Multiscale Feature-Clustering-Based Fully Convolutional Autoencoder for Fast Accurate Visual Inspection of Texture Surface Defects

Hua Yang, *Member, IEEE*, Yifan Chen, Kaiyou Song, and Zhouping Yin, *Member, IEEE*

***Abstract*— Visual inspection of texture surface defects is still a challenging task in the industrial automation field due to the tremendous changes in the appearance of various surface textures. Current visual inspection methods cannot simultaneously and efficiently inspect various types of texture defects due to either the low discriminative capabilities of handcrafted features or their time-consuming sliding-window strategy. In this paper, we present a novel unsupervised multiscale feature-clustering-based fully convolutional autoencoder (MS-FCAE) method that efficiently and accurately inspects various types of texture defects based on a small number of defect-free texture samples. The proposed MS-FCAE method utilizes multiple FCAE subnetworks at different scale levels to reconstruct several textured background images. The residual images are obtained by subtracting these texture backgrounds from the input image individually; then, they are fused into one defect image. To maximize the efficiency, each FCAE subnetwork utilizes fully convolutional neural networks to extract the original feature maps directly from the input images. Meanwhile, each FCAE subnetwork performs feature clustering to improve the discriminant power of the encoded feature maps. The proposed MS-FCAE method is evaluated on several texture surface inspection data sets both qualitatively and quantitatively. This method achieves a *Precision* of 92.0% while requiring only 82 ms for input images of 1920 × 1080 pixels. The extensive experimental results demonstrate that MS-FCAE achieves highly efficient and state-of-the-art inspection accuracy.**

***Note to Practitioners*—Most conventional visual inspection methods can address only one specific type of texture defect, while multiscale feature-clustering-based fully convolutional autoencoder (MS-FCAE) can simultaneously and accurately inspect various types of texture surface defects, such as those of thin-film transistor liquid crystal displays, wood, fabrics, and ceramic tiles. Furthermore, MS-FCAE requires only a small number of surface texture samples to learn a robust network model, and its training requires no defect samples. This is extremely important for industrial applications because identifying and labeling defect samples is difficult. Moreover, MS-FCAE can be applied to online visual inspection utilizing a graphics processing unit-based parallel processing strategy.**

***Index Terms*— Convolutional autoencoder (AE), deep neural network (DNN), feature clustering, texture surface, visual inspection.**

I. Introduction

IN THE industrial automation field, the raw materials employed or complex manufacturing techniques can result in various types of texture defects on product surfaces, such as wood [1], fabrics [2], ceramic [3], and thin-film transistor liquid crystal displays (TFT-LCDs) [4]. Based on the specific criteria, such as the texture pattern and texture structure [5], compared with surrounding defect-free regions, areas that include texture defects can be regarded as local regions with irregular brightness variations. To control the product quality, all types of texture defects should be inspected efficiently during the manufacturing process because texture defects result in visibly imperfect products, which directly affect users' experiences.

Visual inspection is a well-known, noncontact, sensor-based technology that has the advantages of being both flexible and highly accurate. Therefore, it has been widely applied to automated optical inspection (AOI) [6] equipment to perform automatic surface texture defect inspection and reduce product defect rates. Generally, visual inspection consists of the following steps. First, an image of the product surface, called a "texture image," is captured under illuminated conditions using a digital camera. Then, the inspected region of interest in the captured image is determined via template matching methods [7]–[9]. Subsequently, a visual inspection method is employed to separate texture defect regions from defect-free regions in the inspected image. Finally, the identified texture defects are classified through vision classification methods [10], [11], as shown in Fig. 1. Visual inspection technologies can be used to effectively identify some but not all types of texture defects, due to the limitations in the performance of visual inspection methods for texture defects.

The texture defects in inspected images can have a variety of characteristics, such as different types, large appearance changes, low contrast, irregular brightness variations, variable

Manuscript received October 14, 2018; accepted December 5, 2018. This paper was recommended for publication by Associate Editor C. C. L. Wang and Editor D. Popa upon evaluation of the reviewers' comments. This work was supported in part by the Major Project Foundation of Hubei Province under Grant 2016AAA009 and in part by the National Science Foundation of China under Grant 51875228, Grant 51475193, and Grant 51327801. *(Corresponding author: Hua Yang.)*

The authors are with the State Key Laboratory of Digital Manufacturing Equipment and Technology, School of Mechanical Science and Engineering, Huazhong University of Science and Technology, Wuhan 430074, China (e-mail: huayang@hust.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TASE.2018.2886031

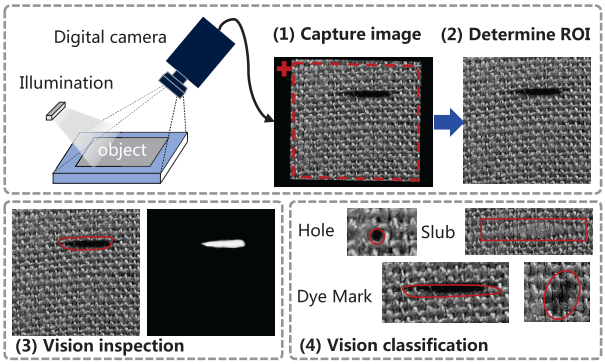


Fig. 1. Flowchart of visual inspection technology for texture defects.

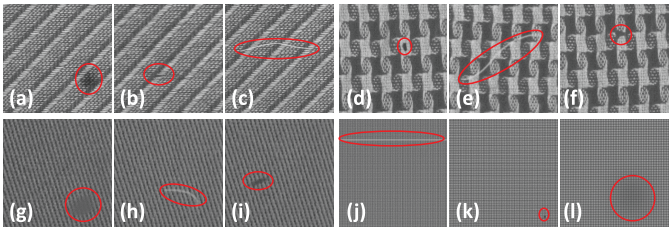


Fig. 2. Various types of texture defects. (a)–(c) Defects on a screen surface. (d)–(f) Defects on a knitted-wool scarf surface. (g)–(i) Defects on a cotton fabric surface. (j)–(l) Defects on TFT-LCD panels.

shapes, and insufficient defect samples, as shown in Fig. 2. In addition, the resolution of inspected images is always high, which requires visual inspection methods to be highly efficient to complete the online inspection within the available time. Therefore, visually inspecting texture defects is still a challenging task in industrial product quality control. During the past few decades, many surface texture defect inspection methods have been proposed and developed to overcome these challenges. These methods are divided into two main categories based on the feature extraction strategy: conventional methods and deep learning methods.

The majority of the conventional methods utilize handcrafted features to represent texture defects. There are four subcategories according to the textural abnormalities detection methods [12]: statistical approaches, structural approaches, spectral approaches, and model-based approaches. The statistical approaches define a spatial distribution of gray values using various representations to describe the texture characteristics. Ng [13] proposed a revised Otsu method to select an optimal threshold automatically in a defect image using a histogram with a unimodal or bimodal distribution. Vujasinovic *et al.* [14] proposed a gray-level co-occurrence matrix texture analysis method to analyze the breast tumor images. However, the approaches using a pixel intensity statistical model can address only substantial defects. The structural approaches represent the texture through texture elements or texture primitives that often model or generalize the spatial placement rules after extracting the texture elements from nondefect images. Mallik-Goswami and Datta [2] proposed using a critically selected structuring element for morphological operations on fabric defect images. These types of approaches usually achieve good performance on repetitive texture patterns. The spectral approaches describe textures in a transformed domain and then process them with filters or filter banks, including fast Fourier transforms (FFTs), wavelet transforms, and Gabor transforms. Tsai and Huang [15] proposed using low-pass filtering with curvature analysis (LCA) to remove periodic repetitive patterns using FFTs and preserve the defective region in the restored image. Aiger and Talbot [16] proposed the phase-only transform (PHOT) to remove the regularities from the image while preserving only the irregular patterns considered to represent the defects. These types of approaches utilize global information in the spectral domain to automatically identify the defects on randomly textured surfaces. The model-based methods describe texture images by modeling special distributions of patterns or features with specific models. Xie and Mirmehdi [17] proposed using texture exemplars (TEXEMS) to establish a Gaussian mixture model (GMM) to obtain the distributions of color textures and detect the defects on randomly textured surfaces. Zhou *et al.* [18] proposed a dictionary learning framework for detecting textile fabric defects that well approximated training samples through a linear summation of their elements. These types of approaches utilize training samples to learn a mathematical model for inspection, and they are well able to address some complex texture defects. Overall, most of the conventional visual texture defect inspection methods use handcrafted features extracted from each texture image patch to differentiate between defect regions and defect-free regions. The performances of these methods rely heavily on the discriminative ability of the extracted handcrafted features of each texture image patch. However, at present, no unified handcrafted feature representation is appropriate for all types of texture images. Thus, conventional texture defect visual inspection methods are unable to achieve good performances on multiple surface texture types simultaneously.

