Yinan Wang
Mem. ASME
Department of Industrial
and Systems Engineering,
Rensselaer Polytechnic Institute,
Troy, NY 12180
e-mail: wangy88@rpi.edu

Wenbo Sun
Transportation Research Institute,
University of Michigan,
Ann Arbor, MI 48109
e-mail: sunwbgt@umich.edu

Jionghua (Judy) Jin
Fellow ASME
Department of Industrial
and Operations Engineering,
University of Michigan,
Ann Arbor, MI 48109
e-mail: jhjin@umich.edu

Zhenyu (James) Kong
Fellow ASME
Grado Department of Industrial
and Systems Engineering,
Virginia Tech,
Blacksburg, VA 24060
e-mail: zkong@vt.edu

Xiaowei Yue[1]
Mem. ASME
Grado Department of Industrial
and Systems Engineering,
Virginia Tech,
Blacksburg, VA 24060
e-mail: xwy@vt.edu

MVGCN: Multi-View Graph Convolutional Neural Network for Surface Defect Identification Using Three-Dimensional Point Cloud

Surface defect identification is a crucial task in many manufacturing systems, including automotive, aircraft, steel rolling, and precast concrete. Although image-based surface defect identification methods have been proposed, these methods usually have two limitations: images may lose partial information, such as depths of surface defects, and their precision is vulnerable to many factors, such as the inspection angle, light, color, noise, etc. Given that a three-dimensional (3D) point cloud can precisely represent the multidimensional structure of surface defects, we aim to detect and classify surface defects using a 3D point cloud. This has two major challenges: (i) the defects are often sparsely distributed over the surface, which makes their features prone to be hidden by the normal surface and (ii) different permutations and transformations of 3D point cloud may represent the same surface, so the proposed model needs to be permutation and transformation invariant. In this paper, a two-step surface defect identification approach is developed to investigate the defects' patterns in 3D point cloud data. The proposed approach consists of an unsupervised method for defect detection and a multi-view deep learning model for defect classification, which can keep track of the features from both defective and non-defective regions. We prove that the proposed approach is invariant to different permutations and transformations. Two case studies are conducted for defect identification on the surfaces of synthetic aircraft fuselage and the real precast concrete specimen, respectively. The results show that our approach receives the best defect detection and classification accuracy compared with other benchmark methods. [DOI: 10.1115/1.4056005]

Keywords: 3D point cloud analytics, graph convolutional neural network, defect identification, sensing, monitoring and diagnostics

1 Introduction

The surface defects (e.g., scratch, dent, and protrusion) caused during manufacturing processes, such as machining or additive manufacturing, may further cause structural failures due to the reduced material strength and the excessive local stress around the defective region [1,2]. Surface defect identification is an important task in the fields of advanced manufacturing and maintenance. Human-based visual inspection is traditionally used in surface quality monitoring. However, such inspection is often time-consuming and has some subtle defects being ignored. With the rapid development of sensing and data analytics methods, automatic sensing-based inspection techniques have been increasingly used because of their high accuracy and efficiency.

Various types of sensors have been equipped in advanced manufacturing systems to facilitate automatic quality inspection and process control. Image-based quality inspection is one of the popular ways of defect detection and identification [3]. For example, techniques in digital image processing, such as various filters [4–6], wavelet transformation [7,8], and morphological algorithms [9], have been adapted into defects detection; in addition, image decomposition methods are developed to inspect defective regions [10,11]; advanced machine learning methods have also shown their strength in detecting and classifying diverse surface defects from inspection images [12–15]. However, the noise level of the images, which may arise from inevitable disturbances such as the light, angle, color, sensor stability, etc., has a significant influence on the accuracy of image-based inspection methods. For example, light and color usually vary among different environments and objects, resulting in a reduction in the accuracy and generalization ability of image-based methods. Moreover, two-dimensional (2D) image inspections lead to a loss of three-dimensional (3D) information. Alternatively, the advances in 3D laser scanning technologies can measure the 3D point cloud of objects, and the examples of the 3D point cloud are shown in Fig. 8. Compared with images, the 3D point cloud has three advantages: (1) it can quickly and accurately measure the 3D coordinates of object surface; (2) the measurement performance is robust to the object's surface colors and light variations/disturbances; and (3) the resolution of 3D scanner (density of points) can be adjusted according to the dimension of the potential defects. Therefore, increasing research interests have arisen in using 3D point cloud measurements for product surface defect inspection [16–18].

Although 3D point cloud has many advantages to benefit surface defect identification, there are still three critical challenges in data analysis of 3D point cloud: (1) The 3D point cloud is unstructured, which means that applying the data transformations, such as permutations, rotations, or translations, to a point set might change the coordinates and the order of points but will not affect the shape of the corresponding object. We also demonstrate the

[1]Corresponding author.
Manuscript received September 6, 2021; final manuscript received October 11, 2022; published online December 2, 2022. Assoc. Editor: Ran Jin.

transformation invariance in Fig. 1 to better illustrate this property, in which three columns from left to right demonstrate the rotation, translation, and permutation invariances, respectively. Therefore, the proposed method should be invariant to different transformations of the same set of points; (2) the defective regions are sparsely distributed over the non-defective surface. Thus, the defect's characteristics are prone to be diluted by the non-defective points around the defective area; (3) each sample of the 3D point cloud contains at least thousands of points, which makes pointwise manual data labeling infeasible or inapplicable in practice. To the best of our knowledge, there is little work successfully addressing all these challenges simultaneously. The existing methods either sacrifice the 3D information or incorporate human efforts in feature extraction, such as projecting the 3D point cloud into different 2D planes to transform the 3D information into a set of images [19] or building the classification model based on manually extracted features or labeled points [20].

To address the aforementioned challenges, we propose a unified machine learning framework to conduct surface defect identification using the 3D point cloud. The contributions of the proposed methodology include:

(1) A two-step method consisting of defect detection and classification is proposed to tackle the sparsity of defects, in which the first detection step is used to detect potential defective regions for separating the local area of defective regions from the global area of non-defective regions, and the second classification step is used to further identify the type of defects detected in the first step.
(2) A multi-view graph convolutional neural network (MVGCN) is proposed as the classification method to extract and exploit features from the detected local view (defective regions) and global view (non-defective regions) separately. A novel unsupervised learning method is proposed for defect detection, which does not require manual labeling. The influence of the detection results on the classification performance is also discussed.
(3) Both defect detection and classification are conducted by graph-based methodologies to handle the 3D point cloud directly. The selection of the number of neighbors in building graphs is discussed. We also proved theoretically that the proposed method is invariant to different permutations and transformations of the 3D point cloud.

It is worth noting that although they share a similar name, our proposed MVGCN is different from multi-view learning (MVL) [21]. Intuitively speaking, the methods proposed in the MVL focused on fusing different features for the same object. For example, when predicting citywide crowd flow, multi-view features include weather, time of the day, multi-scale temporal correlation (daily, weekly, monthly, and quarterly), etc. [22]; when predicting the possible links among nodes in a graph, the researchers proposed to both consider the current linkage relationship among nodes and the features of each node [23]; when conducting clustering, the raw information of each node is considered as one view and the transformations of the raw information (e.g., fast Fourier transform (FFT)) are considered as other views [24]. Differently, the multi-view in our work refers to different regions of the 3D point cloud, for example, the defective and non-defective regions in the 3D point cloud. The motivation to extract features from different regions is to preserve the informative features in the sparsely distributed defective regions and prevent them from being diluted by the features from the non-defective regions.

The remainder of this paper is organized as follows. Section 2 reviews existing research works for surface defect identification and related applications using the 3D point cloud. Section 3 proposes the two-step framework for surface defect identification and illustrates its properties in dealing with the 3D point cloud. In Sec. 4, we demonstrate the performance of the proposed framework on the 3D point cloud dataset of synthetic fuselage surface, which is further compared with the state-of-the-art baseline methods. In Sec. 5, we validate the strength of our proposed method on the real 3D point cloud dataset collected by scanning the surface of the precast concrete specimen. Finally, we conclude this paper in Sec. 6.

2 Literature Review

Surface defect identification has been studied based on various formats of measurement data. In this section, we mainly focus on surveying the state-of-the-art surface defect identification works

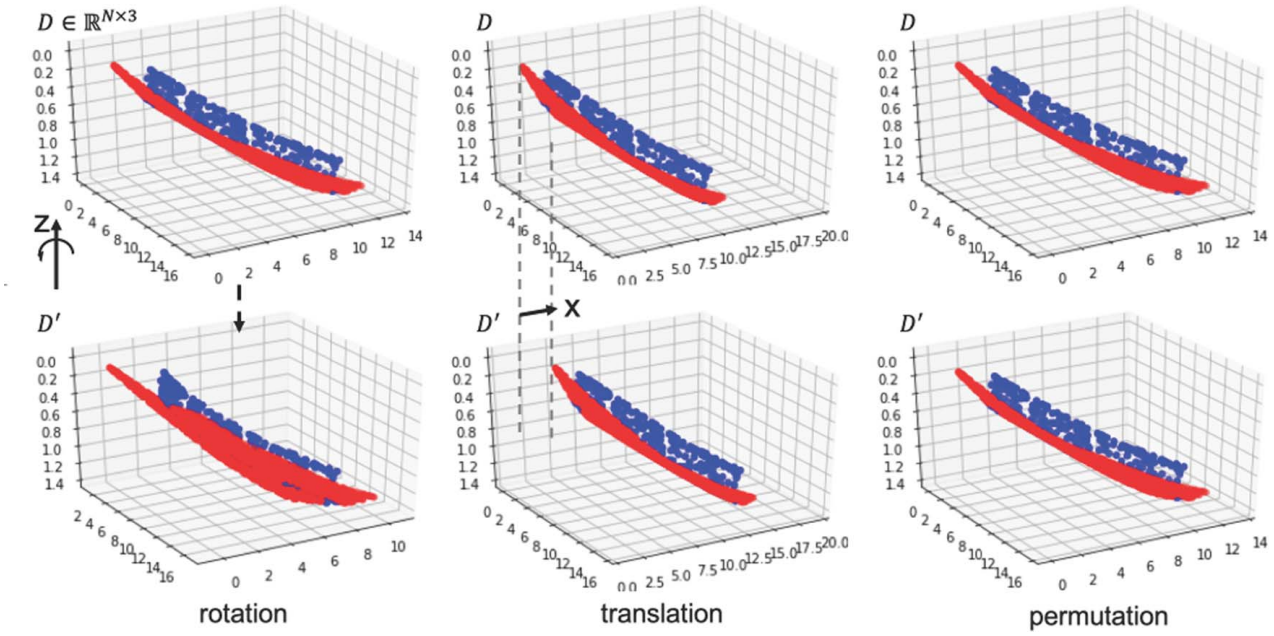

Fig. 1 Demonstration of transformation invariance property. Matrix *D* stores the set of points in the 3D point cloud, and its entries will change after rotation, translation, and permutation (either the value or order), but the represented object keeps the same.

using 3D point clouds (Sec. 2.1) and introduce some related applications of 3D point clouds in other related domains (Sec. 2.2). The limitation of these methods is also discussed at the end of each sub-sections.

2.1 Three-Dimensional Point Cloud-Based Surface Defect Identification. There are increasing research interests in conducting surface defect inspections using 3D point cloud measurements. For example, an automatic vision-based inspection system was proposed for detecting and characterizing defects on the airplane exterior surface, in which the undesired defects were detected by the curvature and normal direction information calculated over a given 3D point cloud [16]. The iterative closest point (ICP) algorithm was applied to analyze 3D point cloud data for assessment of the dimensional accuracy of manufactured parts [18]. In addition, a new approach based on the spectral graph theory was proposed to quantify the dimensional deviations and to localize the spatial defects in additive manufacturing [17]. Different types of dimensional deviations were further classified using the selected machine learning approaches [20]. Furthermore, the local surface curvature extracted by the principal component analysis (PCA) was applied to detect the damages on cooling tower shells [25].

The aforementioned works are mainly based on manually extracted or selected features to conduct defect detection and classification, which is not efficient for real-time surface defect identification or makes it hard to accurately capture 3D information on defects' shapes and dimensional size. The following sub-sections will survey the related 3D point cloud data analysis methods that have been recently developed in the machine learning field.

