13th International Conference on
Structural Health Monitoring of Intelligent Infrastructure
DOI: 10.3217/978-3-99161-057-1-083

Developing a Deep Learning-Based Method to Segment Bridge Members by using 2D Cross Sectional Point Clouds

Nao Hidaka [1], Naofumi Hashimoto [1], Ei Watanabe [2], Daisuke Uchiyama [1]
[1] Nagoya Institute of Technology, Gokiso-cho, Showa-ku, Nagoya, Aichi, Japan
[2] Aichi Prefectural Government, 3-1-2, Sannomaru, Naka-ku, Nagoya, Aichi, Japan
email: hidaka.nao@nitech.ac.jp, n.hashimoto.704@stn.nitech.ac.jp, ei_watanabe@pref.aichi.lg.jp, d.uchiyama.703@stn.nitech.ac.jp

ABSTRACT: For efficient maintenance and repair, 3D modeling of bridges for converting analytical models and visualizing deformation locations is being advanced. Instead of manually creating models from drawings and ledgers, automating model generation from point cloud data, capable of capturing as-is geometry quickly and widely, can improve efficiency. Generating 3D modeling of bridges from point cloud data requires segmenting each member, but dynamically setting thresholds for shape features and positional relationships is challenging due to point density and missing points. While deep learning can dynamically set thresholds, in the case of point cloud data, it is impractical to prepare sufficient training data, and the number of inputting points is inadequate for setting appropriate thresholds. Therefore, this research focuses on two aspects: most bridges consist of members with swept cross sections along longitudinal direction, and deep learning classification methods for 2D images are highly developed. The aim is to segment members based on deep learning on 2D cross-sectional point cloud data obtained by slicing along longitudinal direction. This reduces the number of inputting points and increases training data. Additionally, fine cross sections enable segmentation close to 3D. The multiple patterns of learning methods, training data processing, and procedures of segmentations are compared to identifying highly accurate segmentation methods.

KEY WORDS: Point cloud; Deep learning; Segmentation; Cross section; Steel truss bridge.

1 INTRODUCTION

A vast number of existing bridges are rapidly aging. Since it is not practical to rebuild all of them at the same time, strategic renewal through life cycle extension is required. To extend the life cycle of bridges, 3D models of bridges are created. These models can be converted numerical analysis models [1] and can visualize deformations [2]. However, in cases of old bridges, as-build drawings are often unavailable. In addition, conditions of bridges inevitably changed since its construction due to various factors. Therefore, it is necessary to construct 3D model based on dimensions data instead of relying on drawings, but manual measurement is time-consuming and prone to various human errors.

Therefore, a method to efficiently create 3D models from point cloud data, capable of capturing as-is 3D geometry as a set of points quickly and widely, has begun to attract attention. It can make a significant contribution to efficiency. Qin et al. [3] sliced the point cloud data of a PC box girder bridge vertically from the ground and used the density of each obtained point cloud as a threshold to divide the superstructure and substructure for Building Information Modeling (BIM). Schatz et al. [4] semi-automatically divided the point cloud data of a PC box girder bridge into substructure, girders, bearing pavement, drainage facilities, etc. based on template matching, and created an Industry Foundation Classes (IFC) model. The authors [5] performed Finite Element Method (FEM) modeling of a steel truss bridge by segmenting a fine section along a longitudinal direction and dividing the point cloud of the section based on Euclidean distance. The segmentation and component determination processes in these papers are a mixture of manual processing based on human visual judgment and automatic processing based on threshold values such as shape features and positional relationships. In general, it is not easy to set the threshold dynamically in automatic processing because of the effects of point density, missing points, and other factors. In recent years, deep learning has attracted attention as a method for dynamically setting threshold values, and there are several cases where it has been applied to point cloud data processing [6, 7]. However, when targeting large-scale bridges with a wide variety of geometries, the classification is roughly divided into upper and lower structures, and the lack of training data and the number of input points are insufficient.

Therefore, this research focuses on the characteristics of bridges, which generally have many structures with swept cross sections of each member along a longitudinal direction, and the fact that classification methods for 2D images are relatively well-developed. In this research, point cloud data of 2D cross sections sliced along the longitudinal direction is used to classify members using deep learning. The number of input points can be reduced, and the number of training data can be increased by inputting point cloud data of cross sections. In addition, a finer cross-sectional view leads to an almost 3D segmentation. The optimal learning method, processing procedures, and multiple proposed patterns are compared and validated, and a highly accurate segmentation method is considered.

2 STRUCTURE OF THE MODEL

2.1 Literatures about segmentation by using deep learning

Deep learning-based image classification and segmentation often uses Convolutional Neural Network (CNN), which obtains features by convolving surrounding pixel features. If the number of pixels is reduced by convolution, the

DOI: 10.3217/978-3-99161-057-1-083

segmentation results are restored to the original number of pixels by increasing the number of pixels based on the convolved features again. Typical models include U-Net [8], which performs domain segmentation at the pixel level; Feature Pyramid Networks (FPNs) [9] which combine high-resolution and low-resolution features to exploit multi-scale information; DeepLav v3+ [10] which applies convolution with different expansion rates in parallel. As an example of member segmentation of point cloud data of bridges, Saovana et al. [11] implemented segmentation of point cloud data by segmenting photographs of bridges from multiple viewpoints into members using U-Net and then projecting the results onto the point cloud data generated by Structure from Motion (SfM). The results are then projected onto the point cloud data generated by SfM to implement point cloud data segmentation.

On the other hand, point cloud data differs from images in that pixels are not arranged in a regular and continuous manner, the same shape and color data can be obtained even if the order of the points is changed, and the three-dimensional coordinates provide a large degree of freedom. To cope with these problems, a transformation matrix is obtained from the features and applied to control the posture, and MaxPooling is applied to eliminate the effect of reordering. The segmentation of point cloud data is similar to that of a CNN. Typical models include PointNet [12], PointNet++ [13], and Dynamic Graph CNN (DGCNN) [14]. PointNet does not perform convolution to obtain features for all input points. PointNet++, an advanced version of PointNet, reduces the number of points and obtains features from the points in the neighborhood of the point, which is similar to the convolution process. DGCNN also obtains features from neighboring points, but the number of points does not change at any layer.

