



# Bridge point cloud semantic segmentation based on view consensus and cross-view self-prompt fusion

Yan Zeng<sup>a,b</sup>, Feng Huang<sup>c,d</sup>, Guikai Xiong<sup>a,e</sup>, Xiaoxiao Ma<sup>a,b,\*</sup>, Yingchuan Peng<sup>c,d</sup>,  
Wenshu Yang<sup>c,d</sup>, Jiepeng Liu<sup>a,b</sup>

<sup>a</sup> School of Civil Engineering, Chongqing University, Chongqing 400045, China

<sup>b</sup> Key Laboratory of New Technology for Construction of Cities in Mountain Area (Chongqing University), Ministry of Education, Chongqing 400045, China

<sup>c</sup> Sixth Engineering Branch of China Railway Major Bridge Engineering Group Company Limited, Wuhan 430101, China

<sup>d</sup> Fengcheng Branch of China Railway Major Bridge Engineering Group Company Limited, Fengcheng 331104, China

<sup>e</sup> Chongqing Academy of Surveying and Mapping, Chongqing 401121, China

## ARTICLE INFO

### Keywords:

Bridge  
Point cloud semantic segmentation  
SegGPT  
View consensus  
Prompt fusion

## ABSTRACT

Point cloud semantic segmentation has been widely applied for bridge inverse modeling. However, existing methods are either labor-intensive or exhibit poor generality for real-world bridges. To address these limitations, this paper presents a bridge semantic segmentation method based on a pre-trained visual model. A viewpoint selection method based on view consensus is proposed to evaluate and optimize the viewpoints. The key insight is ensuring two adjacent viewpoints share a substantial consensus with high component visibility. The proposed self-prompt augmentation strategy enhances the performance of the multi-view image segmentation by fusing the initial prompt with cross view 2D masks. Bridge components are extracted through hard voting and further refined via post-processing. Experimental results demonstrate our method achieves state-of-the-art performance on real-world bridges. It provides reliable semantic information of bridge point cloud data for bridge inspection and maintenance applications.

## 1. Introduction

Bridge infrastructure is a vital element of civil engineering, essential for modern transportation and economic activities. However, the deterioration due to environmental factors and heavy loads reduces the serviceability and reliability of bridges and further may jeopardize public safety. This necessitates the development of effective bridge health monitoring (BHM) and regular maintenance. Traditional techniques rely on periodic visual inspections by trained inspectors, which are subjective, labor-intensive, and prone to errors. To overcome these limitations, recent approaches [1–5] have focused on generating Bridge Information Model (BrIM) from 3D point cloud data (PCD) obtained by 3D reality capturing techniques, such as laser scanning, photogrammetry, and unmanned aerial vehicles [6–8]. BrIM is a comprehensive representation of bridge assets [9–12], serving as an invaluable resource for maintenance throughout the bridge's lifecycle.

However, the transition from PCD to BrIM remains challenging, particularly due to the labor-intensive process of point cloud semantic segmentation (PCSS). Identifying and labeling bridge components is

time-consuming, limiting its scalability [13–16]. Various methods have been proposed to address this challenge, yielding significant performance improvements. However, these methods face additional obstacles in real-world applications. For instance, due to the lack of comprehensive bridge PCSS datasets, many previous methods [1,17–19] are tailored to certain bridge types. Additionally, some approaches [20–24] require extensive labeling works or intricate parameter adjustment. Although, efforts [25–28] to generate synthetic point clouds have enriched the training data, these methods still demand substantial manual effort.

Recently, large visual models (LVMs) have revolutionized the computer vision (CV) community. These models, pre-trained on massive datasets, enable a wide range of downstream tasks through zero-shot transfer and prompt tuning [29–34]. While LVMs have achieved remarkable success in 2D domains, their application in 3D PCSS remains largely unexplored. Efforts have been made to adapt 2D models for 3D tasks [35–39]. A common approach involves generating viewpoints around the target object and employing well-designed pipelines with supervised processes to leverage multi-view images. However, the

\* Corresponding author at: School of Civil Engineering, Chongqing University, Chongqing 400045, China.

E-mail address: [Maxx@cqu.edu.cn](mailto:Maxx@cqu.edu.cn) (X. Ma).

<https://doi.org/10.1016/j.autcon.2025.106003>

Received 26 September 2024; Received in revised form 10 January 2025; Accepted 20 January 2025

Available online 29 January 2025

0926-5805/© 2025 Elsevier B.V. All rights are reserved, including those for text and data mining, AI training, and similar technologies.

inherent modal gap between 2D and 3D data poses significant challenges, and studies focusing on bridge PCSS using multi-view learning remain scarce.

Two primary issues arise in this context: viewpoints evaluation and viewpoints information utilization. First, not all viewpoints contribute equally to bridge PCSS. Specifically, only component-related viewpoints are critical for accurate segmentation. Motivated by this, we propose a component-aware viewpoint evaluation method to identify and select the most informative views. Second, integrating information from different viewpoints is essential for improving 3D segmentation performance. Inspired by the observation that adjacent viewpoints often share substantial visual consensus, we combine information from multiple perspectives. This integration ensures that the segmentation process fully leverages the strengths of 2D models in a multi-view context.

In this paper, we present a novel bridge segmentation technique leveraging the capabilities of 2D LVMs. For viewpoint evaluation, the viewpoints for bridge PCD multi-view projection are generated redundantly. To eliminate irrelevant viewpoints, a heuristic viewpoint selection method based on view consensus is proposed. The key lies in this method is the information evaluation for each viewpoint, which involves two critical steps: visible points extraction and visible information evaluation. Instead of directly searching in 3D points, we check visible regions in voxel space and evaluate the visible information by view consensus rate. Viewpoints with reasonable consensus are then prioritized and selected. To further integrate information across different viewpoints, we designed an efficient prompt-augmentation strategy using cross-view self-prompt fusion. This strategy incorporates the inference mask from the previous image as a self-prompt and fuses it with a reliable initial prompt, enhancing segmentation performance of the 2D LVM. By employing hard voting and post-processing, the bridge PCSS results are obtained under zero-shot learning.

Our work achieves bridge PCSS for not only one type of bridge with no need for training data. The main contributions of this paper are as follows:

- (1) Zero-shot bridge PCSS: We introduce a method that achieves zero-shot 3D bridge PCSS based on SegGPT. The proposed PCSS method demonstrates high generality across different bridge types, achieving significant quantitative results on beam and suspension bridges.
- (2) Viewpoint selection method based on view consensus for multi-view projection: We propose a viewpoint selection method based on visible information inherent in each viewpoint and the variances in information between neighboring viewpoints for multi-view projection.
- (3) Cross-view self-prompt fusion for 2D inference: A novel prompting scheme is designed to generate augmented prompts for multi-view images both in terms of quantity and quality.

This paper is structured as follows: [Section 2](#) reviews the state-of-the-art methods. [Section 3](#) details the methodology, where [Section 3.1](#) gives an overview of the whole method, [Section 3.2](#) elaborates on the viewpoint selection for bridge PCD multi-view projection, [Section 3.3](#) explains the implementation of cross-view self-prompting scheme for 2D inference by SegGPT, and [Section 3.4](#) outlines the process of coarse segmentation by hard voting mechanism and fine segmentation by post-processing. [Section 4](#) validates the proposed bridge PCSS method on 2 different types of bridge PCD. [Section 4.2](#) presents the experimental results in quantitative and qualitative. [Section 4.3](#) exhibits validation experiments for three key processes of the proposed method. And

[Section 4.4](#) provides the results of stress test on sparse data. [Section 5](#) discusses the findings and outlines future research directions. Finally, [Section 6](#) draws on the highlights of the research.

## 2. Related works

This section shows the state-of-the-art works related to the 3D bridge PCSS and 2D LVMs.

### 2.1. 3D bridge PCSS

Numerous studies have focused on developing methodologies and algorithms to detect and recognize components in the scene of bridge. Broadly, existing bridge PCSS methods can be categorized into unsupervised and supervised approaches.

Unsupervised methods utilize PCD processing algorithms and contextual information. These include region-growing algorithms [40], clustering algorithms [41], optimization-based methods [42], and heuristic-based methods [1,2,17–19]. However, these methods face significant challenges. They often rely on local features, which are frequently similar across different bridge components, making segmentation difficult. Moreover, in outdoor bridge scenes with large data volumes and noisy environmental conditions, these methods suffer from high computational costs and are sensitive to noise and outliers. Furthermore, these methods are limited to specific bridge types, resulting in poor generality.

Supervised methods utilize labeled data to train models for different segmentation tasks. Notable models include PointNet [43] and PointNet++ [44]. These deep learning-based methods have spurred significant interest in bridge PCSS. Kim et al. [20,21] employed PointNet to segment piers and decks, which is applicable to curved or inclined bridges with components in different shapes. However, to preserve spatial resolution within limited memory, PCD is partitioned into small subspaces for training, resulting in the loss of global features. Lee et al. [22] used a graph-based hierarchical DGCNN (HGCNN) model to improve boundary detection accuracy and increase the intersection over union (IoU) of electric poles segmentation. Nonetheless, due to insufficient training data, this method neglects the segmentation of girders, pier caps, diaphragms and parapets.

In recent years, some scholars have explored various methods to address the issue of insufficient training data. Xia et al. [23] combined local feature descriptors with machine learning to achieve satisfactory segmentation of bridge components with minimal sample learning. However, these local feature descriptors rely on the computation of point cloud normal, which are sensitive to point cloud density. Other scholars have proposed synthesizing bridge PCD to enrich samples. Jing et al. [24] created an arch bridge dataset to train BridgeNet, demonstrating the utility of synthetic training data and the segmentation capability of the new network. Yang et al. [25] used both real bridge PCD and synthesized PCD for training, achieving component segmentation of large-scale bridge PCD. Lamas et al. [27] proposed an automatic method for synthesizing truss bridge PCD and achieved both semantic and instance segmentation of truss bridge based on the synthesized dataset. While such methods obviate the need for manual collection of real bridge PCD and enrich training samples safely, they require a huge demand for forward modeling and involve laborious manual annotation work. To alleviate the burden of manual annotation, Martens et al. [45] proposed a cross-domain segmentation approach employing artificial neural networks. This method performs semantic segmentation in the image domain and then transfers the results to the PCD domain. However, constrained by image resolution and camera

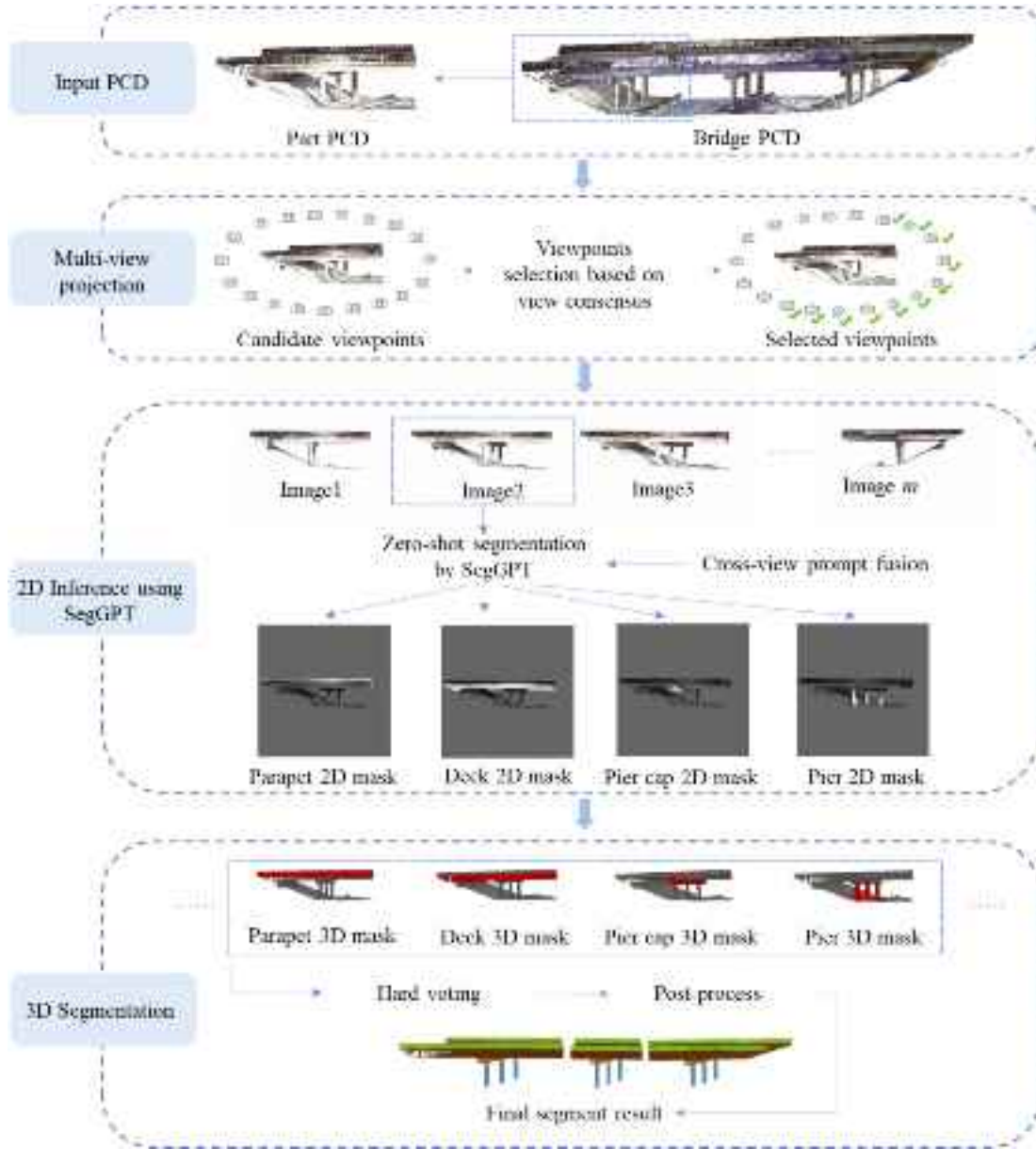


Fig. 1. Overall pipeline of the proposed method.

viewpoints, this method may suffer from contextual information loss, leading to suboptimal segmentation outcomes.