2.2 Survey of Three-Dimensional Point Cloud Analysis in Machine Learning. A general analysis framework for surface defect identification consists of two steps—defect detection and classification. Most existing works introduced in Sec. 2.1 focused on detecting defects or anomaly regions without further classifying them into different classes. Recently, machine learning models for 3D point cloud classification have been rapidly developed, which mainly follow three technical routes: image-based, voxel-based, and point-cloud-based. Image-based models at first transform 3D point cloud datasets into a set of 2D images by projection or rendering and then apply image classification methods, such as using CNN to analyze 2D images [26]. Differently, the voxel-based method is to represent a 3D point cloud with a 3D voxel grid, which is classified using 3D CNN [27]. It is worth noting that both image-based and voxel-based methods try to represent the unstructured 3D point cloud in a structured data format, such as matrices or tensors. Such a representation enables the application of existing models to the 3D point cloud, but it also leads to extra computational costs in preprocessing the 3D point cloud and an information loss during the transformation. In contrast, the recently developed PointNet was one attempt to directly classify the unstructured 3D point cloud using the multi-layer perceptron as the building block [28]. Furthermore, graph-based neural work, such as the dynamic graph convolutional neural network (DGCNN) [29], was also proposed to work directly on 3D point cloud. These models preserve the complete 3D information and extract features directly from the unstructured 3D point cloud for classification. Thus, they also can be invariant to different permutations and transformations of 3D point clouds.

In the literature, these 3D point cloud classification models are mainly designed to identify the shapes of various objects (such as chairs, tables, planes, cars, etc.), which is generally an easier task than distinguishing different types of random surface defects. There are unique challenges in identifying various types of surface defects. First, the defects are usually sparsely distributed over the non-defective surface. Thus, compared with classifying various types of objects, the informative features used to identify different defects are rather sparse. Furthermore, surface defects' features are easy to be diluted by the features representing the overall shape when using the existing single-step classification model. Thus, the existing methods are not directly applicable to the surface defect classification. In the next section, we propose a new two-step surface defect identification framework, which aims to detect and classify the subtle differences among 3D point clouds that contain various types of surface defects.

3 Proposed Methods for Surface Defect Identification Using Three-Dimensional Point Cloud

Surface defects are often sparsely distributed over free-form surfaces, which makes the features of surface defects prone to be hidden by non-defective regions. To overcome this obstacle, we propose a two-step surface defect identification framework as shown in Fig. 2, which includes two steps analyses: (1) an unsupervised defect detection method by proposing an adaptive threshold method based on local surface variation (Sec. 3.1) and (2) a supervised defect classification by proposing MVGCN (Sec. 3.2). The term "multi-view" consists of a "global view" component that models the global variations from non-defective regions and a "local view" component that focuses on variations from defective regions. In this way, surface variations of defective and non-defective regions can be quantified separately.

3.1 Adaptive Threshold for Defect Detection Based on Local Surface Variation. In the first step of the proposed method, unsupervised surface defect detection is used to identify the potential defect points, which is a prerequisite for the following surface defect classification. As discussed in Sec. 1, the proposed defect detection method should be invariant to different transformations of the unstructured 3D point cloud (e.g., permutation) and insensitive to inevitable measurement noise. Moreover, it should not require labeled data for training. To meet these requirements, we propose an adaptive thresholding method by incorporating K-nearest neighbor (KNN) graph model [30], which will be discussed in detail in the following sub-sections.

3.1.1 Distance Measures in Building K-Nearest Neighbor Graphs. The raw 3D point cloud measured/collected by a laser scanner is usually a set of unstructured points stored in a matrix format. Suppose that $D \in \mathbb{R}^{N\times 3}$ represents the raw 3D point cloud, in which N is the number of points, and three columns in matrix D correspond to the 3D Cartesian coordinates of $\mathbf{x} = (x, y, z)$. To guarantee that the set of 3D point cloud is invariant to permutations of the points, a KNN graph is built for each point based on its distances to the closest k' neighboring points. There are different measures of pairwise distance in the 3D point cloud. In this

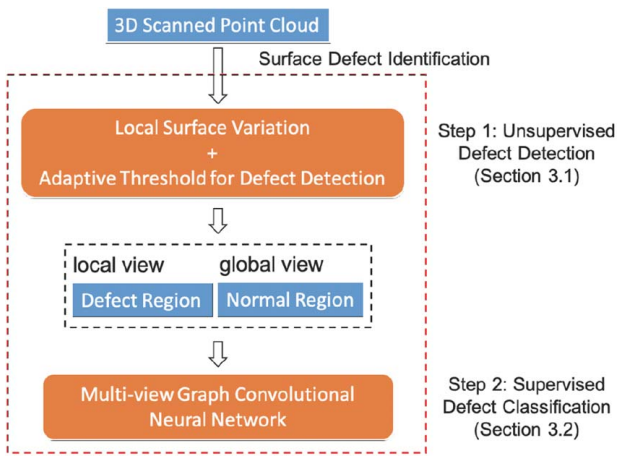

Fig. 2 Framework of surface defect identification

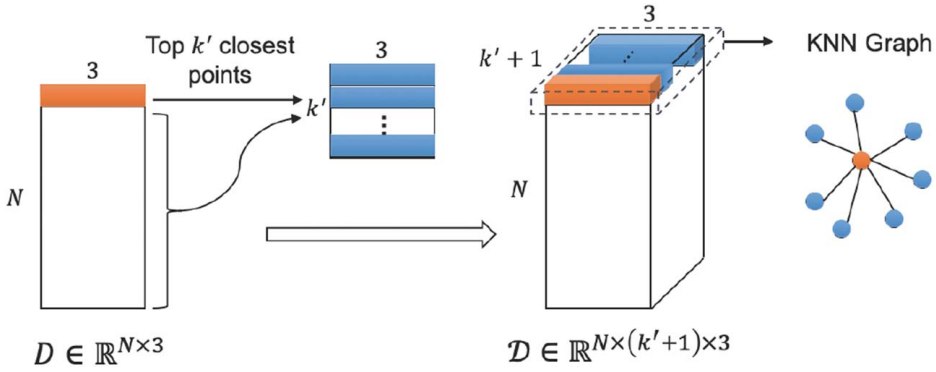

Fig. 3 The process to build KNN graph

paper, Euclidean and geodesic distances [31] are selected to measure the pairwise distance among points when building the KNN graphs. The influence of these two distance measures on the performance of unsupervised defect detection is discussed in Sec. 4.3.

The idea of geodesic distance is to use the length of the shortest curve between two points on the 3D surface as their distance. It can quantify the distance more accurately by fully exploiting the geometric structure on the target surface, while the Euclidean distance sacrifices partial geometric information when evaluating the distance. However, calculating the geodesic distance is more time-consuming because it requires first reconstructing the surface from the 3D point cloud and then finding the length of the shortest curve. In this paper, the heat method is used to approximate the pairwise geodesic distance from the reconstructed 3D surface [32].

3.1.2 Local Surface Variation Based on K-Nearest Neighbor Graph. The reorganized 3D point cloud connects each point to the top k' closest points and forms a star graph. The reorganized 3D point cloud is denoted as $\mathcal{D} \in \mathbb{R}^{N\times(k+1)\times 3}$. The process of building KNN graphs on the 3D point cloud is demonstrated in Fig. 3.

KNN graphs store the local neighbors of each point in the 3D point cloud. Compared with non-defective surfaces, which are mostly smooth, a surface with defects has more considerable variations. Intuitively, variations indicate the roughness of the surface. The next step is to estimate the local surface variation at each point, which is determined by the center point and its neighbors. Suppose that the point set X containing one center point $\mathbf{x}$ and its k' nearest local neighbors are defined as $X \in \mathbb{R}^{(k'+1)\times 3}$. The covariance matrix of X is defined as $\mathrm{Cov}(X) = \frac{1}{k'+1}(X-\overline{X})^{\top}(X-\overline{X})$, where the $\overline{X}$ represents the centroid of the point set. The local surface variation [33] at point $\mathbf{x}$ in a neighborhood of size k' is defined as

$$\sigma_{k'}(\mathbf{x}) = \frac{\lambda_0}{\lambda_0+\lambda_1+\lambda_2} \tag{1}$$

where $\lambda_0 \le \lambda_1 \le \lambda_2$ are the three eigenvalues of the matrix $\mathrm{Cov}(X)$. Among these three eigenvalues, λ_1 and λ_2 correspond to the eigenvectors spanning the tangent plane at point $\mathbf{x}$, and λ_0 corresponds to the eigenvector approximating the surface normal at point $\mathbf{x}$. Thus, $\sigma_{k'}(\mathbf{x})$ indicates the variation along the normal direction at point $\mathbf{x}$, and a larger value indicates a more significant variation. When $\sigma_{k'}(\mathbf{x}) = 0$, all the points in X are on the same plane. When $\sigma_{k'}(\mathbf{x}) = 1/3$, all the points in X are completely isotropically distributed. In Proposition 1, we show that the defined local surface variation is a feature invariant to rotations and translations on X. The proof of Proposition 1 is given in Appendix A.

Proposition 1. *Let X be the point set covering all the k' nearest neighbors of* x. *The local surface variation $\sigma_{k'}(\mathrm{x})$ is invariant with respect to any rotation and translation of X.*

Note that point set X is formulated by the KNN graph, which is invariant to different permutations of the 3D point cloud. Consequently, the local surface variation is also invariant to the permutation of the point set X.

3.1.3 Adaptive Thresholding Defect Detection. Based on Eq. (1), the local surface variation is calculated over all the KNN graphs in $\mathcal{D}$ and makes it a pointwise feature to the original 3D point cloud. We use $\Sigma_{k'}(D) = \{\sigma_{k'}(\mathbf{x}), \mathbf{x} \in \mathcal{D}\}$ to denote the set of pointwise local surface variations of the 3D point cloud. From the definition of local surface variation, we know that λ_0 quantitatively represents the variation along the normal direction of the tangent plane. We have $\sigma_{k'}(\mathbf{x}) \in [0, 1/3]$, and the spatial variation of the points increases with the increase of $\sigma_{k'}(\mathbf{x})$. The assumptions of the proposed method can be summarized in three aspects: (1) the proposed framework is designed for surface defects identification for the large-scale product, and most of its surface is smooth or only with slightly complex geometry designs; (2) the measurement noise, geometry designs, and subtle structures on the large-scale product have a similar influence on the local surface variations for both non-defective and defective surfaces; and (3) the local surface variations of points around defects are usually higher than those on a non-defective surface.

The unsupervised method is preferable for surface defect detection because it does not require a time-consuming labeling process. The threshold-based method is selected to detect defects by clustering all the points into two groups, which is defined in Eq. (2).

$$f(\mathbf{x}) = \begin{cases} 1, & \sigma_{k'}(\mathbf{x}) \ge \delta \\ 0, & \sigma_{k'}(\mathbf{x}) < \delta \end{cases} \tag{2}$$

In Eq. (2), the value of $f(\mathbf{x})$ represents either a point $\mathbf{x}$ belongs to the defective regions ($f(\mathbf{x}) = 1$) or the non-defective surface ($f(\mathbf{x}) = 0$), and δ represents the threshold.