2.2 Structure of the deep learning model in this research

Since the aforementioned PointNet++ [13] and DGCNN [14] are candidates for deep learning models in this research, the details of these model configurations are described in this section.

The model structure of PointNet++ is shown in Figure 1. First, $\mathbf{c}_1$ is obtained by randomly sampling n_1 points from the cross-sectional point set $\mathbf{c}_0$. Next, a point $\mathbf{p}_{c1}$ in $\mathbf{c}_1$ is used to search k_1 neighbor points within r_1, and then vectors from $\mathbf{p}_{c1}$ to the neighbor points as feature values. The obtained feature values are convoluted by using Conv1d, BatchNormlize and Relu function. This process is repeated 4 times to obtain a set of cross-sectional points $\mathbf{c}_1$, $\mathbf{c}_2$, $\mathbf{c}_3$, and $\mathbf{c}_4$ reduced by random sampling and the features associated with $\mathbf{c}_4$. After that, 3 neighbor points from $\mathbf{c}_3$ to $\mathbf{c}_4$, are detected and their feature values are convoluted by using Conv1d, BatchNormlize and Relu function. This process is repeated until the $\mathbf{c}_0$ features are updated, and finally they are convolved with the classified features and output as random variables by applying the Logsoftmax function.

The model structure of DGCNN is shown in Figure 2. In PointNet++, the neighbor points were obtained based on the position coordinates, but in DGCNN, the k_n neighbor points are obtained based on all the feature values changed by convolution, not limited to the position coordinates, and the vector from the reference point to the neighbor points and the original feature values are integrated and convolved. The

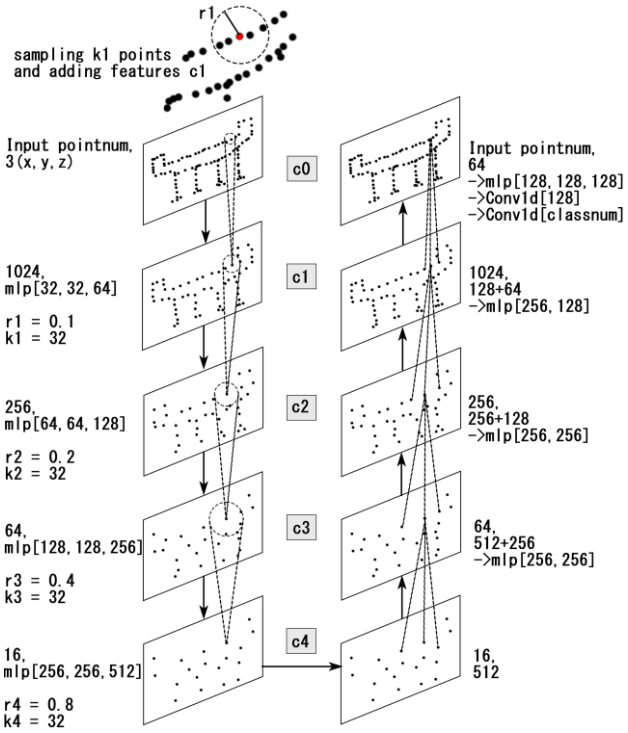


Figure 1. Model structure of PointNet++.

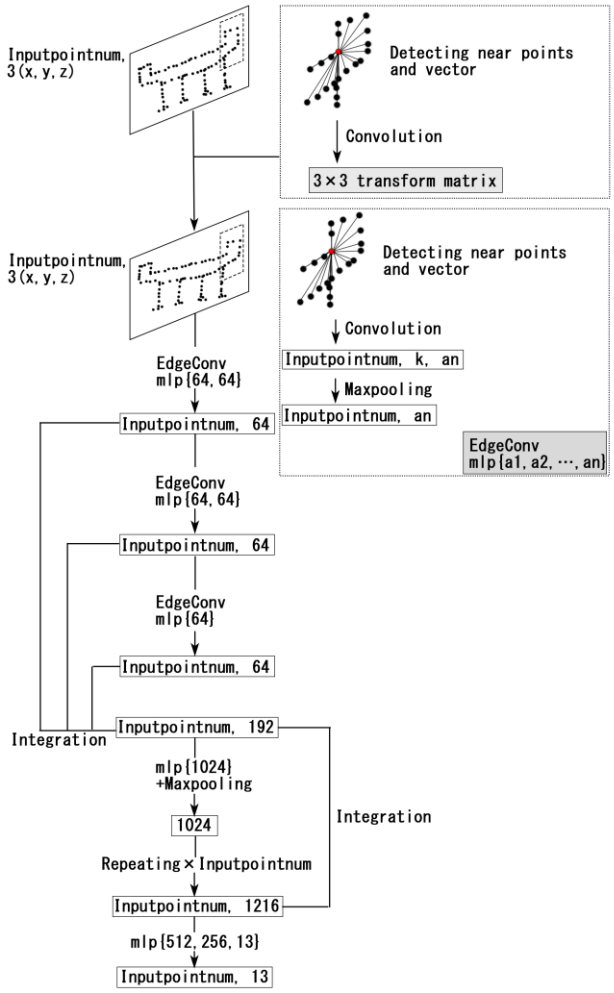


Figure 2. Model structure of DGCNN.

DOI: 10.3217/978-3-99161-057-1-083

number of input points does not change in either layer, but the *xyz* coordinates also change due to convolution. After repeating this process 3 times, the feature values obtained in each step are integrated. Next, MaxPooling is applied, and replicated for the number of input points. Finally, after several convolutions, the number of classified features is obtained and output as a random variable by applying the Logsoftmax function. Although the paper [14] describes a categorical vector that provides labels for each input data as additional information, it is not used in this research.