In recent years, deep neural networks (DNNs) have been widely employed in the AOI field due to their ability to extract powerful feature representations compared with the handcrafted features. There are three DNN subcategories according to the network type, namely, supervised approaches, transfer learning approaches, and unsupervised approaches. Supervised approaches use large amounts of labeled defect and defect-free samples to train a DNN model from scratch. Mehta *et al.* [19] proposed end-to-end DNNs by using multitasking fully convolutional networks to detect soiling on solar panels. Yu *et al.* [20] presented a novel combination of lightweight DNNs for segmentation and a final lightweight DNN for classification. The segmentation DNNs are used to quickly localize coarse defect regions via dense pixelwise classification; then, the classification DNN further inspects the fine defect regions. Ronneberger *et al.* [21] proposed a fully convolutional network, named UNet, which uses a top-down architecture to build a high-level feature map with high resolution for biomedical image segmentation. Zhou *et al.* [22] proposed a deeply supervised encoder–decoder network, named UNet++, for medical image segmentation. In this paper, the encoder and decoder subnetworks are connected through some nested, dense skip pathways to obtain features with high semantic representation for medical image segmentation. However, it can be extremely difficult or even impossible to

obtain sufficient labeled defect samples to train a DNN in practical industrial applications, which limits the performance of the supervised approaches. Transfer learning [23] is a common method to solve this problem. This type of approach first pretrains a DNN model using a large data set, for instance, the ImageNet data set. Then, the pretrained model undergoes further fine-tuning using some industrial defect samples. Ren *et al.* [24] proposed a generic transfer learning approach that transfers a pretrained DNN to extract features and then utilizes a pixelwise prediction module to detect wood texture defects. To detect pavement distress, Gopalakrishnan *et al.* [25] pretrained a single-layer neural network classifier using the ImageNet data set and then fine-tuned it using a small defect data set. Kim *et al.* [26] performed a deep analysis of the effects of fine-tuning and revealed that fine-tuning not only amplifies meaningful features but also deactivates unneeded features in the training data. However, due to the very large mismatch between the source domain (ImageNet) and the target domain (texture images), transfer learning approaches can exhibit inferior inspection accuracy; they still require labeled defect samples. To avoid using any labeled defect samples, unsupervised autoencoder (AE) approaches [5], [27] have been employed in AOI. However, these types of approaches usually detect texture defects using a sliding window, which generates large numbers of image patches, resulting in extremely redundant computations and low efficiency. Consequently, they are unsuitable for online applications. Therefore, it is essential to develop a novel fast highly accurate unsupervised deep learning method that can inspect texture defects in real time.

In this paper, we propose a novel unsupervised multiscale feature-clustering-based fully convolutional AE (MS-FCAE) method to effectively and simultaneously inspect various types of texture defects using only a small number of defect-free texture surface samples. The proposed MS-FCAE method utilizes multiple FCAE subnetworks at different scales to reconstruct several types of textured backgrounds. The residual images are obtained by subtracting these texture backgrounds from the input images. Then, the residual images are fused into one defect image. To achieve high time efficiency, each FCAE subnetwork utilizes a fully convolutional neural network to extract the original feature maps directly from the input image. Meanwhile, each FCAE subnetwork uses feature clustering to improve the discriminant power of the encoded feature maps. Moreover, MS-FCAE significantly improves the inspection accuracy for hard texture defects.

The remainder of this paper is organized as follows. In Section II, the related works regarding AE-based defect inspection methods and clustering methods are introduced. In Section III, the procedure of the proposed MS-FCAE method is elaborated and discussed in detail. In Section IV, we report the results of several experiments. Finally, our conclusions are presented in Section V.

II. Related Works

In this section, we briefly introduce and discuss the related works regarding AEs and their extensions. Then, we present popular feature-clustering methods.

A. AEs and Their Extensions

An AE network is a type of artificial neural network that has been widely used for unsupervised learning in the fields, such as target recognition [28], [29], object detection [30], scene description [31], and shape retrieval [32]. An AE network can be trained to learn a representation for a set of data without any labeled ground-truth images or human intervention. In general, an AE network consists of an encoder module and a decoder module. First, the encoder module maps the input data to a latent representation using nonlinear mapping functions, such as $sigmod$, $tanh$, and $relu$. After mapping, the latent representation can be considered as a feature code that contains the compressed information of the input data. Then, the decode module uses the feature code to reconstruct the output data by performing a reverse mapping. During optimization, the goal of the AE objective function is to make the output data equal the input data. After training, the AE network has a strong ability to reconstruct the input data that are similar to the samples in the training data set. To improve the nonlinear representation in both the encoder and decoder modules, the AE method was improved through a deep network to yield the multilayer perceptron (MAE) method. However, both AEs and MAEs are based on the fully connected layers, which leads to some severe problems during image processing, such as ignoring the structural information in the image, forcing each feature to be global, and requiring redundant parameters.

Recently, deep convolutional neural networks (DCNNs) [33] have demonstrated outstanding strengths in a wide range of machine vision applications, including detection, segmentation, and tracking. Inspired by the successes of DCNNs, the convolutional AE network (CAE) [34] was developed based on MAE by using DCNNs to share its weights among local regions. A common CAE network uses the following steps. During feature extraction and encoding, it extracts the features of the input image using the convolution layer(s) and then encodes them into a feature code in the fully connected layer(s). In the decoding and reconstruction steps, the feature code is first decoded to a feature map using fully connected layer(s), and then, the feature map is reconstructed to match the input image using a deconvolutional layer(s). A CAE can be trained by optimizing the reconstruction loss function, for instance, by minimizing the mean squared error calculated by subtracting the reconstructed image from the input image. Ke *et al.* [35] proposed an ACAE based on a CAE to generate a template of samples and then detect abnormal information by comparing test images with the adaptive template.

Later, Vincent *et al.* [36] proposed an improved AE, namely, the denoising AE (DAE), which takes a partially corrupted input during training and can obtain a robust powerful representation from the raw noisy data. Similarly, the convolutional DAE (CDAE) is capable of reconstructing a clean image from the one destroyed by salt or pepper noise. To take advantage of CDAE, Li *et al.* [27] proposed a representation for patterned fabric that worked even when only limited defect data were available. Chalapathy *et al.* [37] proposed the robust CAE (RCAE) method to detect anomalies. This approach achieved good performance on a range of real-world data

sets. Mei *et al.* [5] proposed a multiscale CDAE (MSCDAE) method that can be trained using only defect-free patches. The MSCDAE method employs a three-layer pyramid to perform multiscale defect inspection. Nevertheless, the MSCDAE method requires a separate model to be trained and tested for each layer; therefore, it is time-consuming and leads to excessive memory consumption.

Although both the CAE method and its extensions achieve good performances without requiring labeled training data, they have serious computational efficiency deficiencies. For the common CAE/CDAE methods, there are two severe issues when the input is a full image, due to the existence of the fully connected layer. On the one hand, defects are usually local abnormalities compared with their surrounding areas; however, the feature code contains the global information of the whole input image. On the other hand, the feature code vector is large due to the high-resolution input, which leads to the curse of dimensionality [38]. Therefore, these methods usually inspect defects in sliding windows, using image patches. However, this patchwise inspection method results in redundant computation, which causes these methods to be extremely time-consuming.

Thus, in this paper, to effectively address all types of texture surface defects simultaneously while requiring only a limited number of defect-free texture samples, the proposed MS-FCAE method utilizes a novel FCAE subnetwork to efficiently reconstruct multiscale background images in a fully convolutional manner. Furthermore, this method uses a feature clustering method to further enhance the background reconstruction accuracy of FCAE and improve the inspection accuracy for hard texture defects.