The overall process to distinguish defective regions from non-defective ones is summarized in Fig. 4, in which the most critical issue is to determine an appropriate threshold by considering the natural measurement noise (i.e., equipment and environment noise) and other local subtle structural variations (i.e., joint rivets) in the surface. We propose to use the statistics (e.g., the upper bound of 80% confidence interval) of local surface variations on the non-defective surface as the threshold. The rationales of this adaptive thresholding method are justified by the following four folds: (1) The 3D point cloud of defective and non-defective surfaces is collected using the same equipment under the same environment. So the increase of local surface variation caused by the measurement noise has little influence and can be ignored; (2) when the environment or equipment changes, we can repeat the data collection of non-defective surface to adaptively calibrate the threshold using updated statistics; (3) the selected statistics of the local surface variations are determined by the major features of the non-defective surface, such as the smooth surface and commonly appeared subtle structures (i.e., rivet joints), so as to

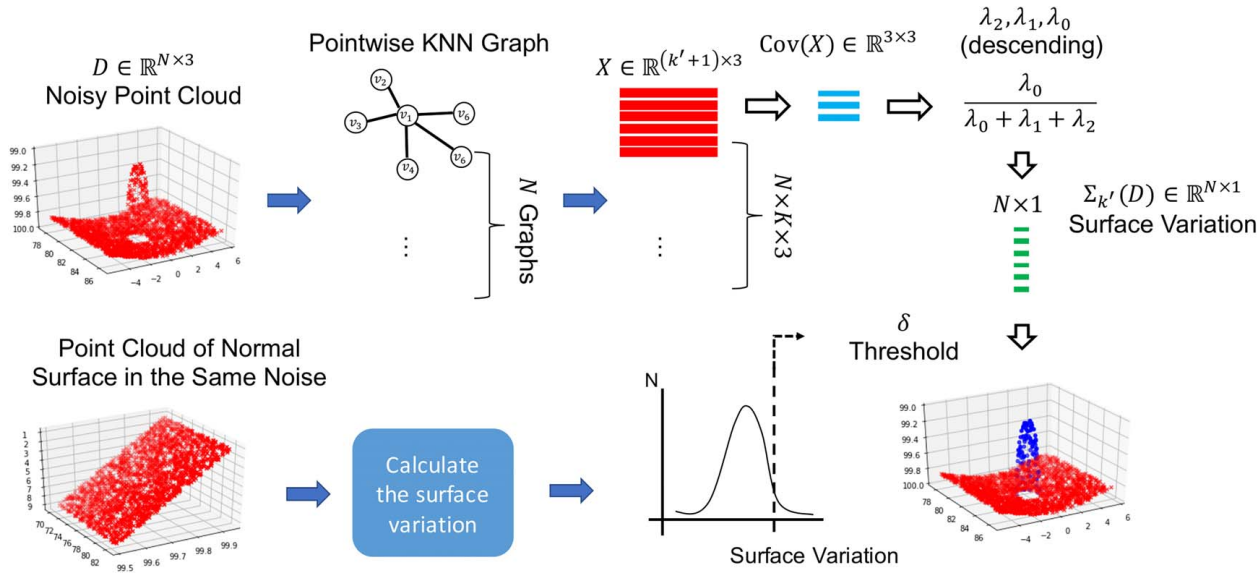

Fig. 4 Visualization of adaptive threshold defect detection

eliminate the influence from outliers; and (4) the threshold calibration can be efficiently conducted either on the experimental non-defective samples or on the non-defective area in the target product identified by the domain expert.

3.2 Multi-view Graph Convolutional Neural Network for Defect Classification. Based on the results from the first step of unsupervised defects detection (Sec. 3.1), a new MVGCN classifier is proposed in the second step for defect classification. The proposed MVGCN overcomes the following two challenges in defect classification using 3D point clouds: (1) the classification model should be invariant to different permutations and transformations of 3D point clouds and (2) the defects are randomly scattered over the inspected surface, which causes the difficulty to capture the subtle features of sparse defects. To better illustrate the proposed MVGCN, the entire process of feature extraction and defect classification is visualized in Fig. 5, which consists of graph convolutional layer (GCL), multi-view graph convolutional layer (MVGCL), and fully-connected (FC) layer. In this section, we first introduce the formulation of GCL and then demonstrate the specific design of MVGCL for defect classification.

3.2.1 Graph Convolutional Neural Network. As introduced in Sec. 3.1.1, to keep the representation of 3D point cloud to be invariant to different permutations of points, the KNN graph model is used to reformulate 3D point cloud as $\mathcal{D} \in \mathbb{R}^{N\times(k+1)\times 3}$, which can be regarded as a big graph that consists of the N number of KNN

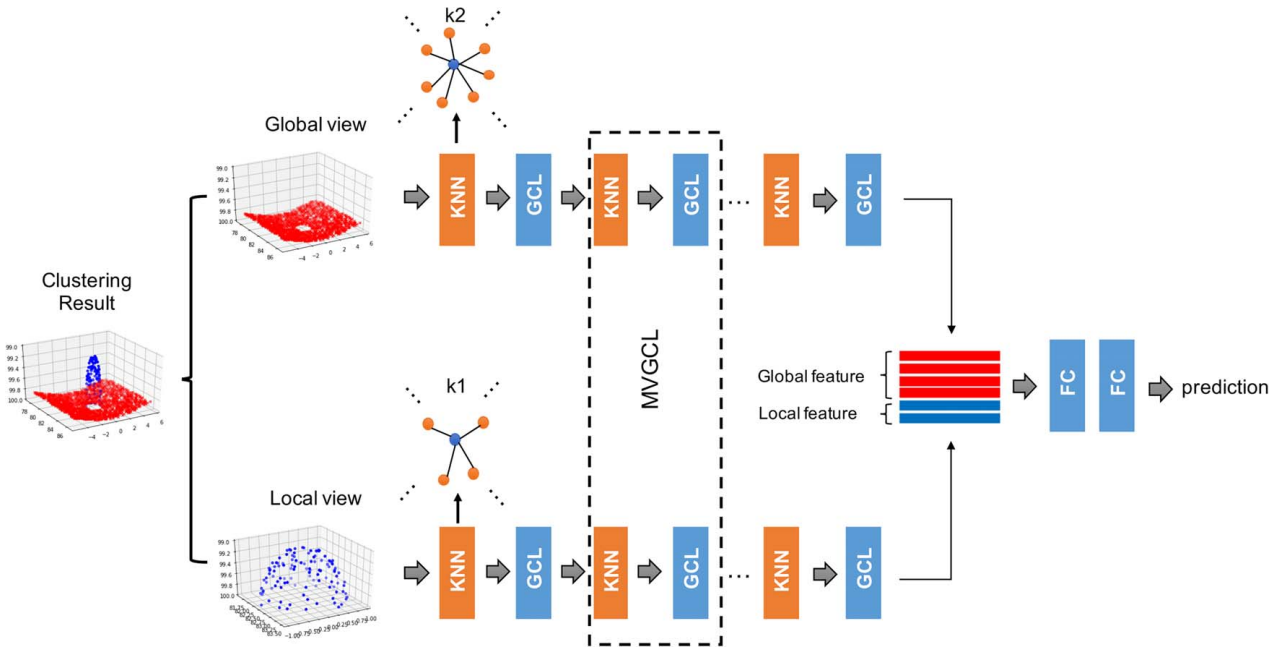

Fig. 5 Visualization of multi-view graph convolutional neural network. The detected defective region (local view) and non-defective region (global view) are fed into the proposed MVGCL, which are kept disjoint during the feature extraction over multiple MVGCLs. The extracted global feature and local feature are then concatenated and further fed into the FC layer for feature fusion and prediction.

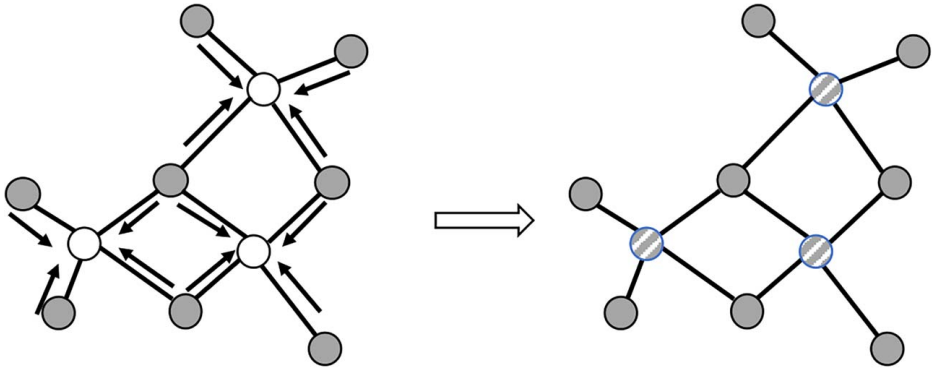

Fig. 6 Visualization of graph convolution operation on three center nodes (nodes in white). The features from neighbors (nodes in gray) propagated through edges and aggregated with the center nodes (nodes in white).

graphs for all the N points. Because the topology of the KNN graph contains the geometric features around the corresponding center point, it is essential to consider the graph topology when designing the model. Therefore, the graph convolutional neural network (GCN) [29] is adapted to extract the geometric features over the pre-built KNN graph for classification. It is worth noting that we also consider Euclidean distance or geodesic distance when building the KNN graphs. The influence of these two distance measures on the performance of supervised defect classification is discussed in Sec. 4.4.

The information flow within a single graph convolutional layer is visualized in Fig. 6, in which we set $k=4$ and select three center points (white nodes on the left) as examples to demonstrate how the information from adjacent nodes (gray nodes) influences the center points. If we stack multiple GCLs together, the output from the previous layer will be fed into the next layer, which enables the information from distant nodes to propagate over multiple hops to influence a node with no direct connection. The mathematical expression of the lth GCL is shown in Eq. (3).

$$\begin{aligned} \mathbf{x}_i^l &= \sum_{(i,j)\in\mathcal{D}} h\left(\Theta_1^l(\mathbf{x}_j^{l-1}-\mathbf{x}_i^{l-1})+\Theta_2^l\mathbf{x}_i^{l-1}\right) \\ X^l &= \left[\mathbf{x}_1^l,\ldots,\mathbf{x}_N^l\right]^\top \end{aligned} \tag{3}$$

where $\mathbf{x}_i^l \in \mathbb{R}^{d_l}$ represents the feature vector of the node i generated by the lth layer, and we have the input of node i in the first layer as $\mathbf{x}_i^0=(x_i, y_i, z_i)$ and $X^0=X\in\mathbb{R}^{N\times 3}$ as the entire point set in the first layer; $X^l\in\mathbb{R}^{N\times d_l}$ represents the output features from the lth layer, and d_l is the dimension of output features at the lth layer; $\mathcal{D}$ represents the entire graph, (i, j) represents the edge with one end connecting to node i and the other end connecting to its neighbors j, and $\mathbf{x}_j^{l-1}$ represents the features of its neighbors at layer $(l-1)$; Θ_1^l, Θ_2^l represent the trainable parameters in the lth layer, and $h(.)$ represents the activation function, such as ReLU. The aggregation of $\Theta_1^l(\mathbf{x}_j^{l-1}-\mathbf{x}_i^{l-1})$ and $\Theta_2^l\mathbf{x}_i^{l-1}$ ensures that the graph convolutional operation captures both the local differences by comparing the center point with its direct neighbors and the global geometries by propagating the features from distant nodes.

3.2.2 Multi-View Graph Convolutional Neural Network. From the perspective of defect classification, the model should pay sufficient attention to the features of defects. However, due to the sparsity of defective regions, it is likely that the center point of a KNN graph belongs to the defective region while its neighbors belong to the non-defective surface. In this case, the resultant KNN graph is mixed with a defective center point and non-defective neighbor points. Based on Eq. (3), the features of the defective regions will be diluted by the features of the non-defective neighbor points. Furthermore, with the increment of layers in multi-layer GCN, the information from distant nodes will be propagated and aggregated through the edges. Consequently, every pair of points are reachable from each other no matter whether they are directly connected or not. Therefore, the information between every pair of points, no matter whether they are defective or non-defective, can influence each other. To tackle this problem, we propose to cut the entire graph into two sub-graphs based on the clustering results from the first step using the unsupervised defect detection method (Sec. 3.1). We refer to the sub-graph containing most of the points from the detected defective regions as the local view graph and the sub-graph containing points from the non-defective surface as the global view graph. These two sub-graphs are used to construct separate GCN models, in which the information in two sub-graphs is not connected to avoid unnecessary information exchange. These two sub-graphs are formally defined in Eq. (4).

$$\begin{cases} \mathbf{x}_i\in\mathcal{D}_{\text{local}}, & f(\mathbf{x}_i)=1 \\ \mathbf{x}_i\in\mathcal{D}_{\text{global}}, & f(\mathbf{x}_i)=0 \end{cases} \tag{4}$$

where $\mathcal{D}_{\text{local}}\in\mathbb{R}^{N_1\times(k_1+1)\times 3}$, $\mathcal{D}_{\text{global}}\in\mathbb{R}^{N_2\times(k_2+1)\times 3}$, $N_1+N_2=N$; and $f(.)$ is the adaptive threshold defect detection method introduced in Sec. 3.1.

