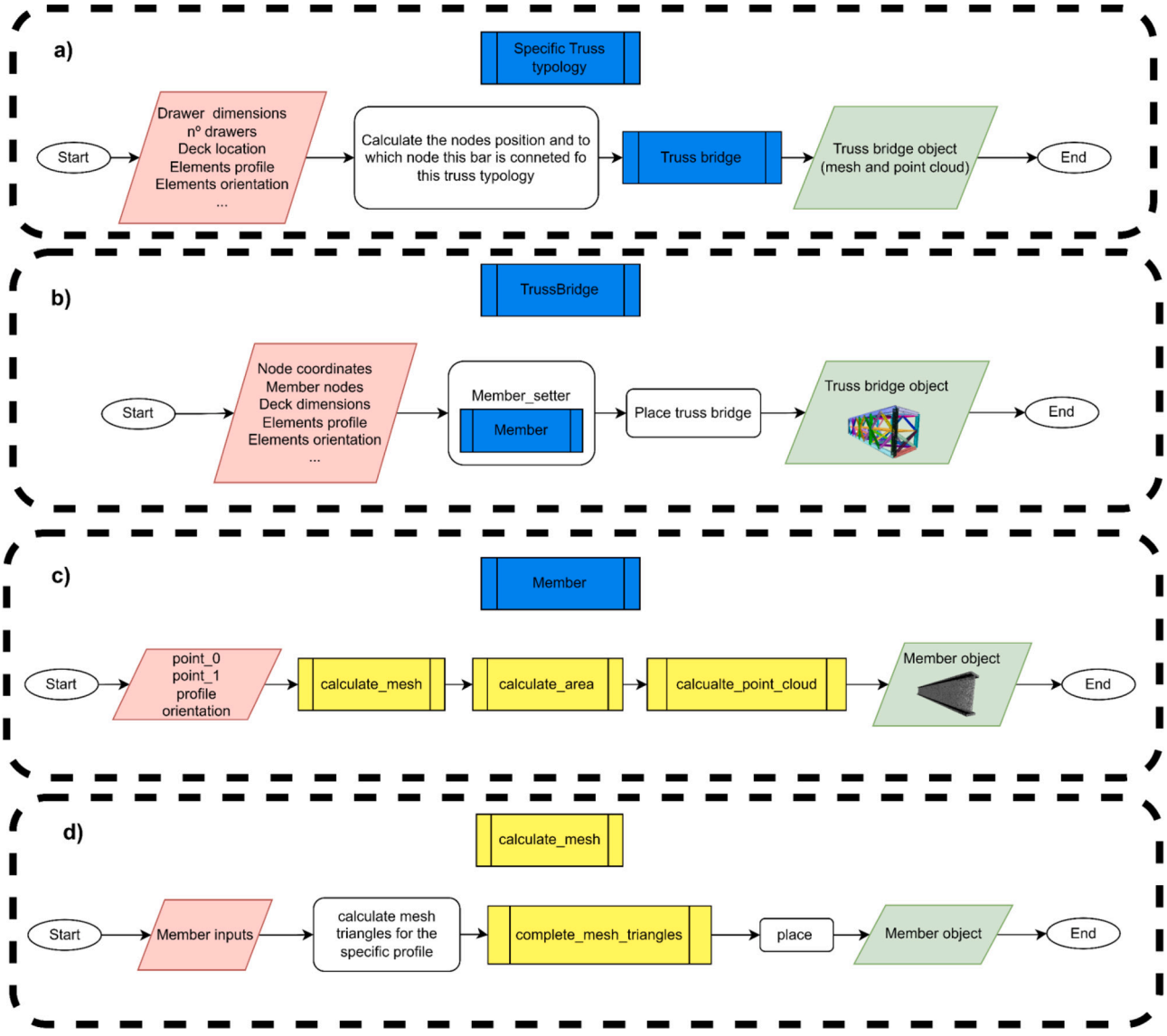


Fig. 2. Synthetic truss bridge model generation.

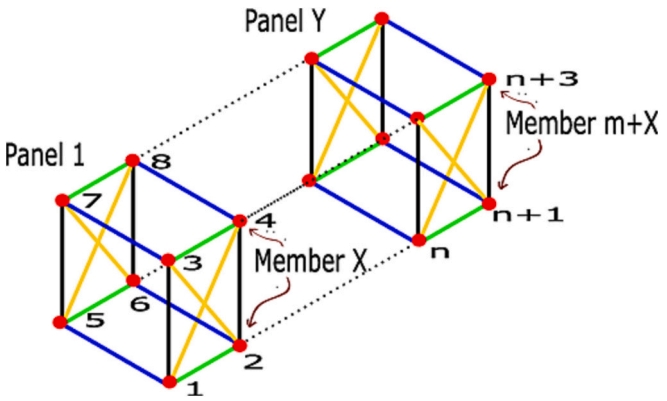


Fig. 3. Schematic relation between nodes and members of a truss.

class, is the generation of members through the class *Member*. The inputs required in this procedure are the coordinates of their corresponding nodes, their profile, and their orientation. Each member is created independently from one another, following the workflow of Fig. 2c.

As it can be seen, the first step is the computation of the member mesh. The workflow of the mesh generation is described in Figure 2dand a schematic example is shown in Fig. 4.

The mesh structure can be seen as an extrusion of the specified profile in the Z axis, where the extrusion length is determined by the Euclidean distance between the nodes to which the members is connected, called p_0 and p_1. First, the mesh triangles of the bottom face are defined, as shown in Fig. 4a. These triangles depend on the type of profile of the member being created. For instance, a square profile only needs two triangles, while other, more complex profiles, require more. However, the vertices of all implemented profile types are defined in a clockwise manner so that the following steps can be performed in an automated manner. The next step is to define the top face. Its vertices are obtained by increasing setting the Z coordinate of the bottom face vertices as the extrusion length. Meanwhile, its triangles are calculated by assessing the number of nodes of the bottom face triangles and inversing their normal (counter-clockwise), as shown in Fig. 4b. Finally, the two faces are connected through the lateral triangles. These triangles are profile independent as they are based on the node indexes of the top and bottom faces, creating a pair of triangles per edge of the profile used, as shown in Fig. 4c.

As a note, the currently supported profile typologies are I, T, L, U, and solid rectangular profiles. They are selected by inputting a label that specifies the type of standardized profile that the member uses (e.g. IPN 220), which is used to access a list of .csv files that contains the dimensions required to generate the vertices and triangles of the bottom face of each type. At the moment, the standardized profiles accessible for automated generation are CH, HD, HL, HLZ, HP, HSS, IPN, L, PFC, T, U, UB, UBP, UC, UPE, and UPN, which are the most commonly used profiles. These profiles are defined in a local XY axis system where the opening of the profile, if applicable, points to the positive X axis. The reasoning behind this is to have a consistent rotation matrix for all profiles that is easy to use.

Once all the triangles have been assembled into the mesh that represents the member, the next step is to place it in the target location, as shown in Fig. 4d. This positioning is determined by the input nodes p_0 and p_1 and the input *orientation* that describes the axial rotation of the member. To do this, the mesh is centred at (0,0,0). Then, a rotation matrix, composed of translation and rotation, is applied. The translation is directly marked by the centre of $\overline{p_0.p_1}$. The rotation, however, is more complex. All members need to be oriented in a manner that is easily mutable according to the *orientation* parameter and that is consistent for ease of use. To achieve this, the rotation matrix performs a ZYZ rotation that places the Z' axis parallel to the main axis (**MA**) of the member, which is equal to $\overline{p_0.p_1}$. This procedure can be seen in Fig. 5. First, the system is rotated $\widehat{Z}$ in the +Z axis, which makes X' parallel to the projection of the main axis in the XY plane ($\mathbf{MA_h}$), as shown in Fig. 5a and Fig. 5b. Then, the system rotates $\widehat{Y}$ in the +Y′ axis, resulting in the Z' axis parallel to the main axis (**MA**), as shown in Fig. 5c. By performing the rotation in this manner, the Y′ axis is now contained in the XY plane, and the +X" axis points downwards. This means that, in the case of a U profile, the opening of the U points downwards after these two rotations (X"). Then, the last rotation *orientation* along its main axis (**MA**) is what allows the tailoring of the rotation of the member in an intuitive and easy manner, as shown in Fig. 5d. For instance, through this setup, a 90° orientation would mean that the opening of the U (now **X"'**) is contained in the XY plane, with a 180° meaning that it points upwards, and so on.

This procedure makes it possible to define the orientation of the members of each face to be decide independently, with one exception: the two vertical faces have a mirror configuration. In this way, defining the configuration of one vertical face, the other is determined automatically.

The generation of the deck follows the same procedures as the members but with the following exceptions. The deck is also created as an extruded member, but its profile is always a rectangle. The length of the rectangle is the width of the bridge, and its height is the input that defines the thickness of the deck. For placement, there are defined two virtual nodes in order to use the member methods. These nodes are two points at the ends of the bridge, located at a height determined by the input that defines the height at which the deck is placed.

At this point, the mesh has been created, placed and oriented, effectively modelling the geometry and location of the member.

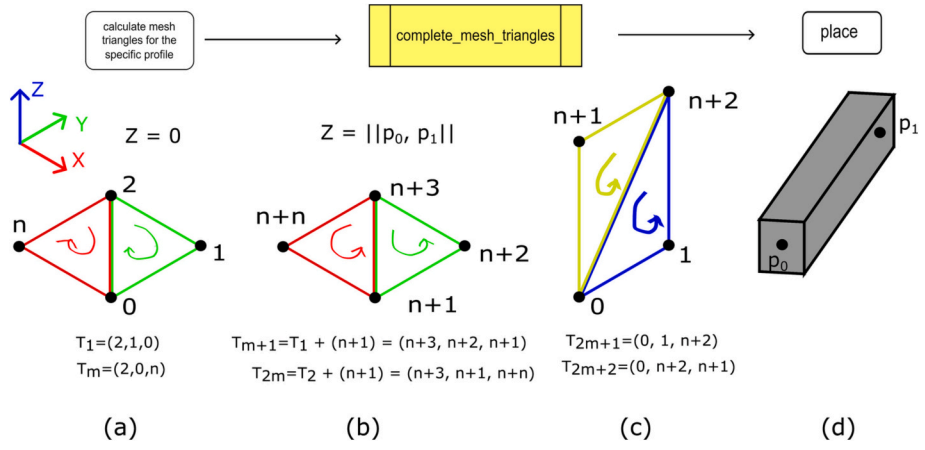


Fig. 4. Calculate_mesh schematic example.