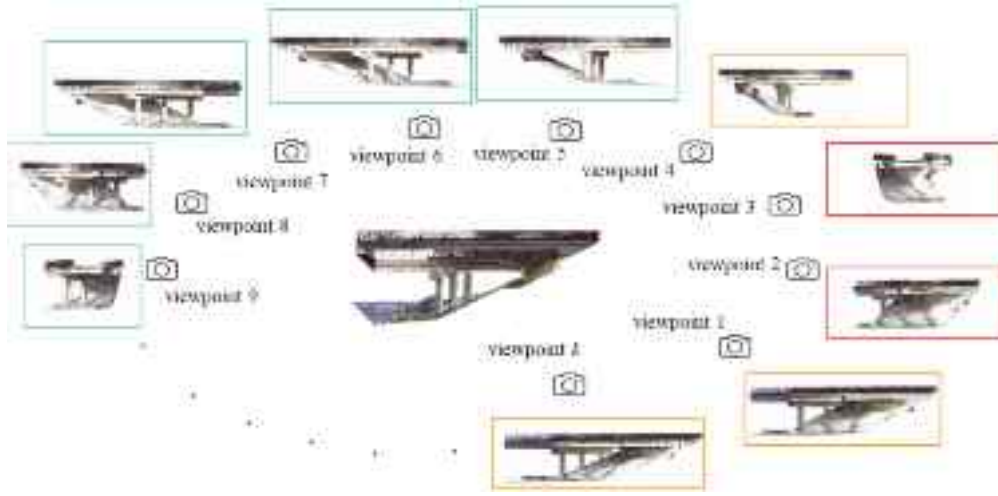
Although the progress of supervised methods brings new impetus to the development of bridge PCSS, the scarcity of labeled data remains a significant challenge. Moreover, there is currently no evidence to ascertain whether algorithms trained on a specific bridge scene can be applied to novel bridge scenes with different component types. Hence, overcoming the challenge of inadequate training data to achieve PCSS of multi-type bridges stands as a crucial research direction in the realm of PCSS.

## 2.2. 2D LVMs

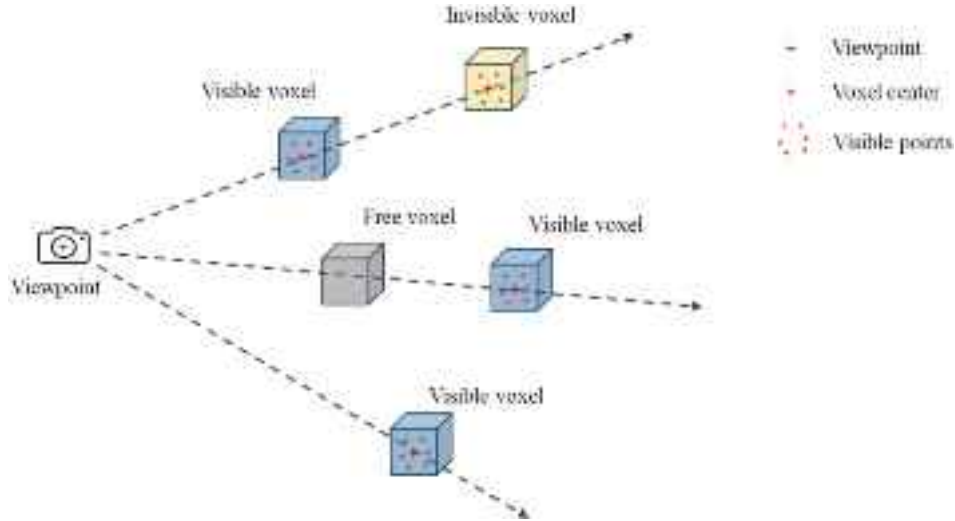
Foundation models have revolutionized artificial intelligence (AI) in recent years. These models are pre-trained on web-scale datasets, enabling powerful zero-shot generalization across a wide range of downstream tasks. Inspired by the greatest success of foundation models

in the natural language (NLP), the CV community has undergone a significant shift towards the development of the LVMs. Prominent examples include SAM [31], OneFormer [32], SEEM [33], and SegGPT [34]. These models are pretrained on extensive datasets to function as task-agnostic models by prompt tuning. One of the standout models, SegGPT, is designed to segment various entities within a given context by integrating different segmentation tasks into a unified in-context learning framework. SegGPT achieves this by framing its training process as a context coloring problem using a random coloring strategy. During training, the model is compelled to reference contextual information to accomplish assigned tasks, rather than relying on specific colors. This approach allows SegGPT to perform arbitrary segmentation tasks through in-context inference. Additionally, the predicted image results can serve as plug-and-play keys for specific applications. By specifying context-specific prompts without compromising the model's generality, SegGPT opens the possibilities for various applications.

Given its success in the 2D realm, LVMs have naturally expanded to



**Fig. 2.** Different images from different viewpoints of bridge PCD. The viewpoints within the red frame are occluded viewpoints; the viewpoints within the yellow frame are transitional viewpoints with partial occlusion; and the viewpoints within the green box are unobstructed viewpoints. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)



**Fig. 3.** Illustration of visible voxels and visible points.

the 3D vision domain. Recently, many researchers are dedicated to adding knowledge of additional modalities to widespread the LVMs application such as lifting 2D foundation models to 3D. A key challenge of this emission is the cross-modal reversible representation between 3D PCD and 2D images. A common approach for PCD involves rendering 2D multi-view images using multiple specific cameras poses and then mapping the features extracted from the 2D base models back to the 3D PCD. Bradski et al. [46] initially proposed the idea of using 2D images to recognize the 3D objects, and Su et al. [47] combined it with deep learning for 3D understanding in MVCNN. Such multi-view methods also present impressive performance in 3D shape classification [48,49], shape retrieval [50] and segmentation [51–53]. Notably, many works render several views of 3D objects and feed these images into a pre-trained 2D network for 3D vision tasks. Leveraging a pre-trained network can compensate for the general scarcity of labeled 3D datasets. ParSLIP [34] achieves zero-shot 3D part segmentation by leveraging pretrained image-language models. It renders the PCD from  $K$  predefined uniformly spaced camera poses through simple rasterization. PointCLIP [36] encodes PCD by projecting it into multi-view depth maps

and aggregates the view-wise zero-shot predictions to achieve knowledge transfer from 2D to 3D. PointCLIP V2 [37] improves visual encoder by a realistic shape projection module to generate more realistic depth maps.

All these previous methods rely on fixed rendered viewpoints of the 3D objects, only aiming to cover the object without optimization discussion regarding the camera positions. However, this often leads to occlusion when dealing with complex structures like bridges. To address this issue, our study proposes a novel viewpoint selection method before multi-view projection. This approach effectively discards the occluded viewpoints, resulting in higher quality multi-view images.

### 3. Methods

In this section, we introduce the overall framework. And then, we demonstrate the design of each module in detail.

### 3.1. Overall pipeline

Figure 1 depicts our overall pipeline, which comprises three main parts: multi-view projection, 2D inference with SegGPT and 3D segmentation. In the multi-view projection, we propose a viewpoint selection method based on view consensus, allowing us to render colour images that clearly capture various components. These images are then fed into SegGPT, where a cross-view self-prompt fusion mechanism is designed to enhance the 2D inference performance. During the 3D segmentation, we back-project 2D inference results into 3D space and obtain the coarse segmentation results by hard voting and a post-processing method is employed to refine the segmentation results. The following sections provide a detailed explanation of the viewpoint selection based on view consensus, 2D inference with cross-view self-prompt fusion and 3D segmentation.

### 3.2. Viewpoint selection based on view consensus

In multi-view projection, we aim to render  $m$  images from different viewpoints of the bridge. Typically, viewpoints setup is either circular [48] or spherical [50]. In this paper, candidate viewpoints are uniformly distributed around the bridge PCD, considering the structural characteristics of the bridge. However, due to interference from irrelevant information, the visual information provided by each viewpoint is not equally important for the segmentation task. As illustrated in Fig. 2, images marked by the red frame are obstructed by retaining walls and fail to provide any component information, merely increasing computational burden. In contrast, images within the green frame clearly capture the components from different angles, significantly aiding in comprehensive segmentation. Meanwhile, the yellow-framed images are from transitional viewpoints with partial obstruction, providing some useful information but potentially leading to confusion. Therefore, a good selection strategy should prioritize those viewpoints that can gain images like those marked by the green frame, which clearly capture the components without obstruction by irrelevant interference and maintain substantial consensus with adjacent viewpoints. Towards this end, this paper proposes a viewpoint selection method based on view consensus including two steps: visible points extraction and visible information evaluation.

#### 3.2.1. Visible points extraction

Inspired by ray tracing, visible points extraction is completed indirectly by identifying visible voxels (Fig. 3). The whole process includes: voxelization, visible voxels check, and visible points extraction.

**3.2.1.1. Voxelization.** We first partition the PCD  $\mathbf{P}$  into a spatial voxel grid  $G$  with voxel size  $l$ . Each voxel  $v_i$  has one index  $I_{v_i} = (ix, iy, iz)$ , which denotes the index of the voxel along the x-axis, y-axis and z-axis. The occupancy status  $O = \{o_i, i = 1, \dots, N_v\}$  of these voxels are also recorded.  $o_i \in \{0, 1\}$ , which is a binary value to define whether a voxel contains points. For each voxel  $v_j$  of these occupied voxels  $V_{occupied} = \{v_j | o_j = 1\}$ , we compute its voxel center  $v_j = (v_{jx}, v_{jy}, v_{jz})$  and record the points in the voxel  $v_j = \{p_k | p_k \in \mathbf{P} \text{ and } p_k \text{ in } v_j\}$ .

**3.2.1.2. Visible voxels check.** To check the visibility of the voxels, we follow the efficient voxel traverse algorithm [54] to get voxels in the path between each viewpoint and each occupied voxel. Since the viewpoints are located outside the voxel grid  $G$ , our path starts from the voxel center towards each viewpoint. Given the voxel center  $v_j = (v_{jx}, v_{jy}, v_{jz})$  and the viewpoint  $u_i = (u_{ix}, u_{iy}, u_{iz})$ , the direction of the path can be easily calculated as  $v_j - u_i$ . Then, the path can be discretized into

intervals of  $t$ , each of which spans one voxel. The equation of the path is formulated as follows:

$$r_{ji} = v_j + t(v_j - u_i), t \geq 0 \quad (1)$$

We start with the occupied voxel and record the voxels visited in the path. In detailed, we calculate the time  $T_x$  at which the path visits next voxel along the x-axis, while  $T_y$  and  $T_z$  represent the times that the path visits next voxel the y-axis and z-axis, respectively. The minimum of these three values will determine the path crosses which vertical voxel boundary or remains in the current voxel in the next step. We use  $\Delta t_x = l / (v_{jx} - u_{ix})$  to update the  $T_x$ , which represents the time interval for the path to travel entire length of one voxel along x-axis. Similarly,  $\Delta t_y = l / (v_{jy} - u_{iy})$  represents the time it takes to walk the width of a voxel along the y-axis and  $\Delta t_z = l / (v_{jz} - u_{iz})$  represents the time it takes to move the height of a voxel along the z-axis. We take the index  $I_{v_j} = (jx, jy, jz)$  to present the visited voxel  $v_j$ , and use the variables  $\Delta x, \Delta y$  and  $\Delta z$ , initialized to 1 or -1, to indicate whether  $jx, jy$  and  $jz$  increase or decrease along the path.

Through this procession, we obtain all the voxels traversed along the path from the voxel  $v_j$  to viewpoint  $u_i$ . And then, using the occupancy status  $O$  mentioned before, we can count the number of occupied voxels along the path. If there are any other occupied voxel stands in the path, the voxel  $v_j$  is considered invisible from the viewpoint  $u_i$ .