The expression of the proposed MVGCL is given in Eq. (5).

$$\begin{aligned} \mathbf{x}_i^l &= f(\mathbf{x}_i)\sum_{(i,j)\in\mathcal{D}_{\text{local}}} h\left(\Theta_{1,\text{local}}^l(\mathbf{x}_j^{l-1}-\mathbf{x}_i^{l-1})+\Theta_{2,\text{local}}^l\mathbf{x}_i^{l-1}\right) \\ &\quad +(1-f(\mathbf{x}_i))\sum_{(i,j)\in\mathcal{D}_{\text{global}}} h\left(\Theta_{1,\text{global}}^l(\mathbf{x}_j^{l-1}-\mathbf{x}_i^{l-1})+\Theta_{2,\text{global}}^l\mathbf{x}_i^{l-1}\right) \\ X^l &= \left[\mathbf{x}_1^l,\ldots,\mathbf{x}_N^l\right]^\top \end{aligned} \tag{5}$$

where $\Theta_{1,\text{local}}$ and $\Theta_{2,\text{local}}$ are the set of trainable parameters used to extract features from the local view graph; $\Theta_{1,\text{global}}$ and $\Theta_{2,\text{global}}$ are used to extract features from the global view graph. The advantages of separately modeling the local and global view graphs include two aspects: (1) The center node and its neighbors all belong to either the defective regions (local view graph) or the non-defective surface (global view graph), which prevents the defect features from being blurred or diluted by features from non-defective areas and (2) the local and global view graphs are disjoint. Therefore, when increasing the number of layers in the MVGCN model, the features will only propagate and aggregate within each graph. In this way, the sparse defect features will not be influenced by features from distant non-defective areas in a deeper model.

The entire process of MVGCN is summarized in Fig. 5. The detected defective and non-defective regions from the first step are fed into the model and processed by the MVGCL. One example of MVGCL is demonstrated in the dashed box in Fig. 5. The extracted global features and local features are then

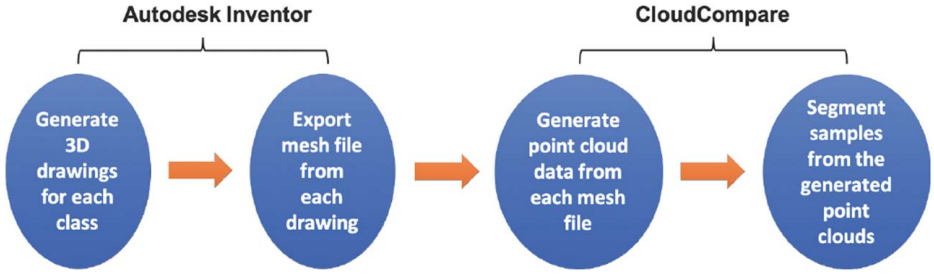

Fig. 7 The process of generating 3D point cloud data

concatenated and fused by the FC layers. The predicted defect type is finally generated using the fused features.

4 Case Study 1

In this section, we generate a synthetic 3D point cloud dataset that mimics the surface defects on the aircraft fuselage and validate the effectiveness of the proposed method. In the following context, we first introduce the procedure of generating and preprocessing the 3D point cloud dataset (Sec. 4.1). Then we test the performance of the proposed method on the synthetic dataset (Secs. 4.2, 4.3, and 4.4). The code and dataset will be released upon publication.

4.1 Surrogated Defects Generation

4.1.1 Build Computer-Aided Design Model to Generate Three-Dimensioanl Point Cloud With Surface Defects. The aircraft fuselage with a cylinder shape is used as an example in this study. We create a computer-aided design (CAD) model to mimic the defects on the cylinder surface of an aircraft fuselage. According to the visual inspection guidelines for aircraft [2], we select three typical types of symbolic surface defects, which are scratch, dent, and protrusion. This sub-section will discuss the process of building a CAD model to generate 3D point cloud data for each type of defect, which is summarized in Fig. 7.

Autodesk Inventor is used to generating CAD drawings of the fuselage and defects. Specifically, we create a cylinder having 200 inches in diameter and 100 inches in height. Then, different types of defects are manually created on the surface of the fuselage. The dimension range of each type of defect is summarized in Table 1, and the specific dimensions of individual defects vary randomly by following a uniform distribution within its corresponding range. Existing research works built up simulation models and found out that defects, especially dents, with such dimensions typically influence the mechanic properties of the metal fuselage, such as its compressive and shear failure loads [34,35]. Thus, accurately identifying these defects can prevent severe structural failures, reduce maintenance costs, and improve the reliability of the fuselage. We also mimic the possible real-world scenarios when generating these different types of defects. In particular, for the dent, we first create a solid object with an arbitrary shape within the given dimension range. Then we partially insert the solid object into the fuselage surface to mimic the process that one solid object hits the fuselage surface. Finally, we remove the intersection volume between the solid object and the fuselage to create one surrogated dent on the fuselage surface. Similarly, for the scratch, we generate every single scratch using an arbitrary solid object to scrape over the surface of the fuselage along a random trajectory. For the protrusion, we squeeze the cylinder surface and insert a solid object with an arbitrary shape to mimic the deformed protrusion above the fuselage surface.

Table 1 Dimension ranges of synthetic defects (inches)

Defect	Scratch	Dent	Protrusion
Width	[1, 3]	[1, 3]	[1, 3]
Depth	[0.5, 1.5]	[0.5, 2]	[0.5, 2]

4.1.2 Data Generation. After generating the CAD drawings of surface defects, the 3D point cloud is further sampled and segmented from these CAD drawings via CLOUDCOMPARE software. The examples of segmented 3D point cloud samples in each defect class are shown in Fig. 8.

In this case study, the surrogated 3D point cloud with surface defects contains a total of 200 samples, in which each of four classes (i.e., dent, scratch, protrusion, and non-defective) contains 50 samples. When initially segmenting these 200 samples, the number of points in each sample may vary and is controlled to contain slightly more than 2000 points. Then, random sampling is conducted on each sample to ensure the same sample size of 2000 points among all samples. As the initial number of points in each sample is controlled to be slightly more than 2000 during segmentation, randomly sampling 2000 points will not significantly influence the overall shape of the sample. The generated entire dataset is denoted as $\{(D_0, y_0), \ldots, (D_{199}, y_{199})\}$, in which $D_i \in \mathbb{R}^{2000\times 3}$ represents one sample of the 3D point cloud and y_i is the corresponding class label. Considering the shape of each 3D point cloud sample is only influenced by the relative positions of its points, we then center each data sample via $D_i - \overline{D}_i$, in which $\overline{D}_i$ represents the mean coordinates along each axis of the sample.

After segmenting 3D point cloud samples from the CAD drawings, the noise is added to these samples to consider the inevitable measurement errors in a practical scanning process. The measurement uncertainty is mainly influenced by two factors, the dimension of the target object and the accuracy of the 3D scanner. The measurement accuracy of the commercial 3D scanner is around $(14 + 14/m)\ \mu$m, which means that the base measurement error equals 14 μm plus the proportional error of 14 μm for every meter of the measured object's largest dimension. Recall that the dimension of the aircraft fuselage is set as 200 inches in diameter and 100 inches in height. According to the largest dimension of 200 inches, the scanning error of the fuselage is around 84 μm. Also, considering the range of 3D point cloud coordinates and the dimension of the fuselage, we have that one measurement unit length in the coordinate system is approximately 2.5 cm. The measurement error is thus around 0.003 units in the 3D point cloud coordinate system. Based on the analysis, we add the normally distributed noise ϵ with standard deviation σ ranging from 0 to 0.006 units to mimic the measurement errors under normal scanning conditions. The surrogated 3D point cloud is generated based on Eq. (6).

$$\begin{aligned} &\widehat{D}_i = (D_i - \overline{D}_i) + \epsilon \\ &\epsilon \in \mathcal{N}(0, \sigma I) \\ &\sigma = \{0, 0.001, 0.002, 0.003, 0.004, 0.005, 0.006\} \end{aligned} \tag{6}$$

4.2 Baseline Methods and Experimental Settings

4.2.1 Baseline Methods. To demonstrate the strength of the proposed method for defect detection and classification, several

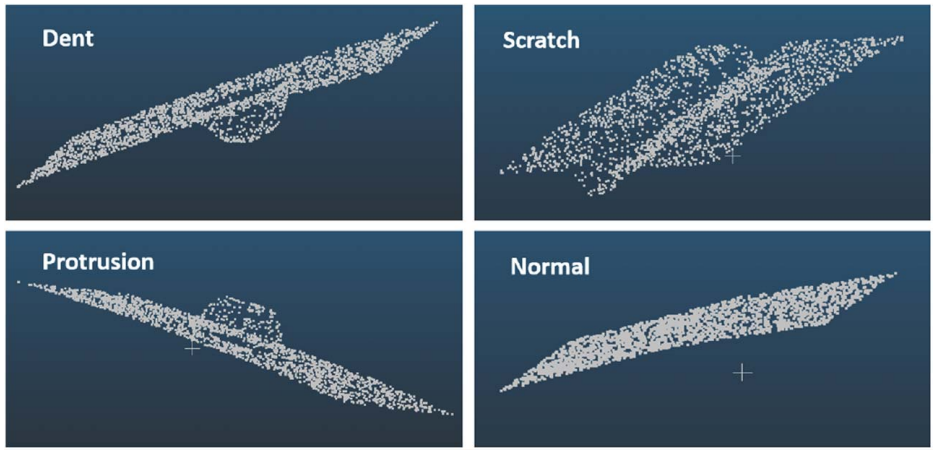

Fig. 8 Visualization of synthetic 3D point cloud

state-of-the-art methods are selected as the baseline models for performance comparison. For comparing different unsupervised defect detection methods, two clustering-based methods are selected, i.e., K-means clustering (KMC) [36] and hierarchical clustering [37] methods. The basic idea of these clustering methods is to group data points so that the formed clusters can minimize the dissimilarities among data points within the same cluster and, meanwhile, maximize the dissimilarities between different clusters. The dissimilarities among points can be measured by pre-defined metrics. In our case study, two types of metrics, i.e., pairwise Euclidean distance and pairwise normal direction vector similarity, are selected. The performances of baseline clustering methods are separately demonstrated by using these two metrics. The raw 3D point cloud contains the coordinate information, which can be directly used to calculate the Euclidean distance, and the normal vector of each point can be calculated by the PCA method [38]. For comparing different defect classification methods, PointNet [28] and DGCNN [29] are selected as two baseline methods. These two methods are cutting-edge methods in point cloud classification and have been widely validated in different datasets. The goal of selecting these two benchmark methods is to demonstrate the effectiveness of the proposed MVGCN in capturing the features of various types of sparsely distributed surface defects.

4.2.2 Experimental Settings. In the comparison studies, 160 samples are randomly selected from the total 200 samples as the training dataset to learn the parameters in the proposed MVGCN, and the rest 40 samples are used as the testing dataset to assess the model performance in defect classification. Different levels of noise are added to both training and testing datasets, as introduced in Sec. 4.1.2. The comparison analyses are repeatedly conducted on all datasets with different levels of noise.

The hyperparameters are carefully selected and optimized in the experiments. In unsupervised defect detection, the threshold δ is determined by the upper bound of 80% confidence interval of the pointwise local surface variation $\sigma_{k'}(\mathbf{x})$ that is calculated using non-defective training samples. The proper number of neighbors, namely, k', in the KNN graph is determined by qualitatively evaluating the detection results on the training samples. The selection of k' is discussed in Sec. 4.3.2. In the supervised defect classification, the hyperparameters are selected by the grid-search within a certain range. For example, the number of hidden layers in all methods is searched within 10, because a neural network with too many layers tends to overfit the training data. The number of neighbors k, k_1, k_2 in DGCNN and the proposed MVGCN are searched within 30, which is detailedly introduced in Sec. 4.4.3. To make a fair comparison among those models, these two sets of parameters are kept the same: (1) the number of layers in baseline methods PointNet and DGCNN, and the proposed MVGCN are set with the same value of $L=5$; (2) the number of neighbors in the DGCNN and the proposed MVGCN are equal, i.e., $k=k_1=k_2=15$.