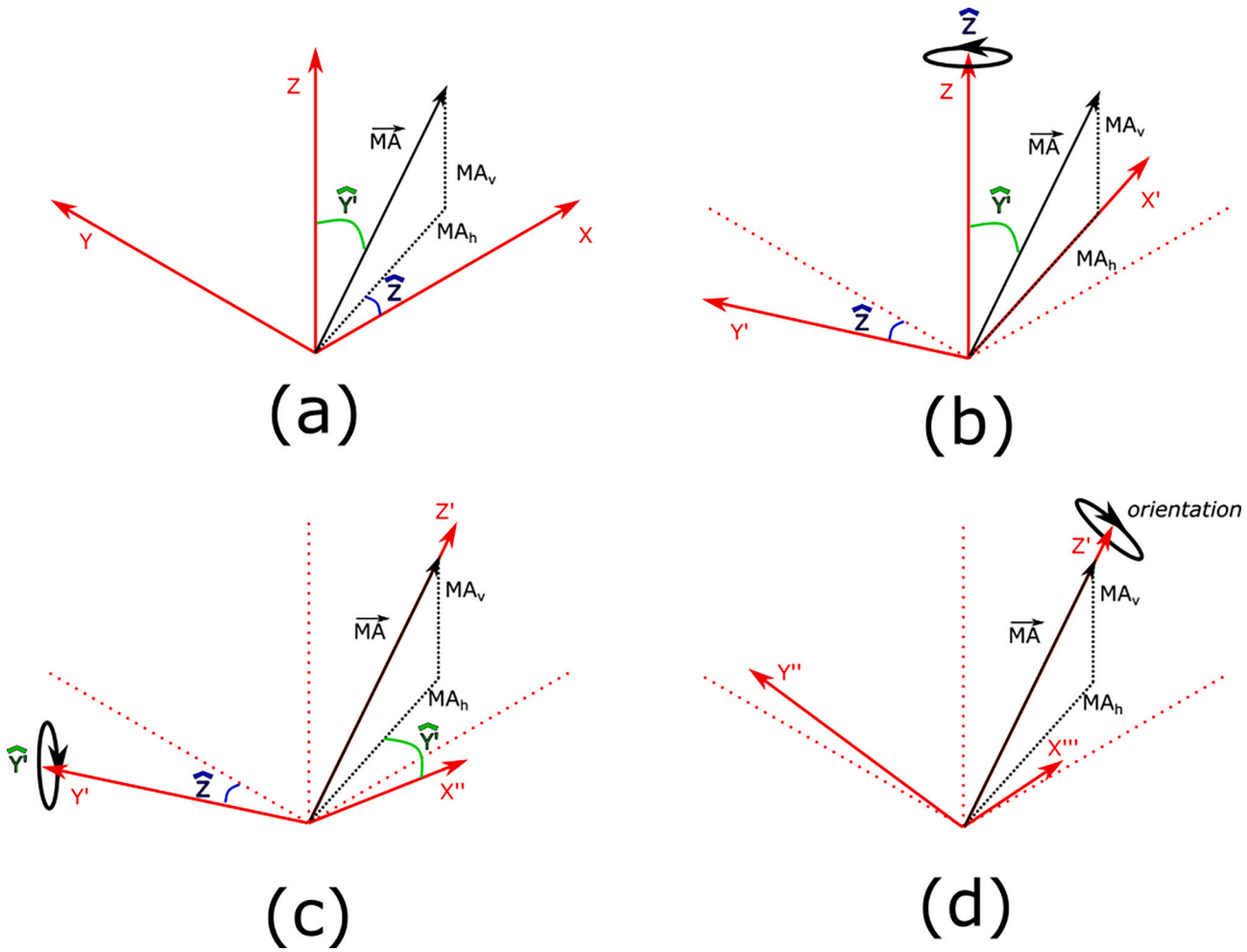


Fig. 5. Rotation to target axis. MA is the main axis of the member, which is $\overline{p_0.p_1}$.

Therefore, the next step is to generate the synthetic point cloud. Two different methods are used:

- Uniform point clouds. This method uses an Open3D [47] function that uniformly samples points that lie on the surfaces of the mesh. The input parameters is the final number of points. To calculate that number, the area of the mesh is computed as the sum of the area of each triangle calculated with Heron's formula. The number of points is determined using the input *density* and the calculated area.
- Point clouds with occlusions. This method simulates the operation of a terrestrial laser scanner (TLS) to create occlusions in synthetic point clouds. The procedure involves specifying the positions of the virtual TLS in the target environment, along with defining the angular pitch to emulate a scanning. These parameters are defined in the input *cameras,* which is a list with the parameters of each virtual TLS acquisition. This process allows us to generate a synthetic point cloud that accurately replicates occlusions observed in real-world scans. For its implementation we utilized the Open3D ray casting function. We launch rays from each virtual scanner position and compute the impact points where the rays intersect with the surrounding mesh surfaces. The ray casting function makes it possible to determine the impact points of rays launched on a camera on a mesh. The cameras are the virtual TLS positions and the point of impact of the virtual rays are the points of the synthetic point cloud.

Once this procedure is repeated for each member, the truss can now be assembled and positioned at the target location. So far, the creation of the members has used the origin as reference. This means that if no translation or rotation is applied, the bridge would be centred at (0, 0, 0) with its main orientation matching the positive X axis and no vertical or lateral tilt. Therefore, the last step of the point cloud generation is to place the assembled bridge in the position and orientation specified in the *centre* and *orientation* inputs passed to the *TrussBridge* class.

Lastly, the point cloud is saved in LAS or LAZ format [48]. The semantic information about the type of member is saved in the *classification* field. The number assigned for each type of member is an input of *SpecifyTrussBridge* class together with its profile and orientation. The instance information is saved in the *user_data* field with a unique index for each member object.

Table 1 shows the inputs required to generate a synthetic truss bridge point cloud.

3.2. Model architecture

The use of deep learning algorithms on point clouds has grown over the last decade. There are works exploring various fields, from new architectures to testing their applicability in specific areas [49]. One of the objectives of this paper is in line with the latter: to investigate the applicability of this type of architectures in the field of linear infrastructure, concretely in truss bridges. The purpose is to obtain a semantic and instance segmentation, since the objective is to recognise not only each type of member, but also each member independently. For this reason, the architecture used is an adaptation of JSNet [50] developed by Lin Zhao et al. JSNet is a state-of-the-art instance segmentation

Table 1
Synthetic truss bridge inputs.

General		Members	
typology	typology name	*chord*	profile name, orientation
n° of panels	N	*vertical*	profile name, orientation
		diagonal	
centre	(x, y, z)	*vertical parallel*	profile name, orientation
orientation	$(\hat{z}, \hat{y}, \hat{x})$	*bottom diagonal*	profile name, orientation
density	R^+	*bottom parallel*	profile name, orientation
cameras	(x, y, z), angular pitch	*inner diagonal*	profile name, orientation
deck position	R^+		
Panel dimensions			
height	R^+		
length	R^+		
width	R^+		

network that ranked as the best of the architectures tested in the dataset S3DIS [16] in 2019, and currently [14] remains in the top 10 at the time of writing [51].

The model proposed by Lin Zhao et al. consists of a pre-processing, an instance and semantic segmentation by JSNet, and a post-processing. The pre-preprocess consists of cropping the point cloud in parallelepipeds of equal size and randomly selecting N_a number of points. The semantic segmentation JSNet output is of type $N_a x\, C$, being C the number of semantic classes. The instance segmentation JSNet output is of the type $N_a x\, K$, being K the dimension of the embedding vector that represents the instance relationship of points. To obtain the final instance labels of each parallelepiped, Lin Zhao et al. used the algorithm mean-shift clustering. This method requires a unique parameter, called *bandwidth*. Finally, the post- processing consists of merging those parallelepipeds and remapping the instance labels using the algorithm *GroupMerging* proposed by Want et al. [52].

The pre-processing used in JSNet does not include a sub-sampling performed by a voxelization. They only randomly select N_a points from each parallelepiped. This can be a problem in point clouds with not uniform point distribution. Subsampling by voxelization is a widely used method. We have incorporated it before the cropping process.

As our data contains objects that require a larger number of points to identify them accurately, modifications are made to the architecture of JSNet, which is originally designed to segment objects of S3DIS. Specifically, the encoder and decoder blocks of the model are each given an additional layer to accommodate for the increased amount of data. Additionally, the depth of these layers is adjusted to ensure optimal performance of the modified architecture. These modifications are necessary to effectively process the larger number of points required for accurate object segmentation in our data, resulting in a more robust and accurate model. Fig. 6 shows the architecture of the encoder and decoder block.

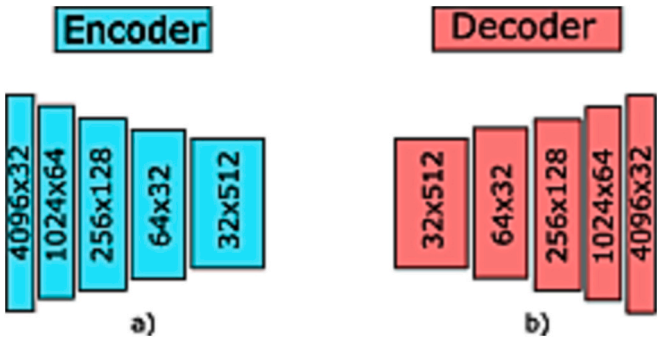


Fig. 6. a) Encoder and b) Decoder block of the adapted version based on JSNet.

3.3. Training and testing

On the one hand, we want to test the effect of the variety of training data. The primary approach involves training with a vast amount of real data, while there exist no limitations on the quantity of synthetic data. Consequently, it becomes interesting to explore the impact of varying data volumes on the training process.

On the other hand, we introduce two distinct methods for generating synthetic point clouds: one involving occlusions and the other without occlusions. This aspect raises the question of how the model's performance and response may vary under these different conditions.

To test the effect of synthetic point clouds on models, the data is organised as follows. The synthetic models are defined and two point clouds are created for each of them and saved in separately: one with occlusions and other without. Let's design the dataset without occlusions as $\boldsymbol{U}$ and the dataset with occlusions as $\boldsymbol{O}$. Then, the data is split in k folders with equal number of point clouds. The distribution is the same for both. Let's design each folder as $\boldsymbol{U_i}$ and $\boldsymbol{O_i}$, respectively, being $\boldsymbol{i}$ the folder number. In this way, folders $\boldsymbol{U_i}$ and $\boldsymbol{O_i}$ contain the same bridge models.

With this data distribution, models are trained and tested as follows. A model is trained with the dataset in the folder $\boldsymbol{U_i}$ to $\boldsymbol{U_{i+(n-1)}}$, being $\boldsymbol{n}$ the number of folders used for training, and tested in folders $\boldsymbol{U_k}$, $\boldsymbol{O_k}$ and in real point clouds called $\boldsymbol{R}$. This real point clouds are the two point clouds analysed in our previous work with the heuristic method [15], with which this new method is compared. In this way, for the same $\boldsymbol{n}$ and $\boldsymbol{k}$ values, there are two models trained and tested in the same truss bridge models, which their only difference is that the point clouds used in one have occlusions and in the other does not. Moreover, by changing $\boldsymbol{n}$ we can see the effect of training with different volume of data. Besides, the k-fold cross validation method is applied by changing the $\boldsymbol{k}$ value for each dataset $\boldsymbol{O}$ or $\boldsymbol{U}$ and $\boldsymbol{n}$ value. The scheme of this process is shown in Fig. 7.

For each model, the training data is further split into training and validation data. Epoch by epoch, all the training data is used to train the model, and all the validation data is used to calculate the different metrics that reveals the training process. The metrics obtained from the validation data are utilized to eventually apply an early stop to the training if there is no progress. The model chosen to perform the test is the one with the best validation results.

The test process is performed by applying the entire architecture: pre-processing, segmentation and post-process. However, the training processing does not use the pre- and post- processing methods. For each point cloud of training data, only one parallelepiped is selected. The parallelepipeds are randomly taken and they are different for each point cloud and for each training epoch. Besides, data augmentation technics are applied to the parallelepipeds. The cropped point cloud is randomly rotated and gaussian noise is added. The same processes are applied to the validation data.

The difference between the testing and the training data is because using the full point clouds to train and apply the pre- and post-processing is very time-consuming and does not significantly contribute to the training. After all, each point cloud contains only one type of truss bridge. Therefore, each parallelepiped of the same point cloud is made up of members with the same distribution that do not introduce new information into the network. However, the testing process must be performed by applying the pre- and post-processing methods as the whole process needs to be evaluated.

4. Results

The purpose of this section is to present the results obtained from the methodology described through Section 3. More specifically, two main outcomes can be analysed: (i) the generation of synthetic data of truss

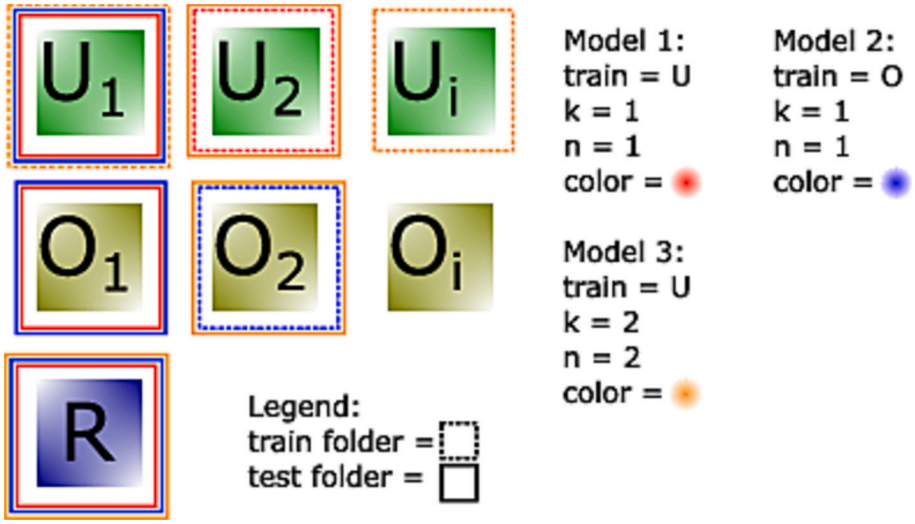


Fig. 7. Train and test split.

bridges; (ii) and the point cloud segmentation process. Therefore, it has been split into Section 4.1 for the synthetic data, and Section 4.2 for the segmentation.