**3.2.1.3. Visible points extraction.** For each viewpoint, we traverse all occupied voxels and filter out the occupied voxels that are unobstructed. These unobstructed voxels are regarded as visible voxels. For each visible voxel  $v_j$  of the viewpoint  $u_k$ , we extract the corresponding points based on the previously recorded voxel-point relationship  $v_j.P_k$  denotes the visible points of the viewpoint  $u_k$ .

#### 3.2.2. Visible information evaluation

Without any semantic information, evaluating component visibility directly from spatially discrete 3D points is challenging. To address this, this paper employs a heuristic approach to identify visibility cues of components, which are then used to calculate the visibility rate of components. Subsequently, viewpoint selection is performed based on the calculated view consensus rate.

**3.2.2.1. Component visibility.** Component visibility cues within the visible points are firstly detected by the method proposed in [55]. Initially, the visible points of each viewpoint are divided into distinct segments according to their x and y coordinate values. Given a viewpoint  $u_k$  and its visible points  $\mathbf{P}_k.P_{kc} = (x_{kc}, y_{kc}, z_{kc})$  is the centroid of the visible points  $\mathbf{P}_k$ .  $\mathbf{n}_c = (x_{kc} - x_k, y_{kc} - y_k, z_{kc} - z_k)$  denotes the vector from viewpoint  $u_k$  to centroid  $p_{kc}$ . Each point  $p_{ki} = (x_{ki}, y_{ki}, z_{ki}) \in \mathbf{P}_k, i = 1 \dots n_k$  is then assigned to segment  $S_{s(p_{ki})}$  according to the following formula:

$$s(p_{ki}) = \frac{\theta(\mathbf{n}_c, \mathbf{n}_i)}{\Delta \alpha}, i = 1 \dots n_k \quad (2)$$

where  $\mathbf{n}_i = (x_{ki} - x_k, y_{ki} - y_k, z_{ki} - z_k)$  denotes the vector from viewpoint  $u_k$  to the point  $p_{ki}$  and  $\theta(\cdot, \cdot)$  denotes the angle between two vectors. Note that the range of the angle is  $[0, 2\pi)$ . And we define points in the same segment  $S_s$  as  $\mathbf{P}_{ks}$ :

$$\mathbf{P}_{ks} = \{p_{ki} \in \mathbf{P}_k | s(p_{ki}) = s\}, s = 1 \dots n_{seg} \quad (3)$$

Then we further divide the points of each segment into different bins based on their distance to the viewpoint. Specifically, for any point  $p_{ksi} =$

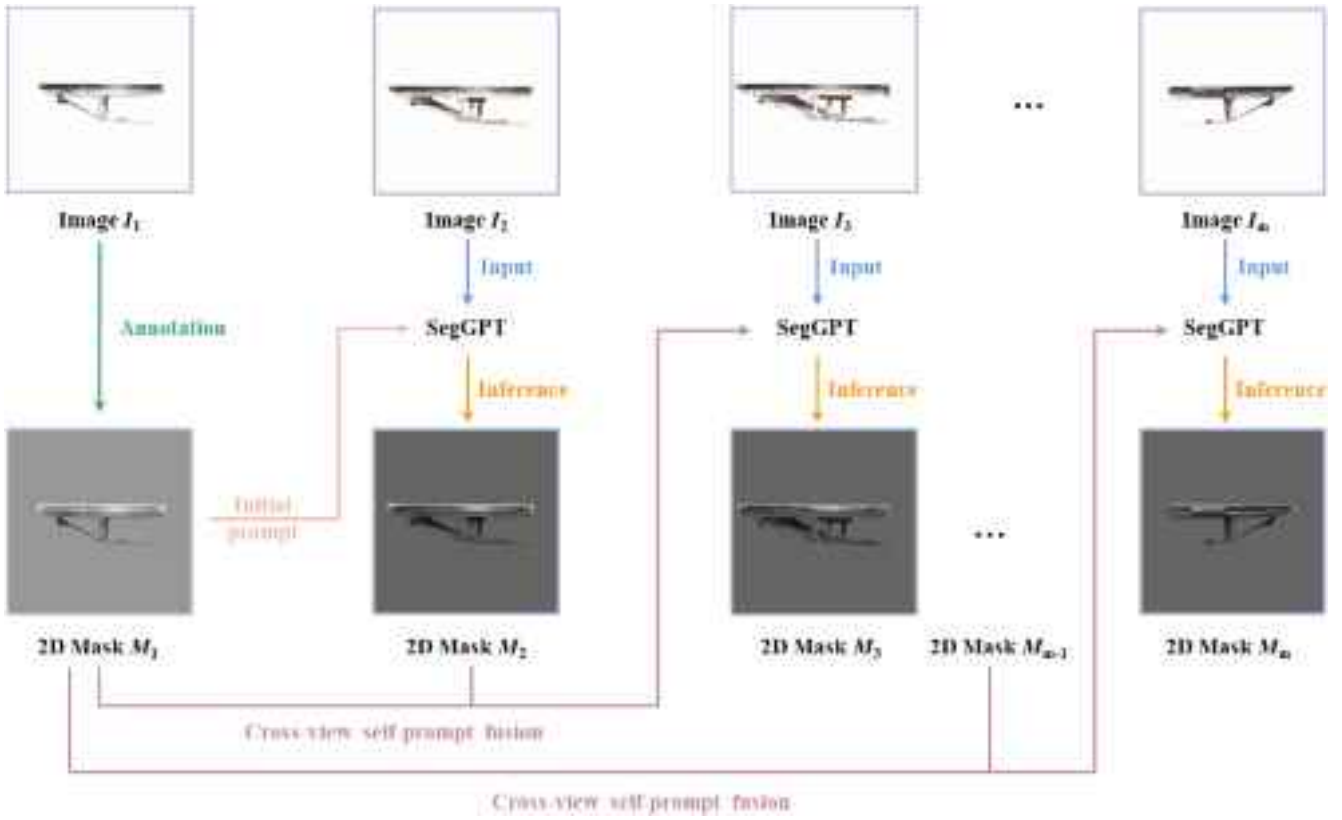


Fig. 4. 2D inference with cross-view self-prompt fusion.

$(x_{ksi}, y_{ksi}, z_{ksi}) \in P_{ks}$ , if  $d_j^{\min} \leq \sqrt{(x_{ksi} - x_k)^2 + (y_{ksi} - y_k)^2} < d_j^{\max}$ , then it is divided to the bin  $b_j^s$ .  $d_j^{\min}$  is the closest distance from the bin  $b_j^s$  to the viewpoint  $u_k$  and  $d_j^{\max}$  is the farthest distance from the bin  $b_j^s$  to the viewpoint  $u_k$ . The visible points in bin  $b_j^s$  is denoted as  $P_{kb_j^s}$ .

Through this way, visible points become organized based on their segment and bin IDs. And then we map the points in the same bin to their corresponding 2D points using the following formula:

$$P_{kb_j^s} = \left\{ p_i = \left( \sqrt{x_i^2 + y_i^2}, z_i \right) \mid p_i = (x_i, y_i, z_i) \in P_{kb_j^s} \right\} \quad (4)$$

Evidently, a bin may contain many 2D points or none. We take the lowest point in  $P_{kb_j^s}$  as the prototype point to present each non-empty bin  $b_j^s$ . Therefore, the segment can be represented by a set of 2D prototype points from each non-empty bin. The least squares line fitting is then applied to obtain the segment line for each segment based on their 2D prototype points efficiently. Finally, calculating the distance from each point in  $P_{kb_j^s}$  to the segment line and using a distance threshold  $d_{threshold}$ , we can identify the component visibility cues  $P_{kg}$  from the entire visible points  $P_k$ . Subsequently, the visibility rate  $r_k$  of the viewpoint  $u_k$  can be calculated as follows:

$$r_k = \frac{\text{size}(P_{kg})}{\text{size}(P_k)} \quad (5)$$

where  $\text{size}(\cdot)$  denotes the number of points in a point set.

The importance of each viewpoint can be evaluated by the component visibility rate. If the components visibility rates of all viewpoints are relatively similar, it indicates that each viewpoint provides useful information for the segmentation task. In that case, we only apply uniform selection to reduce the number of viewpoints. Conversely, when significant differences in component visibility rates are observed, we further select the most appropriate viewpoints by the view consensus

rates.

**3.2.2.2. View consensus.** For each viewpoint, the view consensus with preceding and succeeding viewpoints is also assessed in addition to the component visibility. In our work, we take the visibility rate difference to present the difference in visibility information between two viewpoints and employ the differences with the preceding and succeeding viewpoints to present the viewpoint consensus rate of a viewpoint.

For a viewpoint  $u_i$  with visibility rate  $r_i$ , the difference between  $u_i$  and the preceding viewpoint  $u_{i-1}$  is denoted as  $|r_i - r_{i-1}|$ , and the difference between  $u_i$  and succeeding viewpoint  $u_{i+1}$  is denoted as  $|r_i - r_{i+1}|$ . The viewpoint consensus rate for viewpoint  $u_i$  is then calculated as follows:

$$c = \frac{|r_i - r_{i-1}|}{|r_i - r_{i+1}|}, r_i > r_{threshold} \quad (6)$$

where  $r_{threshold}$  denotes the visibility threshold. When visible rate of a viewpoint exceeds this threshold, we consider that the viewpoint to have component visibility. Based on empirical knowledge, this study sets the visibility threshold to 0.2 and calculates the view consensus rate for each viewpoint with a visibility rate greater than 0.2. And  $m$  viewpoints were selected as the final projection viewpoints with consensus rate between [0.2, 5].

The pseudocode of viewpoint selection is presented in Algorithm 1. Firstly, the time complexity of the voxelization for the PCD  $P$  with  $n$  points is  $o(n)$ . Then, we extract visible points from visible voxels for each candidate viewpoint with a time complexity of  $o(n_1 \times n_2)$ . Finally, we compute the visibility rate for each candidate viewpoint and the view consensus rate for those viewpoints that meet the visibility rate requirement to select the final viewpoint, with a complexity of  $o(n_1) + o(n_3)$ . Therefore, the overall time complexity is  $o(n) + o(n_1 \times n_2) + o(n_1) + o(n_3)$ . Note that,  $n_1$  is the number of candidate viewpoints,  $n_2$  is the number of visible voxels, and  $n_3$  is the number of visible points. Since

the number of visible voxels and viewpoints is much smaller than the number of points in PCD, our algorithm is an efficient viewpoint selection method with a time complexity of  $O(n)$ .

**Algorithm 1** Pseudocode of viewpoint selection.

---

**Algorithm 1.** Viewpoint selection based on view consensus

---

**Input:** Bridge PCD  $\mathbf{P}$  with  $n$  points, Candidate viewpoints  $U_{\text{candidate}} = \{u_i, i = 1, \dots, m\}$

**Output:** Selected viewpoints  $U_{\text{selected}} = \{u_j, j = 1, \dots, m\}$

```

1:  $U_{\text{selected}} = \emptyset$ 
2: PCD voxelization by a voxel grid  $G$  with voxel size  $l$ . Each voxel  $v_i$  has one index  $I_{v_i} = (ix, iy, iz)$  and occupancy status  $o_i \in \{0, 1\}$ . // Voxelization
3: for each occupied voxel  $v_j \in V_{\text{occupied}} = \{v_i | o_i = 1\}$  do
4:   compute voxel center  $v_j$  and record all points in the voxel  $vp_j = \{p_k | p_k \in \mathbf{P} \text{ and } p_k \text{ in } v_j\}$ ;
5: end for
6: for each candidate viewpoint  $u_i$  do
7:   visible points  $P_i = \emptyset$ ; // Visible points extraction
8:   for each occupied voxel  $v_j \in V_{\text{occupied}}$  do
9:     initialize  $r_{ji}$  according to Eq. (1);
10:    initialize time intervals  $\Delta t_x, \Delta t_y, \Delta t_z$ ; move direction  $\Delta x, \Delta y, \Delta z$ ;
11:    initialize  $v_{\text{current}} \leftarrow I_{v_j}; T_x \leftarrow l / (v_{jx} - u_{ix}); T_y \leftarrow l / (v_{jy} - u_{iy}); T_z \leftarrow l / (v_{jz} - u_{iz})$ ;
12:    while  $v_{\text{current}}$  in the voxel grid  $G$  do
13:      update  $v_{\text{current}}$  by methods proposed in [53] and add  $v_{\text{current}}$  to  $r_{ji}$ ;
14:    end while
15:    if  $\sum_{v_j \in V_{\text{occupied}}} o_j = 1$  then add  $vp_j$  to visible points  $P_i$ ;
16:    end if
17:  end for
18:  divide visible point  $p_i \in P_i$  into segment  $s(p_i)$  and bin  $b_i^s$ ; // visibility rate calculation
19:  calculate the  $P_{\text{avg}}$  according to Eq. (4) and adopt the lowest point as the prototype point.
20:  fit  $line_s$  and get component visibility cues  $P_{\text{ls}}$  and calculate  $r_i$  according to Eq. (5);
21: end for
22: for each viewpoint  $u_i$  with  $r_i > 0.2$  do // viewpoint selection
23:   calculate the view consensus rate  $c_i = |r_i - r_{i-1}| / |r_i + r_{i-1}|$ ;
24:   if  $c_i \in [0.2, 0.5]$  then add  $u_i$  to  $U_{\text{selected}}$ ;
25: end if
26: end for
27: return selected viewpoints  $U_{\text{selected}} = \{u_j, j = 1, \dots, m\}$ 

```

---

Finally, we renders the bridge PCD  $\mathbf{P}_{n \times 3}$  with  $n$  points into  $m$  images of size  $h \times w \times 3$  from these  $m$  selected viewpoints by a renderer  $\mathbf{R}: \mathbf{R}^{n \times 3} \rightarrow \mathbf{R}^{m \times h \times w \times 3}$ . The renderer includes a rasterizer and a shader. Initially, the rasterizer transforms points from world coordinates to view coordinates given a camera defined by  $u_i (i = 0, 1, \dots, m)$ . Then each 3D point is assigned to a specific pixel in the image and the index maps  $PM_i (i = 1, \dots, m)$  between PCD and pixels of each image are recorded. Moreover, the shader interpolates RGB colors of the points in the same pixel as the RGB value of this pixel. In our work, we use Pytorch3D [38] differential renderer and the parameters are the  $m$  azimuth angles and  $m$  elevation angles from the  $m$  camera viewpoints.