We also investigated how different distance measures in building the KNN graphs influence the performance of unsupervised defect detection and supervised defect classification. Euclidean distance and geodesic distance are separately used in building KNN graphs, and the corresponding performance of the proposed MVGCN and the benchmark DGCNN are discussed and compared. Finally, the sensitivity analysis is conducted to investigate how the performance in unsupervised defect detection will influence the supervised defect classification. All the experiments are implemented by PyTorch [39] and run on NVIDIA GeForce GTX 1080Ti GPUs.

4.3 Defect Detection Results

4.3.1 Comparison of Defect Detection Results. We compare the defect detection performance among our proposed adaptive threshold clustering method and the two baseline methods introduced in Sec. 4.2.1. The comparison results are demonstrated in Fig. 9, in which the ground truth detection results are shown in the first row (dots for defective regions and crosses for non-defective regions). The proposed adaptive thresholding method is implemented by using Euclidean distance and geodesic distance to build KNN graphs, respectively. The detection results are shown in the second and third rows, from which we can see that the proposed adaptive thresholding method can successfully identify the defective regions and receive comparable performance when using different distance measures. For comparison, the K-means clustering method fails to identify all the defective regions when using the Euclidean distance (KMC-v1) among points. It can partially identify the defective regions when using the cosine similarities of normal vectors (KMC-v2). Furthermore, the incorrectly clustered points when using KMC-v2 are mainly from the bottom part of the defective regions. The normal vectors at the bottom of the defective regions have a similar direction to the normal vectors in the non-defective surface. Considering this phenomenon, the third baseline method is to use a two-step clustering method by successively applying the KMC-v2 and hierarchical clustering. The hierarchical clustering tries to further identify the bottom of the defective region from the "non-defective" region given by the KMC-v2. The results in the last row demonstrate that this two-step clustering method has some improvements in identifying dents but fails to detect other types of defects. In summary, our proposed unsupervised adaptive thresholding method outperforms all the baseline methods in defect detection, which provides accurate detection results for proceeding with the next defect classification using the MVGCN method.

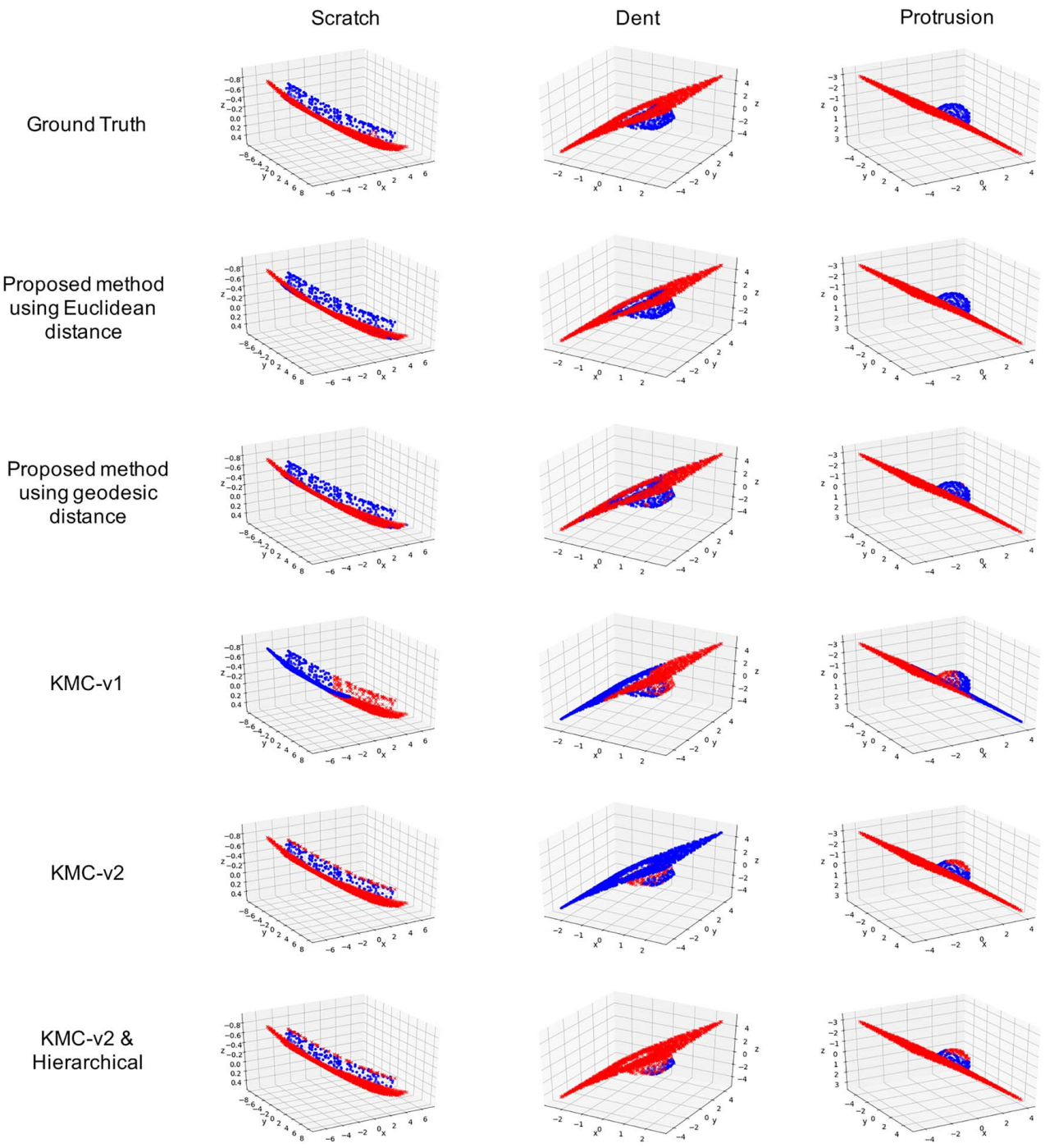

Fig. 9 Visualization of defect detection results

4.3.2 Influence of the Number of Neighbors in K-Nearest Neighbor Graphs on the Defect Detection. The proposed adaptive thresholding method for defect detection is directly influenced by the KNN graphs formulated from the 3D point cloud. The KNN graph is built for each point by finding its closest k' neighbors in the 3D point cloud. We further discuss how the number of neighbors k' in the KNN graph will influence the defect detection performance. The detection results under different values of k' are qualitatively demonstrated in Fig. 10, where we use the KNN graphs built upon the Euclidean distance as an example. From Fig. 10, we can conclude that when the number of neighbors in the KNN graph is small, such as $k' = 5$, the local surface variation calculated on it tends to underestimate the surface roughness at each point. Thus, some points in the defective region might be ignored. Such misidentification will be mitigated with the increase in the number of neighbors, which can be observed by comparing the detected results highlighted in rectangle dashed boxes. On the contrary, increasing the number of neighbors in building KNN graphs might overestimate the local surface variation in the non-defective region. Thus, some points might be misidentified as defects, which can be observed by comparing the detected results highlighted in round dashed boxes. In our case, we set $k' = 50$ to ensure that more points in defective regions are correctly identified while allowing misidentifying fewer points in the non-defective regions.

4.4 Defect Classification Results

4.4.1 Comparison of Defect Classification Results. The defect detection results are further fed into the proposed MVGCN for

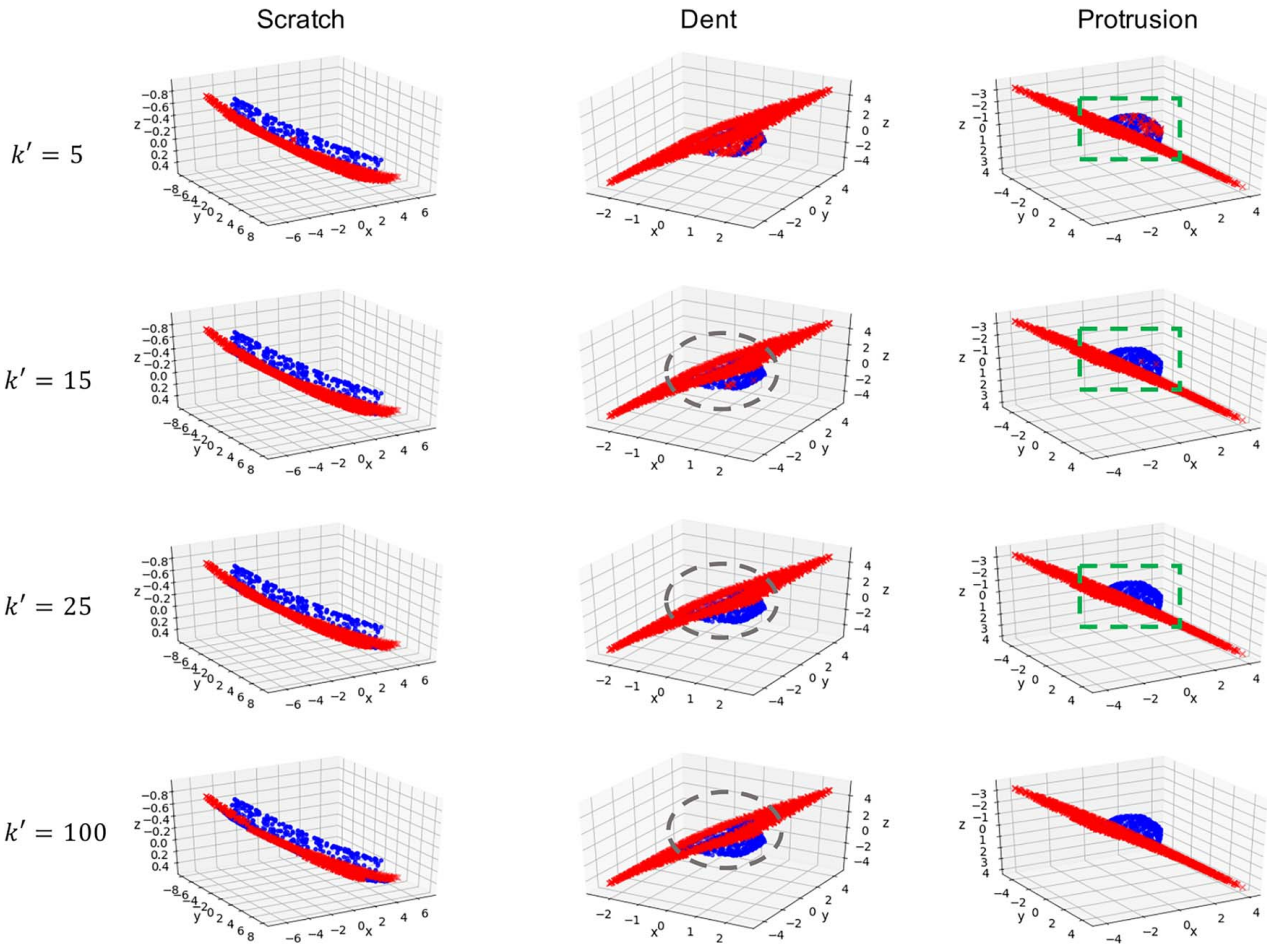

Fig. 10 Influence of the number of neighbors on the defect detection

defect classification, and its performance is compared with two baseline methods PointNet [28] and DGCNN [29]. Here, we also consider two distance measures, Euclidean distance and geodesic distance, when building the KNN graphs and demonstrate their influence on the performance of the proposed MVGCN and the baseline method DGCNN (PointNet does not require reorganizing the 3D point cloud as KNN graphs).

Classification Results Using Euclidean Distance. When using the Euclidean distance to build KNN graphs, it partially sacrifices the geometric information when evaluating the distance between two points in the target surface. Thus, in addition to using three coordinates of the raw 3D point cloud for defect classification, we propose to add the normal direction vector at each point to compensate for the geometric features. The rationale for adding the normal direction vector is that it contains the information on local surface curvature, which can compensate for the geometric features. The experiments are conducted using the raw 3D point cloud (without normal) and augmented 3D point cloud (with normal) as inputs, respectively. The comparison results are demonstrated in Table 2, from which the superior performances of the proposed method over the baseline methods are summarized as: (1) the proposed MVGCN method consistently outperforms the baseline methods no matter whether the pointwise normal vector is included in the input or not; (2) when using the raw 3D point cloud as the input (without normal direction vectors), the proposed method is more robust to different levels of noise, compared with the baseline methods; and (3) using the normal direction vectors can improve the accuracy and robustness of all the methods. However, even without including the normal direction vectors in the input, the proposed MVGCN shows comparable performance to the baseline methods using the normal direction vectors. The significant improvement indicates that the proposed MVGCN can capture some information embedded in the normal direction vectors by only using raw 3D point clouds.