3 PREPARING DATA FOR DEEP LEARNING

3.1 Case study

A two-span continuous through-type truss bridge in Aichi Prefecture, Japan, is the case study. A photograph of the bridge is shown in Figure 3. The bridge length is 136.9m with a span length of 2@67.9m. A full width is 14.3 m with sidewalks on both sides. The effective width of the roadway is 7.5 m and that of the sidewalk is 2.0 m without widening. A slab thickness is 200 mm with a pavement of 80 mm thickness (roadway) and of 30 mm thickness (sidewalk). In addition, the bridge is straight and has a symmetric cross slope. A general bridge drawing is shown in Figure 4. The members to be segmented are "Upper chord", "Lower chord", "Brace (tensile)", "Brace (compression)", "Main girder", "Cross beam", "Upper lateral bracing", "Lower lateral bracing", "Sway bracing", "Gate", "Handrail", "Mounted components", and "Slab".

3.2 Generating point cloud data from 3D CAD data

The 3D CAD data was manually created based on the drawings of the bridge shown in Section 3.1. Furthermore, for each 3D CAD component, point cloud data is obtained by randomly sampling points on its surface. In this case, noise and missing points are not generated. In addition, a uniform density (1 [pts./cm^2] in this case) was set for all members in order to avoid extreme bias in density. It is also possible to assign to each point the normal vector of the plane from which it was generated as a parameter. The generated point cloud data is shown in Figure 5. Each point in the point cloud data has a label number that corresponds to only one of the 13 types of components mentioned above, and the colors of the points in the figure correspond to the labels of the components (Figure 6). The total number of points was 64,016,718.

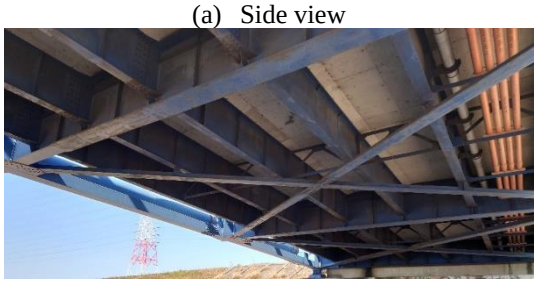
3.3 Creating cross sectional point clouds

The following shows the flow of the method for acquiring cross-sectional point clouds.

First, the line that corresponds to a longitudinal direction is determined. Since the bridge in this research has straight linear and the width is not widened, a line passing through the centroid of the point cloud data and having the direction of the first principal component vector obtained by principal component analysis is defined as the "longitudinal direction line".

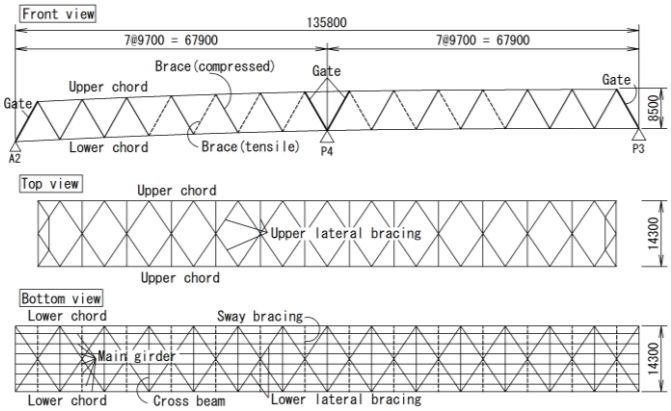
Next, the point **p** is shifted pitch (0.1[m] in this case) from the starting point of the longitudinal direction line, and the plane that contains **p** and is perpendicular to the line along the longitudinal direction line is defined as the "cutting plane". Points within *d* (0.05 [m] in this case) of the cutting plane are detected, and these points are projected onto the cutting plane.

(a) Side view

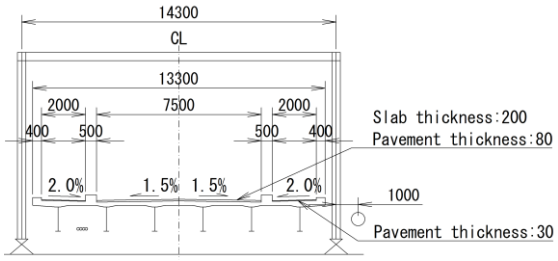


(b) Bottom view

Figure 3. Pictures of the case study bridge.



(a) Front, Top, Bottom view



(b) Cross sectional diagram

Figure 4. General bridge diagram (Unit: mm).

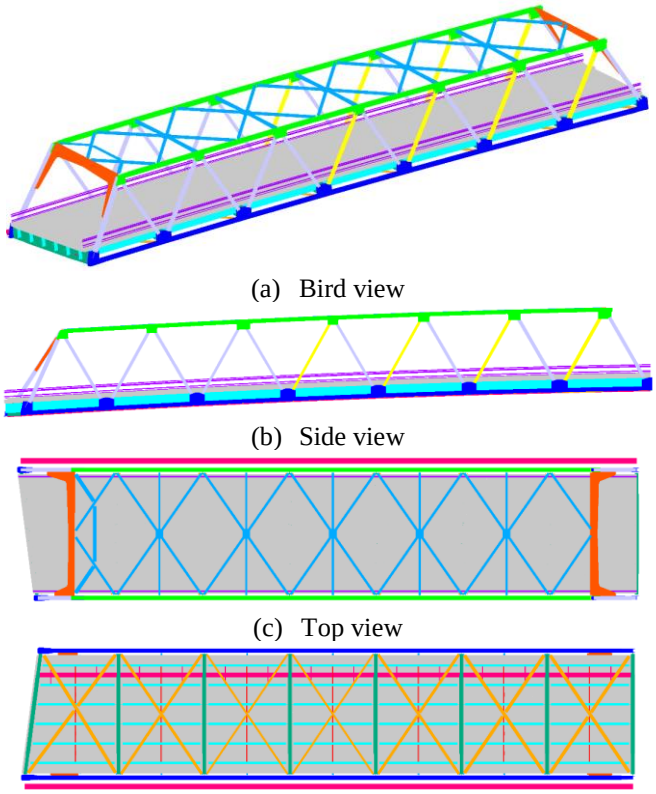
This process is repeated until **p** reaches the end point (Figure 7).