### 3.3. 2D inference with cross-view self-prompt fusion

After comparing several released pre-trained LVMs, we identified SegGPT as an optimal choice to perform 2D inference for bridge multi-view images. This model exhibits robust contextual inference capabilities, enabling prompt tuning for diverse segmentation tasks in different domains. Thereby, it is essential to provide suitable prompts to SegGPT for inferring accurate mask of components in those images. Additionally, it is capable to inference numerous target images using one or multiple prompt images along with corresponding masks. Therefore, a single accurately annotated image can serve as a prompt for all images and more prompts may augment the inference ability of the SegGPT

theoretically.

This paper proposes a cross-view self-prompt fusion mechanism during the 2D inference stage to augment the prompts for SegGPT. In this mechanism, the fine-annotated mask of the first image is employed as the initial prompt  $\text{prompt}_{\text{init}}$ . This initial prompt contains accurate component information, offering a high-quality semantic prompt for SegGPT. Additionally, as the viewpoints selected in Section 3.2 have substantial view consensus, the inference result of the previous image can also serve as a reliable prompt. Consequently, this paper fuses the initial prompt with the mask of the previous viewpoint to provide reliable augmented prompts for SegGPT.

Specifically, when segmenting the image  $I_i$  of the viewpoint  $u_i$ , we take both the inference result  $M_{i-1}$  of the  $u_{i-1}$  and the initial prompt  $\text{prompt}_{\text{init}}$  as the prompt for the current segmentation task. Specifically, the cross-view self-prompt fusion mechanism can provide the following prompt for the  $I_i$ :

$$\text{Prompt}_i = \{\text{prompt}_{\text{init}}, M_{i-1}\}, i = 2, \dots, N \quad (7)$$

Note that the augmented prompt fed to SegGPT is an image set. Therefore, when segmenting the second viewpoint's image, only the initial prompt is used as the prompt for this round of segmentation, i.e.  $\text{Prompt}_2 = \{\text{prompt}_{\text{init}}\}$ .

Fig. 4 illustrates the 2D inference with cross-view self-prompt fusion for parapet segmentation. The first image is annotated with precise component information, such as the parapet shown in the figure. This fine-annotated mask is fed into SegGPT along with the image to inference the mask. And then, for each subsequent mask inference of image, we use the cross-view self-prompt fusion mechanism to generate augmented prompt which fuses the initial prompt and previous mask. This process is repeated until the last image is inferred.

### 3.4. 3D segmentation

#### 3.4.1. 3D Mask inverse projection

The 2D inference step described in Section 3.3 is iteratively repeated until images from all viewpoints are segmented. And then, all 2D image masks are inverse projected into the 3D point cloud space according to the mapping relationship between the point cloud and the image pixels described in Section 3.2. We obtain 3D masks corresponding to each 2D image mask by following formula:

$$M_i' = PM_i^{-1}(M_i) \quad (8)$$

Points of these 3D masks obtained by inverse projection are coarse, containing numerous extraneous noise points and suffering from over-segmentation. Directly merging multiple 3D segmentation masks is not only in excessive redundancy and introduces noise, but also imposes a computational burden on subsequent fine segmentation processes. Therefore, this paper adopts a hard voting strategy for all 3D segmentation masks. This strategy involves counting the number of times a point appears in different 3D segmentation masks and using a voting parameter  $k$  to filter out the points most likely representing bridge components, as shown in the following equation:

$$k = \underset{[0, N]}{\text{arginf}} P_{\text{target}} \quad (9)$$

$$P_{\text{vote}} = \left\{ p_j \in \mathbf{P} \mid \sum_{i=1}^N E(p_j, M_i) > k \right\} \quad (10)$$

where  $P_{\text{target}}$  is the ground truth of the components,  $k$  is the best hard-voting parameter, and  $P_{\text{vote}}$  is the coarse segmentation result obtained by hard-voting;  $E(p_j, M_i)$  is an existence indicator to judge one point is present in one 3D mask or not, if  $p_j$  belongs to  $M_i$ ,  $E(p_j, M_i)$  equals to 1, else equals to 0, formulated as  $E(p_j, M_i) = \begin{cases} 1, p_j \in M_i \\ 0, p_j \notin M_i \end{cases}$ .



Fig. 5. Data preparation for 10 RC bridges.

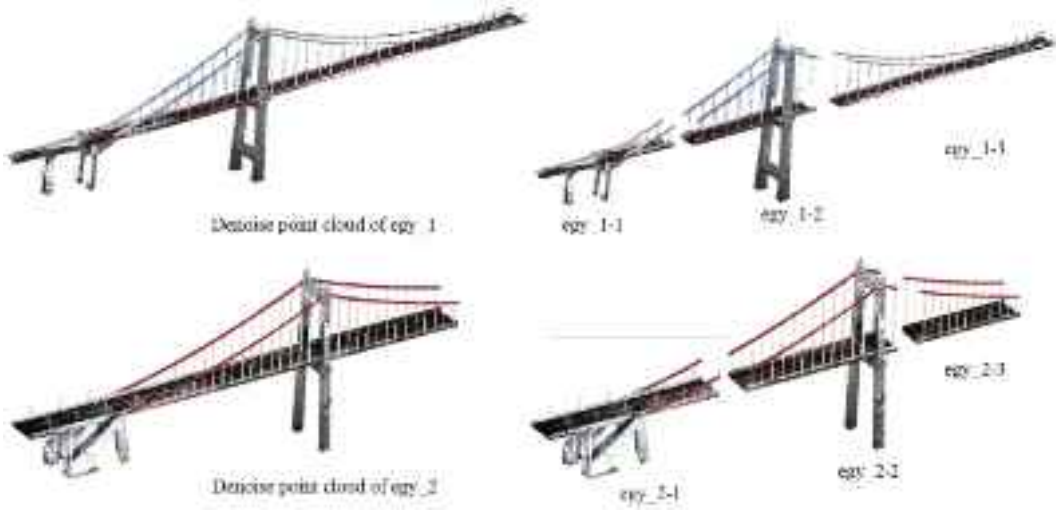


Fig. 6. Data preparation for two suspension bridges.

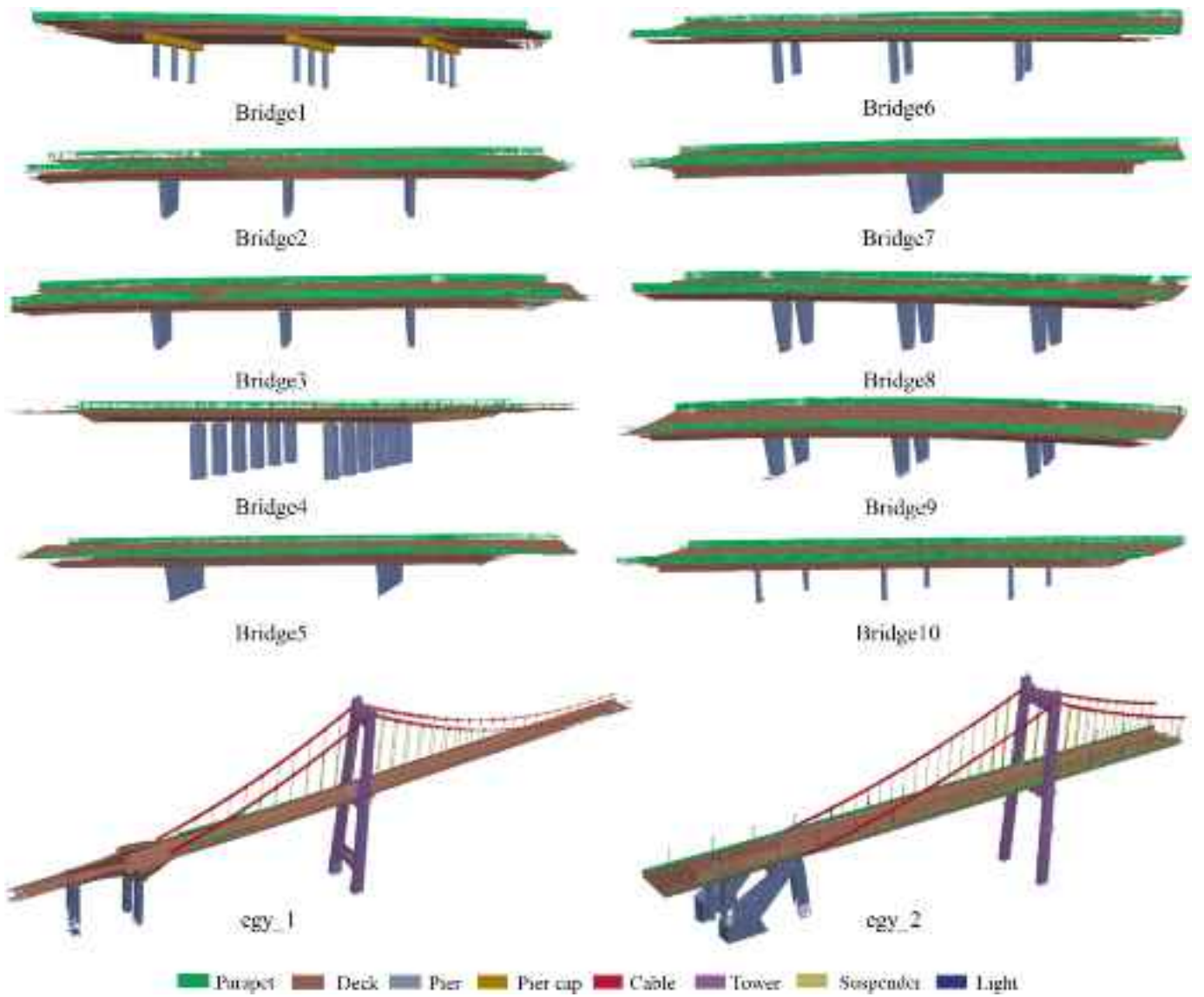


Fig. 7. Segmentation results by our proposed method.

**Table 1**  
Quantitative results of 10 RC bridges.

	Parapet	Deck	Pier	Pier cap	Avg.
Pr(%)	99.29	99.34	99.80	99.79	99.55
R(%)	97.28	99.71	98.65	97.86	98.37
IoU(%)	96.95	99.05	98.46	97.66	98.03
F1(%)	98.25	99.52	99.22	98.82	99.00

#### 3.4.2. Post-processing

The outcome derived from hard voting often manifests as an over-segmented result, with connected components and noise from surrounding regions. Relying on a single rule for post-processing over-segmented point cloud has limitations, often resulting in incomplete or inaccurate segmentations. We thus propose a post-processing method that applies different rules to refine the components with distinct features. For components with prominent planar features (i.e. suspenders, decks, pier caps, and piers), we employ the Random Sample Consensus (RANSAC) algorithm for multi-plane detection. Spatial relationships between the segmentation targets are then used to extract the desired components. For components exhibiting significant linear features (i.e. cables, suspensions, and lights), we adopt the method proposed in [56]. Specifically, points with strong linear characteristics are identified based on their neighbors' geometric structures, and the region growing algorithm based on the principal component analysis is applied to obtain segmentation targets. Finally, an outlier filtering algorithm is applied to eliminate irrelevant points, resulting in a refined segmentation outcome.