B. Clustering Methods

Clustering methods divide a set of objects into groups called clusters; objects in the same cluster are more similar to each other than to those in the different clusters. Because clustering is unsupervised and requires no labeled training data, it has been applied to computer vision tasks, such as object recognition [39] and object detection [40].

The clustering methods proposed over the past few decades can be generally divided into four categories: connectivity-based, centroid-based, distribution-based, and density-based. Connectivity-based clustering is also called hierarchical clustering. These approaches use a distance measurement to form the different clusters. Sibson *et al.* [41] proposed the single-link cluster method, which groups clusters by combining the two clusters that contain the closest pair of objects. However, these methods are not very robust to outliers and have high computational complexity. Centroid-based clustering methods aim to find clusters' central vectors, which are not necessarily members of the data set. Forgy [42] proposed the most popular clustering method, namely, k-means, which groups objects into k clusters, and each object belongs to the cluster with the nearest mean value. One drawback of these methods is that they require a predefined number of clusters. Distribution-based clustering methods define clusters as objects that are most likely to belong to the same statistical distribution. The most well-known distribution-based clustering method is the GMM [43]. This method models the objects using a fixed number of Gaussian distributions and utilizes the expectation–maximization algorithm [44] to solve it. The shortcoming of distribution-based clustering methods is that they assume objects to obey some distribution, which is a rather strong assumption. Density-based clustering methods define clusters as areas with higher density than the remainder of the data set. Ester *et al.* [45] proposed the most popular density-based clustering method, namely, density-based spatial clustering of applications with noise, which clusters objects by grouping the objects that satisfy a density criterion. Despite the existence of so many high-quality and widely used clustering methods, these methods are all based on a clustering strategy or on similarity measurements, and they do not change the data distribution. Recently, Xie *et al.* [46] proposed the deep embedded clustering (DEC) method to address this limitation. DEC simultaneously learns feature representations and clustering using a DNN. Moreover, it can change the object distribution to become more discriminative.

To improve the background reconstruction accuracy, it is essential to enhance the discriminability of the encoded features. That is, the feature distribution must be changed during training. Thus, in this paper, we employ the DEC method [46] to enhance the discriminative ability of the encoded features in the proposed MS-FCAE method.

III. Proposed MS-FCAE Method

In this section, the proposed MS-FCAE method is introduced in detail. First, the overall DNN architecture of MS-FCAE is briefly introduced. Then, its three modules, including the feature extraction network, the multiscale FCAE network, and the result fusion module, are presented in detail. Furthermore, the MS-FCAE unsupervised training process, which requires only a few defect-free samples, is introduced. Finally, the parameter setup of MS-FCAE is reported and discussed.

A. Network Architecture of MS-FCAE

The conventional CAE methods are a time-consuming approach, which limits their application in the real-time automated inspection. Meanwhile, due to the poor discriminative ability of learning features, they cannot simultaneously inspect various types of texture defects. In this paper, we propose a novel DNN, called MS-FCAE, to efficiently and accurately inspect texture defects. MS-FCAE is an unsupervised deep learning method and can be trained using only a few unlabeled defect-free samples. Fig. 3 shows the overall network architecture of MS-FCAE, which consists of three new modules: a feature extraction network, a multiscale FCAE network, and a result fusion module.

The feature extraction network is constructed using several convolutional layers and used to extract feature maps with different receptive fields directly from the input image rather than using the time-consuming sliding-window strategy that requires image patches as input. Therefore, it extracts features from all the patches of an image simultaneously

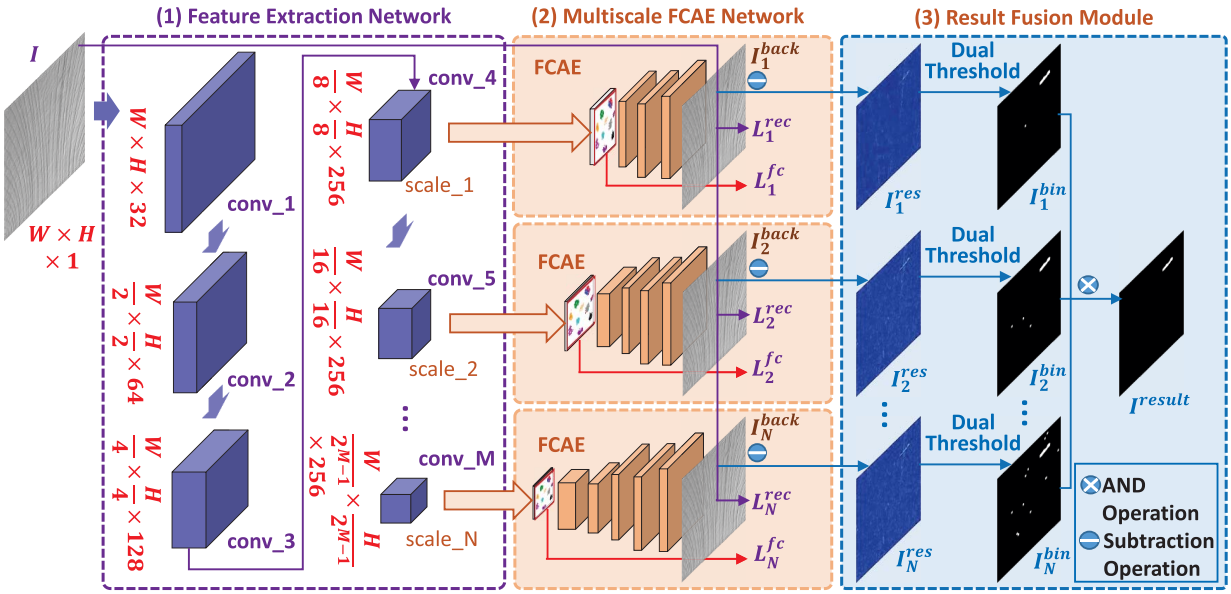


Fig. 3. Overall architecture of the proposed MS-FCAE method. MS-FCAE consists of the feature extraction network, the multiscale FCAE network, and the result fusion module. Each FCAE subnetwork consists of an encoding module, a feature clustering module, and a decoding module. During the testing phase, the reconstructed background images are fused by the result fusion module to obtain the final defect image.

without any redundant computation, which guarantees its high time efficiency. Subsequently, based on the extracted multiscale feature maps output by the feature extraction network, we propose a multiscale FCAE network, which consists of multiple FCAE subnetworks that reconstruct multiple texture background images at different scale levels. Furthermore, in each FCAE subnetwork, a feature clustering module is proposed to enhance the discriminability of the encoded features to improve the background reconstruction accuracy. Using the multiscale FCAE network, MS-FCAE efficiently inspects the defects at different scales, similar to a Gaussian pyramid, which improves the inspection accuracy for various types of texture defects. Finally, based on the texture background images reconstructed by the multiscale FCAE network, to reduce false detections and improve the inspection accuracy, the result fusion module is proposed to obtain the final inspection result. The result fusion module first obtains multiple residual images by subtracting the texture background images from the input image. Then, these residual images are fused into a single result image that indicates the texture defects.

Due to the fully convolutional layers used in both the feature extraction network and the multiscale FCAE network, the computational complexity of MS-FCAE is very low. Moreover, because it reconstructs different scale texture backgrounds, MS-FCAE can accurately inspect texture defects, even hard defects.

B. Feature Extraction Network

The feature extraction network is constructed to extract feature maps with different receptive fields directly from the whole input image. To achieve high accuracy and efficiency, the feature extraction network is designed as a deep fully convolutional neural network. It encodes the input image into multiple feature maps with different scales.

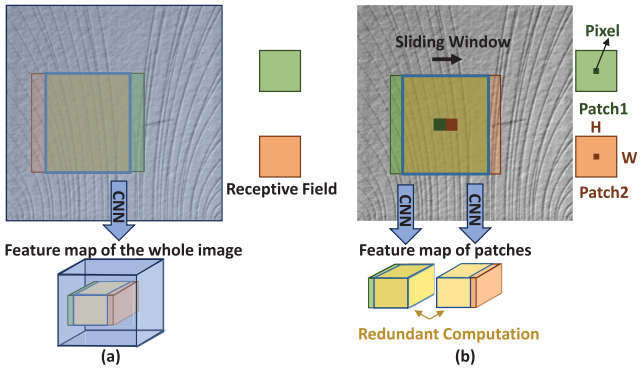


Fig. 4. Schematic of the redundant computation. (a) In the patchwise method, redundant computation (yellow region) occurs for every patch (red and green region). (b) No redundant computation exists during feature extraction from the entire image (blue region).