4.1. Synthetic data

500 synthetic point clouds of truss bridges are created. These 500 point clouds correspond to 250 different models randomly defined, which 125 have a Bailey truss configuration and 125 a Brown Truss. From each of these models, two point clouds are generated: one with uniformly distributed points and the other simulating occlusions, making a total of 500 point clouds. The parameters that define the synthetic point clouds are shown in Table 2, Table 3, Table 4 and Table 5.

Table 2 shows the range of values used to randomly define the input parameters of each point cloud. For each set of parameters, one point cloud is created using the input *density* and the other with the input *cameras*, which correspond to a uniform point cloud and a point cloud with occlusions, respectively. Furthermore, there are members that are not always created to introduce more variability in the data. The presence or not of the members is determined by the column "Probability of having this member".

Different standardized profiles are available for the definition of the truss members. However, not all possible combinations among them were used. Table 3 shows the considered profiles for each type of truss member. Also, for a better view of how an isolated member is represented both as a mesh and point cloud, Fig. 8 presents some examples of profile morphologies.

The generation of uniform point clouds only requires the parameters *density*. However, to simulate occlusions, we utilize parameters called cameras, which define the positions of the virtual TLS. In a real case study, the TLS placements are determined primarily by accessible locations and secondarily by positions that offer valuable structural information. Considering this, the virtual TLS positions are strategically placed in four areas: on the deck when it is not on top of the bridge, under the bridge, and at both sides of the bridge. Considering these factors, a range of values is defined to randomly select the input parameters for the TLS positions. Table 4 presents these parameters, including the minimum number of scanners or the distances between the bridge and the virtual positions, among others.

The deck position can be positioned at different heights to further diversify the topologies available, as it is shown in Table 5.

Each point cloud is saved with semantic and instance information. The points of a member are saved with the unique index of their instance. Also, there are 4 types of semantic classes: deck, chord, parallel and diagonal.

Table 2
Range of truss bridge parameters corresponding to the inputs required as it is shown in Table 1.

General			Members		
Name	Value	Unit	Name	Value	Probability of having this member
typology	Bailey or Truss	m	*chord*	Table 3, Table 1(0°, 90°,180° or 270°)	100%
n° of panels	4–10		*vertical diagonal*	Table 3, Table 1 (0°, 90°,180° or 270°)	70%
centre	0–100, 0–100, 0–100	m	*vertical parallel*	Table 3, Table 1 (0°, 90°,180° or 270°)	100%
orientation	±180, ±20, ±5	°	*horizontal diagonal*	Table 3, Table 1 (0°, 90°,180° or 270°)	70%
density	1000	points/m^2	*horizontal parallel*	Table 3, Table 1 (0°, 90°,180° or 270°)	100%
cameras	Table 4		*inner diagonal*	Table 3, Table 1 (0°, 90°,180° or 270°)	70%
deck position	Table 5				
Panel dimensions					
Name	*Value*	*Unit*			
height	3–5	m			
length	3–5	m			
width	2–5	m			

Fig. 9 shows some examples of this synthetic data, in which there are Bailey truss and Brown truss configurations, with several deck positions, several profiles, absence of certain members and the effect of the occlusions. Besides, colours show the instance and semantic information attach to each point.

For a deeper understanding of the disparities between the point clouds containing occlusion and the ones without, examine Fig. 10. It is shown how some elements have fewer points in some areas or no points

Table 3
Profile considered for each type of truss member.

Type of truss member	Profile	Principal dimension [mm]
Chord	IPN, UB, UBP, UC, U, UPE, UPN, CH, HD, HL, HLZ, HP	≥ 300
Vertical diagonal	IPN, L, HD, HL, HLZ, HP	≤ 200
Vertical parallel	IPN, HD, HL, HLZ, HP	≤ 300
Horizontal diagonal	IPN, L, HD, HL, HLZ, HP	≤ 200
Horizontal parallel	IPN, L	≤ 300
Inner diagonal	L	≤ 200

Table 4
TLS virtual positions. XYZ are the main axes of the bridge, being X the direction of travel, Y the width and Z the axis perpendicular to the deck plane.

TLS on the deck		TLS under the bridge	
n° of scanners	2–3	n° of scanners	2–4
Distance to the deck in Z	0.5–1.5 [m]	Distance in Z to the bridge bottom face	3–10 [m]
TLS on the right size of the bridge		TLS on the left size of the bridge	
n° of scanners	2–4 [m]	n° of scanners	2–4
Distance to the bridge right face in Y	3–10 [m]	Distance to the bridge left face in Y	3–10 [m]
Distance to the center of the bridge in Z	(+2) - (−10)	Distance to the center of the bridge in Z	(+2) - (−10)
General parameters			
minimum n° of scanners	6		
angle pitch	5		
standard deviation of normal noise (x,y,z)	1.0, 0.2, 0.0		
X position	Equidistributed along the bridge		

Table 5
Deck position.

Probability of being at that Z position	Position
70%	0.5-*height/2*
15%	0
15%	*height*

at all.

4.2. Segmentation

Regarding the training and testing dataset configurations explained in Section 3.3, the specific setups are as follows. The datasets ***U*** and ***O*** are split in 5 folders of 50 point clouds each. Switching the folders designated for testing enables the implementation of a k-fold validation with k = 5. The number of folders used for training, designed as ***n***, vary from 1 to 4. In total, there are height different training configurations, and each of them is tested using a cross-validation of k = 5.

The real point cloud dataset used are the two point clouds analysed in our previous work with the heuristic method [15]. Only the points segmented with the heuristic method have been used as ground truth, applying a mask to the rest of the points.

For testing and validation purposes, the following metrics are computed. For semantic segmentation evaluation, overall accuracy (oAcc), mean accuracy (mAcc), and mean IoU (mIoU) are calculated considering all the categories together, as shown in Eqs. (1), (2) and (3), respectively.

$$oAcc = \frac{\sum_{i}^{C} TP_i}{\sum_{i}^{C}(TP_i + FN_i)} \tag{1}$$

$$mAcc = \frac{1}{|C|}\sum_{i}^{|C|}\frac{TP_i}{TP_i + FN_i} \tag{2}$$

$$mIoU = \frac{1}{|C|}\sum_{i}^{|C|}\frac{TP_i}{TP_i + FN_i + FP_i} \tag{3}$$

C: semantic classes.
TP_i: true positives of class i at point level.
FN_i: false negatives of class i at point level.
FP_i: false positives of class i at point level.

For instance segmentation, mean precision (mPrec) and mean recall (mRec) with IoU threshold (T) 0.5, and coverage (cov) and weighted coverage (wCov) are calculated as shown in Eqs. (4), (5), (6) and (7), respectively. The metric used to evaluate the early stop is cov since it considers all members equally, irrespective of their size. Otherwise, the deck would have much more weight than the others when in fact it is the least interesting member.

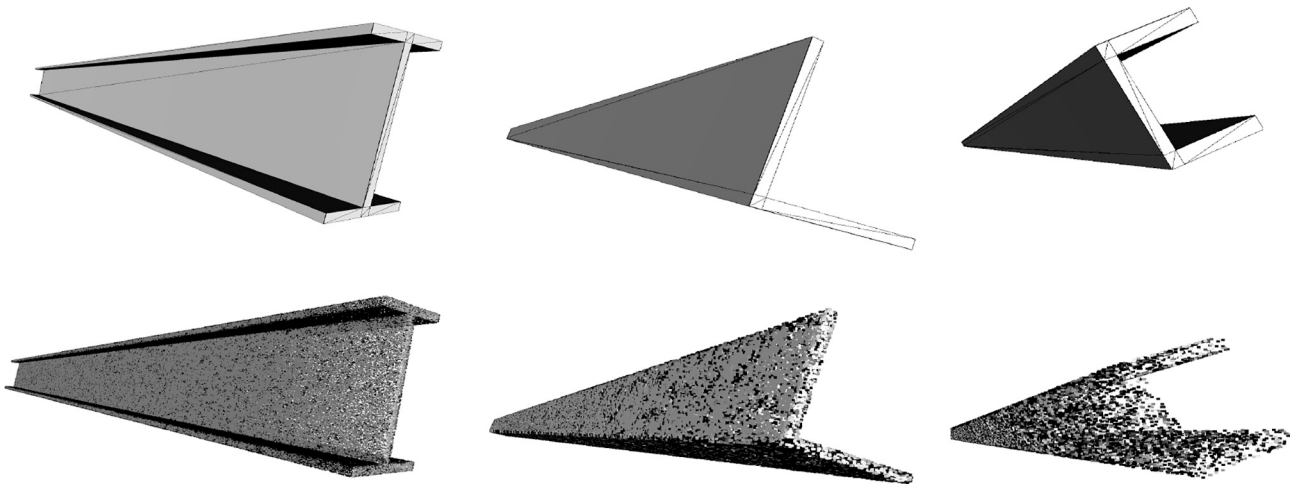


Fig. 8. Synthetic mesh and point cloud profiles I, L and U.

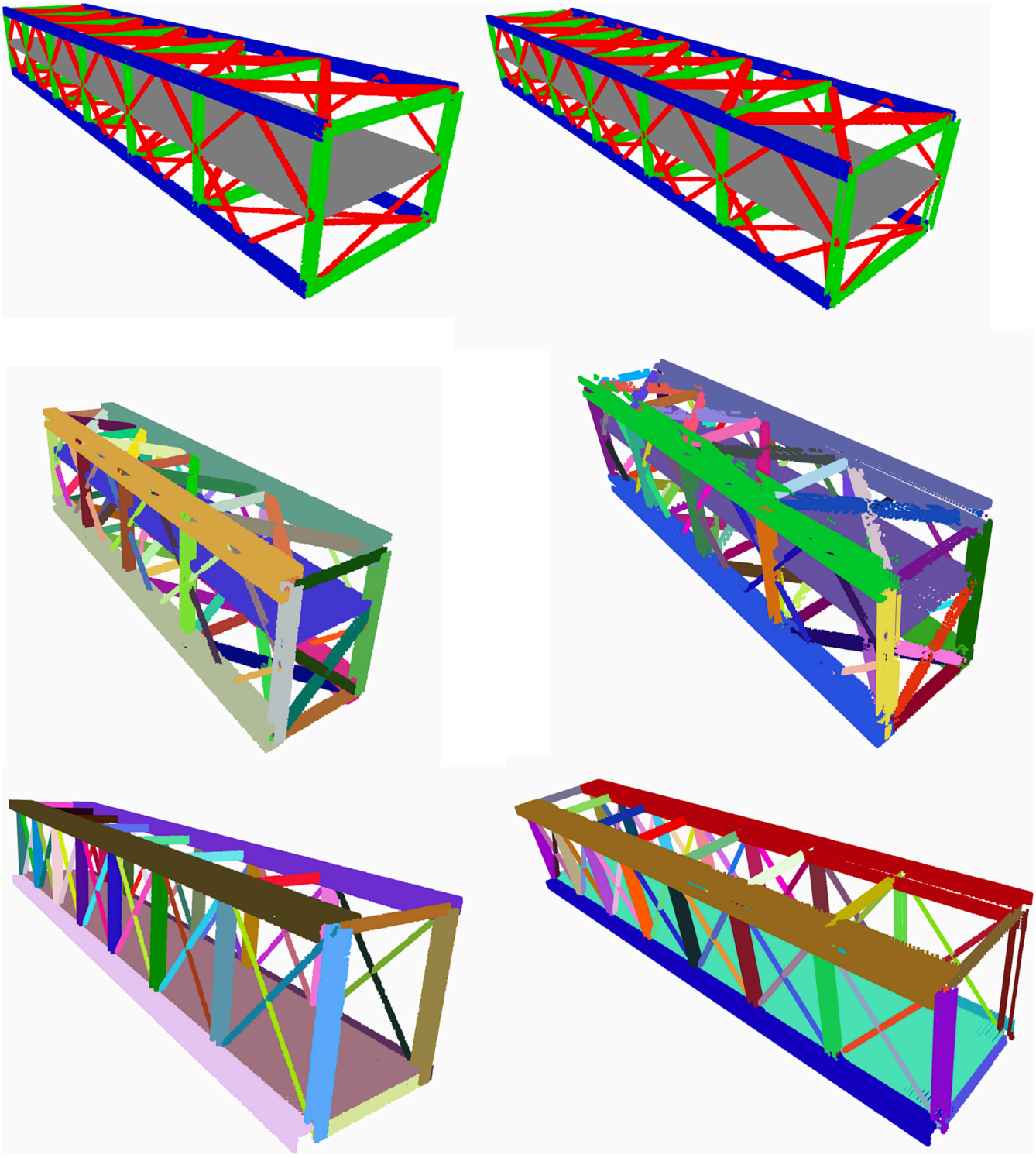


Fig. 9. Synthetic data. Point clouds uniformly sampled are shown in column one. Column two are the same bridge models but with the occlusion simulation method.