## 4. Experiments and results

### 4.1. Data preparation

To validate the reliability and generalizability of the proposed method, numerical experiments were conducted using two bridge datasets. The first type of bridge PCD from an open-source dataset consists of 10 reinforced concrete (RC) bridges near Cambridgeshire in the United Kingdom provided by [17], which is publicly available on the website (<https://zenodo.org/record/1240534>). We excluded the on-site traffic and environmental points (i.e. trees and vegetation) from the raw point clouds. Additionally, each bridge PCD was divided into two or three sections along the traffic direction. The second type pertains to 2

suspension bridges in Chongqing, China. The same removal process has also been applied. Fig. 5 gives the data preparation for 10 RC bridges. And Fig. 6 presents the data preparation for the suspension Bridges named *egy\_1* and *egy\_2*.

### 4.2. Segmentation results

#### 4.2.1. Quantitative results

We selected four most commonly adopted metrics, i.e., Precision (*Pr*), Recall(*R*), Intersection over Union (*IoU*) and F1 score (*F1*), to evaluate the performance of our framework quantitatively. Eqs. (11)–(14) give the definitions of these metrics.

$$Pr = \frac{TP}{TP + FP} \quad (11)$$

$$R = \frac{TP}{TP + FN} \quad (12)$$

$$IoU = \frac{TP}{TP + FP + FN} \quad (13)$$

$$F1 = 2 \times \frac{Pr \times R}{Pr + R} \quad (14)$$

Where *TP* denotes the “True Positive”, representing the number of points correctly segmented. *FP* is the “False Positive”, the number of points erroneously segmented to the class. *FN* means the “False Negative”, indicating the number of points belonging to this class but not segmented out. Precision is the ratio of *TP* to the segmented points number, which indicates the reliability of the segmentation results. Recall is calculated by *TP* and ground truth, which reflects the completeness of the segmentation results. *IoU* and *F1* are applied to evaluate the performance quantitatively considering both two aspects. Fig. 7 presents the segmentation results by our proposed method.

Table 1 presents the quantitative segmentation results of the proposed bridge PCSS method across 10 RC bridges. The results demonstrate high segmentation accuracy for components including the parapet, deck, pier and pier cap. The average Precision for the deck and pier is 99.34 % and 99.80 %, respectively, with corresponding average Recall of 99.71 % and 98.65 %. Therefore, the *IoU* and *F1* scores for these components are notably high. Specifically, the *IoU* for the deck reaches 99.05 %, with an *F1* score of 99.52 %. The pier segmentation

**Table 2**  
Quantitative results of two suspension bridges.

	Parapet	Deck	Pier	Suspender	Cable	Tower	Light	Avg.
Pr(%)	81.32	97.99	100	96.56	95.5	99.84	99.84	95.86
R(%)	87.75	97.11	99.51	94.52	95.47	99.29	98.94	96.08
IoU(%)	72.41	95.22	99.51	91.33	91.26	99.12	98.78	92.51
F1(%)	82.33	95.73	99.76	95.46	95.37	99.56	99.39	95.37

**Table 3**  
Comparison with the method in [23].

Ref.	Pr (%)			R (%)			IoU (%)		
	Deck	Pier	Avg.	Deck	Pier	Avg.	Deck	Pier	Avg.
[23]	96.34	98.58	96.74	97.57	<b>98.94</b>	97.84	94.10	<b>97.61</b>	94.72
Ours	<b>99.34</b>	<b>99.80</b>	<b>99.55</b>	<b>99.71</b>	98.65	<b>99.53</b>	<b>99.03</b>	95.54	<b>97.29</b>

**Table 4**  
Comparison with the method in [27].

Ref.	Deck				Pier				Parapet			
	Pr (%)	R (%)	IoU (%)	F1 (%)	Pr (%)	R (%)	IoU (%)	F1 (%)	Pr (%)	R (%)	IoU (%)	F1 (%)
[27]	96.7	97.7	94.6	97.2	97.5	90.6	88.5	93.9	94.7	90.7	86.3	92.6
Ours	<b>99.3</b>	<b>99.7</b>	<b>99.0</b>	<b>99.5</b>	<b>99.8</b>	<b>98.7</b>	<b>95.5</b>	<b>99.2</b>	<b>99.6</b>	<b>99.5</b>	<b>97.3</b>	<b>98.3</b>

**Table 5**  
Comparison with the method in [25].

Ref.	IoU (%)		
	Pier	Pier cap	Parapet
[25]	99.36	94.60	88.43
Ours	<b>99.71</b>	<b>98.65</b>	<b>99.53</b>

achieves an F1 score of 99.22 %, though its IoU is slightly lower at 98.46 %. The parapet and pier cap also get excellent average precision, with values of 99.29 % and 99.79 %, respectively. However, their

segmentation results show slightly lower scores for Recall, IoU, and F1. Specifically, the average Recall for the parapet is 97.28 %, the average IoU is 96.95 %, and the average F1 is 98.25 %. The segmentation results for pier cap yield scores of 97.86 %, 97.66 %, and 98.82 % for Recall, IoU, and F1, respectively.

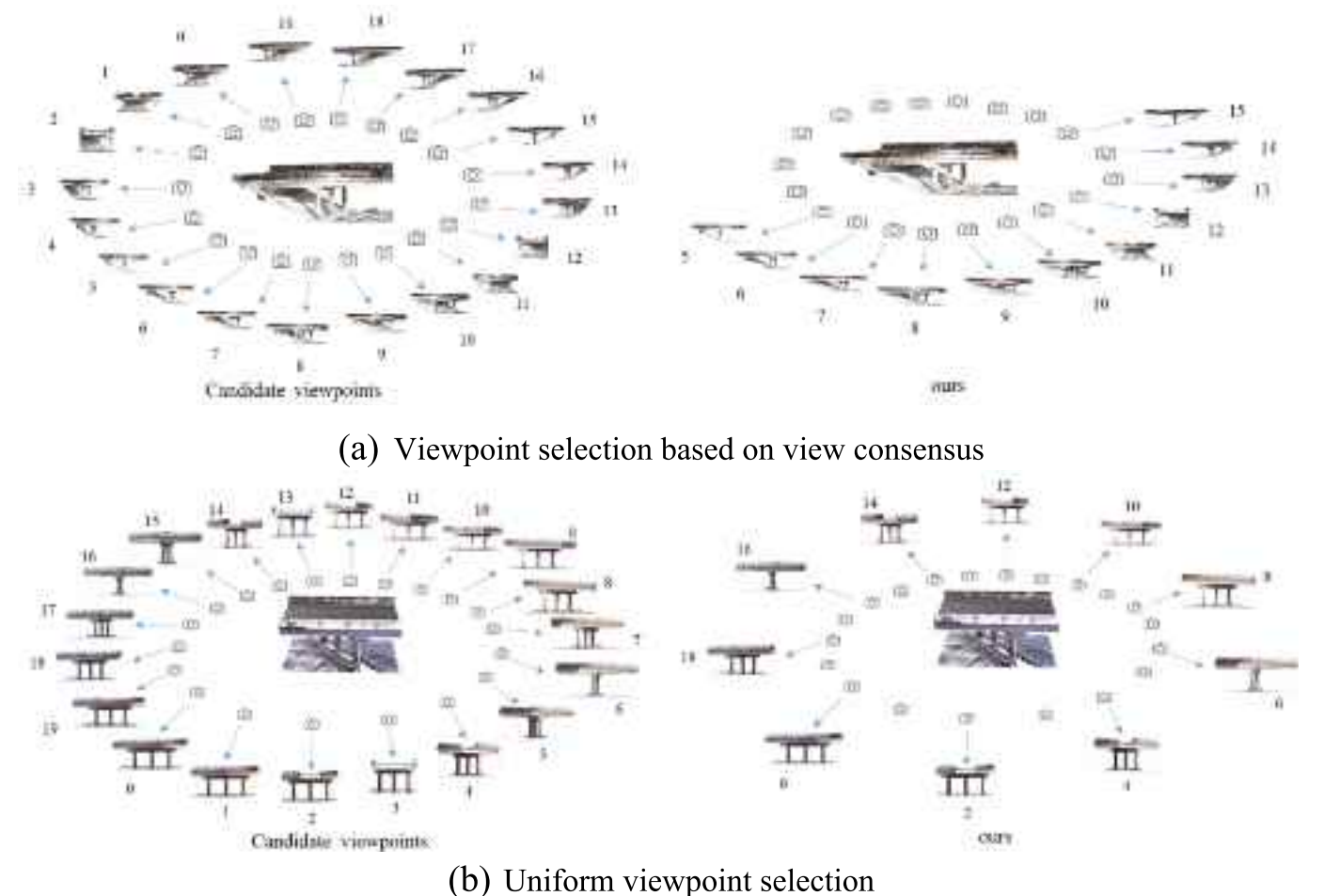
Similarly, the quantitative segmentation results for two suspension bridges' components (i.e. parapet, deck, pier, suspender, cable, tower and light) are provided in Table 2. The average Precision for the pier, tower and light is remarkably high, with the value of 100 %, 99.84 % and 99.84 %, respectively. And their corresponding Recalls are 99.51 %, 99.29 % and 98.94 %. Components like deck are 97.99 % and 97.11 %

**Table 6**  
Comparison of state-of-the-art methods for bridge segmentation.

Ref.	Methodology	Adopt to multiple bridge types	Recognition Capability							
			Deck	Par-apet	Pier	Pier cap	Tow-er	Cable	Susp-ender	light
[17]	unsupervised	No	Yes	No	Yes	Yes	No	No	No	No
[20,21]	supervised	No	Yes	No	Yes	No	No	No	No	No
[23]	supervised	No	Yes	No	Yes	Yes	No	No	No	No
[24,27]	supervised	No	Yes	Yes	Yes	Yes	No	No	No	No
[25]	supervised	No	Yes	Yes	Yes	No	No	No	No	No
Ours	unsupervised	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

**Table 7**  
Computation time (minutes) statistic of the proposed method.

Point cloud No.	Point number	Component types	Viewpoint number	Multi-view projection	2D inference	3D segmentation	Total
Bridge1-1	5,480,927	4	11	3.01	1.16	20.12	24.29
Bridge2-1	1,738,018	3	7	1.86	0.51	4.39	6.76



**Fig. 8.** Visual results of proposed viewpoint selection method.

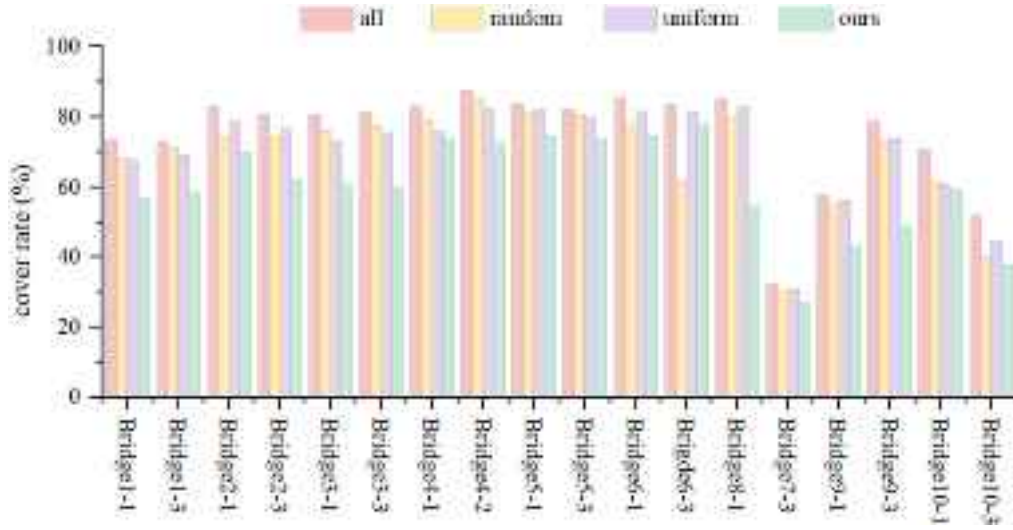


Fig. 9. Cover rate of different viewpoint selection.

respectively in average precision and average recall because of occlusion by other components (i.e. the deck on tower). The quantitative results for the suspender and cable are also satisfactory. However, the segmentation performance for the parapet is poorer, which may be caused by the missing data.

Table 3 reports the quantitative comparison between our method and the method in [23], both tested on 10 RC bridges for segmentation experiments. According to the results provided in their paper, we compared Precision, Recall and IoU for the deck and pier. The comparison shows that our method achieves better performance in deck. We also compared our method with the latest supervised segmentation network, RandLA-BridgeNet [27]. This study utilized the 10 RC bridges without excluding background. According to the results provided by their paper, we compare four metrics for the deck, pier and parapet. As

shown in Table 4, our results outperform in all evaluation metrics. Furthermore, Table 5 presents the comparison between our method and a supervised method improved by data augmentation with synthetic data [25]. Although they achieved further segmentation of the girder, our method still demonstrates superior performance in the pier, pier cap and parapet. These quantitative comparisons confirm that our method outperforms the cutting-edge methods across all segmentation metrics.