From the point cloud data described in Section 3.2, 695 cross-sectional point clouds were created. Examples are shown in Figure 8. The position of the braces differs depending on the cross section, and those irregular cross sections such as cross beams and sway bracings, are also included.

3.4 Normalization process for deep learning

When training a cross-sectional point cloud, the coordinate system is modified. Define a local coordinate system for the cutting plane as shown in Figure 7. The cutting plane is the *xy*-coordinate plane of the local coordinate system. The world

DOI: 10.3217/978-3-99161-057-1-083



(a) Bird view

(b) Side view

(c) Top view

(d) Bottom view

Figure 5. Point cloud generated from 3D CAD data.

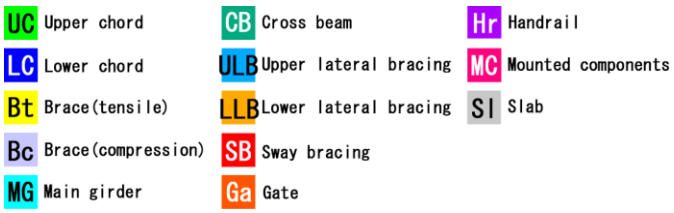


Figure 6. Legend of point colors.

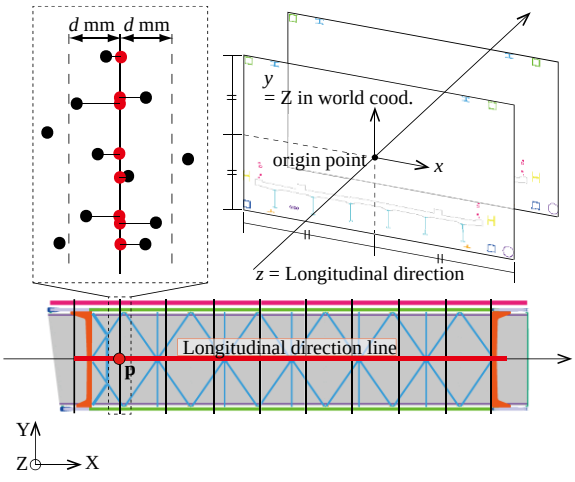
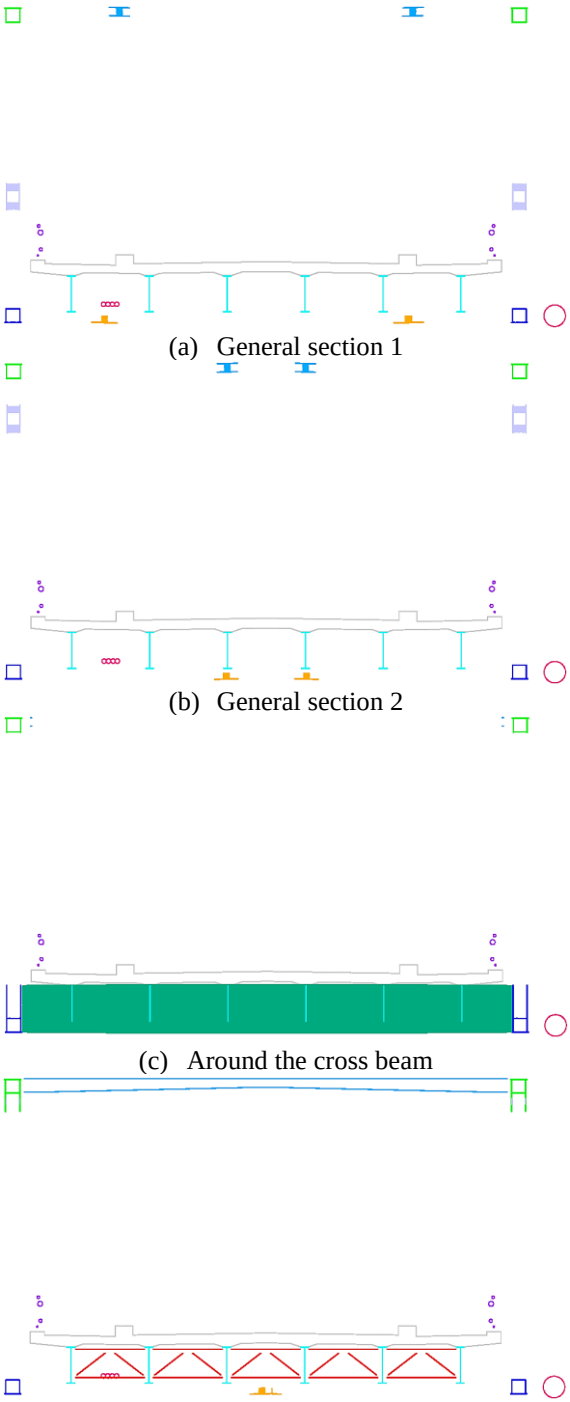


Figure 7. Detecting cross-sectional point clouds.

coordinate Z-axis corresponds to the *y*-axis of the local coordinate system. The origin point is adjusted so that the midpoint between min. and max *x*-coordinates and *y*-coordinates as shown in Figure 7. In order to prevent overlearning, the *x*-coordinate is flipped with respect to the *y*-axis with a probability of 50%.



(a) General section 1

(b) General section 2

(c) Around the cross beam

(d) Around the Sway bracing

Figure 8. Examples of cross-sectional point clouds from 3D CAD data.