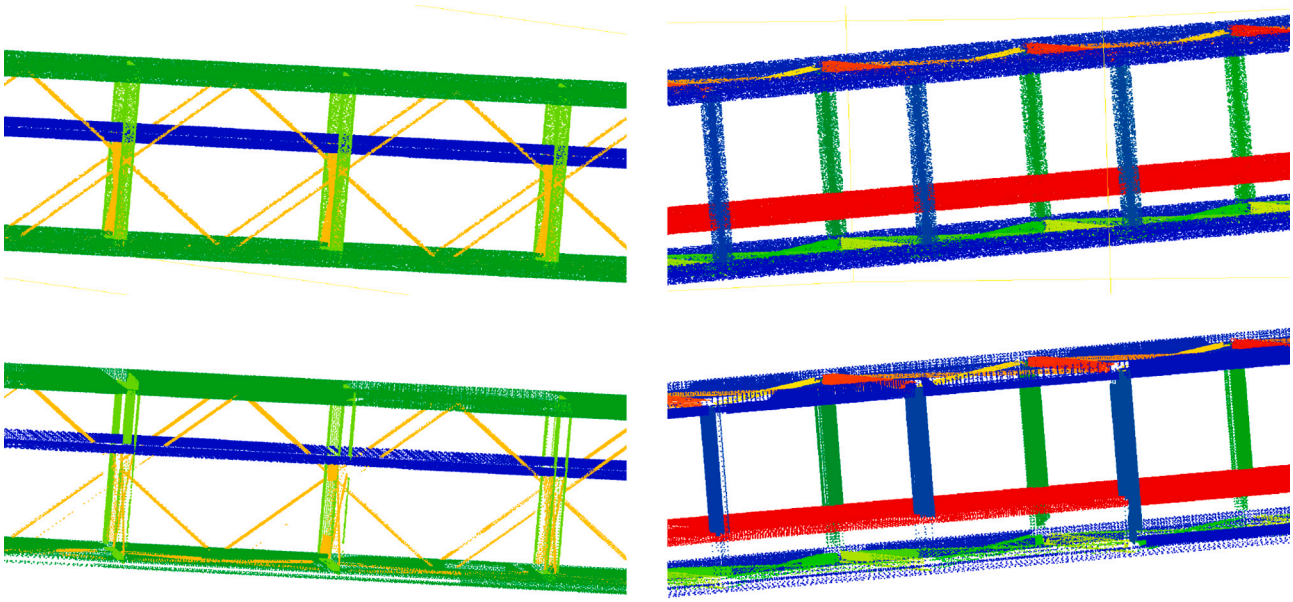


Fig. 10. Details of synthetic point clouds with occlusions in the top row, and without occlusions in the bottom row.

$$mPrec = \frac{1}{|M|}\sum_{j}^{|N|}\left(\left(IoU_j \geq T\right)\rightarrow 1\right) \wedge \left(\left(IoU_j < T\right)\rightarrow 0\right) \quad (4)$$

$$mRec = \frac{1}{|N|}\sum_{j}^{|N|}\left(\left(IoU_j \geq T\right)\rightarrow 1\right) \wedge \left(\left(IoU_j < T\right)\rightarrow 0\right) \quad (5)$$

$$cov = \frac{1}{|N|}\sum_{j}^{|N|} IoU_j \quad (6)$$

$$wCov = \sum_{j}^{|N|} \frac{|N_j|}{\sum_{k}^{|N|}|N_k|} * IoU_j \quad (7)$$

N: ground truth instances.

M: predicted instances.

$$IoU_j = max_n IoU\left(N_j, M_n\right)$$

IoU_j: maximum intersection over union between the ground truth instance j with the predicted instances.

T: IoU threshold between a ground truth instance and its predicted instance to be consider a true positive.

p_j: number of points of the j real instance.

p_t: number of total points.

The implementation details of the training and test of the network are as follows. The point clouds are down-sampled by voxelization with a grid equal to 0.05 m, cropped into overlapping parallelepipeds of 10x10x10 m, and each parallelepiped contains 16,384 points. The

Table 6
Mean number of epochs and training time of each training strategy.

Training dataset (TD)	*n° of folders used for training (n)*	epochs	time [h]
Occlusions (O)	1	890 ± 297	9 ± 2
Occlusions (O)	2	672 ± 242	13 ± 4
Occlusions (O)	3	689 ± 227	21 ± 6
Occlusions (O)	4	630 ± 66	25 ± 2
Uniform (U)	1	873 ± 398	13 ± 2
Uniform (U)	2	670 ± 398	11 ± 2
Uniform (U)	3	670 ± 164	16 ± 3
Uniform (U)	4	774 ± 125	24 ± 3

Table 7
Test result in real point clouds in %.

TD	*n*	Semantic metrics			Instance metrics			
		oAcc	mAcc	mIoU	mPrec	mRec	cov	wCov
O	1	84	85	54	30	12	21	39
O	2	85	87	55	30	13	25	41
O	3	86	87	55	**36**	**18**	**28**	43
O	4	83	86	53	35	18	27	**43**
U	1	86	87	55	24	8	17	35
U	2	84	86	54	28	12	22	40
U	3	**87**	**88**	**56**	31	14	24	41
U	4	84	86	53	30	13	22	38

Table 8
Test result in synthetic dataset with occlusions in %.

TD	*n*	Semantic metrics			Instance metrics			
		oAcc	mAcc	mIoU	mPrec	mRec	cov	wCov
O	1	96	95	90	83	75	69	85
O	2	97	96	92	86	83	76	88
O	3	97	96	93	88	89	79	89
O	4	**98**	**97**	**94**	**89**	**91**	**81**	**90**
U	1	93	91	84	70	58	56	78
U	2	95	94	89	80	79	69	84
U	3	96	94	90	81	85	73	86
U	4	96	95	91	82	87	75	87

Table 9
Test result in synthetic dataset without occlusions (uniform) in %.

TD	*n*	Semantic metrics			Instance metrics			
		oAcc	mAcc	mIoU	mPrec	mRec	cov	wCov
O	1	92	91	84	76	68	62	79
O	2	93	92	86	79	77	68	82
O	3	95	93	88	81	83	72	84
O	4	95	94	88	82	86	74	85
U	1	94	93	87	75	63	60	81
U	2	**96**	**95**	**91**	86	82	74	**87**
U	3	95	94	89	86	86	77	86
U	4	95	94	90	**87**	**88**	**78**	87

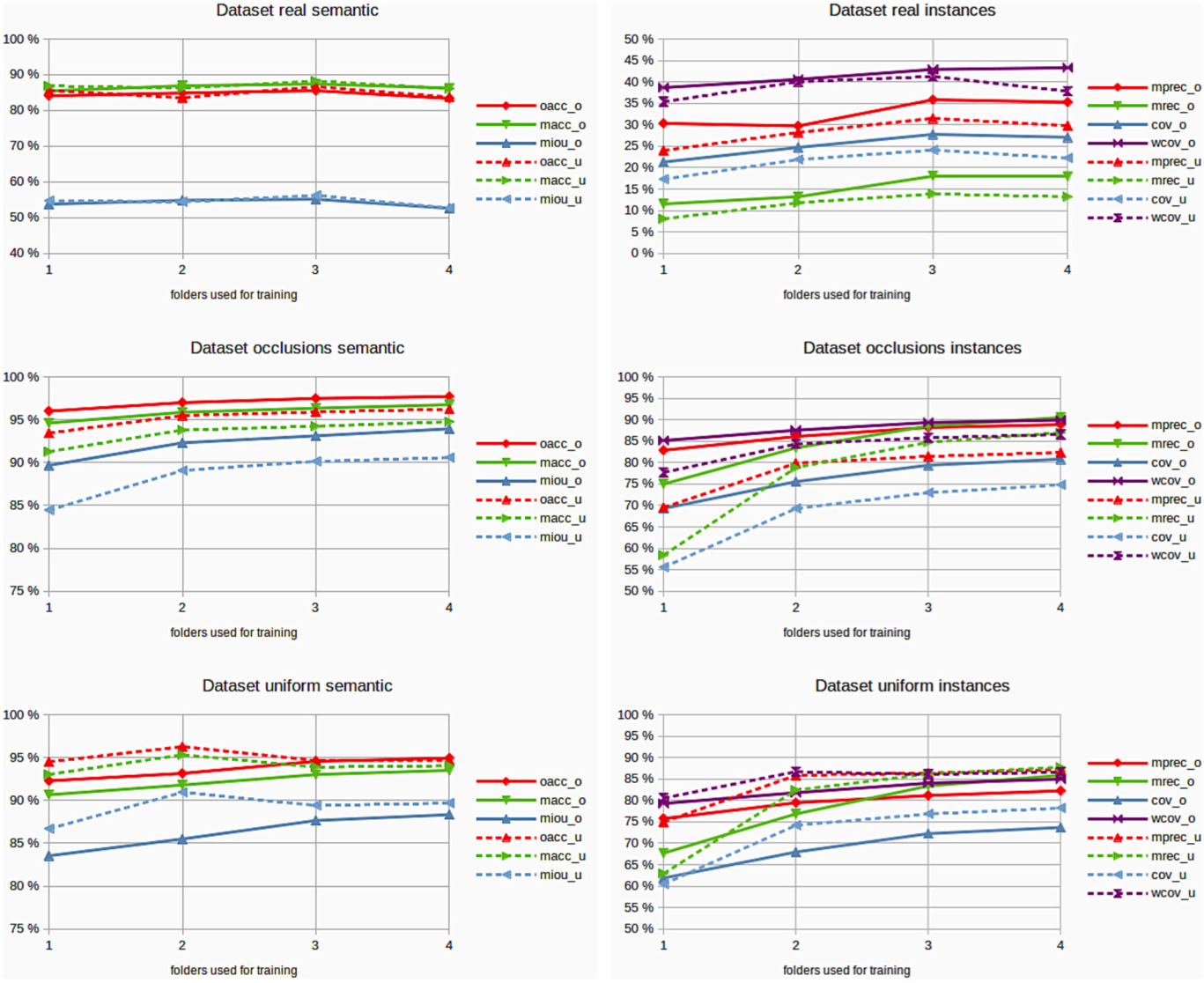


Fig. 11. Graphics comparing the metrics obtained with the different amounts of data used for training. The legends indicates the metric plus the dataset used for training, being _o or _u the datasets ***O*** and ***U***, respectively.

network is trained for 2000 epochs with a batch size of 5 and an early stop of 100, using the metric Cov. The grid size utilized in the *Group-Merging* method for merging the crops during post-processing is 0.1 m, which is twice the size of the grid used in the pre-processing stage. The other implementation details are the same as those used in JSNet. The experiments have been carried out at the Centro de Supercomputación de Galicia (CESGA) [53], using 40G of RAM and a GPU NVIDIA A100.