As shown in Fig. 3, there are M convolutional layers in the feature extraction network, from the first layer conv_1 to the Mth layer conv_M. The value of M is determined by the number of FCAE subnetworks. To reduce the number of parameters required by the feature extraction networks, all the convolutional layers adopt a small 3×3 convolutional kernel filter to extract features from their respective feature maps. To enrich the feature map representations, the number of channels in the first three convolutional layers gradually increases (i.e., from 32 to 64 and then to 128, respectively). After the third layer conv_3, the number of channels in the subsequent convolutional layers is set to 256 because the number of parameters is dramatically increased when the number of channels is too large. The initial convolutional layers (i.e., conv_1) with 1×1 filters and stride 1 extract features directly from the whole input image. Thus, given an input image of size $W \times H$, where W and H denote the width and height of the input image, respectively, the resolution of the first layer conv_1 is also $W \times H$. Then, to reduce the resolution

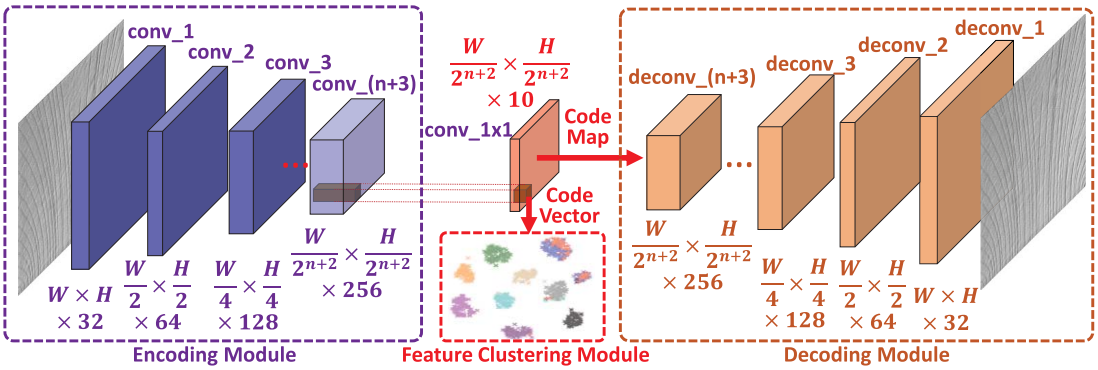


Fig. 5. Architecture of one of the FCAE subnetworks in MS-FCAE.

and enlarge the receptive fields of the subsequent feature maps (i.e., conv_2 and conv_M), a fixed number of convolutional layers with 3×3 filters and a stride of 2 are added to the feature extraction network. Because the convolutional layers have a stride of 2, there is no need to add any pooling layers to the feature extraction network. The resolutions of the subsequent feature maps (conv_2 to conv_M) are $(W/2^{m-1})\times(H/2^{m-1})$, where $m \in (2,\ldots,M)$ is the index of the mth layer.

The receptive field is defined in the input image space as a region with a certain size that contains a particular DNN feature. In the proposed MS-FCAE method, the receptive fields represent the sizes of the image patches from which the individual feature map elements are calculated. Because the input image of MS-FCAE is the whole input image, the feature extraction network extracts all the features of different-sized patches in the input image at the same time without requiring any redundant computation, as shown in Fig. 4(a). Different scales of feature maps in the feature extraction network represent different receptive field sizes. The formula is defined as follows:

$$R_{m-1} = [(R_m - 1) \times \mathrm{Str}_m] + K_m \tag{1}$$

where R_m and R_{m-1} denote the receptive fields of the deeper mth feature map and the shallower $(m-1)$th feature map, respectively. Str_m and K_m are the stride and kernel size of the convolutional layer between the mth feature map and the $(m-1)$th feature map, respectively. This equation shows the relationship between the receptive fields of two neighboring convolutional feature maps. That is, a shallower layer has a smaller receptive field. Thus, MS-FCAE uses feature maps with different scales to address different types of textures with differently sized structures. The previous patchwise methods, such as [5], [34], and [37], involved large amounts of redundant computation. Before the encoding step, patch1 and patch2 are separately processed by the same DNN. The yellow region is the common region between patch1 and patch2, which will be computed twice during the same forward propagation. Therefore, MS-FCAE is more efficient than the previous patchwise methods.

Using this fully convolutional feature extraction network, MS-FCAE encodes the input image into feature maps with different scales. Extracting features from all the patches in the input image at the same time without any redundant computation guarantees MS-FCAE's high efficiency.

C. Multiscale FCAE Network

In MS-FCAE, the novel multiscale FCAE network is proposed to reconstruct multiple texture background images with different levels of texture details at different scales by using the extracted feature maps of the feature extraction network. Fig. 3 shows how multiple FCAE subnetworks are grafted onto the corresponding different-scale feature maps of the feature extraction network to represent the different receptive fields in the input image. To guarantee the semantic representation ability of the latent feature to improve the background reconstruction accuracy, the network depth of the encoding module of each FCAE subnetwork should not be too shallow. Therefore, the first FCAE subnetwork is grafted onto conv_4 with the resolution $(W/8)\times(H/8)$ at scale_1. Similarly, other FCAE subnetworks are grafted onto the subsequent convolutional layers (i.e., from conv_5 to conv_M at scales from scale_2 to scale_N, respectively) to decode differently scaled feature maps. N is the total number of FCAE subnetworks in MS-FCAE, and it both influences the defect inspection accuracy and affects the model complexity. A larger N value results in higher inspection accuracy but increases the model complexity. N also determines the total number of convolutional layers M in the feature extraction network, that is, $M = N + 4$. To obtain a good tradeoff between the defect inspection accuracy and the model complexity, we recommend a setting, such that $1 \le N \le 5$.

The architecture of the nth FCAE subnetwork at the scale_n level is shown in Fig. 5. This subnetwork consists of three modules: an encoding module, a feature clustering module, and a decoding module. To achieve high time efficiency, the FCAE subnetwork first utilizes the fully convolutional encoding module to encode the feature maps of the feature extraction network into code maps, rather than using the time-consuming sliding-window approach. Moreover, each FCAE subnetwork uses the new feature clustering module to further improve the discriminative ability of these code maps. Finally, the encoded code maps are decoded to texture background images through the decoding module at different scale levels.

After combining the encoding module and the decoding module, the image reconstruction process can be defined using

the following expression model:

$$\begin{aligned} &\phi : \mathcal{I} \to \mathcal{F} \\ &\varphi : \mathcal{F} \to \mathcal{I} \\ &(\phi, \varphi) \;=\; \arg\min_{(\phi,\varphi)} \|\boldsymbol{I} - \varphi(\phi(\boldsymbol{I}))\|^2 \end{aligned} \tag{2}$$

where $\boldsymbol{I} \in R^{W \times H}$ denotes the input image in the $\mathcal{I}$ domain, which is mapped to $\boldsymbol{Z} = \phi(\boldsymbol{I}) \in R^{W' \times H' \times C} = \mathcal{F}$; $\mathcal{F}$ indicates the corresponding feature map in the $\mathcal{F}$ domain; and W', H', and C denote the width, height, and channel number of the encoded feature map, respectively. The operators of both the encoding module and decoding module are represented by the following equations:

$$\begin{aligned} \boldsymbol{Z} &= \sigma(\boldsymbol{W} \circ \boldsymbol{I} + \boldsymbol{b}) \\ \boldsymbol{I}' &= \sigma'(\boldsymbol{W}' \circ \boldsymbol{I} + \boldsymbol{b}') \end{aligned} \tag{3}$$

where "$\circ$" is the convolution operator; $\boldsymbol{I}' \in R^{W \times H}$ is the reconstructed image; and σ represents the activation function, such as $sigmod$, $tanh$, or $relu$. Here, $\boldsymbol{W}$ and $\boldsymbol{W}'$ are the parameter matrices of the convolutional kernels in the encoding and decoding modules, respectively, and $\boldsymbol{b}$ and $\boldsymbol{b}'$ are the bias vectors in the encoding and decoding modules, respectively. The FCAE subnetwork can be trained by optimizing the reconstruction object function, e.g., by using the mean square error.