4 IMPLEMENTATIONS AND DISCUSSIONS

4.1 Overview

In this chapter, several cases are implemented and discussed using training data described in Chapter 3 and test data obtained

DOI: 10.3217/978-3-99161-057-1-083

from the actual steel truss bridge by the stationary laser scanner. The implementation environment is shown in Table 1, and the common parameters used for verifying multiple cases are listed in Table 2. The number of input points and the batch size must fit in the GPU's memory. If this limit is exceeded, computation is offloaded to the main memory, significantly increasing computation time. In addition, the batch size must be 2 or greater due to the use of BatchNormalize function as explained in Section 2.2. The number of training samples is 10,336, calculated by dividing the total number of points in the cross-sectional point cloud by the number of input points.

The implementation cases are organized in Table 3. During training, validation is included in each epoch. No fixed distinction is made between training and validation data; instead, 8,000 samples are randomly selected from the 10,336 available for each epoch. The Intersection over Union (IoU) for each member in the cases is shown in Figure 9. The values shown represent the results from the epoch in which the average IoU across all members was the highest. Computation times for each case are summarized in Table 4.

4.2 Point cloud for using the implementation

In this section, the measured point cloud data of the actual bridge without member labels is mentioned. It is used for the test of the trained model. The point cloud data (1,065,353,413 points, Figure 10) measured with a stationary laser scanner Leica RTC360 (resolution: 3mm@10m, accuracy: 1.9mm@10m) from 27 locations on the underpass and road surface of one span of the bridge described in Section 3.1. 5,000 cross-sectional point clouds are created as the validation data by slicing at regular intervals (0.02 [m] in this case) along the longitudinal direction. Note that although this point cloud data contains color information, it is not used because there is no color information in the training data. Normals were obtained by calculating them with the Point Cloud Library [15], a point cloud data processing library. The normalization described in Section 3.4 is also applied to this cross-sectional point cloud.

4.3 Comparing PointNet++ or DGCNN

First, the effectiveness of PointNet++ and DGCNN in segmenting point cloud data of cross sections sliced along the longitudinal direction is evaluated. The construction of model is shown in Figure 1 and Figure 2. As shown in the results of the application to the point cloud data presented in Section 4.2 (Figure 11 and Figure 12), PointNet++ was more accurate in the test. Although DGCNN resulted in a higher IoU and faster computation time during training, there were many places where other members were misidentified as the main girders in the test. PointNet++ was able to distinguish between two types of cross-sections of the braces with high accuracy, although in some cases the slabs were misidentified as the cross beams in areas where the data quality was low due to limitations of the measurement environment. This is likely due to the fact that DGCNN have over-trained the training data. In addition, it seems that PointNet++ is more versatile in handling 2D point cloud data with image-like features, as its behavior more closely resembles the convolution process used in image-based deep learning. For generality, PointNet++ is used in the subsequent validations, although it takes more time.

Table 1. Development Environment.

Common	
CPU	Intel(R) Xeon(R) Silver 4214R CPU @ 2.40GHz 2.39 GHz (2 processors)
Memory	224GB
GPU	NVIDIA GeForce RTX 3080 (10GB)
OS	Windows 11 Enterprise 24H2 64bit

Slicing point cloud	
Platform	Microsoft Visual Studio Community 2022 64bit
Library	Point Cloud Library (PCL) 1.12.0 64bit [15]
Language	C++

Deep learning	
Platform	Microsoft Visual Studio Code
Library	Pytorch 2.5.1 cuda 12.1
Language	Python 3.10.5

Table 2. Common parameters of deep learning.

Num. of input points	6164
Batch size	8
Num. of training data	10336
Num. of epoch	32
Optimization function	Adam
Loss function	Cross Entropy Loss

Table 3. Implementation cases.

Sec.	Model	Sampling	Scale Norm.	Normal vector	Weights for loss
4.3	*	sample	Yes	No	All 1
4.4	PointNet++	*	Yes	No	All 1
4.5	PointNet++	sample	*	No	All 1
4.6	PointNet++	sample	Yes	*	All 1
4.7	PointNet++	sample	Yes	No	*

* is the comparing topic

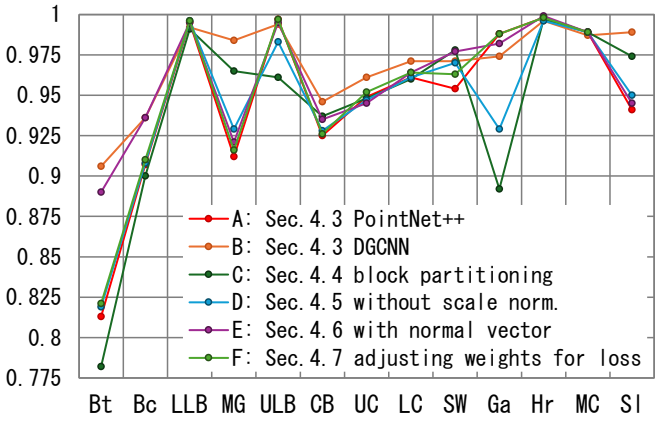


Figure 9. IoU of training in each case.

Table 4. Computation time of each case.

	Train	Test		Train	Test
A	18h.	6h.	D	16h.	9h.
B	9h.	4.5h.	E	15.5h.	11h.
C	16h.	36h.	F	15.5h.	9h.

DOI: 10.3217/978-3-99161-057-1-083

4.4 Comparing sampling methods as input data

The number of input points is 6144, but the number of cross-sectional point clouds is approximately 50,000 to 100,000 points. Therefore, sampling is required. Here, two types of sampling "block partitioning" and "random sampling" are selected and compared the accuracy. For block partitioning, one point is randomly selected from the cross-sectional point cloud, from which points within a 5 m block around it are extracted and sampled so that the total number of points is 6144. In addition, the coordinates that origin is at the center of the block are calculated and added as new features. Random sampling was performed by randomly selecting 6144 points from the entire cross-sectional point cloud and *x* and *y* coordinates are adjusted so that a distance to the farthest point from the origin point is 1. As shown in the test results (Figure 13), block partitioning is able to detect the slabs without misidentifying them even in areas where the quality of the measurement data is low, but the accuracy of other members such as the braces and the cross beams is low.