Table 6 shows the time and number of epochs that each training strategy needs calculated from the k-fold cross validation method.

Table 7, Table 8 and Table 9 show the metrics obtained for each model in the dataset ***R***, ***O*** and ***U***, respectively.

To compare the models in a more visual way, Fig. 11 has been introduced. It contains graphics showing how the use of different training data size affects. In addition, the graphs also compare the effect of training with ***U*** or ***O***.

We will call **X_y** the model trained with the data set (***X***) and n = y. For example, the model trained with ***O*** and ***n*** = 3 is the model ***O_3***.

The best metrics in each dataset are obtained by those models trained on the same dataset and with the maximum amount of data. This is observed in Fig. 11 and Table 8 and Table 9, in which the best models are ***O_4*** and ***U_4***, respectively. Note that in Table 9, ***U_2, U_3*** and ***U_4*** have similar semantic metrics, but the instance metrics of ***U_4*** are better. In the tests with dataset ***R***, the best model is ***O_3.***

Table 10
Comparation of metric *cov* and *wCov* obtained with different models in the datasets ***U***, ***O*** and ***R*** expressed in %.

Models being compared	*cov*			*wCov*		
	U	*O*	*R*	*U*	*O*	*R*
U_2 vs ***U_1***	18	20	27	7	8	13
U_3 vs ***U_2***	3	5	10	-1	2	3
U_4 vs ***U_3***	2	2	−8	−1	1	−8
O_2 vs ***O_1***	9	8	16	3	3	5
O_3 vs ***O_2***	6	5	12	1	2	6
O_4 vs ***O_3***	2	2	−2	1	1	1
O_1 vs ***U_1***	2	20	19	−2	9	9
O_2 vs ***U_2***	−9	8	11	−6	4	1
O_3 vs ***U_3***	−6	8	13	−2	4	4
O_4 vs ***U_4***	−6	7	18	−2	4	13

To compare the effect of using different data size and different dataset for training, Table 10 has been included, which shows the differences of metric *cov* and *wCov* obtained with all the models in the three datasets. These metrics are selected for being the most relevant to split the truss into its elements. Regarding the data size used for training,

there is a considerable improvement in all models in all tests by increasing the number of point clouds used for training from 50 to 100. However, by increasing from 100 to 150 and from 150 to 200, the improvement is much lower, even having a considerable decline in the tests with ***R***.

Regarding the dataset used for training, except when only 50 point clouds are used for training, there is an average improvement of around 7% in *cov* and 4% in *wCov* if the dataset used for training and testing is the same. However, in the tests with dataset ***R***, models trained with ***O*** obtain an improvement of around 3% in *wCov*. It is true that comparing ***O_4*** vs ***U_4*** it increases again, but it is due to a decline of the *wCov* of ***U_4*** rather than an improvement of ***O_4***, as it is shown in Fig. 11. Analysing the metric *cov*, this improvement is higher, at around the 14%. It is interesting to note that, for metric *cov*, the improvement obtained from training with occlusions is twice as much in the tests with dataset ***R*** s as on ***O***. These improvements in *cov* and *wCov* make sense since ***R*** has occlusions, so the dataset ***O*** (occlusions) is more like ***R*** (real) than the dataset ***U*** (uniform).

With the aim of visualising the results obtained and to analyse them from a qualitative perspective, Fig. 12 shows some of the segmented synthetic point clouds with occlusions obtained with the model trained with $\boldsymbol{n} = 3$ and dataset ***O***, which is the model with the best results in ***R***. They are coloured according to two different pallets: the first one (top row) represents the semantic segmentation, where members of the same class are coloured in the same way; the second one (bottom row) shows the instance segmentation, which means that every individual member is randomly coloured.

To complete the qualitative analysis, the errors are also visualised to study where they are. Fig. 13 shows different semantic and instance errors in the same point clouds of Fig. 12. This visualization shows, as well as the metrics, that there are more instance errors than semantic ones. Furthermore, it also indicates that how both errors tend to accumulate in the same areas: the nodes. Note that instance errors are those points of the correctly segmented instance that do not belong to the real instance. Moreover, all the points of the incorrect segmented instances are errors. An incorrect instance is an instance with a IoU < 0.5 with all real instance. For example, if a ground truth instance is segmented into three instances and all of them have a IoU < 0.5 with the ground truth instance, all the points of the segmented instances are displayed as errors.

The methodology is also compared with the heuristic method presented in [15], that to the best of our knowledge, are the only two methods of truss bridges point cloud segmentation. For comparison, the heuristic method is tested in three randomly selected point clouds of the dataset ***O***. Relevant measurements are manually taken in the point clouds to feed the inputs into the algorithm. As result, the metrics in Table 11 are obtained. Note that deck points are not considered into this analysis since the heuristic method is not designed to segment those points.

With the aim of analysing the qualitative differences between the presented deep learning method and the heuristic one, the segmentation results and the errors of the heuristic approach are visualised in the same way as in the deep learning method, as shown in Fig. 14 and Fig. 15, shown the labels and the errors, respectively. The heuristic method, unlike the deep learning approach, leaves points without labelling, coloured in black in Fig. 14 and Fig. 15. For the computation of metrics, these non-labelled points are considered errors, except for the deck points, which are not considered. In the first point cloud, the main errors appear in nodes, as with the deep learning method. However, in the second point cloud, the faces are not correctly split, causing errors in the segmentation. The deck is considered as the upper face, which means that the actual upper face is not analysed, and some parts of the deck are segmented as beams. In the third face, the bottom chords are not well segmented: the lateral areas of the deck are segmented as chords.

Before comparing the results of the presented deep learning method with those of the heuristic method, the following should be considered. First, it is important to notice that this method requires a series of measurements to be taken in the point cloud. Second, the deck points have been removed from the calculation of the metrics since the heuristic method does not label deck points. Despite all these licenses granted to the heuristic method, the metrics obtained, shown in Table 11, are 44% of oAcc and 45% of cov. These metrics show how the new method improves on the old one in the dataset ***O***. Besides, the new method is easier to use since it does not require to make measurements in the point clouds, and it is able the label deck points. It is also interesting to discuss the results from a qualitative point of view. These analysis shows that the main failure sizes are the nodes, as shown in Fig. 13. This is because nodes are union zones where it is difficult to precisely determine to which member the points of these places belong. This error occurs also in the heuristic method, although the main error of this one is the large number of points it leaves unlabelled. The reason why there become unlabelled points is because the heuristic method was designed as a conservative method, unlike the new method, which segments all the points.

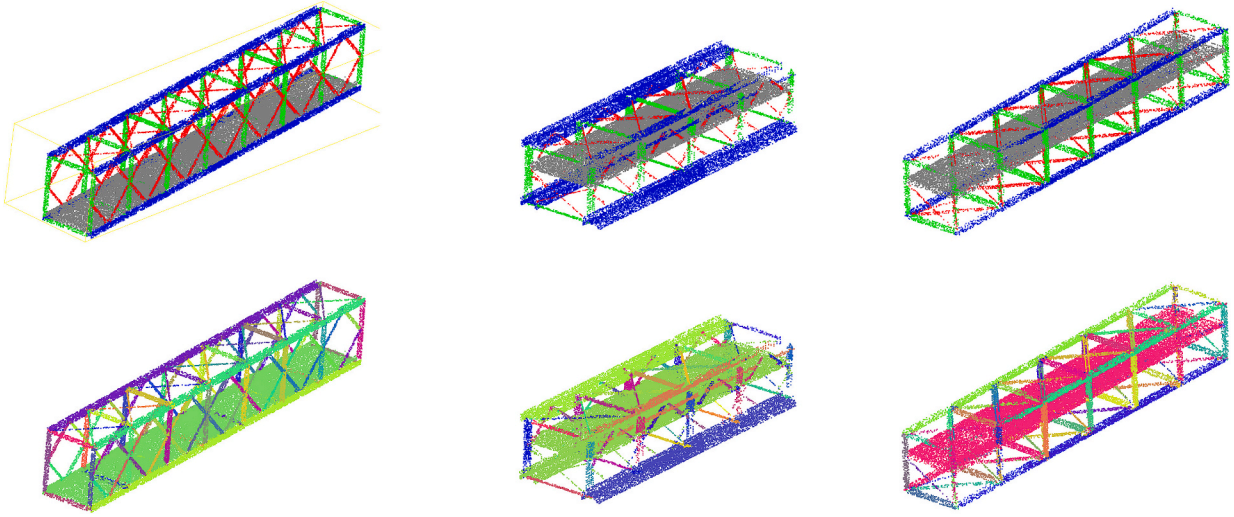


Fig. 12. Semantic and instance segmented synthetic point clouds with occlusions (***O***) performed by the model ***O_3***. Top row are semantic segmented point clouds where the deck is coloured in grey, chords in blue, parallels in green, and verticals in red. Bottom row are instance segmented point clouds, with every individual member randomly coloured. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

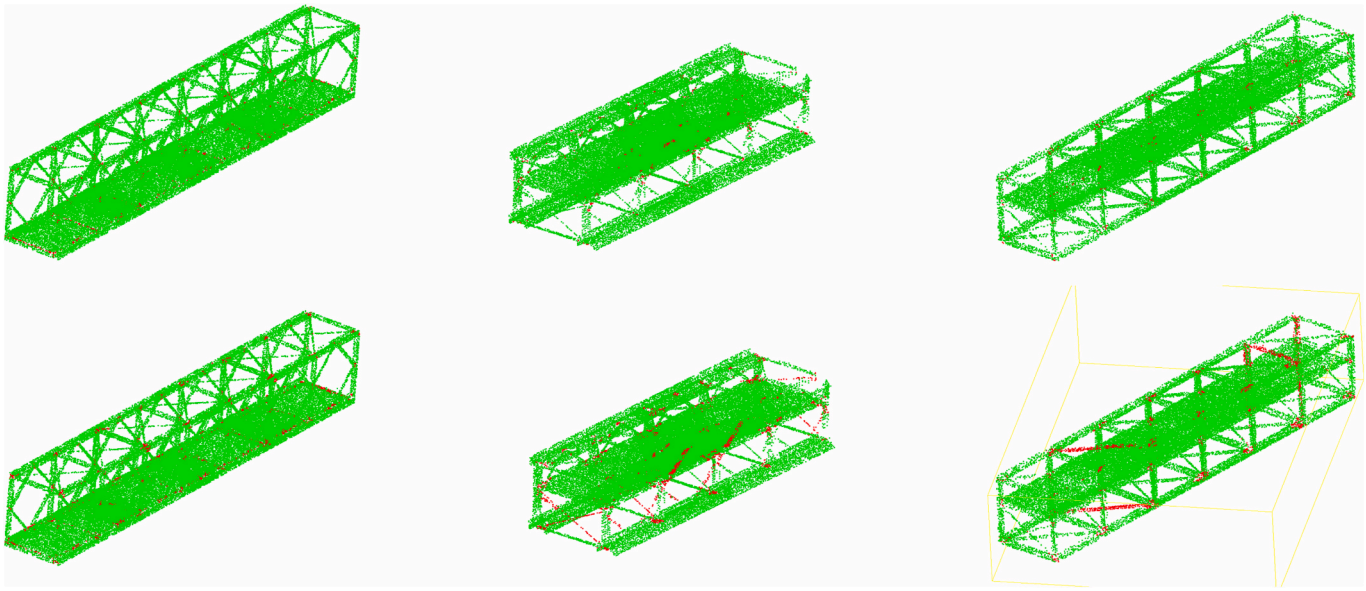


Fig. 13. Segmentation errors in synthetic point clouds with occlusions (***O)*** performed by the model ***O_3.*** First row shows semantic errors. Second row instance errors. True positives in green, errors in red. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Table 11
Average metrics of the heuristic and the deep learning approach in %.

Method	Semantic metrics			Instance metrics			
	oAcc	mAcc	mIoU	mPrec	mRec	Cov	wCov
DL **n = 3** dataset **O**	97	96	93	88	89	79	89
Hueristic	44	57	32	61	53	45	46

In the same way, the results obtained in the real point clouds are shown to discuss them from a qualitative point of view. Fig. 16 and Fig. 17 show the results and errors performed by the deep learning model ***O_3***, respectively.