Classification Results Using the Geodesic Distance. When using the geodesic distance to build KNN graphs, each KNN graph fully captures the geometric features in the target surface. The performance of PointNet will not be influenced, and the comparison

Table 2 Defect classification results (KNN graphs are built by on Euclidean distance)

	Noise level	$\sigma = 0$	$\sigma = 0.001$	$\sigma = 0.002$	$\sigma = 0.003$	$\sigma = 0.004$	$\sigma = 0.005$	$\sigma = 0.006$
Without normal	PointNet	60.0	60.0	62.5	67.5	55.0	62.5	60.0
	DGCNN	85.0	75.0	77.5	87.5	82.5	85.0	77.5
	MVGCN	**90.0**	**90.0**	**90.0**	**90.0**	**90.0**	**92.5**	**87.5**
With normal	PointNet	85.0	82.5	82.5	85.0	85.0	85.0	80.0
	DGCNN	90.0	90.0	92.5	90.0	92.5	**95.0**	90.0
	MVGCN	**95.0**	**95.0**	**92.5**	**92.5**	**95.0**	92.5	**92.5**

Table 3 Defect classification results (KNN graphs are built on geodesic distance)

Noise level	$\sigma = 0$	$\sigma = 0.001$	$\sigma = 0.002$	$\sigma = 0.003$	$\sigma = 0.004$	$\sigma = 0.005$	$\sigma = 0.006$
DGCNN	100.0	90.0	95.0	**92.5**	92.5	**92.5**	92.5
MVGCN	**100.0**	**90.0**	**95.0**	90.0	**95.0**	90.0	**95.0**

results between the DGCNN and the proposed MVGCN are given in Table 3, from which we can conclude that the proposed MVGCN receives a comparable performance with the baseline method DGCNN. If we compare the results in Tables 2 and 3, we can see an apparent improvement in performance in both methods when using the geodesic distance to build KNN graphs. Such improvement demonstrates that the KNN graphs built on the geodesic distance can better capture the geometric features on the target surface compared with those graphs built upon the Euclidean distance. However, from Table 3, we can further observe an obvious decay in the performance when the noise is introduced to the 3D point cloud. Recall that the geodesic distance calculates the length of the shortest trajectory between two points on the reconstructed surface. The noise in the point cloud degrades the accuracy of surface reconstruction and thus degrades the accuracy of geodesic distance.

4.4.2 Selection of Distance Measures in Building K-Nearest Neighbor Graphs. When comparing the distance measures in building KNN graphs, we can conclude that (1) although using the Euclidean distance sacrifices partial geometric features on the target surface when building KNN graphs, calculating the Euclidean distance is time-efficient, which takes around 0.07 s to generate the pairwise distance among 2000 points. Such a time cost is also insensitive to the number of points in the 3D point cloud. (2) The KNN graphs built on geodesic distance can accurately capture the geometric features on the target surface. However, it is prone to be influenced by the noise in the 3D point cloud. Calculating the geodesic distance is also more time-consuming, which takes around 7 s to generate the pairwise distance among 2000 points. The time cost is linearly proportional to the number of points. (3) When using the Euclidean distance to build the KNN graphs, including the normal direction vectors can compensate for the geometric features and improve the performance of defect classification. Thus, when selecting the distance measures in practice, we would recommend using the geodesic distance for those products with a small scale and a complex shape such that the geodesic distance can accurately capture the geometric features with an acceptable computational cost. Also, denoising the 3D point cloud is highly recommended before calculating the pairwise geodesic distance. For those large-scale products requiring inline defect identification, we would recommend using the Euclidean distance considering the trade-off between accuracy and time efficiency. The normal direction vectors can be further included to compensate for the loss of geometric features.

4.4.3 Influence of the Hyperparameter on the Defect Classification. Both our proposed MVGCN and the baseline method DGCNN require reorganizing the 3D point cloud into KNN graphs. Thus, the number of neighbors in building KNN graphs is an important hyperparameter that will influence the classification results. We have k which represents the number of neighbors used in the DGCNN, and k_1 and k_2 are the numbers of neighbors in the proposed MVGCN. To give an insight on how to properly select the values of k, k_1, k_2, we conduct a grid-search of these values within 30 and with a step size of 5. In our proposed MVGCN, we suppose $k_1 = k_2$ to reduce the searching space, and the Euclidean distance is used to build KNN graphs in searching the hyperparameters. The results are demonstrated in Fig. 11, in which the triangle line and the star line denote the change in classification accuracy using the proposed MVGCN and the baseline DGCNN, respectively. To ensure a fair comparison and best performance, we have $k = k_1 = k_2 = 15$, which receives the highest accuracy using both methods. The number of neighbors mainly influences the defective features because the geometric shapes of the non-defective surfaces are almost consistent. Thus, a large value of k might decay the performance of DGCNN because it will include some points from non-defective regions as neighbors. Our proposed MVGCN is designed to separate the defective and non-defective regions in building KNN graphs, which prevents the features from defective regions from being diluted by non-defective regions. Thus, its performance is more stable with different numbers of neighbors.

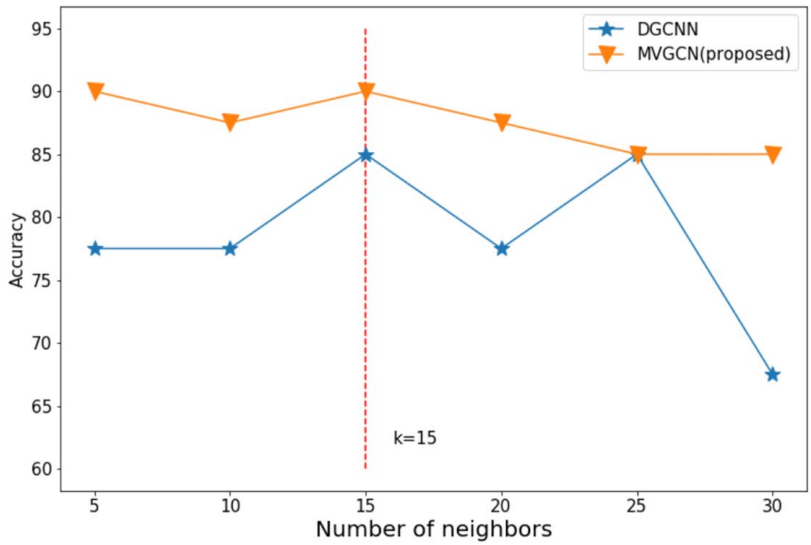

Fig. 11 Influence of the number of neighbors on the classification results (using Euclidean distance)

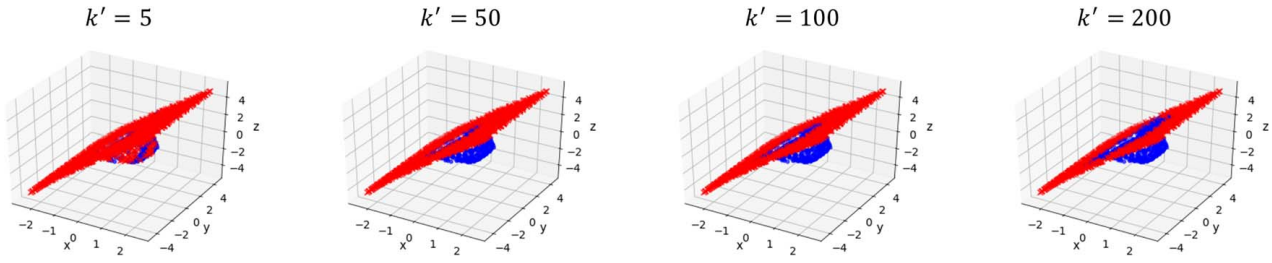

Fig. 12 Examples of defect detection under different values of k'

4.4.4 Influence of the Defect Detection Results on the Defect Classification. The proposed framework is a two-step method that feeds the defect detection results (step 1) into the defect classification (step 2). Therefore, in this sub-section, we will give an insight into how the defect detection performance will influence the classification results. Following the discussion in Sec. 4.3.2, we use the 3D point cloud without noise as the example and set the number of neighbors, k', as 5, 50, 100, 200, respectively, to build the KNN graphs for defect detection. Euclidean distance is used as the distance measure. The defect detection results are qualitatively demonstrated in Fig. 12, from which we can see that when the value of k' is small (i.e., $k' = 5$), the defective region cannot be fully identified, and when the value of k' is large (i.e., $k' = 200$), the partial non-defective region might be misidentified as the defective region.

We further explore the performance of the proposed MVGCN using these four detection results. The KNN graphs used in the classification are also built upon Euclidean distance, and the normal direction vectors are included as the compensation for geometric information. The results are given in Table 4, in which the overestimated defective region (e.g., $k' = 100$, 200) in the first step will decay the classification performance in the second step. Such an observation reveals the intuition in designing the two-step method. If we consider the extreme case when most of the points are identified as the defective region (i.e., $k' \gg 200$), the operation designed for the local view will converge to the baseline DGCNN, while the operation designed for the global view cannot provide additional information in defect classification (refer to Fig. 5). We also notice that underestimating the defective region does not have an apparent influence on the classification performance (i.e., $k' = 5$). It shows that given the partially identified defective region, the operations designed for the local view in our proposed MVGCN can still extract informative features in defect classification.

Table 4 Influence of the defect detection results on the classification performance

The number of neighbors (k')	5	50	100	200
Without normal	90.0	**90.0**	82.5	80.0
With normal	95.0	**95.0**	90.0	87.5

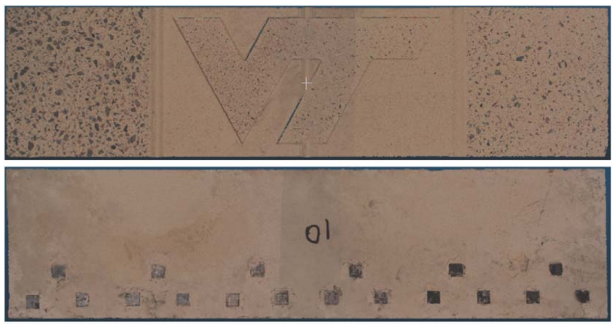

Fig. 13 Visualization of the scanned 3D point cloud. Upper: the front surface of the precast concrete specimen; bottom: the back surface of the precast concrete specimen.

4.4.5 Potential Bottleneck of Computational Cost for Inline Inspection. When the proposed method is applied to inline process a high-resolution 3D point cloud for the large-scale product, the concerns of computational cost might arise because generating KNN graphs requires the pairwise distance among all the points. This issue can be mitigated or even solved from four aspects: (1) when using the Euclidean distance to build KNN graphs, its computational cost is not sensitive to the number of points in the 3D point cloud as long as the computation device has enough memory to tackle the matrix operation; (2) when inspecting the surface of the large-scale product, such as aircraft fuselage, the collected 3D point cloud for the entire product can be segmented into different samples to reduce the number of points in each sample and make it compatible with computational resources; (3) when calculating the pairwise distance for each sample, parallel computing and matrix operation can be adopted to further improve the computation efficiency; and (4) some approximation algorithms to calculate the geodesic distance can be explored to reduce the computation cost.

5 Case Study 2

In this section, we further validate the performance of our proposed method using the real 3D point cloud dataset containing the surface defects on the precast concrete specimen. In the following context, we first introduce the collection of the raw 3D point cloud and the generation of the surface defects dataset (Sec. 5.1). Then we test the performance of the proposed method on the generated surface defects dataset (Secs. 5.2 and 5.3).

5.1 Surface Defects Data Collection. The structured-light 3D scanner is used to collect the raw 3D point cloud from the surface of the precast concrete specimen, and the inline quality inspection system is further built using the raw 3D point cloud, which mainly focuses on ensuring the overall dimension of the product and checking the locations of key elements [40]. The raw 3D point cloud (front and back surfaces) of the specimen is visualized in Fig. 13. There are two types of defects, dent and uneven, existing on the surface of the product. The surface defects dataset is further built by sampling and segmenting the raw 3D point cloud via CLOUDCOMPARE Software. The examples of segmented 3D point cloud samples are visualized in Fig. 14.