1) Encoding Module: The encoding module of FCAE is utilized to encode the feature maps of the feature extraction network into code maps, as shown in Fig. 5. The encoding module consists of $(n+4)$ convolutional layers (i.e., conv_1 to conv_$(n+3)$ and conv_1 $\times$ 1). Note that layers conv_1 through conv_$(n+3)$ belong to the feature extraction network. Here, to form a complete encoding and decoding procedure, they are also treated as belonging to the FCAE subnetwork. For the conv_$(n+3)$ layer, the resolution of the feature map is $(W/2^{n+2}) \times (H/2^{n+2})$.

Most current CAE methods utilize a fully connected layer to connect the last convolutional features of the encoding module directly to the code map—the middle layer between the encoding module and the decoding module. This fully connected approach can cause the curse of dimensionality and affect the efficiency of these CAE methods. In the proposed FCAE subnetwork, to dramatically reduce the network parameters in the fully connected layer, a special convolutional layer conv_1 $\times$ 1 with a 1 $\times$ 1 convolutional kernel is utilized to perform the same operation as the fully connected layer.

Fig. 6(a) shows the structure and parameter of the convolution process for the code map. Through a 1 $\times$ 1 convolutional operation with stride 1, the feature map in the feature extraction network with size $W_{\text{conv}} \times H_{\text{conv}} \times C_{\text{conv}}$ is processed to the code map with a size of $W_{\text{code}} \times H_{\text{code}} \times C_{\text{code}}$, where W_{conv}, H_{conv}, and C_{conv}, respectively, denote the width, height, and number of channels in the input feature map; W_{code}, H_{code}, and C_{code} denote the width, height, and number of channels in the output code map, respectively; and $W_{\text{code}} = W_{\text{conv}}$ and $H_{\text{code}} = H_{\text{conv}}$. For example, for the first FCAE subnetwork, the input feature map is conv_4, and the output feature map is conv_1 $\times$ 1; thus, $W_{\text{code}} = (W/8)$, $H_{\text{code}} = (H/8)$,

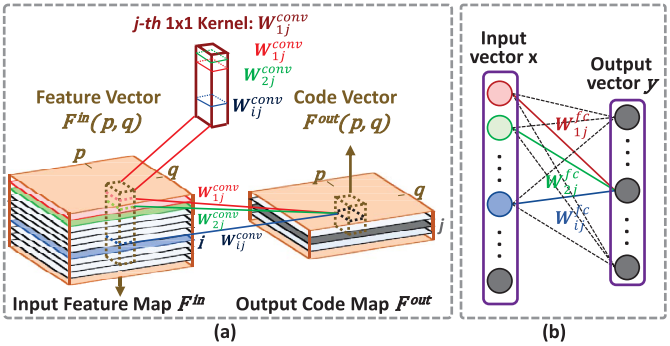


Fig. 6. Architecture of the encoding module in FCAE (a) compared with the fully connected layer (b) commonly used in current CAE methods.

and $C_{\text{conv}} = 256$. As shown in Fig. 6(a), $\boldsymbol{F}^{\text{out}}(p,q)$ denotes the feature vector at position (p,q) of the code map, where p and q denote the horizontal and vertical values, respectively. To obtain the medium-length code vectors, the channel number of the code maps is set to $C_{\text{code}} = 10$ in all the FCAE subnetworks. For every channel in the output feature matrix, the convolutional process is defined as follows:

$$\boldsymbol{W}_j^{\text{conv}} \circ \boldsymbol{F}^{\text{in}} + b_j^{\text{conv}} = \boldsymbol{F}_j^{\text{out}} \tag{4}$$

where $\boldsymbol{W}_j^{\text{conv}}$ and b_j^{conv} are the jth convolution kernel weight matrix and bias in the kernel set, respectively; $\boldsymbol{F}^{\text{in}}$ is the input feature matrix; $\boldsymbol{F}_j^{\text{out}}$ denotes the jth channel of the output feature; and $\circ$ is the convolution process. For every position (p,q) in the output feature matrix, the convolutional operation is a multiplication

$$\boldsymbol{W}_j^{\text{conv}} \boldsymbol{F}^{\text{in}}(p,q) + \boldsymbol{b}^{\text{conv}} = \boldsymbol{F}^{\text{out}}(p,q,j) \tag{5}$$

in which $\boldsymbol{F}^{\text{in}}(p,q)$ denotes the feature vector in position (p,q) of the input feature map and $\boldsymbol{F}^{\text{out}}(p,q,j)$ is the feature value in position (p,q) and channel j of the output code map. To integrate the weights and bias, we use

$$\boldsymbol{W}^{\text{conv}} \boldsymbol{F}^{\text{in}}(p,q) + \boldsymbol{b}^{\text{conv}} = \boldsymbol{F}^{\text{out}}(p,q) \tag{6}$$

$\boldsymbol{W}^{\text{conv}} = [\boldsymbol{W}_1^{\text{conv}}, \boldsymbol{W}_2^{\text{conv}}, \ldots \boldsymbol{W}_{C^{\text{out}}}^{\text{conv}}]$ is a parameter matrix composed of kernel weight vectors and $\boldsymbol{B} = [\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots \boldsymbol{b}_{C^{\text{out}}}]$ is the bias vector. For comparison purposes, Fig. 6(b) shows the fully connected layer commonly used in the current CAE methods. It is an N-input and M-output fully connected layer, where $\boldsymbol{w}_j^{\text{fc}} = [\boldsymbol{w}_1^{\text{fc}}, \boldsymbol{w}_2^{\text{fc}}, \ldots \boldsymbol{w}_M^{\text{fc}}]$ and b_j^{fc} denote the weight vector and bias of the jth output neuron in the fully connected network. Its value can be expressed as follows:

$$\boldsymbol{w}_j^{\text{fc}} \boldsymbol{x} + b_j^{\text{fc}} = \boldsymbol{y}_j \tag{7}$$

where $\boldsymbol{x} = [x_1, x_2, \ldots, x_N]$ is the input vector and $\boldsymbol{y}_j$ denotes the jth output. Rewriting this in matrix form

$$\boldsymbol{W}^{\text{fc}} \boldsymbol{x} + \boldsymbol{b}^{\text{fc}} = \boldsymbol{y} \tag{8}$$

where $\boldsymbol{W}^{\text{fc}} = [\boldsymbol{w}_1^{\text{fc}}, \boldsymbol{w}_2^{\text{fc}}, \ldots, \boldsymbol{w}_M^{\text{fc}}]$ denotes the weight matrix composed of weight vectors, $\boldsymbol{b}^{\text{fc}} = [b_1^{\text{fc}}, b_2^{\text{fc}}, \ldots, b_M^{\text{fc}}]$ denotes the bias vector, and $\boldsymbol{y}$ is the output vector. Note that (6) and (8) are identical, which proves the equality between the fully connected layer and the convolutional operation with

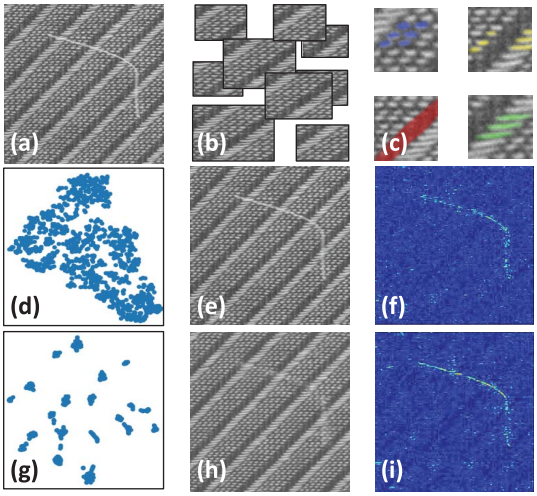


Fig. 7. Effect of feature clustering. (a) Input defect image. (b) Patches in the pixelwise methods or receptive field region in MS-FCAE. (c) Textons of the textures. (d)–(f) Distribution of code vectors in the feature space, the reconstructed background image, and the residual image without feature clustering, respectively. (g)–(i) Distribution of code vectors in the feature space, the reconstructed background image, and the residual image with feature clustering, respectively. When feature clustering is applied, the code vectors are more discriminative.

a 1×1 kernel. Using this convolutional layer, conv_1×1, the number of parameters is dramatically reduced.