To compare the effect of training with synthetic dataset with occlusions in real point clouds, Fig. 18 and Fig. 19 are introduced to show the segmentation and errors performed by the deep learning model ***U_3***, respectively. We chose ***U_3*** because is the model with the best metrics trained with the dataset ***U.***

Comparing the results obtained with model ***O_3*** and ***U_3*** shown in Table 7, we see that the semantic metrics obtained by ***U_3*** are a bit better than ***O_3***. However, the instance metrics of ***O_3*** are much better than ***U3.*** Such as in the synthetic dataset, the semantic results are better than the instance ones, which shows the positive effect of training with occlusions. Regarding the figures, it is shown that the 2nd bridge is better analysed from both models than the 1st one.

Our interpretation suggests that the second point cloud bears more resemblance to the synthetic data, as it contains fewer diagonals compared to the first point cloud. The primary weakness of the deep learning algorithm is its inability to provide semantic labels for non-structural elements like railings, lampposts, or walkways since these elements are not simulated in the synthetic data. These non-structural elements are not considered as errors in the images due to the lack of labels from the heuristic method. However, these errors might make it difficult to create a structural model from the segmented data.

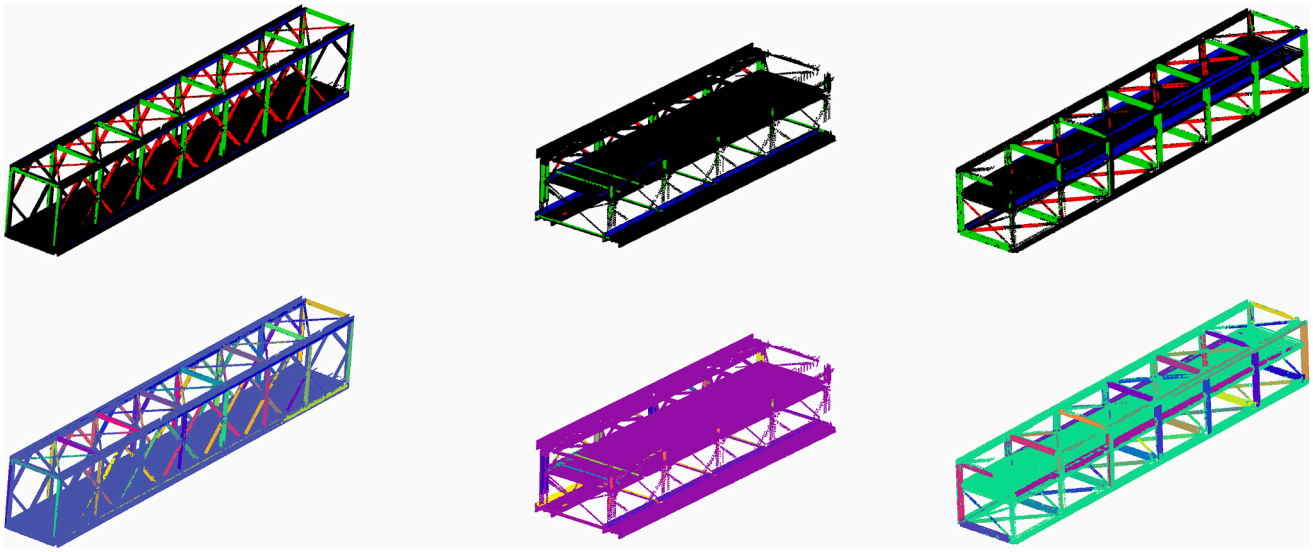


Fig. 14. Segmented synthetic point clouds with occlusions (***O***) using the heuristic method [15]. Pallet of colours of Fig. 12. Non-segmented points are coloured in black.

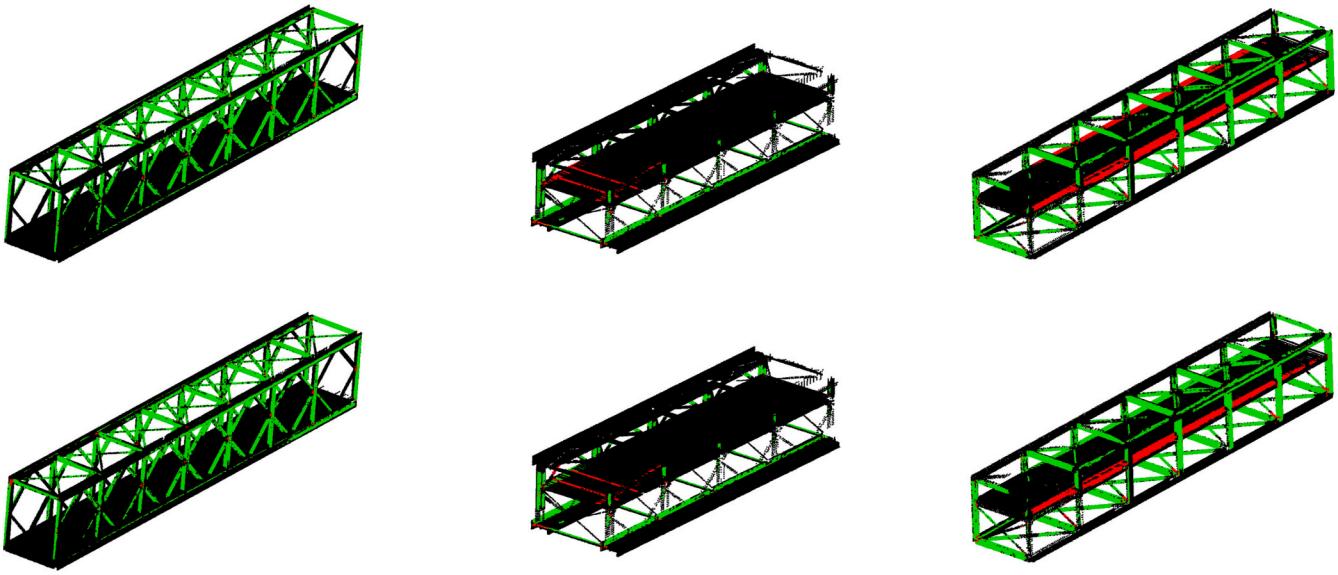


Fig. 15. Segmentation errors in synthetic point clouds with occlusions (***O***) using the heuristic method [15]. Pallet of colours of Fig. 13. Non-segmented points in black.

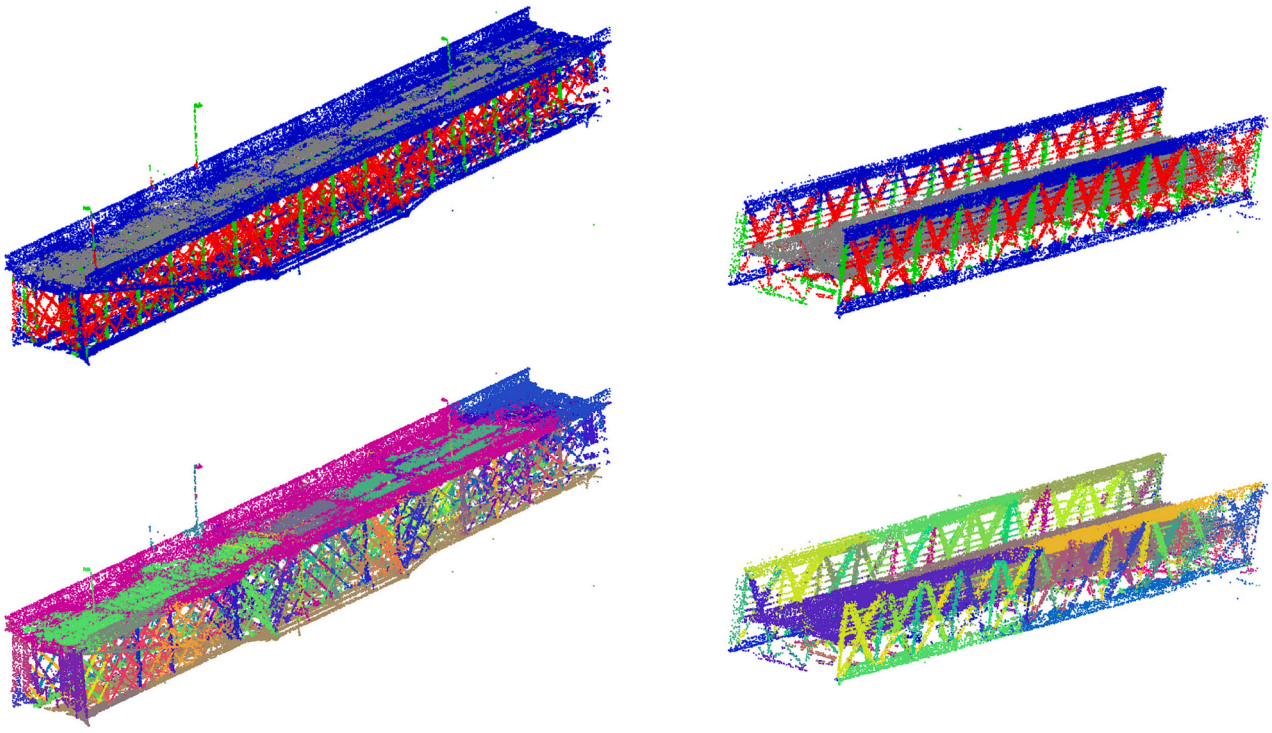


Fig. 16. Segmented real point clouds performed by the model ***O_3***. Pallet of colours of Fig. 12. Dataset of [15].

5. Conclusions

This paper presents a methodology for generating synthetic point clouds of truss bridges. Moreover, this data is used to demonstrate the applicability of a deep learning approach to simultaneously obtain a semantic and instance segmentation of truss bridges point clouds.

The generation of point clouds of synthetic truss bridges are made from the specification of the dimensions and components of them. These specifications are the typology, the profiles of the members, their orientation, and the deck position. In addition, it is also possible to specify which members form the truss, as there may be multiple topologies for the same typology of truss. For instance, it is possible to create a Bailey truss with or without Sant Andrew's cross, which chords are U profiles with their open face pointing up or down. Moreover, it is possible to create point clouds with a uniform distribution of points or simulate the TLS positions to create occlusions. The dataset is generated automatically by selecting these parameters randomly from several predefined options, which allows for a dataset with high variability. Some results are shown in Fig. 9.

The process of generating synthetic point clouds works as follows. First, the nodes of the bridge are calculated. Then, each member is created as an extrusion of the specified profile and placed in its place in the truss. To this end, each member is defined from its pair of nodes, its profile, and its orientation. The dimensions of the profile are read from a

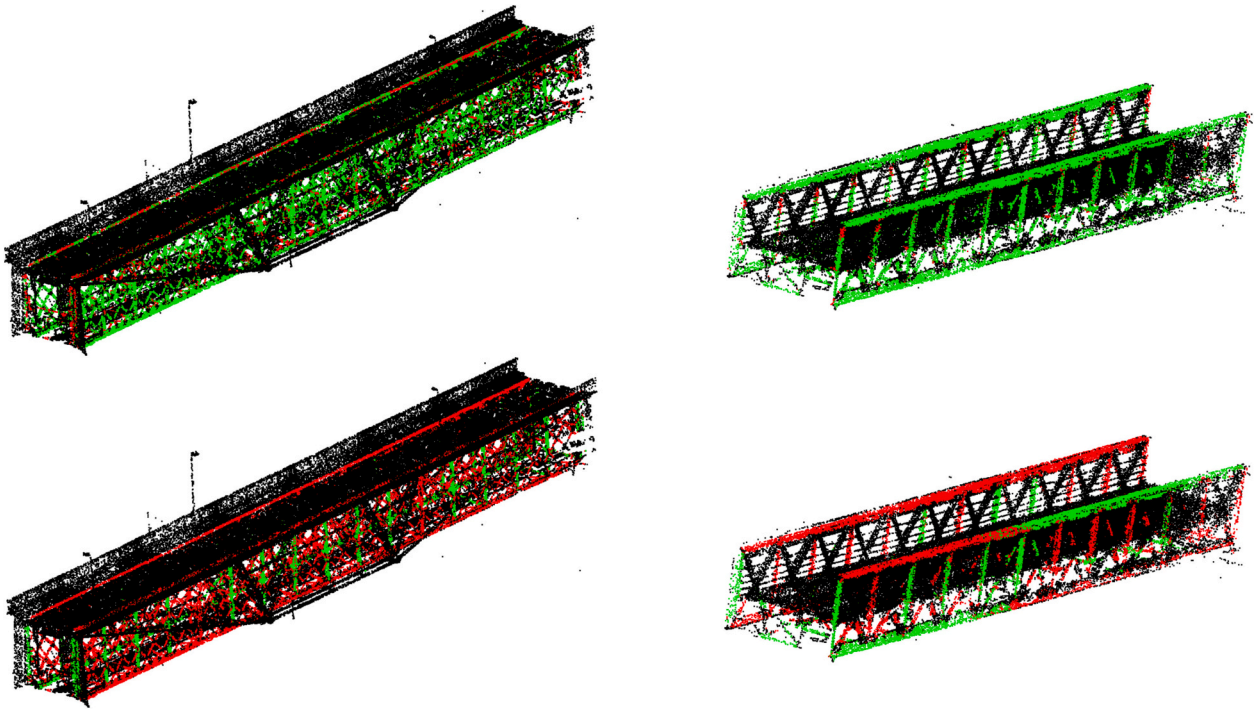


Fig. 17. Segmentation errors in real point clouds performed by the model ***O_3***. Pallet of colours of Fig. 13. Point without ground truth are coloured in black. Dataset of [15].