In this case study, the surface defects dataset contains a total of 120 samples, in which each of the three classes (i.e., dent, uneven, and normal) contains 40 samples. Similar to the processing process in Sec. 4.1, when initially segmenting these samples, each sample is controlled to contain slightly more than 2000 points, and random sampling is then conducted to ensure each sample contains exactly 2000 points. In the experiments, 72 samples are randomly

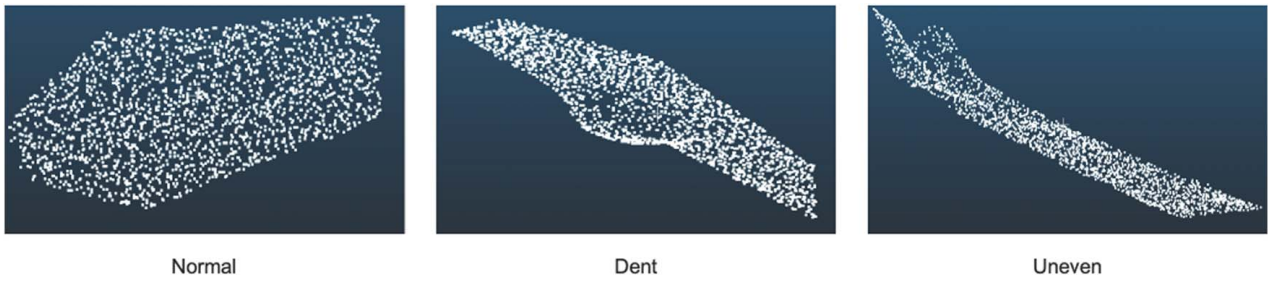

Fig. 14 Visualization of the generated 3D point cloud samples

selected from the total 120 samples as the training data, and the rest 48 samples are used as the testing data to validate and compare the performances. The baseline methods and experiment settings are kept the same as the cast study 1, which can be referred to in Secs. 4.2.1 and 4.2.2, respectively.

5.2 Defect Detection Results. The defect detection results are compared among our proposed adaptive thresholding method and the baseline methods introduced in sec. 4.2.1, which are qualitatively demonstrated in Fig. 15. The ground truth detection results are shown in the first row (dots for defective regions and crosses

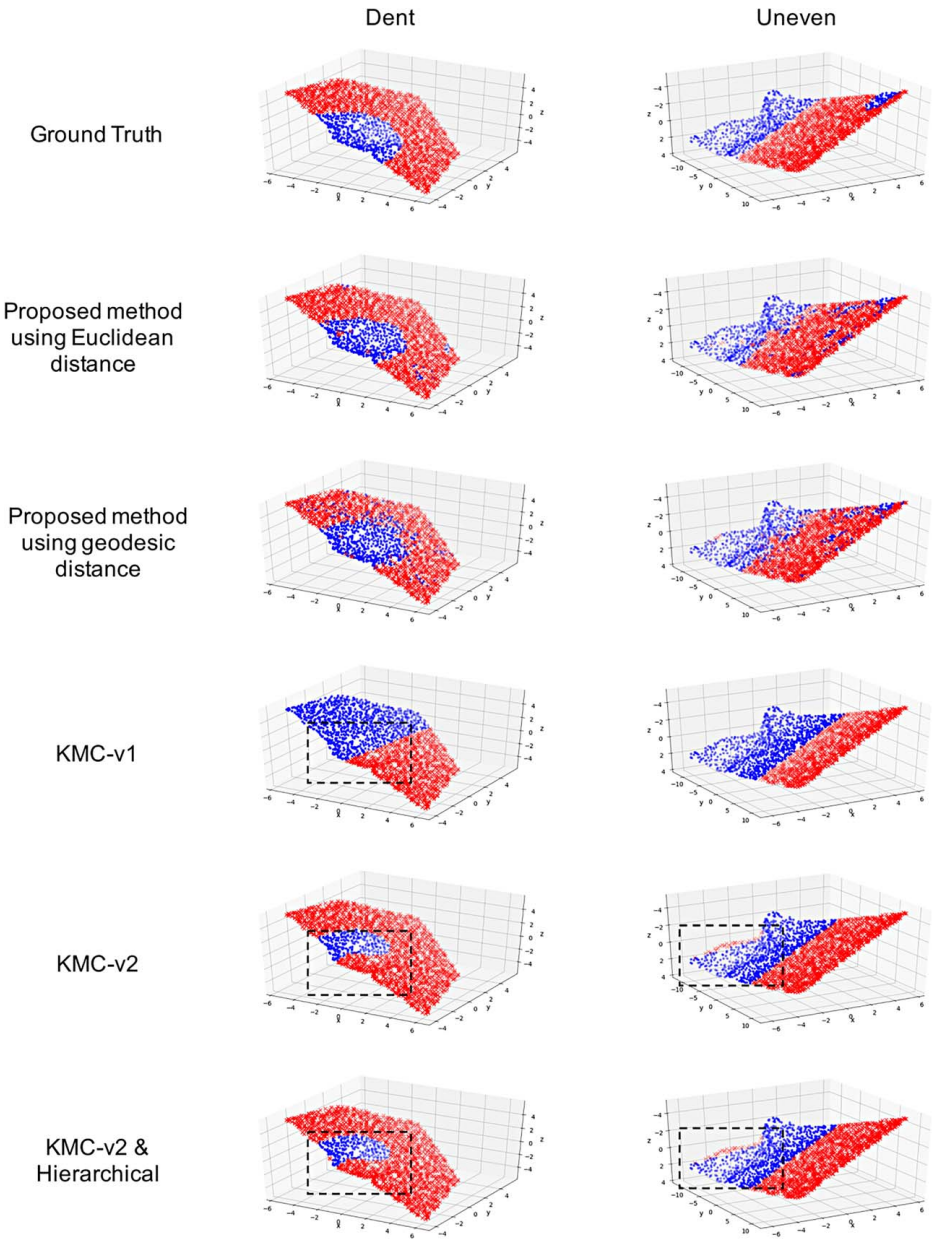

Fig. 15 Visualization of defect detection results

for non-defective regions), and the detection results using different methods are shown in the following rows. By comparison, we can conclude that when using the proposed adaptive thresholding method with either Euclidean distance or the geodesic distance, the defective regions can be successfully identified. On the contrary, the baseline methods tend to misidentify the defective regions in the dent and uneven samples and overestimate the defective regions in uneven samples. The examples of these failure scenarios are highlighted in rectangle dashed boxes in Fig. 15. In summary, our proposed unsupervised adaptive thresholding method outperforms the baseline methods in identifying surface defects from the real 3D point cloud samples.

5.3 Defect Classification Results. The proposed MVGCN takes the detected defective and non-defective regions as the input for the defect classification. When building the KNN graphs in the baseline DGCNN and the proposed MVGCN, the Euclidean distance and geodesic distance are separately used. The normal direction vector at each point is optionally included in the input to compensate for geometric features. The defect classification results are summarized in Table 5, from which we can conclude that our proposed MVGCN significantly outperforms the benchmark methods and receives a consistently outstanding performance with different distance measures (Euclidean distance or geodesic distance) and different inputs (with or without normal direction vector).

If we compare the results in Table 5 vertically, we can find out that the normal direction vector improves the performance of baseline methods, and such improvement is more significant for the PointNet (a similar phenomenon can also be observed in Table 2). In comparison, the normal direction vector does not influence the performance of the proposed MVGCN significantly. The reasons are (1) compared with the synthetic 3D point cloud dataset used in Sec. 4, the real 3D point cloud is collected by scanning the flat surface of the specimen and contains fewer types of defects, which has less complex geometric features and (2) the graph convolution operation is designed to extract geometric features directly from the coordinates of neighbor points. Thus, compared with the methods built upon the graph convolution operation (DGCNN and MVGCN), including the normal direction vector can provide more beneficial information to the PointNet. (3) Furthermore, the design of the proposed MVGCN pays extra attention to the geometric features in the defective region when conducting classification, which could extract adequate features from the coordinates to classify these three types 3D point cloud.

If we compare the results in Table 5 horizontally, we can find out that using the geodesic distance improves the performance of DGCNN while using Euclidean distance and geodesic distance do not have a significant difference in the proposed MVGCN. Before discussing the reasons, we first analyze the mechanisms of the geodesic distance in benefiting the defect classification as follows: (1) the KNN graphs built on the geodesic distance consist of neighbor points with the shortest curves on the surface, which tend to have most of the points in one graph from the same type of surface (defective or non-defective) and (2) for the 3D point cloud collected from the surface with a complex geometric structure, the KNN graphs built on the geodesic distance can better represent the structure of the local surface. Given these two mechanisms, the reasons for observations in Table 5 are summarized in three folds: (1) for the DGCNN, the KNN graphs built upon the geodesic distance can better preserve the geometric features in the defective regions; (2) the design of the MVGCN ensures that the features in the defective regions are preserved and extracted, which is not influenced by the selection of distance measures; (3) compared with the synthetic 3D point cloud segmented from the surface of a cylinder, the real 3D point cloud is segmented from a flat surface, on which the neighbor points found by the Euclidean distance contain adequate geometric features of the surface.

To this point, we can summarize that the results on the real 3D point cloud further demonstrate and validate the strength and effectiveness of the proposed two-step surface defect inspection framework, especially the effectiveness of the proposed MVGCN.

Table 5 Defect classification results (KNN graphs are built on Euclidean distance)

		DGCNN		MVGCN	
	PointNet	Euclidean	Geodesic	Euclidean	Geodesic
Without normal	75.0	70.8	81.3	**97.9**	93.8
With normal	83.3	75.0	85.4	**95.8**	93.8

6 Conclusion

This paper proposed a unified two-step method for surface defect inspection using 3D point cloud. Specifically, an unsupervised adaptive threshold clustering method is proposed to first detect and separate the potential defective regions from the non-defective surface based on the pointwise local surface variations. The detection results are further fed into the multi-view deep learning model to keep track of features from both defective and non-defective regions for defect classification. The strength of the proposed framework is validated using both a synthetic 3D point cloud dataset consisting of three defects types and one normal surface (sampled from the created CAD model of aircraft fuselage surface) and a real 3D point cloud dataset consisting of two defects types and one normal surface (scanned and sampled from the precast concrete specimen). The comparison results show that the proposed approach outperforms the state-of-the-art methods in both defect detection and classification. The advantages of the proposed method can be summarized in three aspects: (1) the proposed method extracts features directly from the 3D point cloud instead of relying on manually selected features, which preserves the inherent informative features in the 3D point cloud and improve the defect identification efficiency; (2) the proposed method has been proved to be invariant to different permutations and transformations of the 3D point cloud; and (3) the proposed method uses the two-step and multi-view deep learning structures, which preserve the features from sparsely distributed defective regions.

From the practitioners' perspective, the proposed surface defect identification method can create values for both quality control and maintenance in the manufacturing system. For example, the proposed method for detecting and classifying defects can be used in the inline 3D laser inspection equipment to improve the efficiency and detection power of surface defects for quality control. Moreover, the classification of defect shapes can also be used for the evaluation of the surface quality for the large-scale product, i.e., aircraft fuselage, which will provide the guidance to setup a proactive maintenance plan to improve the aircraft operational safety.

The future work can be summarized in two directions. First, the proposed method is designed for large-scale products, and one assumption is that most of their surfaces are smooth or with slightly complex geometry design, which limits its wide applications to other products with complex surfaces. The idea of the two-step method is transferable, and for products with a more complex outer surface, novel unsupervised learning methods need to be explored to accurately identify points of defective regions. Second, in the proposed two-step method, the adaptive thresholding clustering method requires calibration when the environment, equipment, or target product change, which introduces extra time costs before the inspection. Also, the calculation of geodesic distance is time-consuming, which makes it inapplicable to inline inspection. We will further explore the time-efficient and accurate single-step method to conduct inline defect inspection for large-scale products. Besides, we will investigate how to use the

active machine learning method to improve the efficiency of information acquisition for quick surface defect inspection [41].