Additionally, Table 6 presents the qualitative comparison between our method and the current state-of-the-art (SOTA) methods. This comparison demonstrates that our method surpasses these SOTA approaches at the component level, effectively identifying more types of components. Moreover, our method also exhibits substantial generalizability across different bridge types in segmentation tasks.

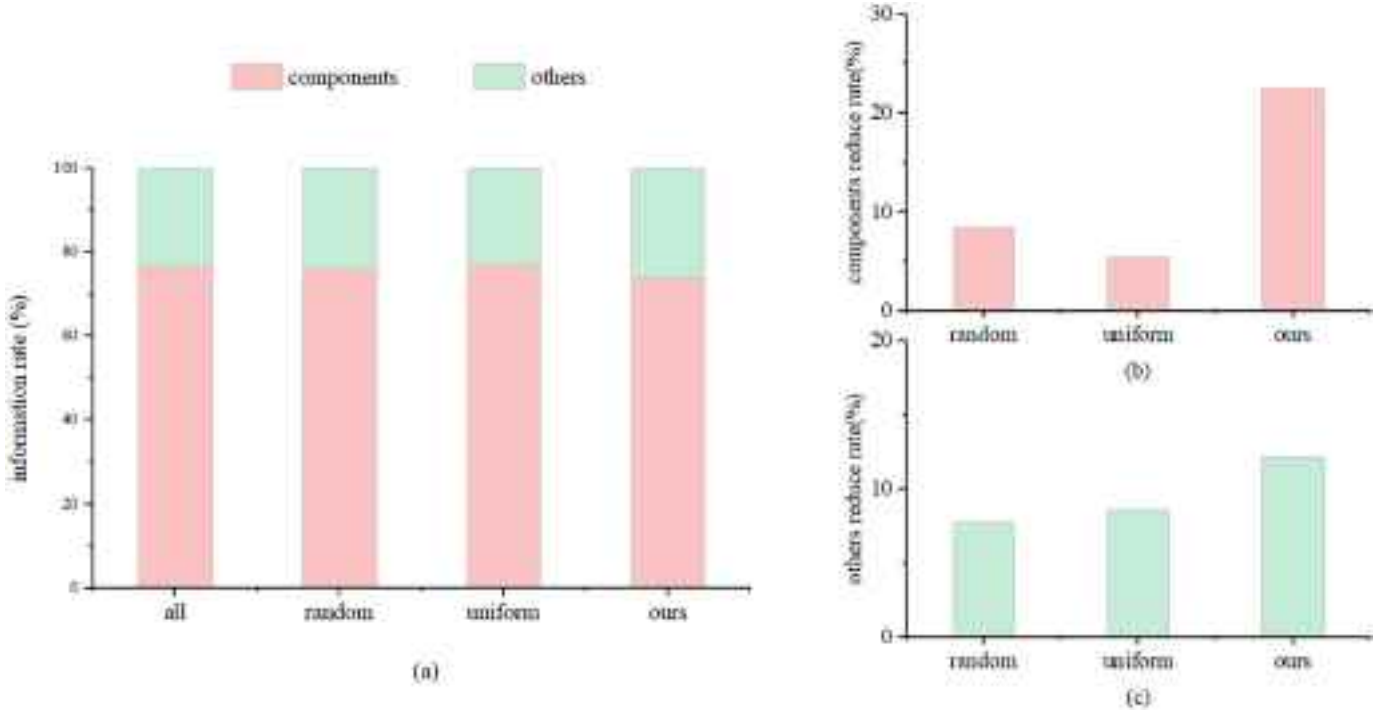
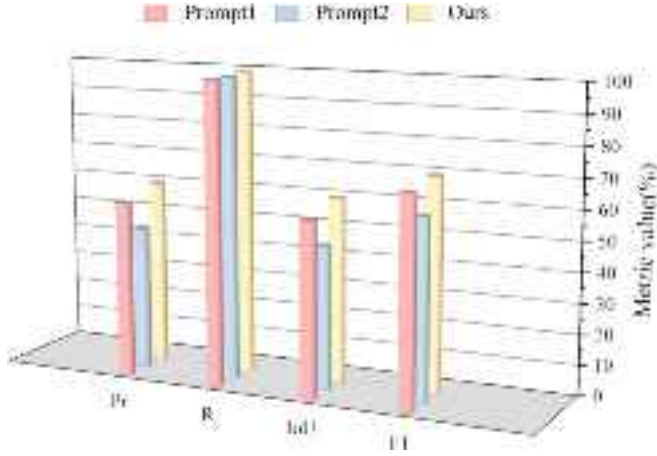


Fig. 10. Information of visible points in different situations. (a) Red bars show the proportion of the component points in visible points; Others are in green. (b) Components points reduce rate in visible points after different viewpoint selection; (c) Other points reduce rate in visible points after different viewpoint selection. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

**Table 8**

Average and variance of the overlap comparison results.

Overlap	All	Random	Uniform	Ours
Average (%)	16.10	13.91	12.24	<b>15.40</b>
Variance ( $\times 10^{-3}$ )	2.96	3.41	2.41	<b>1.78</b>

**Fig. 11.** Results of different prompt strategies before post-processing. Prompt1 only takes the initial prompts. Prompt2 only takes the last view masks as the prompt. Ours takes the set of initial prompt and last predict mask as the prompt.**Table 9**

Comparison results of self-prompting methods.

	Pr (%)	R (%)	IoU (%)	F1 (%)
Self-Prompting I	43.40	99.77	43.37	54.47
Self-Prompting II	50.86	99.92	50.83	63.64
Ours	<b>66.34</b>	<b>99.99</b>	<b>66.33</b>	<b>77.91</b>

#### 4.2.2. Runtime statics

The experiments in this study were conducted on a computer equipped with a 12th Gen Intel(R) Core(TM) i5-12400F CPU, an Nvidia GeForce GTX 1660 Super GPU, and 32 GB of RAM. Table 7 presents the runtime of our proposed segmentation method on the original point clouds of Bridge1-1 and Bridge 2-1. Bridge1-1, a high-density dataset with 11 viewpoints and 4 different component types, required a total processing time of 24.29 min. Bridge 2-1, characterized by a lower density, involved only 7 viewpoints and 3 component types, achieving

significantly higher efficiency with a processing time of just 6.76 min. Overall, these results indicate that increases in point cloud density, selected viewpoints number, and component types reduce computational efficiency. However, it is anticipated that the use of more advanced hardware configurations could substantially enhance computational performance.

#### 4.3. Validation experiments

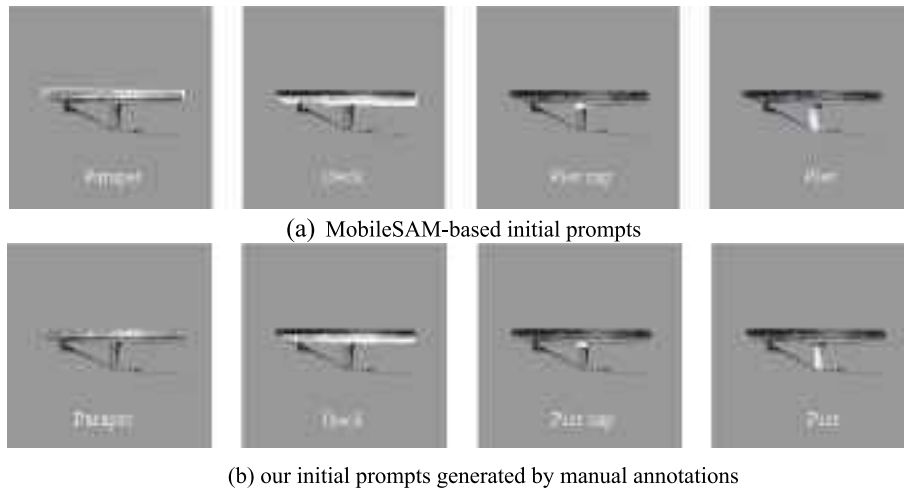
##### 4.3.1. Viewpoint selection

The visual results of the proposed viewpoint selection method based on view consensus are presented in Fig. 8. 20 candidate viewpoints are initially generated uniformly around the bridge initially. Fig. 8(a) presents the viewpoint selection based on view consensus and Fig. 8(b) presents the viewpoint selection strategy when the component visibility rate is similar. As shown in Fig. 8, the number of viewpoints selected by our method was fewer than the original set. This indicates that our method effectively reduces the number of viewpoints, which is significantly releasing redundancy.

We also compare our method with random selection and uniform selection. In the random viewpoint selection, the number of viewpoints is matched to that selected by our method, while the uniformly spaced group selected 10 viewpoints by choosing every other viewpoint starting from the first one. Fig. 9 depicts the coverage rate of visible points obtained by different selections. As expected, all candidate viewpoints without selection, due to their greater quantity, exhibited a higher coverage rate than the selected viewpoints. Selection resulted in a slight reduction in coverage rate, with random and uniform selection showing similar levels of coverage. Our method yielded a lower rate, indicating these viewpoints are more focused.

Figure 10(a) shows that the proportions of component points and others are similar among the visible points obtained viewpoints with and without selection. It means viewpoint selection does not change the information composition of visible points. However, reducing the number of viewpoints inevitably decreases visible information. Fig. 10 (b) and (c) illustrate the loss of component points and others after applying different selection methods. This comparison reveals that our proposed method results in the greatest reduction of other information, indicating that our viewpoints are less affected by noise.

Finally, we calculated the average overlap rate between adjacent viewpoints and their variance (Table 8). The results show that the average overlap rate of the viewpoints selected by our method is higher than that of the other two methods, with the smallest variance in overlap between viewpoints. This reveals that the viewpoints selected by our method exhibit reliable view consensus, with smooth information

**Fig. 12.** Visual comparisons between MobileSAM-based initial prompts and our manually annotated initial prompts. The semantic mask are hilighted in white.

**Table 10**

Comparison results of different initial prompt generation methods.

	Pr (%)	R (%)	IoU (%)	F1 (%)
MobileSAM-based initial prompts	62.68	99.74	62.57	74.35
Manually anotated initial prompts	66.34	99.99	66.33	77.91

transitions between viewpoints. As a result, these viewpoints can clearly capture various components while ensuring consistent component visibility.

#### 4.3.2. Cross-view self-prompt fusion

The cross-view self-prompt fusion adopted in this paper enhances the initial prompts by incorporating predictions from the previous viewpoint. Fig. 11 illustrates the segmentation results obtained using only the initial prompts, only the predictions from the previous viewpoints, and the prompts obtained by our method. Prompt1 represents using only the initial prompts for segmentation, resulting in an average precision of 57.7 %, Recall of 99.73 %, IoU of 57.61 %, and F1 score of 68.70 %. Prompt2 represents using the predictions from the previous viewpoints as segmentation prompts, resulting in an average precision of 47.36 %, Recall of 99.48 %, IoU of 47.34 %, and F1 score of 59.70 %. The cross-