4.5 Comparing presence or absence of scale normalization

According to Section 4.4, since block partitioning without scale normalization has a higher detection rate of the slabs, the additional case without scale normalization with random sampling is implemented. As shown in the test results (Figure 14), the accuracy of the slab segmentation was improved, but the accuracy of the braces was significantly reduced. The second span, where segmentation accuracy is low, was not originally intended to be measured. However, it was partially captured during scanning of the adjacent first span. As a result, the point cloud data is of poor quality, with low density and many missing points. Although it is necessary to develop a learning model that can be applied to point cloud data of low measurement quality as a future challenge, this paper concludes that random sampling and scale normalization are effective for segmentation accuracy in high measurement quality areas.

4.6 Comparing presence or absence of normal vector

In the previous explanations, only *xyz* coordinates were used for the input point cloud data, but the case with additional normal vector was also verified. As shown in the test results (Figure 15), the reason for the poor results in the case where normals were added is that, as shown in Figure 16, normals were generated even where the laser scanner would not have been irradiated if generated from 3DCAD, which may have caused a discrepancy between the training data and the test data. Although it is useful to develop a sampling tool that simulates a laser scanner, it is more effective to create a learning model that does not use normal vector, considering the efficiency of training data generation.

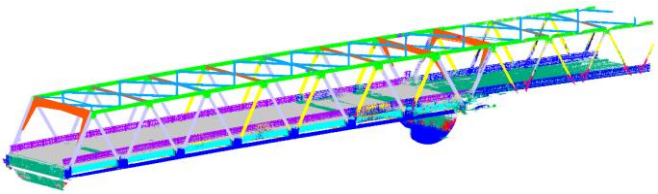
4.7 Adjusting weights for loss calculation

The equation for the CrossEntropyLoss function is shown below:

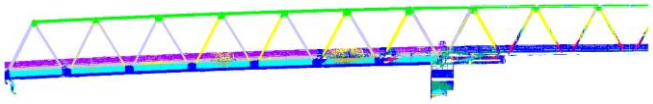
$$Loss = \sum_{n=1}^{N} -w_{yn} x_{n,y_n} \quad (1)$$

The loss is calculated each member and they are sum up. N is the number of points and x_{n,y_n} is a random variable in the output data. In the previous cases, w_{yn} was set to 1 for all members, but in this case, it is adjusted for each member. This parameter is used to prevent bias in the accuracy of the classification depending on the size of the member that is

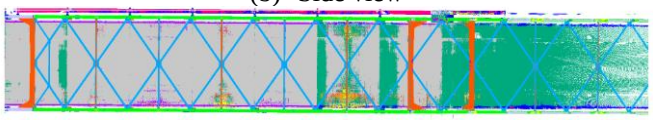
Figure 10. The point cloud data measured with a stationary laser scanner



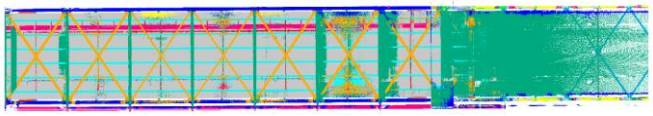
(a) Bird view



(b) Side view

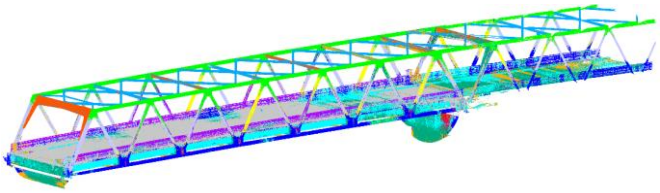


(c) Top view

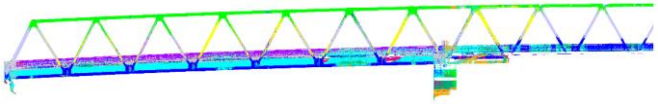


(d) Bottom view

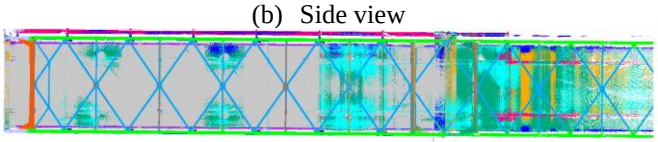
Figure 11. Result of case A: PointNet++.



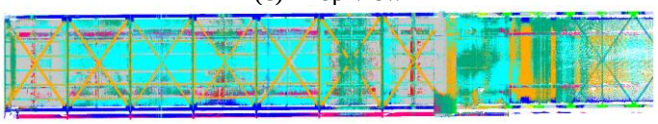
(a) Bird view



(b) Side view



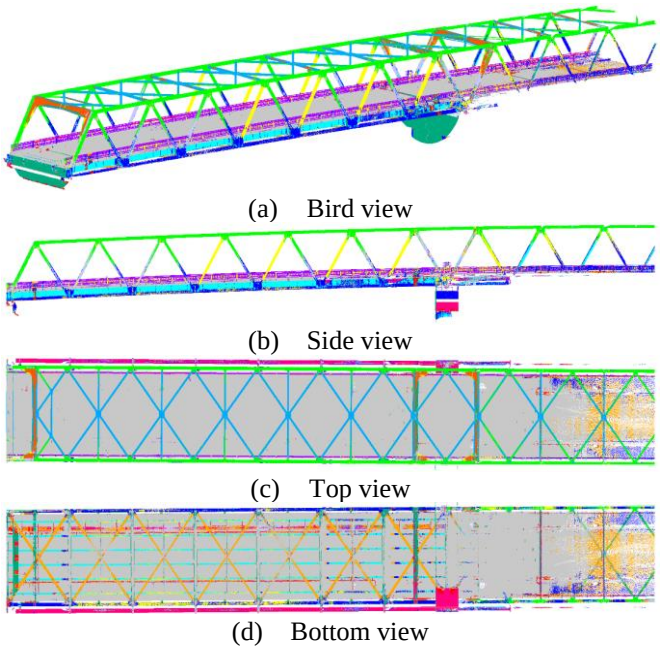
(c) Top view



(d) Bottom view

Figure 12. Result of case B: DGCNN.