Acknowledgment

This work was partially financially supported by the Department of Defense (DoD) MEEP program under award N00014-19-1-2728. Dr. Yue's research was partially supported by the National Science Foundation under award 2035038 and the Grainger Frontiers of Engineering Grant Award from the National Academy of Engineering.

Conflict of Interest

There are no conflicts of interest.

Data Availability Statement

The datasets generated and supporting the findings of this article are obtainable from the corresponding author upon reasonable request.

Appendix A: Proof of Proposition 3.1

Without loss of generality, suppose the point set X rotate along the axis z by θ deg, the rotation matrix is defined as

$$R = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{A1}$$

With the rotation matrix R, we have the rotated point set as XR, and the covariance of XR is

$$\begin{aligned} \mathrm{Cov}(XR) &= \frac{1}{k'+1}\left[(X-\bar{X})R\right]^{\top}\left[(X-\bar{X})R\right] \\ &= \frac{1}{k'+1}R^{\top}(X-\bar{X})^{\top}(X-\bar{X})R \\ &= R^{\top}\mathrm{Cov}(X)R \end{aligned} \tag{A2}$$

Suppose we have the eigenvalue and eigenvector of $\mathrm{Cov}(X)$ as λ, $\mathbf{p}$, respectively, they satisfy $\mathrm{Cov}(X)\mathbf{p} = \lambda\mathbf{p}$. Also, we have $R^{\top}\mathbf{p} \neq \mathbf{0}$, $R^{\top}R = I$. We can derive the eigenvalues and eigenvectors of $\mathrm{Cov}(XR)$ as follows:

$$\begin{aligned} \left(R^{\top}\mathrm{Cov}(X)R - \lambda I\right)\left(R^{\top}\mathbf{p}\right) &= R^{\top}\mathrm{Cov}(X)RR^{\top}\mathbf{p} - \lambda R^{\top}\mathbf{p} \\ &= R^{\top}\mathrm{Cov}(X)\mathbf{p} - \lambda R^{\top}\mathbf{p} \\ &= R^{\top}\left(\mathrm{Cov}(X)\mathbf{p} - \lambda\mathbf{p}\right) \\ &= \mathbf{0} \end{aligned} \tag{A3}$$

From Eq. (A3), we have the $\mathrm{Cov}(XR)$ has the eigenvalues λ, which is the same as $\mathrm{Cov}(X)$. So the rotation will not change the local surface variation of point set X. The translation of the point set X along with the vector $\mathbf{v} = (a, b, c)$ is basically adding constants a, b, c along each axis of all the points. The translation will not change the covariance matrix of X. So it will not change the eigenvalues and local surface variation. The proof completes.

References

[1] Chiu, Y. P., and Liu, J. Y., 1970, "An Analytical Study of the Stress Concentration Around a Furrow Shaped Surface Defect in Rolling Contact," ASME J. Lubr. Technol., **92**(2), pp. 258–263.

[2] United States, F. A. A., 1997, "Visual Inspection for Aircraft," U.S. Dept of Transportation, Federal Aviation Administration, Advisory Circular, pp. 43–204.

[3] Liu, C., Law, A. C. C., Roberson, D., and Kong, Z. J., 2019, "Image Analysis-Based Closed Loop Quality Control for Additive Manufacturing With Fused Filament Fabrication," J. Manuf. Syst., **51**, pp. 75–86.

[4] Tsai, D. M., and Wu, S. K., 2000, "Automated Surface Inspection Using Gabor Filters," Int. J. Adv. Manuf. Technol., **16**(7), pp. 474–482.

[5] Kumar, A., and Pang, G., 2002, "Defect Detection in Textured Materials Using Gabor Filters," IEEE Trans. Ind. Appl., **38**(2), pp. 425–440.

[6] Park, Y., and Kweon, I. S., 2016, "Ambiguous Surface Defect Image Classification of Amoled Displays in Smartphones," IEEE Trans. Ind. Inform., **12**(2), pp. 597–607.

[7] Zhang, Y., Lefebvre, D., and Li, Q., 2017, "Automatic Detection of Defects in Tire Radiographic Images," IEEE Trans. Autom. Sci. Eng., **14**(3), pp. 1378–1386.

[8] Ngan, H. Y., Pang, G. K., Yung, S., and Ng, M. K., 2005, "Wavelet Based Methods on Patterned Fabric Defect Detection," Pattern Recognit., **38**(4), pp. 559–576.

[9] Karimi, M. H., and Asemani, D., 2014, "Surface Defect Detection in Tiling Industries Using Digital Image Processing Methods: Analysis and Evaluation," ISA Trans., **53**(3), pp. 834–844.

[10] Ng, M. K., Ngan, H. Y. T., Yuan, X., and Zhang, W., 2014, "Patterned Fabric Inspection and Visualization by the Method of Image Decomposition," IEEE Trans. Autom. Sci. Eng., **11**(3), pp. 943–947.

[11] Yan, H., Paynabar, K., and Shi, J., 2017, "Anomaly Detection in Images With Smooth Background Via Smooth-Sparse Decomposition," Technometrics, **59**(1), pp. 102–114.

[12] Yan, H., Yeh, H.-M., and Sergin, N., 2019, "Image-Based Process Monitoring Via Adversarial Autoencoder With Applications to Rolling Defect Detection," 2019 IEEE 15th International Conference on Automation Science and Engineering (CASE), Vancouver, Canada, Aug. 22, pp. 311–316.

[13] Li, Y., Zhao, W., and Pan, J., 2017, "Deformable Patterned Fabric Defect Detection With Fisher Criterion-Based Deep Learning," IEEE Trans. Autom. Sci. Eng., **14**(2), pp. 1256–1264.

[14] Cheon, S., Lee, H., Kim, C. O., and Lee, S. H., 2019, "Convolutional Neural Network for Wafer Surface Defect Classification and the Detection of Unknown Defect Class," IEEE Trans. Semicond. Manuf., **32**(2), pp. 163–170.

[15] Wang, Y., Guo, W. G., and Yue, X., 2022, "Tensor Decomposition to Compress Convolutional Layers in Deep Learning," IISE Trans., **54**(5), pp. 481–495.

[16] Jovancevic, I., Pham, H.-H., Orteu, J., Gilblas, R., Harvent, J., Maurice, X., and Brethes, L., 2017, "3D Point Cloud Analysis for Detection and Characterization of Defects on Airplane Exterior Surface," J. Nondestruct. Eval., **36**(4), pp. 1–17.

[17] Rao, P. K., Kong, Z., Duty, C. E., Smith, R. J., Kunc, V., and Love, L. J., 2015, "Assessment of Dimensional Integrity and Spatial Defect Localization in Additive Manufacturing Using Spectral Graph Theory," ASME J. Manuf. Sci. Eng., **138**(5), p. 051007.

[18] Decker, N., Wang, Y., and Huang, Q., 2020, "Efficiently Registering Scan Point Clouds of 3D Printed Parts for Shape Accuracy Assessment and Modeling," J. Manuf. Syst., **56**, pp. 587–597.

[19] Xie, Q., Lu, D., Huang, A., Yang, J., Li, D., Zhang, Y., and Wang, J., 2021, "Rrcnet: Rivet Region Classification Network for Rivet Flush Measurement Based on 3-D Point Cloud," IEEE Trans. Instrum. Meas., **70**, pp. 1–12.

[20] Samie Tootooni, M., Dsouza, A., Donovan, R., Rao, P. K., Kong, Z. J., and Borgesen, P., 2017, "Classifying the Dimensional Variation in Additive Manufactured Parts From Laser-Scanned Three-Dimensional Point Cloud Data Using Machine Learning Approaches," ASME J. Manuf. Sci. Eng., **139**(9), p. 091005.

[21] Xu, C., Tao, D., and Xu, C., 2013, "A Survey on Multi-View Learning," preprint arXiv:1304.5634.

[22] Sun, J., Zhang, J., Li, Q., Yi, X., Liang, Y., and Zheng, Y., 2022, "Predicting Citywide Crowd Flows in Irregular Regions Using Multi-view Graph Convolutional Networks," IEEE Trans. Knowl. Data Eng., **34**(5), pp. 2348–2359.

[23] Li, Z., Liu, Z., Huang, J., Tang, G., Duan, Y., Zhang, Z., and Yang, Y., 2019, "MV-GCN: Multi-view Graph Convolutional Networks for Link Prediction," IEEE Access, **7**, pp. 176 317–176 328.

[24] Xia, W., Wang, Q., Gao, Q., Zhang, X., and Gao, X., 2022, "Self-Supervised Graph Convolutional Network for Multi-view Clustering," IEEE Trans. Multimedia, **24**, pp. 3182–3192.

[25] Makuch, M., and Gawronek, P., 2020, "3D Point Cloud Analysis for Damage Detection on Hyperboloid Cooling Tower Shells," Remote Sens., **12**(10), p. 1542.

[26] Su, H., Maji, S., Kalogerakis, E., and Learned-Miller, E., 2015, "Multi-view Convolutional Neural Networks for 3D Shape Recognition," 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, Dec. 7, pp. 945–953.

[27] Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., and Xiao, J., 2015, "3D Shapenets: A Deep Representation for Volumetric Shapes," 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, June 8, pp. 1912–1920.

[28] Charles, R. Q., Su, H., Kaichun, M., and Guibas, L. J., 2017, "Pointnet: Deep Learning on Point Sets for 3D Classification and Segmentation," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, July 21, pp. 77–85.

[29] Wang, Y., Sun, Y., Liu, Z., Sarma, S. E., Bronstein, M. M., and Solomon, J. M., 2019, "Dynamic Graph CNN for Learning on Point Clouds," ACM Trans. Graph., **38**(5), pp. 1–2.

[30] Eppstein, D., Paterson, M. S., and Yao, F. F., 1997, "On Nearest-Neighbor Graphs," Discrete Comput. Geom., **17**(3), pp. 263–282.

[31] Hauberg, S. R., Freifeld, O., and Black, M., 2012, "A Geometric Take on Metric Learning," Advances in Neural Information Processing Systems (NeurIPS), Lake Tahoe, NV, Dec. 3, pp. 2024–2032.

[32] Crane, K., Weischedel, C., and Wardetzky, M., 2017, "The Heat Method for Distance Computation," Commun. ACM, **60**(11), pp. 90–99.
[33] Pauly, M., Gross, M., and Kobbelt, L. P., 2002, "Efficient Simplification of Point-Sampled Surfaces," Proceedings of the Conference on Visualization '02, VIS '02, Boston, MA, Oct. 27, pp. 163–170.
[34] Guijt, C. B., and Donne, C. D., 2005, "The Effect of Dents in Fuselage Structures on Fatigue and Static Stability," Proceedings of the Symposium of the International Committee on Aeronautical Fatigue (ICAF), Hamburg, Germany, June 6, Vol. 2, pp. 417–428.
[35] Lang, N., and Kwon, Y., 2007, "Investigation of the Effect of Metallic Fuselage Dents on Compressive Failure Loads," J. Aircr., **44**(6), pp. 2026–2033.
[36] Kanungo, T., Mount, D., Netanyahu, N., Piatko, C., Silverman, R., and Wu, A., 2002, "An Efficient K-Means Clustering Algorithm: Analysis and Implementation," IEEE Trans. Pattern Anal. Mach. Intell., **24**(7), pp. 881–892.
[37] Zepeda-Mendoza, M. L., and Resendis-Antonio, O., 2013, *Hierarchical Agglomerative Clustering*, Springer, New York, pp. 886–887.
[38] Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., and Stuetzle, W., 1992, "Surface Reconstruction From Unorganized Points," Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '92, Chicago, IL, July 1, pp. 71–78.
[39] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., and Killeen, T., 2019, "Pytorch: An Imperative Style, High-Performance Deep Learning Library," Advances in Neural Information Processing Systems (NeurIPS), Vancouver, Canada, Dec. 8, pp. 8026–8037.
[40] Wang, R., Wang, Y., Devadiga, S., Perkins, I., Kong, Z. J., and Yue, X., 2021, "Structured-Light Three-Dimensional Scanning for Process Monitoring and Quality Control in Precast Concrete Production," PCI J., **66**(6), pp. 17–32.
[41] Lee, C., Wang, X., Wu, J., and Yue, X., 2022, "Failure-Averse Active Learning for Physics-Constrained Systems," IEEE Trans. Autom. Sci. Eng.