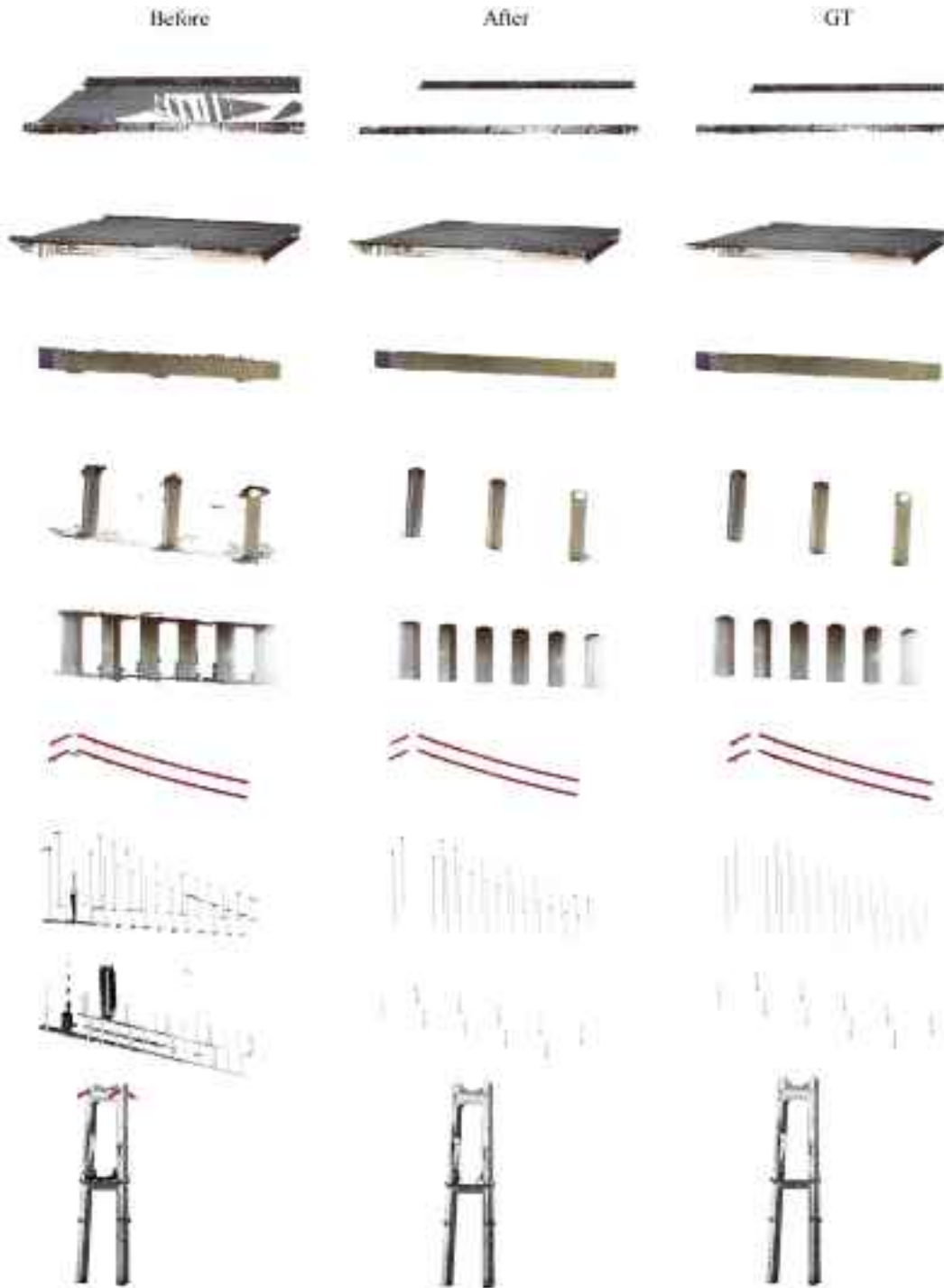


Fig. 13. Visual comparisons among before and after post-processing and ground truth.

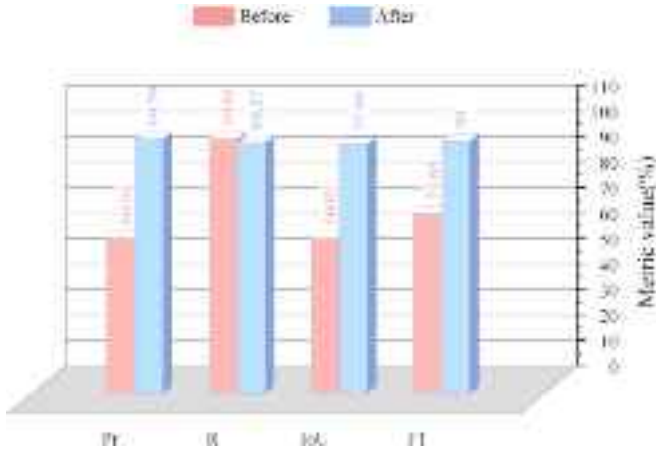


Fig. 14. Quantitative comparisons of before and after post-processing.

view self-prompt fusion mechanism proposed in this paper demonstrates advantages across various evaluation metrics, with an average precision of 60.95 %, Recall of 99.81 %, IoU of 60.82 %, and F1 score of 70.64 %. Comparing these metrics, the cross-view self-prompting strategy improves segmentation performance by mitigating over-segmentation issues, effectively reducing false positives (FP) and false negatives (FN), and achieving superior segmentation performance.

Existing studies on cross-view self-prompting strategies are limited. A similar work based on inverse rendering mask prompts is introduced in [55]. The core idea of this strategy involves projecting the 2D mask onto 3D mask and taking the NeRF-rendered 2D mask given the next view as the prompt. Table 9 provides comparative results of the inverse rendering mask prompt strategy and our cross-view self-prompt fusion. In the comparative experimental setup, the inverse-rendered mask prompt for a novel view is the visible pixels of the 3D mask inverse-rendered from the previous view. The prompting strategy that exclusively uses the inverse-rendered mask prompts is called Self-Prompting I. Self-Prompting II combines the initial prompt with the inverse-rendered mask prompt. The results indicate that the proposed method achieves superior performance in terms of segmentation metrics. This improvement may stem from the use of masks as prompts in SegGPT, as the viewpoint consistency across views provides more comprehensive information.

#### 4.3.3. Initial prompt

In this study, the initial prompts were semantic masks manually annotated on the initial images. To demonstrate that the initial prompt generation can be replaced by simple interactive mask generation, various initial masks were also generated using MobileSAM [57] based on points prompt obtained by mouse click (Fig. 12). Table 10 presents the coarse segmentation accuracy obtained using the two types of initial prompts. The results confirm that automatically generated masks can also serve as reliable initial prompts.

#### 4.3.4. Post-processing

Until the step of hard voting, the method is generalized for coarse segmentation of multiple bridge types. In this study, the RANSAC algorithm is employed to refine the components with planar features in the voting results. And the region growing algorithm based on principal component analysis is applied to obtain the fine segmentation results of components with linear features. Fig. 13 presents the visual comparisons before and after post-processing for different components across both bridge types. The results validate that the proposed multi-rules post-processing objectively refines over-segmented voting results. Quantitative comparison results are shown in Fig. 14. After post-processing, the fine segmentation achieves higher Pr while maintaining R. Both quantitative and qualitative results show that our post-processing method achieves satisfactory segmentation outcomes.

#### 4.4. Stress test

We investigate the robustness of our method using point clouds with varying levels of sparsity. The point clouds, downsampled at different ratios (1/5, 1/10, 1/20, 1/50, and 1/100) are illustrated in Fig. 15. The corresponding coarse segmentation results are summarized in Table 11. As shown in the table, the segmentation performance declines slightly with sparser input points. It is attributed to the increasing ambiguity in distinguishing structural components, which leads to over-segmentation in connection regions.

### 5. Discussion

According to the results represented in section 4, this paper achieves zero-shot bridge PCSS with high precision. The zero-shot segmentation capability of the proposed method is demonstrated in three folds: 1) Annotation-free and train-free: The proposed method does not require extensive datasets or labor-intensive annotation; 2) Generalization: The method is versatile for different bridge types not limited to beam bridges and suspension bridges in our experiments; 3) Robustness: Results of the stress test demonstrate that the proposed method can achieve reliable segmentation even with sparse data.

Our viewpoint selection method based on view consensus is demonstrated effective for selecting reliable viewpoints. Each selected viewpoint not only provides a clear view of the components, but also offers reasonable overlaps with adjacent viewpoints, facilitating the segmentation tasks. Moreover, our cross-view self-prompt fusion

Table 11

Comparison results of self-prompting methods.

	Point number	Pr (%)	R (%)	IoU (%)	F1 (%)
Original point cloud	5,480,927	66.34	99.99	66.33	77.91
Downsampled 1/5	1,096,186	61.72	99.93	61.70	74.39
Downsampled 1/10	548,093	61.32	99.89	61.30	74.47
Downsampled 1/20	274,047	57.55	99.72	57.43	70.00
Downsampled 1/50	109,619	57.75	99.29	57.66	70.16
Downsampled 1/100	54,810	47.69	97.88	47.55	59.60

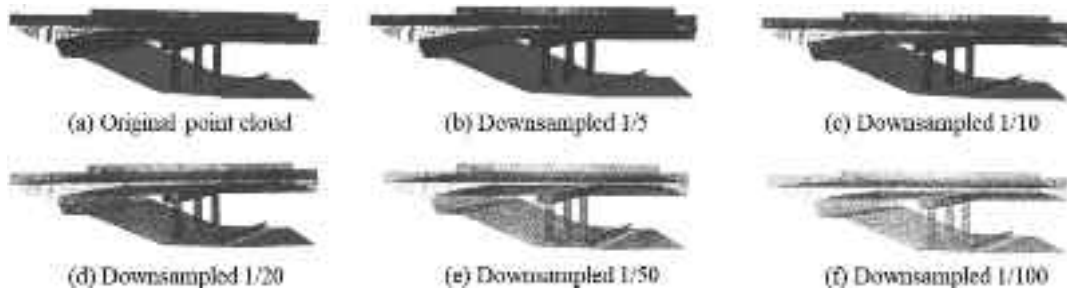


Fig. 15. Visualization of original point cloud and point clouds with varying levels of sparsity.

mechanism significantly enhances the LVM's performance, leading to satisfactory results. In our work, the post-processing method based on multi-rules meets the segmentation requirements for components with different geometric structures.

However, the proposed method has certain limitations. Firstly, although segmentation experiments were conducted on beam and suspension bridges, the method's segmentation capability to other bridge types remains theoretically feasible since the insufficient publicly available datasets. It is helpful to collect more bridge data to evaluate the performance of our method. Secondly, the semantic labels for different components in this study are assigned individually. While this ensures segmentation accuracy and no need for a separate classification module, it leads to a decrease in computational efficiency. Therefore, performing parallel segmentation for multiple components could improve efficiency significantly. Additionally, although our viewpoint selection can discard occluded viewpoints, it only addresses occlusions caused by the external environment. Self-occlusions between components of the bridge can also lead to segmentation inaccuracies. Automatically exploring viewpoints based on active vision would be a valuable direction for future research.

## 6. Conclusion

This paper introduced a zero-shot semantic segmentation method for bridge point clouds by leveraging the prompt-tuning capabilities of SegGPT. The proposed approach addresses the modality gap between 3D PCSS and 2D models through multi-view projection. Subsequently, a viewpoint selection method based on view consensus was developed to evaluate the visible information provided by each viewpoint and optimize the projection viewpoints both in quantitative and qualitative. Additionally, a cross-view self-prompt fusion mechanism was introduced to augment the input prompts, significantly enhancing the segmentation performance of SegGPT. The proposed method achieves high-precision component-level segmentation for RC bridges and suspension bridges. For RC bridges, the overall precision for components such as parapet, deck, pier, and pier cap reached an average of 99.55 %, while for suspension bridges, components including parapet, deck, pier, tower, cable, suspender, and light achieved an average precision of 95.86 %. Stress tests on sparse data further demonstrated the robustness of the proposed method.

In conclusion, this method represents a significant step towards zero-shot semantic segmentation of bridge point clouds, offering a robust and high-precision solution for component-level segmentation of different bridge types. The high-quality semantic segmentation results obtained through this method can effectively support tasks such as bridge dimensional quality inspection and reverse modeling, contributing to improved bridge maintenance and operational management. Further research should focus on expanding datasets to include more bridge types for better generalizability and exploring active vision methods to achieve viewpoint selection without component self-occlusion.

## CRediT authorship contribution statement

**Yan Zeng:** Writing – review & editing, Writing – original draft, Software, Methodology, Investigation. **Feng Huang:** Resources, Funding acquisition. **Guikai Xiong:** Supervision, Project administration. **Xiaoxiao Ma:** Writing – review & editing, Writing – original draft, Methodology, Investigation. **Yingchuan Peng:** Supervision, Resources. **Wenshu Yang:** Software, Resources. **Jiepeng Liu:** Validation, Supervision, Funding acquisition.

## Declaration of competing interest

This manuscript represents an original research and it has not been submitted or published elsewhere. All the authors have approved the manuscript and agree with submission to your Journal. There is no

conflict of interest to declare.

## Acknowledgments

The authors greatly appreciate the financial support provided by the National Key Research and Development Program of China (2021YFF0500903).

## Data availability

Data will be made available on request.