DOI: 10.3217/978-3-99161-057-1-083



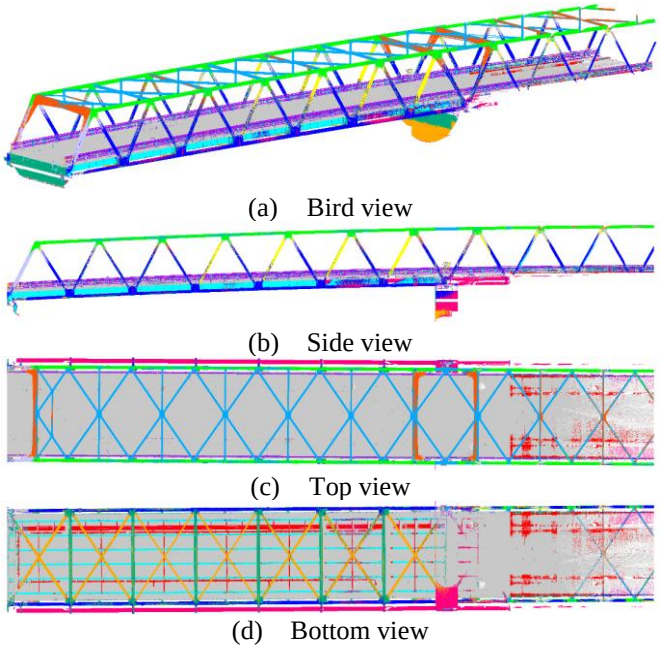
(a) Bird view

(b) Side view

(c) Top view

(d) Bottom view

Figure 13. Result of case C: block partitioning.



(a) Bird view

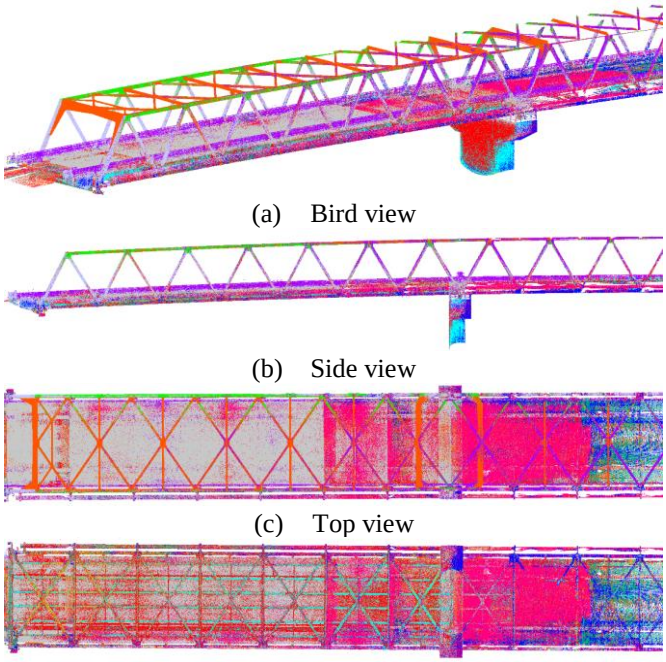
(b) Side view

(c) Top view

(d) Bottom view

Figure 14. Result of case D: without scale normalization.

number of points. So, in this case, w_{yn} is set inversely proportional to the number of points in each member (Table 5). As shown in the test results (Figure 17), the outcomes were generally similar to the case in which all weights were set to 1. However, the segmentation accuracy for the braces was slightly lower. This is believed to be due to the increased weight assigned to members that appear in only a small number of cross-sectional point clouds, such as the gates and the sway bracings, which in turn reduces the accuracy for other members that appear in most cross-sectional point clouds. Therefore, it is more effective to set all weights to 1 without adjusting the weights between members.



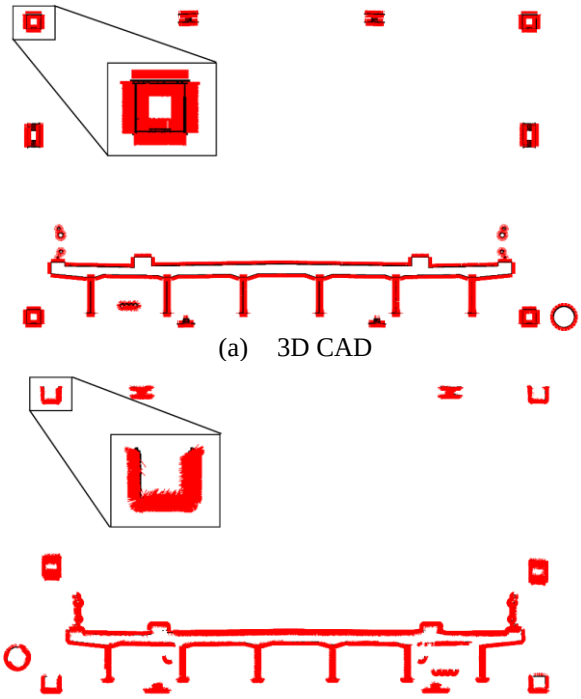
(a) Bird view

(b) Side view

(c) Top view

(d) Bottom view

Figure 15. Result of case E: Using normals.



(a) 3D CAD

(b) Stational laser scanner

Figure 16. Normal vectors of the point clouds.

5 CONCLUTION

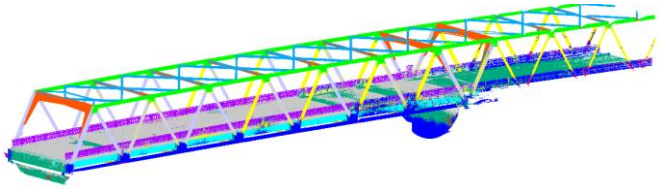
In this research, as a method for segmenting point cloud data of bridges into members using deep learning, using point cloud data of 2D cross sections sliced along the longitudinal direction. Several patterns are proposed and compared for validation to identify a highly accurate segmentation method.