The encoding module of FCAE has two noteworthy advantages. First, during training, the weights are shared due to the convolutional process; moreover, due to the memory-efficient feature extraction, a relatively large batch size can be set for the training step; in other words, a large number of training samples can be utilized for the training step, which causes the convergence of the FCAE to be better than that of the common CAE. Second, during the test phase, the computational speed of FCAE will be faster because the convolutional process can be calculated in parallel using a graphics processing unit (GPU).

2) Feature Clustering Module: In the field of texture surface defects visual inspections, some defects are called "hard defects" because they have patterns or pixel intensities similar to the textured background. The conventional visual inspection methods perform poorly for hard defect inspection because the discriminative ability between the image features of defect samples and those of defect-free samples is insufficiently high. Thus, in FCAE, inspired by [46], a feature clustering module is proposed to enhance the discriminability of the encoded features of the code map. Such clustering can improve the reconstruction accuracy of the texture background image. The basic microstructures in typical texture images are called textons [47]. Fig. 7(c) shows that the texture of fabric or other industrial products is highly complex and usually contains several different kinds of textons. In FCAE, the different code vectors of the encoded code maps correspond to the different regions of the input image, as shown in Fig. 6(a). After being trained, these code vectors contain all the information of the texture textons. In the subsequent decoder module, these code vectors are combined linearly by using convolutional operations to generate the texture background image. Intuitively, a complex texture is a linear combination

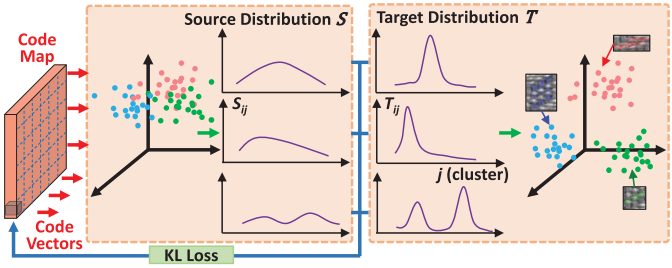


Fig. 8. Schematic of the feature clustering module. The feature clustering module is a branch that occurs after the encoding module. The target distribution $\boldsymbol{T}$ is more concentrated at a cluster than the source distribution vmS. The parameters are updated using the KL Loss between $\boldsymbol{T}$ and $\boldsymbol{S}$.

of various textons. However, in texture images of industrial products, patches are often quite similar, as shown in Fig. 7(b). Thus, the code vectors of the code map will be confused and have little discriminability in the feature space, as shown in Fig. 7(d). The lack of discriminative features makes it difficult to remove these hard defects from the input image and results in reduced inspection accuracy, as shown in Fig. 7(e).

In good discriminative features, the intraclass feature distance calculated by using the similarity measurement function should be small, while the interclass distance should be large. The feature clustering module first calculates the similarity distribution of the code vectors as the source distribution. Then, the target distribution, which has a smaller intraclass feature distance and a larger interclass feature distance, is designed and generated. To create more discriminative features, the network parameters of FCAE, including the feature clustering module, are updated with an objective function that fits the source distribution to the target distribution.

Here, assuming that there are k types of textons in the texture image, the code vectors can be divided into k clusters using a specific similarity measurement. As shown in Fig. 8, the feature clustering module aims to aggregate the code vectors $\boldsymbol{c_i}$ into clusters during training, where $\boldsymbol{c_i}$ is the code vector $\boldsymbol{F}^{\text{out}}(p, q)$ in Fig. 6(a), and $\boldsymbol{c_i} \in R^{C_{\text{code}} \times 1}$, $i \in (1, \ldots, W_{\text{code}} \times H_{\text{code}})$. To simplify the calculation, the centroid feature vector of the jth cluster $\boldsymbol{\mu}_j$ is represented by the code vectors belonging to that cluster. Let $\boldsymbol{\mu} = [\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots \boldsymbol{\mu}_k]$ denote the cluster centroids ($\boldsymbol{\mu}_k \in R^{C_{\text{code}} \times 1}$). To represent the source similarity distribution $\boldsymbol{S}$, the probability $\boldsymbol{S}_{ij}$ of assigning code vector $\boldsymbol{c_i}$ to cluster $\boldsymbol{\mu}_j$ calculated by a Student's t-distribution is employed to compute the similarity measurement between two feature vectors in a low-dimensional feature space [48]

$$\boldsymbol{S}_{ij} = \frac{\left(1 + \|\boldsymbol{c}_i - \boldsymbol{\mu}_j\|^2/\alpha\right)^{-\frac{\alpha+1}{2}}}{\sum_{j'} \left(1 + \|\boldsymbol{c}_i - \boldsymbol{\mu}_{j'}\|^2/\alpha\right)^{-\frac{\alpha+1}{2}}} \tag{9}$$

where $j \in (1, \ldots, k)$, $j' \in (1, \ldots, k)$, and α is the degrees of freedom, set to 1 in this paper.

Smaller intraclass similarity and larger interclass similarity mean that the gap between the feature probabilities belonging to different clusters is larger. For instance, $\{0.8, 0.1, 0.1\}$ is more discriminative than $\{0.4, 0.3, 0.3\}$. Therefore, the target distribution $\boldsymbol{T}$ should be more concentrated. To achieve this

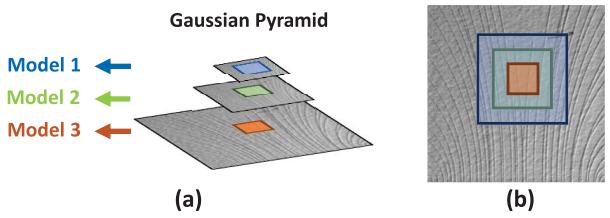


Fig. 9. (a) Schematic of a Gaussian pyramid. (b) Mask corresponding to (a) (best viewed in color).

goal, $\boldsymbol{T_i}$ is calculated by first raising $\boldsymbol{S_i}$ to the second power and then normalizing it by the per-cluster frequency

$$\boldsymbol{T}_{ij} = \frac{\boldsymbol{S}_{ij}^2 / \boldsymbol{f}_j}{\sum_{j'} \boldsymbol{S}_{ij'}^2 / \boldsymbol{f}_{j'}} \tag{10}$$

where $\boldsymbol{f}_j = \sum_i \boldsymbol{S}_{ij}$ are the soft cluster frequencies. To fit $\boldsymbol{S}$ to $\boldsymbol{T}$, an objective function is designed in Section III-E.

After training the feature clustering module as shown in Fig. 7(g), the feature codes mapped from the input image into k clusters will have strong discriminability. Fig. 7(h) and (i) shows that when using the feature clustering module, fewer defects remain in the generated background image and the residuals in the residual map are higher, which significantly improves the accuracy of defect inspection.

3) Decoding Module: For each FCAE subnetwork, the decoding module, which includes multiple deconvolutional layers, is used to upsample the discriminative code map and generate the texture background image, as shown in Fig. 5. The architecture of the decoding module is symmetric with that of the encoding module. Thus, to generate a background image with the same resolution as the input image, the total number e of deconvolutional layers is determined by the scale n of the FCAE subnetwork

$$e = n + 3. \tag{11}$$

The proposed MS-FCAE method utilizes multiple-scale FCAE subnetworks to reconstruct multiple texture background images. These background images, which are reconstructed by the different FCAE subnetwork scales, have different characteristics. The texture background image reconstructed at a large scale contains more texture details and smaller residuals between the input image and the background image. In contrast, the texture background image reconstructed at a small scale contains only the main texture structure, as shown in Fig. 3. This phenomenon can be explained by the pyramid structure of MS-FCAE. A Gaussian pyramid is a classic method that can effectively address multiple scales, and it is used in MSCDAE [5]. In a Gaussian pyramid, the image patches extracted from the input image with the same size become larger during the downsampling process, as shown in Fig. 9. In the proposed FCAE subnetwork, the feature vectors of the different code map scales have differently sized receptive fields and represent differently sized image patches. Due to the different scales of the FCAE subnetworks, MS-FCAE efficiently achieves the same effect as the Gaussian pyramid. Thus, MS-FCAE can address different types of textures that have different texture structure sizes.