## References

- [1] 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>.
- [2] B.J. Perry, Y. Guo, R. Atadero, J.W. van de Lindt, Streamlined bridge inspection system utilizing unmanned aerial vehicles (UAVs) and machine learning, *Measurement* 164 (2020) 108048, <https://doi.org/10.1016/j.measurement.2020.108048>.
- [3] H. Zhang, Y. Zhu, W. Xiong, C.S. Cai, Point cloud registration methods for long-span bridge spatial deformation monitoring using terrestrial laser scanning, *Struct. Control. Health Monit.* 2023 (1) (2023) 2629418, <https://doi.org/10.1155/2023/2629418>.
- [4] S. Chen, L. Truong-Hong, D. Laefer, E. Mangina, Automated bridge deck evaluation through UAV derived point cloud, in: CERI-ITRN2018, 2018, August, pp. 735–740. <http://hdl.handle.net/2451/43478>.
- [5] E. Kaartinen, K. Dunphy, A. Sadhu, LiDAR-based structural health monitoring: applications in civil infrastructure systems, *Sensors* 22 (12) (2022) 4610, <https://doi.org/10.3390/s22124610>.
- [6] S.A. Dabous, S. Feroz, Condition monitoring of bridges with non-contact testing technologies, *Autom. Constr.* 116 (2020) 103224, <https://doi.org/10.1016/j.autcon.2020.103224>.
- [7] D. Reagan, A. Sabato, C. Niezrecki, Feasibility of using digital image correlation for unmanned aerial vehicle structural health monitoring of bridges, *Struct. Health Monit.* 17 (5) (2018) 1056–1072, <https://doi.org/10.1177/1475921717735326>.
- [8] M. Rashidi, M. Mohammadi, S. Sadeghlo Kivi, M.M. Abdolvand, L. Truong-Hong, B. Samali, A decade of modern bridge monitoring using terrestrial laser scanning: review and future directions, *Remote Sens.* 12 (22) (2020) 3796, <https://doi.org/10.3390/rs12223796>.
- [9] S. Jeong, R. Hou, J.P. Lynch, H. Sohn, K.H. Law, An information modeling framework for bridge monitoring, *Adv. Eng. Softw.* 114 (2017) 11–31, <https://doi.org/10.1016/j.advengsoft.2017.05.009>.
- [10] T.H. Kwon, S.H. Park, S.I. Park, S.H. Lee, Building information modeling-based bridge health monitoring for anomaly detection under complex loading conditions using artificial neural networks, *J. Civ. Struct. Heal. Monit.* 11 (5) (2021) 1301–1319, <https://doi.org/10.1007/s13349-021-00508-6>.
- [11] T. Yamane, P.J. Chun, R. Honda, Detecting and localising damage based on image recognition and structure from motion, and reflecting it in a 3D bridge model, *Struct. Infrastruct. Eng.* 20 (4) (2024) 594–606, <https://doi.org/10.1080/15732479.2022.2131845>.
- [12] C.S. Shim, N.S. Dang, S. Lon, C.H. Jeon, Development of a bridge maintenance system for prestressed concrete bridges using 3D digital twin model, *Struct. Infrastruct. Eng.* 15 (10) (2019) 1319–1332, <https://doi.org/10.1080/15732479.2019.1620789>.
- [13] A. Justo, D. Lamas, A. Sánchez-Rodríguez, M. Soilán, B. Riveiro, Generating IFC-compliant models and structural graphs of truss bridges from dense point clouds, *Autom. Constr.* 149 (2023) 104786, <https://doi.org/10.1016/j.autcon.2023.104786>.
- [14] A. Nguyen, B. Le, 3D point cloud segmentation: A survey, in: 2013 6th IEEE Conference on Robotics, Automation and Mechatronics (RAM), IEEE, 2013, November, pp. 225–230, <https://doi.org/10.1109/RAM.2013.6758588>.
- [15] E. Grilli, F. Menna, F. Remondino, A review of point clouds segmentation and classification algorithms, *Int. Arch. Photogramm. Remote. Sens. Spat. Inf. Sci.* 42 (2017) 339–344, <https://doi.org/10.5194/isprs-archives-XLII-2-W3-339-2017>, 2017.
- [16] Y. Xie, J. Tian, X.X. Zhu, Linking points with labels in 3D: a review of point cloud semantic segmentation, *IEEE Geosci. Remote Sens. Mag.* 8 (4) (2020) 38–59, <https://doi.org/10.1109/MGRS.2019.2937630>.
- [17] R. Lu, I. Brilakis, C.R. Middleton, Detection of structural components in point clouds of existing RC bridges, *Comput. Aided Civ. Inf. Eng.* 34 (3) (2019) 191–212, <https://doi.org/10.1111/mice.12407>.
- [18] 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>.
- [19] Y.P. Zhao, H. Wu, P.A. Vela, Top-down partitioning of reinforced concrete bridge components, in: ASCE International Conference on Computing in Civil Engineering 2019, American Society of Civil Engineers, Reston, VA, 2019, pp. 275–283, <https://doi.org/10.1061/9780784482445.035>.

- [20] H. Kim, J. Yoon, S.H. Sim, Automated bridge component recognition from point clouds using deep learning, *Struct. Control. Health Monit.* 27 (9) (2020) e2591, <https://doi.org/10.1002/stc.2591>.
- [21] H. Kim, C. Kim, Deep-learning-based classification of point clouds for bridge inspection, *Remote Sens.* 12 (22) (2020) 3757, <https://doi.org/10.3390/rs12223757>.
- [22] 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>.
- [23] 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>.
- [24] 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>.
- [25] 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>.
- [26] 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>.
- [27] D. Lamas, A. Justo, M. Soilán, B. Riveiro, Automated production of synthetic point clouds of truss bridges for semantic and instance segmentation using deep learning models, *Autom. Constr.* 158 (2024) 105176, <https://doi.org/10.1016/j.autcon.2023.105176>.
- [28] J.L. Xiao, J.S. Fan, Y.F. Liu, B.L. Li, J.G. Nie, Region of interest (ROI) extraction and crack detection for UAV-based bridge inspection using point cloud segmentation and 3D-to-2D projection, *Autom. Constr.* 158 (2024) 105226, <https://doi.org/10.1016/j.autcon.2023.105226>.
- [29] J. Wang, Z. Liu, L. Zhao, Z. Wu, C. Ma, S. Yu, H. Dai, Q. Yang, Y. Liu, S. Zhang, E. Shi, T. Pan, T. Zhang, D. Zhu, X. Li, X. Jiang, B. Ge, Y. Yuan, D. Shen, T. Liu, S. Zhang, Review of large vision models and visual prompt engineering, *Meta-Radiology* (2023) 100047, <https://doi.org/10.1016/j.metrad.2023.100047>.
- [30] C. Zhang, L. Liu, Y. Cui, G. Huang, W. Lin, Y. Yang, Y. Hu, A comprehensive survey on segment anything model for vision and beyond, *arXiv preprint* (2023), <https://doi.org/10.48550/arXiv.2305.08196> arXiv:2305.08196.
- [31] Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A. C., Lo, W. Y., Dollar, P., & Girshick, R. (2023). Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (pp. 4015–4026). Doi:10.48550/arXiv.2304.02643.
- [32] Jain, J., Li, J., Chiu, M. T., Hassani, A., Orlov, N., & Shi, H. (2023). Oneformer: one transformer to rule universal image segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 2989–2998). Doi:10.48550/arXiv.2211.06220.
- [33] X. Zou, J. Yang, H. Zhang, F. Li, L. Li, J. Wang, L. Wang, J. Gao, Y.J. Lee, Segment everything everywhere all at once, *Adv. Neural Inf. Proces. Syst.* 36 (2024).
- [34] X. Wang, X. Zhang, Y. Cao, W. Wang, C. Shen, T. Huang, Segpt: segmenting everything in context, *arXiv preprint* (2023), <https://doi.org/10.48550/arXiv.2304.03284> arXiv:2304.03284.
- [35] M. Liu, Y. Zhu, H. Cai, S. Han, Z. Ling, F. Porikli, H. Su, Partslip: Low-shot part segmentation for 3d point clouds via pretrained image-language models, in: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 21736–21746, <https://doi.org/10.48550/arXiv.2212.01558>.
- [36] R. Zhang, Z. Guo, W. Zhang, K. Li, X. Miao, B. Cui, P. Qiao, H. Li, Pointclip: Point cloud understanding by clip, in: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 8552–8562, <https://doi.org/10.48550/arXiv.2112.02413>.
- [37] X. Zhu, R. Zhang, B. He, Z. Zeng, S. Zhang, P. Gao, Pointclip v2: Adapting clip for powerful 3d open-world learning, *arXiv preprint* 3 (4) (2022), <https://doi.org/10.48550/arXiv.2211.11682> arXiv:2211.11682.
- [38] Ravi, N., Reizenstein, J., Novotny, D., Gordon, T., Lo, W. Y., Johnson, J., & Gkioxari, G. (2020). Accelerating 3d deep learning with pytorch3d. *arXiv preprint arXiv:2007.08501*. Doi:10.48550/arXiv.2007.08501.
- [39] N. Cui, H. Chen, X. Guo, Y. Zeng, Z. Hua, G. Xiong, R. Yue, J. Liu, Self-prompting semantic segmentation of bridge point cloud data using a large computer vision model, *Autom. Constr.* 167 (2024) 105729, <https://doi.org/10.1016/j.autcon.2024.105729>.
- [40] L. Truong-Hong, R. Lindner, Automatically extracting surfaces of reinforced concrete bridges from terrestrial laser scanning point clouds, *Autom. Constr.* 135 (2022) 104127, <https://doi.org/10.1016/j.autcon.2021.104127>.
- [41] S. Chen, L. Truong-Hong, D. Laefer, E. Mangina, Automated bridge deck evaluation through UAV derived point cloud, in: *CERI-ITRN2018*, 2018, August, pp. 735–740. <http://hdl.handle.net/2451/43478>.
- [42] Z. Dong, B. Yang, P. Hu, S. Scherer, An efficient global energy optimization approach for robust 3D plane segmentation of point clouds, *ISPRS J. Photogramm. Remote Sens.* 137 (2018) 112–133, <https://doi.org/10.1016/j.isprsjprs.2018.01.013>.
- [43] C.R. Qi, H. Su, K. Mo, L.J. Guibas, Pointnet: Deep learning on point sets for 3d classification and segmentation, in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 652–660, <https://doi.org/10.48550/arXiv.1612.00593>.
- [44] C.R. Qi, L. Yi, H. Su, L.J. Guibas, Pointnet++: deep hierarchical feature learning on point sets in a metric space, *Adv. Neural Inf. Proces. Syst.* 30 (2017), <https://doi.org/10.48550/arXiv.1706.02413>.
- [45] J. Martens, T. Blum, J. Blankenbach, Cross domain matching for semantic point cloud segmentation based on image segmentation and geometric reasoning, *Adv. Eng. Inform.* 57 (2023) 102076, <https://doi.org/10.1016/j.aei.2023.102076>.
- [46] G. Bradski, S. Grossberg, Recognition of 3-d objects from multiple 2-d views by a self-organizing neural architecture, in: *From Statistics to Neural Networks: Theory and Pattern Recognition Applications*, Springer Berlin Heidelberg, Berlin, Heidelberg, 1994, pp. 349–375, [https://doi.org/10.1007/978-3-642-79119-2\\_17](https://doi.org/10.1007/978-3-642-79119-2_17).
- [47] H. Su, S. Maji, E. Kalogerakis, E. Learned-Miller, Multi-view convolutional neural networks for 3d shape recognition, in: *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 945–953, <https://doi.org/10.48550/arXiv.1505.00880>.
- [48] Esteves, C., Xu, Y., Allen-Blanchette, C., & Daniilidis, K. (2019). Equivariant multi-view networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (pp. 1568–1577). Doi:10.48550/arXiv.1904.00993.
- [49] X. Wei, R. Yu, J. Sun, View-GCN: view-based graph convolutional network for 3D shape analysis, in: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 1850–1859, <https://doi.org/10.1109/CVPR42600.2020.00192>.
- [50] J. Jiang, D. Bao, Z. Chen, X. Zhao, Y. Gao, MLCNN: Multi-loop-view convolutional neural network for 3D shape retrieval, in: *Proceedings of the AAAI Conference on Artificial Intelligence* Vol. 33, No. 01, 2019, pp. 8513–8520, <https://doi.org/10.1609/aaai.v33i01.33018513>.
- [51] A. Hamdi, S. Giancola, B. Ghanem, Mvtn: Multi-view transformation network for 3d shape recognition, in: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 1–11, <https://doi.org/10.48550/arXiv.2011.13244>.
- [52] A. Dai, M. Nießner, 3dmv: Joint 3d-multi-view prediction for 3d semantic scene segmentation, in: *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 452–468, <https://doi.org/10.48550/arXiv.1803.10409>.
- [53] J. Cen, Z. Zhou, J. Fang, C. Yang, W. Shen, L. Xie, D. Jiang, X. Zhang, Q. Tian, Segment anything in 3d with nerf, *Adv. Neural Inf. Proces. Syst.* 36 (2023) 25971–25990, <https://doi.org/10.48550/arXiv.2304.12308>.
- [54] J. Amanatides, A. Woo, A fast voxel traversal algorithm for ray tracing, in: *Eurographics* Vol. 87, No. 3, 1987, August, pp. 3–10, <https://doi.org/10.2312/egtp.19871000>.
- [55] M. Himmelsbach, F.V. Hundelshausen, H.J. Wuensche, Fast segmentation of 3D point clouds for ground vehicles, in: *2010 IEEE Intelligent Vehicles Symposium*, IEEE, 2010, pp. 560–565, <https://doi.org/10.1109/IVS.2010.5548059>.
- [56] B. Yang, Z. Dong, G. Zhao, W. Dai, Hierarchical extraction of urban objects from mobile laser scanning data, *ISPRS J. Photogramm. Remote Sens.* 99 (2015) 45–57, <https://doi.org/10.1016/j.isprsjprs.2014.10.005>.
- [57] Zhang, C., Han, D., Qiao, Y., Kim, J. U., Bae, S. H., Lee, S., & Hong, C. S. (2023). Faster segment anything: towards lightweight sam for mobile applications. *arXiv preprint arXiv:2306.14289*. Doi:10.48550/arXiv.2306.14289.