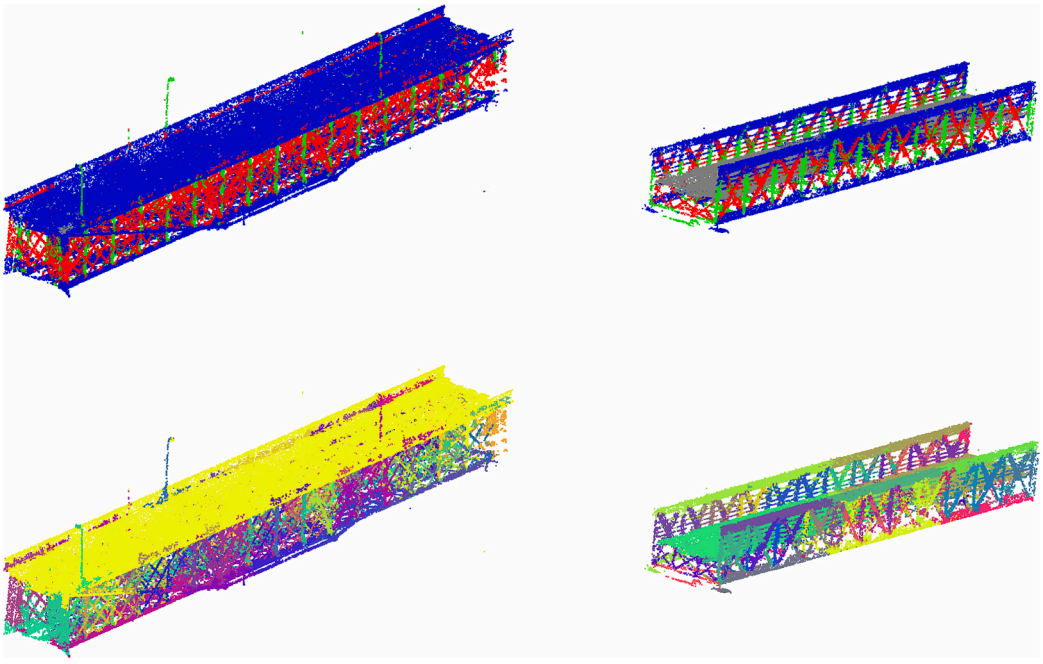


Fig. 18. Segmented real point clouds performed by the model ***U_3***. Pallet of colours of Fig. 12. Dataset of [15].

table in which standardized profiles dimensions are saved. The standardized profiles used are CH, HD, HL, HLZ, HP, HSS, IPN, L, PFC, T, U, UB, UBP, UC, UPE, and UPN, with several sizes available for each of them. This profile is generated in the form of mesh, by creating the triangles that define the profile of the member and its lateral face. Then, the mesh is placed with respect to its node pairs and its orientation. The deck is also generated as a mesh, and its position in the bridge can be modified. Finally, there are two methods for obtaining the point cloud: random points are uniformly generated on the surface of the mesh, or define virtual TLS positions from which ray casting algorithm are applied to calculate the intersection points with the mesh generating a point cloud with occlusions. These processes are automatically performed from the inputs. Note that for this work we only integrated two typologies: Bailey and Brown. However, several typologies might be added taking advantage of the main functions of this method, since it only implies to create new methods for calculating the location of the nodes.

In this work, we only use the synthetic data generator to create point

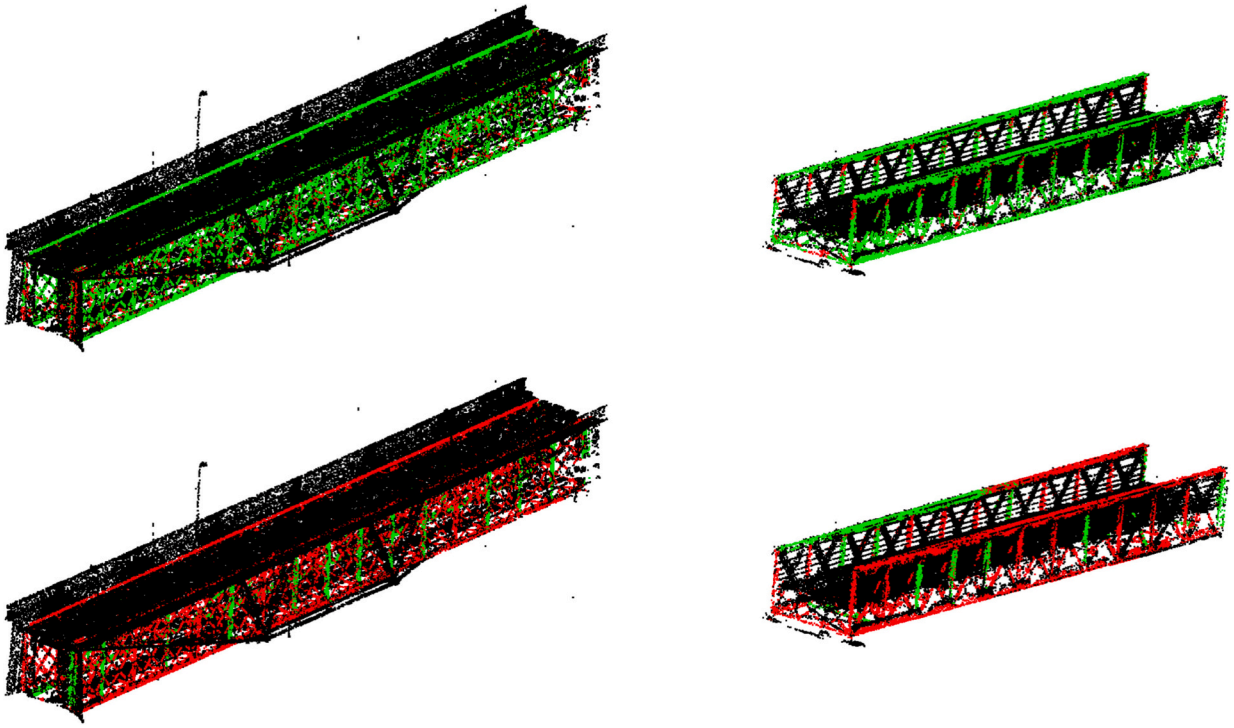


Fig. 19. Segmentation errors in real point clouds performed by the model ***U_3***. Pallet of colours of Fig. 13. Point without ground truth are coloured in black. Dataset of [15].

clouds to train and test a deep learning method. Nevertheless, this method has other applications that need to be explored, as the use of synthetic meshes, or the wire frame structure made from the nodes and the profiles of each member. Ultimately, this method allows the massive generation of various types of geometric data of truss bridges.

On the other hand, this paper demonstrates the well-performance of deep learning approaches for semantic and instance segmentation of truss bridge point clouds. The state-of-the-art of deep learning methods for instance segmentation of point clouds are widely explored, but not in data as complex as truss bridges, where their instances are intertwined to each other. For this purpose, an existing state-of-art architecture is adapted, trained and tested. We use several strategies for training the point clouds to investigate the effect of synthetic data. The models are trained just with the uniform synthetic data (***U***) or with the synthetic data with occlusion (***O***). Moreover, the volume of data used for training varies. The parameter ***n*** indicates the number of folders with data used for training, ranging from 1 to 4. All models are tested in real point clouds, in ***U*** and ***O***.

The obtained results shown in Fig. 11 indicate that the models trained ***O*** are better in the tests with ***R***, showing that training with synthetic point clouds with occlusions has a positive effect in real data. In the tests with synthetic data, using the dataset that matches the testing data yields the most favourable results. However, in the tests with ***O***, the improvement of the models trained with that data compared to those trained with ***U*** is much greater than vice versa. Fig. 11 also indicates the effect of using several amounts of training data. The results indicate that testing on synthetic data there is an improvement when training with more data. However, in the tests with real data, the best models are those trained with $\boldsymbol{n} = 3$ and not $\boldsymbol{n} = 4$, although it is true that their differences are small and that only two real point clouds are available for testing.

Comparing the deep learning method with our previous heuristic, which from the best of our knowledge, are the only two existing methods, we obtained the following conclusions. The deep learning method outperforms the heuristic one in the dataset ***O***, as shown in Table 11. The deep learning model ***U_3*** achieves 96% mAcc and 79% Cov, while the heuristic method 57% mAcc and 45% Cov. However, in real point clouds, ***U_3*** obtains 87% mAcc and 28% Cov, using as ground truth only the points labelled by the heuristic method. The main errors can be due to two reasons:

- The synthetic does not have non-structural elements, as the real data does.
- One of the real bridges is very different from the synthetic typologies.

Future work may consider introducing non-structural members in the synthetic data to make the architecture lean to differentiate between them. Moreover, some members are segmented together, or with certain parts of their adjacent members. This could make it difficult to use these data for modelling purposes. Therefore, refinement methods should be explored to solve this problem. Or from another perspective, approaches that introduce a priori information via axioms into artificial neural networks can be explored.

Funding

This project has received funding from the European Union's Horizon 2020 Research and Innovation Program under grant agreement No. 958171. This work has been partially supported by the Spanish Ministry of Science and Innovation through the PONT3 project Ref. PID2021-124236OB-C33, and through the human resources Grant RYC2021-033560-I funded by MCIN/AEI/10.13039/501100011033 and by European Union NextGenerationEU/PRTR. This document reflects only the views of the authors. Neither the Innovation and Networks Executive Agency (INEA) nor the European Commission is in any way responsible for any use that may be made of the information it contains.

This work has been partially supported by the University of Vigo through the human resources grant: "Axudas para a contratación de personal investigador predoutoral en formacion da Universidade De Vigo 2021" (PREUVIGO-21).

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

Data will be made available on request.

Acknowledgements

Authors would like to thank CESGA [53] for allowing to use their computers to test the methodology.

The library Open3D [47] is used to generate the synthetic data, to visualise it and create some figures.

During the method's development, the software CloudCompare [54] was utilized for generating certain figures and conducting visual inspections.