D. Result Fusion Module for Sample Testing

To reduce false detections and improve inspection accuracy, the result fusion module is constructed to obtain the final inspection result, as shown in Fig. 3.

This network obtains multiple residual images $\boldsymbol{I}_n^{\text{res}}$ by subtracting the texture background images $\boldsymbol{I}_n^{\text{back}}$ from the matching input image $\boldsymbol{I}$, as per the following equation:

$$\boldsymbol{I}_n^{\text{res}} = abs\left(\boldsymbol{I} - \boldsymbol{I}_n^{\text{back}}\right) \tag{12}$$

where $\boldsymbol{I}_n^{\text{res}} \in R^{W \times H}$ is the residual image at scale n and $abs(\cdot)$ is the absolute value operation. In practical situations, to remove noise, a Gaussian filter operation and a morphological open operation are also applied to the residual images.

Next, the binary images $\boldsymbol{I}_n^{\text{bin}}$ are calculated by using a dual-threshold method to segment defects from the residual images $\boldsymbol{I}_n^{\text{res}}$ using the following equation:

$$\boldsymbol{I}_n^{\text{bin}}(i,j) = \begin{cases} 0, & \text{if } T_1^n < \boldsymbol{I}_n^{\text{res}}(i,j) < T_2^n \\ 1, & \text{otherwise} \end{cases} \tag{13}$$

where $i \in (1, \ldots, W)$, $j \in (1, \ldots, H)$, and $T_1^{(n)}$ and $T_2^{(n)}$ are the two thresholds for scale n. The threshold is a critical parameter that directly affects the MS-FCAE's defect inspection performance. Because all the training sample images represent defect-free samples, the two segmentation thresholds should be a bound for the residuals in these training samples. This means that all textures outside the boundaries should be considered as texture defects. To segment the defects robustly, the training residuals are assumed to obey a Gaussian distribution. Thus, the thresholds for the nth scale can be defined as follows:

$$\begin{aligned} T_1^n &= \mu^n - \varepsilon\sigma^n \\ T_2^n &= \mu^n + \varepsilon\sigma^n \end{aligned} \tag{14}$$

where μ^n and σ^n are the mean and standard deviation, respectively, calculated from the training samples. ε can be adjusted to control the segmentation sensitivity. The following experiment shows the influence of ε.

Finally, to limit false detections, the final result image $\boldsymbol{I}^{\text{result}}$ is obtained by fusing these binary images $\boldsymbol{I}_n^{\text{bin}}$ using a logical AND operation, as shown in the following equation:

$$\boldsymbol{I}^{\text{result}}(i,j) = \bigcap_{n=1}^{N} \boldsymbol{I}_n^{\text{bin}}(i,j). \tag{15}$$

This residual image fusion strategy causes the proposed MS-FCAE method to be more robust and more accurate at inspecting various defect types.

E. Two-Phase Model Training Strategy

To reconstruct multiple background images accurately, a multiscale loss function consisting of multiple FCAE loss functions is established to optimize the entire MS-FCAE model. In each FCAE subnetwork, the training phase includes two learning tasks: multiscale image background reconstruction and multiscale feature clustering.

For the image background reconstruction task, the loss function aims to minimize the differences between the input

image and the reconstructed background image. Thus, it is trained using mean square errors as a metric, as shown in the following equation:

$$L_n^{\text{rec}}\left(\boldsymbol{I}, \boldsymbol{I}_n^{\text{back}}\right) = \frac{1}{N_b}\sum_{n_b=1}^{N_b}\left\|{}^{n_b}\boldsymbol{I} - {}^{n_b}\boldsymbol{I}_n^{\text{back}}\right\|^2 + \lambda \sum_{\boldsymbol{w}\in\{\boldsymbol{W}\}}\|\boldsymbol{w}\|_F \quad (16)$$

where ${}^{n_b}\boldsymbol{I}$ denotes the n_bth input image, ${}^{n_b}\boldsymbol{I}_n^{\text{back}}$ represents the n_bth reconstructed background images of the n_b-th input image at scale n, and $n_b \in (1, \ldots, N_b)$, N^b is the batch size. Here, $\boldsymbol{W}$ is the set of parameter matrices in the MS-FCAE model. Finally, $0 < \lambda < 1$ is the penalty factor to control the model complexity.

To enhance the discriminability of the code vectors, the feature clustering task is trained with a feature clustering loss function that aims to minimize the difference between the source distribution $\boldsymbol{S}$ and the target distribution $\boldsymbol{T}$. Thus, the Kullback–Leibler (KL) divergence is used for this loss function

$$L_n^{\text{fc}} = KL(\boldsymbol{T}||\boldsymbol{S}) = \sum_i\sum_j \boldsymbol{T}_{ij}\log\frac{\boldsymbol{T}_{ij}}{\boldsymbol{S}_{ij}}. \quad (17)$$

Combining 16 and 17, the loss function of each FCAE module is as follows:

$$L_n = L_n^{\text{rec}} + \eta_n L_n^{\text{fc}} \quad (18)$$

where $\eta_n \geq 0$ controls the weights of the two loss terms. The η_n parameter can be set to a smaller number at small scales because the encoded features are more discriminative in the deeper layers; thus, it is unnecessary to enhance them using a large weight.

Thus, for MS-FCAE, the loss function is the sum of all the FCAE subnetworks

$$L = \sum_{n=1}^{N} L_n. \quad (19)$$

Because the image background reconstruction loss and the feature clustering loss of FCAE represent opposing optimization directions, it is difficult to obtain the optimum solution simultaneously. Thus, the MS-FCAE model is initialized alternately, as introduced in the following steps. First, because multiscale image background reconstruction is the main task of MS-FCAE, the partial model, which is separate from the feature-clustering modules in FCAE, is trained using only the multiscale image background reconstruction loss. Second, all the training samples are passed through the trained MS-FCAE model to calculate the corresponding feature code vectors. Then, the standard K-means clustering is conducted in the feature space to obtain k initial centroids $\boldsymbol{\mu}$. After these preliminary initializations, the entire MS-FCAE model is trained by jointly optimizing the multiscale image background reconstruction and the multiscale feature clustering. The target distribution $\boldsymbol{T}$ serves as the ground truth for the source distribution $\boldsymbol{S}$, but it also depends on $\boldsymbol{S}$. Therefore, to avoid instability, $\boldsymbol{T}$ should be limited to a certain number of iterations. In practice, $\boldsymbol{T}$ is updated by using all the training data over N^{iter} iterations (set to $N^{\text{iter}} = 2000$ in all the experiments).

Notably, due to the spatial period of texture and the dramatic computation efficiency of MS-FCAE, only a small number of texture surface samples (approximately 50 samples with a size of 256×256 pixels) are needed to perform an effective inspection, and no defect samples are required. This is highly important for industrial applications because it is often difficult to acquire and correctly label defect samples.

F. Parameter Setup

Apart from the weight parameters learned during the training phase, the proposed MS-FCAE method includes $4 + N$ parameters, i.e., N, k, ε, λ, and $(\eta_1, \ldots, \eta_N)$.

The total number N of the FCAE subnetwork influences the defect inspection accuracy and the model complexity. When N is large, the defect inspection accuracy is high because the network constructs more background images. However, a large N value also increases the complexity of the MS-FCAE model. To obtain a good tradeoff between the defect inspection accuracy and the model complexity, we recommend a setting, such that $1 \leq N \leq 5$. To trade off the computational time and accuracy, it is cost-efficient to set N to be 2.

The number of clusters k influences the background reconstruction accuracy. A too-small k value causes the clusters to retain too little texture information; this loss of information will lead to inferior background reconstruction accuracy. However, both the feature vectors and cluster centroids will be dense in a feature space when k is too large, which will cause difficulties in model convergence and thus influence the performance. Therefore, a value of k ranging from 15 to 25 is recommended. In this paper, k is set to 20.