Through the implementation of several cases, it was confirmed that the following settings were effective when point cloud data obtained by pseudo-sampling the surface of a 3D

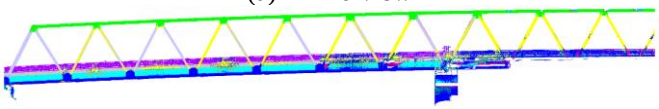
DOI: 10.3217/978-3-99161-057-1-083

Table 5. The weights inversely proportional to points.

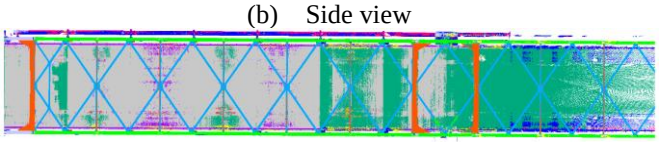
UC	1.76	MG	1.19	SW	2.99	Sl	1.00
LC	1.55	CB	1.60	Ga	2.51		
Bt	2.45	ULB	1.70	Hr	2.37		
Bc	1.63	LLB	2.10	MC	1.94		



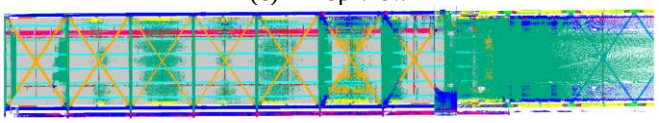
(a) Bird view



(b) Side view



(c) Top view



(d) Bottom view

Figure 17. Result of case F: adjusting weights for loss.

model was used as training data and point cloud data obtained by measuring the same bridge with a laser scanner was used as test data:

- The learning model type PointNet++ is expected to be more versatile because of its higher segmentation accuracy of the test data.
- When the number of input points was reduced to about 5-10% of the total number of points, random sampling was more accurate and faster.
- To increase versatility, it was more effective to apply scale normalization at the time of training model input.
- Normal vectors can cause discrepancies between training data and test data depending on the calculation method, so care must be taken when generating them.
- Weight adjustments during the calculation of the loss function for each member did not have a significant effect.

As future work, training on multiple types of bridges (not limited to steel truss bridges) and performing segmentation across various bridge types is necessary, since the training and test data in this research are from the same bridge, although generated by different methods. In addition, since the bridges covered in this paper have straight liner and no width widening, it is desirable to conduct verification on bridges with curved liners or varying widths.

ACKNOWLEDGMENTS

This research was supported by the Foundation of public interest of Tatematsu (Nao Hidaka).

REFERENCES

[1] M. E. Mabsout, K. M. Tarhini, G. R. Frederick, and C. Tayar, Finite-Element Analysis of Steel Girder Highway Bridges, *Journal of Bridge Engineering*, Vol. 2, No. 3, pp. 83-87, 1997.

[2] T. Yamane, P. Chun, J. Dang, and R. Honda, Recording of bridge damage areas by 3D integration of multiple images and reduction of the variability in detected results, *Computer-Aided Civil and Infrastructure Engineering*, Vol. 38, No. 17, pp. 2391-2407, 2023.

[3] G. Qin, Y. Zhou, K. Hu, D. Han, and C. Ying, Automated Reconstruction of Parametric BIM for Bridge Based on Terrestrial Laser Scanning Data, *Advances in Civil Engineering*, Vol. 2021, No. 1, pp. 8899323, 2021.

[4] Y. Schatz and B. Domer, Semi-Automated Creation of IFC Bridge Models from Point Clouds for Maintenance Applications, *Frontiers in Built Environment*, Vol. 10, 2024.

[5] N. Hidaka, N. Hashimoto, K. Magoshi, T. Nonaka, M. Obata, and E. Watanabe, Construction of a Practical Finite Element Model from Point Cloud Data for an Existing Steel Truss Bridge, *Proceedings of the 23th International Conference on Construction Applications of Virtual Reality (ConVR 2023)*, pp. 1155-1166, 2023.

[6] T. Xia, J. Yang, and L. Chen, Automated Semantic Segmentation of Bridge Point Cloud Based on Local Descriptor and Machine Learning, *Automation in Construction*, Vol. 133, pp. 103992, 2022.

[7] R. Pierdicca, M. Paolanti, F. Matrone, M. Martini, C. Morbidoni, E. S. Malinverni, E. Frontoni, and A. M. Lingua, Point Cloud Semantic Segmentation Using a Deep Learning Framework for Cultural Heritage, *Remote Sensing*, Vol. 12, No. 6, pp.1005, 2020.

[8] O. Ronneberger, P. Fischer, and T. Brox, U-Net: Convolutional Networks for Biomedical Image Segmentation, *Medical Image Computing and Computer-Assisted Intervention (MICCAI 2015)*, pp.234-241, 2015.

[9] T.Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, Feature Pyramid Networks for Object Detection, *Computer Vision and Pattern Recognition*, pp. 2117-2125, 2017.

[10] L. C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation, *Computer Vision and Pattern Recognition*, pp. 833-851, 2018.

[11] N. Saovana, N. Yabuki, and T. Fukuda, Automated Point Cloud Classification Using an Image-Based Instance Segmentation for Structure from Motion, *Automation in Construction*, Vol. 129, pp. 103804, 2021.

[12] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation, *Computer Vision and Pattern Recognition*, pp. 652-660, 2017.

[13] C. R. Qi, L. Yi, H. Su, K. Mo, and L. J. Guibas. Guibas. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space, *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 5105-5114, 2017.

[14] A. V. Phan, M. L. Nguyen, Y. L. H. Nguyen, and L. T. Bui, DGCNN: A convolutional neural network over large-scale labeled graphs, *Neural Networks*, Vol. 108, pp. 533-543, 2018.

[15] R. B. Rusu and S. Cousins, 3D Is Here: Point Cloud Library (PCL), *In 2011 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1-4, 2011.