References

[1] A. Boin, A. McConnell, Preparing for critical infrastructure breakdowns: the limits of crisis management and the need for resilience, J. Conting. Crisis Manag. 15 (2007) 50–59, https://doi.org/10.1111/j.1468-5973.2007.00504.x.
[2] M. Ouyang, Review on modeling and simulation of interdependent critical infrastructure systems, Reliab. Eng. Syst. Saf. 121 (2014) 43–60, https://doi.org/10.1016/J.RESS.2013.06.040.
[3] J.J. Magoua, F. Wang, N. Li, High level architecture-based framework for modeling interdependent critical infrastructure systems, Simul. Model. Pract. Theory 118 (2022), https://doi.org/10.1016/J.SIMPAT.2022.102529.
[4] Keeping European Bridges Safe | EU Science Hub. https://joint-research-centre.ec.europa.eu/jrc-news/keeping-european-bridges-safe-2019-04-05_en, 2023 (accessed July 1, 2022).
[5] M. Soilán, A. Sánchez-Rodríguez, P. Del Río-Barral, C. Perez-Collazo, P. Arias, B. Riveiro, Infrastructures Review of Laser Scanning Technologies and their Applications for Road and Railway Infrastructure Monitoring, 2019, https://doi.org/10.3390/infrastructures4040058.
[6] L. Ma, Y. Li, J. Li, C. Wang, R. Wang, M.A. Chapman, Mobile laser scanned point-clouds for road object detection and extraction: a review, Remote Sens. 10 (2018) 1531, https://doi.org/10.3390/RS10101531.
[7] H. Guan, J. Li, S. Cao, Y. Yu, Use of mobile LiDAR in road information inventory: a review, Int. J. Image Data Fusion 7 (2016) 219–242, https://doi.org/10.1080/19479832.2016.1188860.
[8] S. Gargoum, K. El-Basyouny, Automated extraction of road features using LiDAR data: A review of LiDAR applications in transportation, in: In: 2017 4th International Conference on Transportation Information and Safety, ICTIS 2017 - Proceedings, Institute of Electrical and Electronics Engineers Inc., 2017, pp. 563–574, https://doi.org/10.1109/ICTIS.2017.8047822.
[9] Y. Jing, B. Sheil, S. Acikgoz, Segmentation of large-scale masonry arch bridge point clouds with a synthetic simulator and the BridgeNet neural network, Autom. Constr. 142 (2022), 104459, https://doi.org/10.1016/J.AUTCON.2022.104459.
[10] R. Lu, I. Brilakis, C.R. Middleton, Detection of structural components in point clouds of existing RC bridges, Comp. Aid. Civ. Infrastruct. Eng. 34 (2019) 191–212, https://doi.org/10.1111/mice.12407.
[11] 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.
[12] B. Riveiro, M.J. DeJong, B. Conde, Automated processing of large point clouds for structural health monitoring of masonry arch bridges, Autom. Constr. 72 (2016) 258–268, https://doi.org/10.1016/J.AUTCON.2016.02.009.
[13] N. Gyetvai, L. Truong-Hong, D.F. Laefer, Laser scanning-based diagnostics in the structural assessment of historic wrought iron bridges, in: Proceedings of the Institution of Civil Engineers - Engineering History and Heritage, ICE Publishing, 2018, pp. 76–89, https://doi.org/10.1680/JENHH.17.00018.
[14] Z. Shang, Z. Shen, Flight planning for survey-grade 3D reconstruction of truss bridges, Remote Sens. 14 (2022) 3200, https://doi.org/10.3390/RS14133200.
[15] D. Lamas, A. Justo, M. Soilán, M. Cabaleiro, B. Riveiro, Instance and semantic segmentation of point clouds of large metallic truss bridges, Autom. Constr. 151 (2023), 104865, https://doi.org/10.1016/J.AUTCON.2023.104865.
[16] I. Armeni, O. Sener, A.R. Zamir, H. Jiang, I. Brilakis, M. Fischer, S. Savarese, 3D semantic parsing of large-scale indoor spaces, in: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, IEEE Computer Society, 2016, pp. 1534–1543, https://doi.org/10.1109/CVPR.2016.170.
[17] A. Dai, A.X. Chang, M. Savva, M. Halber, T. Funkhouser, M. Nießner, ScanNet: Richly-annotated 3D reconstructions of indoor scenes, in: Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR, Institute of Electrical and Electronics Engineers Inc., 2017, pp. 2432–2443, https://doi.org/10.1109/CVPR.2017.261.
[18] Papers With Code. https://paperswithcode.com/, 2023 (accessed March 24, 2023).
[19] T. Vu, K. Kim, T.M. Luu, T. Nguyen, C.D. Yoo, SoftGroup for 3D instance segmentation on point clouds, in: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, IEEE Computer Society, 2022, pp. 2698–2707, https://doi.org/10.1109/CVPR52688.2022.00273.
[20] J. Sun, C. Qing, J. Tan, X. Xu, Superpoint Transformer for 3D Scene Instance Segmentation, 2022, https://doi.org/10.48550/arXiv.2211.15766.
[21] M. Zhong, X. Chen, X. Chen, G. Zeng, Y. Wang, MaskGroup: Hierarchical point grouping and masking for 3D instance segmentation, in: IEEE International Conference on Multimedia and Expo, IEEE Computer Society, 2022, https://doi.org/10.1109/ICME52920.2022.9859996.
[22] M. Kolodiazhnyi, D. Rukhovich, A. Vorontsova, A. Konushin, Top-Down Beats Bottom-Up in 3D Instance Segmentation, 2023, https://doi.org/10.48550/arXiv.2302.02871.
[23] J. Schult, F. Engelmann, A. Hermans, O. Litany, S. Tang, B. Leibe, Mask3D for 3D Semantic Instance Segmentation. https://arxiv.org/abs/2210.03105v1, 2022 (accessed March 24, 2023).
[24] M. Jaderberg, K. Simonyan, A. Zisserman, K. Kavukcuoglu, Spatial transformer networks, in: Adv Neural Inf Process Syst, Neural Information Processing Systems Foundation, 2015, pp. 2017–2025. https://arxiv.org/abs/1506.02025v3 (accessed January 3, 2022).
[25] L. Pearlstein, M. Kim, W. Seto, Convolutional neural network application to plant detection, based on synthetic imagery, in: Proceedings - Applied Imagery Pattern Recognition Workshop, Institute of Electrical and Electronics Engineers Inc., 2017, https://doi.org/10.1109/AIPR.2016.8010596.
[26] P.S. Rajpura, H. Bojinov, R.S. Hegde, Object Detection Using Deep CNNs Trained on Synthetic Images. https://arxiv.org/abs/1706.06782v2, 2017 (accessed March 24, 2023).
[27] X. Liu, W. Liang, Y. Wang, S. Li, M. Pei, 3D head pose estimation with convolutional neural network trained on synthetic images, in: Proceedings - International Conference on Image Processing, ICIP, IEEE Computer Society, 2016, pp. 1289–1293, https://doi.org/10.1109/ICIP.2016.7532566.
[28] T. Björklund, A. Fiandrotti, M. Annarumma, G. Francini, E. Magli, Robust license plate recognition using neural networks trained on synthetic images, Pattern Recogn. 93 (2019) 134–146, https://doi.org/10.1016/J.PATCOG.2019.04.007.
[29] Y. Xu, S. Arai, F. Tokuda, K. Kosuge, A convolutional neural network for point cloud instance segmentation in cluttered scene trained by synthetic data without color, IEEE Access. 8 (2020) 70262–70269, https://doi.org/10.1109/ACCESS.2020.2978506.
[30] B. Berenguel-Baeta, J. Bermudez-Cameo, J.J. Guerrero, OmniSCV: an omnidirectional synthetic image generator for computer vision, Sensors 2020 (20) (2020) 2066, https://doi.org/10.3390/S20072066.
[31] Z. Zhou, S.S. Kumar, K. Mallery, et al., Synthetic image generator for defocusing and astigmatic PIV/PTV, Meas. Sci. Technol. 31 (2019), 017003, https://doi.org/10.1088/1361-6501/AB42BB.
[32] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, M. Chen, OpenAI, Hierarchical Text-Conditional Image Generation with CLIP Latents. https://arxiv.org/abs/2204.06125v1, 2022 (accessed March 24, 2023).
[33] D. Griffiths, J. Boehm, SynthCity: A large scale synthetic point cloud. https://arxiv.org/abs/1907.04758v1, 2019 (accessed March 24, 2023).
[34] blender.org - Home of the Blender project - Free and Open 3D Creation Software. https://www.blender.org/, 2023 (accessed March 24, 2023).
[35] G. Curnis, S. Fontana, D.G. Sorrenti, GTASynth: 3D synthetic data of outdoor non-urban environments, Data Brief 43 (2022), 108412, https://doi.org/10.1016/J.DIB.2022.108412.
[36] B. Hurl, K. Czarnecki, S. Waslander, Precise synthetic image and LiDAR (PreSIL) dataset for autonomous vehicle perception, in: IEEE Intelligent Vehicles Symposium, Proceedings, Institute of Electrical and Electronics Engineers Inc., 2019, pp. 2522–2529, https://doi.org/10.1109/IVS.2019.8813809.
[37] A. Gaidon, Q. Wang, Y. Cabon, E. Vig, VirtualWorlds as proxy for multi-object tracking analysis, in: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition. 2016-December, 2016, pp. 4340–4349, https://doi.org/10.1109/CVPR.2016.470.
[38] Y. Cabon, N. Murray, M. Humenberger, Virtual KITTI 2, 2020. https://arxiv.org/abs/2001.10773v1 (accessed March 24, 2023).
[39] J.E. Deschaud, D. Duque, J.P. Richa, S. Velasco-Forero, B. Marcotegui, F. Goulette, Paris-CARLA-3D: a real and synthetic outdoor point cloud dataset for challenging tasks in 3D mapping, Remote Sens. 13 (2021) 4713, https://doi.org/10.3390/RS13224713.
[40] CARLA, Simulator. https://carla.org/, 2023 (accessed March 24, 2023).
[41] G. Andrade, G. Ramos, D. Madeira, R. Sachetto, R. Ferreira, L. Rocha, G-DBSCAN: A GPU accelerated algorithm for density-based clustering, in: Procedia Comput Sci, Elsevier B.V., 2013, pp. 369–378, https://doi.org/10.1016/j.procs.2013.05.200.
[42] 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.
[43] L.U. Roudan, Ioannis Brillakis, Campbell R. Middleton, Detection of Structural Components in Point Clouds of Existing RC Bridges, 2023, https://doi.org/10.5281/ZENODO.1240534.
[44] 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.
[45] X. Yang, E. del Rey Castillo, Y. Zou, L. Wotherspoon, Semantic segmentation of bridge point clouds with a synthetic data augmentation strategy and graph-

structured deep metric learning, Autom. Constr. 150 (2023), 104838, https://doi.org/10.1016/J.AUTCON.2023.104838.
[46] X. Yang, E. del Rey Castillo, Y. Zou, L. Wotherspoon, Y. Tan, Automated semantic segmentation of bridge components from large-scale point clouds using a weighted superpoint graph, Autom. Constr. 142 (2022), 104519, https://doi.org/10.1016/J.AUTCON.2022.104519.
[47] Q.-Y. Zhou, J. Park, V. Koltun, Open3D: A Modern Library for 3D Data Processing, 2018, https://doi.org/10.48550/arxiv.1801.09847.
[48] LAS (LASer) File Format, Version 1.4. https://www.loc.gov/preservation/digital/formats/fdd/fdd000418.shtml, 2023 (accessed June 25, 2021).
[49] Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu, M. Bennamoun, Deep learning for 3D point clouds: a survey, IEEE Trans. Pattern Anal. Mach. Intell. 43 (2021) 4338–4364, https://doi.org/10.1109/TPAMI.2020.3005434.
[50] L. Zhao, W. Tao, JSNet: joint instance and semantic segmentation of 3D point clouds, Proc. AAAI Conf. Artific. Intellig. 34 (2020) 12951–12958, https://doi.org/10.1609/AAAI.V34I07.6994.
[51] S3DIS Benchmark (3D Instance Segmentation) | Papers with Code. https://paperswithcode.com/sota/3d-instance-segmentation-on-s3dis, 2023 (accessed January 24, 2023).
[52] W. Wang, R. Yu, Q. Huang, U. Neumann, SGPN: Similarity Group Proposal Network for 3D, Point Cloud Instance Segmentation, 2023.
[53] Inicio - Cesga - Centro de Supercomputación de Galicia. https://www.cesga.es/, 2023 (accessed February 1, 2023).
[54] CloudCompare. https://www.cloudcompare.org/, 2023 (accessed July 6, 2022).