The segmentation sensitivity ε in the result fusion module during the testing phase influences the final inspection accuracy. A too-small ε value results in the failure to inspect some defect regions correctly, while a too-large ε value leads to large numbers of false detections. In this paper, ε is set to 2.3.

During the training phase, there are $1 + N$ hyperparameters (i.e., λ and $(\eta_1, \ldots, \eta_N)$). The penalty factor λ $(0 < \lambda < 1)$ of the image background reconstruction loss in 16 controls the model complexity, and it is determined by cross validation [49]. After verification, this paper is used $\lambda = 0.005$ for the subsequent experiments.

The η_n $(\eta_n \geq 0)$ in 18 controls the weights of the reconstruction loss and the feature clustering loss in each FCAE subnetwork. η_n can be set to a smaller number at small scales because the encoded features are more discriminative in the deeper layers, and it is unnecessary to enhance them using a large weight. We recommend setting $\eta_n = 1 - 0.2 \times (n - 1)$. In this paper, for example, η_n is set to $\eta_1 = 1$ and $\eta_2 = 0.8$.

For different types of texture surface, the above-analyzed parameters can guarantee that MS-FCAE achieves the good performance. When it is necessary to tune the parameters to obtain better performance, some parameters can be fixed when a type of texture surface is given. The hyperparameters λ and $(\eta_1, \ldots, \eta_N)$ should be set during the training phase to guarantee that the training loss decreases smoothly. These methods can be fixed since they do not have much influence on the final inspection accuracy. The segmentation sensitivity ε in the result fusion module can also be fixed since it will not

TABLE I
TEST SET CONFIGURATION OF THE EIGHT TYPES OF TEXTURE SURFACES

Sample	Material	Defect Hardness	Test Image Number
(a)	screen [50]	simple	562
(b)	seat [50]	hard	543
(c)	scarf [50]	hard	581
(d)	fabric [51]	hard	534
(e)	knitwear [51]	hard	564
(f)	cotton [52]	hard	582
(g)	cement [53]	simple	526
(h)	TFT-LCD [11]	simple	573

influence the training phase. It is only necessary to tune N ($1 \leq N \leq 5$) and k ($15 < k < 25$) to make MS-FCAE to achieve the best possible inspection accuracy.

IV. EXPERIMENTS

This section describes a set of experiments conducted to evaluate the performance of the proposed MS-FCAE method. Specifically, to illustrate the MS-FCAE's efficiency, we compared its inspection time on various image resolutions with several related methods. Second, the influence of the cluster number k is investigated. This value is important because the clusters represent the textons of texture. Third, the relationship between the inspection performance of MS-FCAE and the segmentation threshold is verified. The segmentation threshold parameter directly affects the inspection performance. Finally, the overall inspection performance of the MS-FCAE is compared, both qualitatively and quantitatively, with several outstanding conventional methods and unsupervised methods.

While conducting these experiments, we established a synthetic data set composed of textural samples from various materials, including fabrics, cement tile, and TFT-LCD. In these data, the fabric samples were sourced from the Kylberg Texture data set [50], the Kylberg Sintorn Rotation data set [51], and the KTH-TIPS2 [52]. The cement tile samples were sourced from [53]; the defects are artificially generated, but they are similar to real-world problems and have been widely used for defect inspection verification. The TFT-LCD samples used were sourced from [11]; this data set contains various types of defects of different sizes and scales. Specifically, these data sets can be divided into two categories by the "hardness" of defect inspection. For each type of texture surface data set, we use 50 defect-free samples of this type of texture surface data set to train the corresponding MS-FCAE model and test it on the test set of this type of texture surface. The configuration of the test sets is reported in Table I.

To fairly compare the performances of these methods, this paper adopted three evaluation indicators: *Precision*, *Recall*, and *F1-Measure*, which are calculated as follows:

$$\text{Recall} = \frac{\text{TP}_p}{\text{TP}_p + \text{FN}_p} \times 100\% \tag{20}$$

$$\text{Precision} = \frac{\text{TP}_p}{\text{TP}_p + \text{FP}_p} \times 100\% \tag{21}$$

$$F1 - \text{Measure} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{22}$$

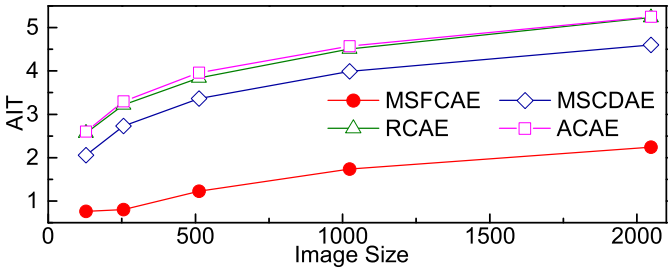


Fig. 10. AITs with different input image sizes.

where FP_p denotes the proportion of falsely detected defect areas in the background region, TP_p denotes the proportion of correctly detected defect areas in the defect region, and FN_p denotes the proportion of undetected defect areas in the defect region. *F1-Measure* is a comprehensive evaluator that utilizes both the *Precision* and *Recall* indicators. All the MS-FCAE models are trained on a computer with four Nvidia Titan Xp GPUs, and all compared experiments were conducted on the same computer, which was equipped with 64 GB of RAM, an Intel Xeon E5-2600 v3 processor, and a Windows 10 64-bit operating system. The speed was evaluated on this computer with the NVIDIA Titan Xps, CUDA 8.0, and cuDNN v6. For each experiment, we trained MS-FCAE five times under the same configuration and tested it under the same configuration. The experimental result was obtained by calculating the mean value of the five results. The MS-FCAE method was implemented using Python.

A. Efficiency Evaluation

An online AOI system must be able to meet real-time inspection demands. To evaluate the speed of the proposed MS-FCAE method, the average inspection time (AIT) was recorded while using various input image sizes. To verify the efficiency of MS-FCAE, we compared it with several existing unsupervised DNN methods, including MSCDAE [5], RCAE [37], and ACAE [35]. The parameters of the compared methods were set to their default parameters—that is, the patch sizes were set to 8×8 pixels, 32×32 pixels, and 32×32 pixels in MSCDAE, RCAE, and ACAE, respectively. The batch size during testing was set to be as large as possible.

The experimental results are shown in Fig. 10 and Table II. Fig. 10 presents the relationship between image resolution and AIT. Note that to show the AIT values intuitively, we display logarithmic values rather than actual values. For each method, image sizes ranging from 128×128 pixels to 2048×2048 pixels were utilized while the patch extraction stride was gradually enlarged, i.e., from 1 to 2, 4, 8, and 16, respectively. Fig. 10 shows that in spite of the Gaussian pyramid in MSCDAE, its AIT values are less than those of RCAE and ACAE. This result occurs because the patch size of MSCDAE is less than the other two, and smaller patches lead to less redundant computation. However, at every image size, MS-FCAE is more than 200 times faster than other methods. This high efficiency is because MS-FCAE requires no redundant computation at all. From Table II, it can be observed that

TABLE II
AITs With Different Image Sizes (Unit: ms)

Method / Resolution	**MS-FCAE**	MSCDAE	RCAE	ACAE
128	5.781	114.207	371.747	398.279
256	6.304	537.634	1652.893	1996.372
512	16.787	2320.186	6906.077	9090.909
1024	54.437	9746.589	32258.079	37037.040
2048	174.216	39658.225	170967.042	178352.203

even when the image size is 1024 × 1024 pixels, the AIT of MS-FCAE is 54 ms, which meets real-time inspection demands. In comparison, other methods are slow and cannot be used in practical applications. As the resolution increases, the AIT values of RCAE, ACAE, and MSCDAE increase rapidly (by more than four times). However, MS-FCAE AIT values increase much more slowly in comparison (approximately 1.1 times from 128 × 128 pixels to 256 × 256 pixels and only 3.2 times for larger images). This result occurs because the number of patches in high-resolution images is very large and GPU memory is limited; thus, the GPU cannot process all the patches simultaneously. Patches can be processed at only a certain batch size. Consequently, the AIT values increase rapidly as the resolution increases. In contrast, in MS-FCAE, the whole image can be processed at the same time and causes a slower increase in AIT.

This experiment verifies that the proposed MS-FCAE is dramatically more efficient than the existing methods. The reason for this efficiency improvement is that the fully convolutional feature extraction network and the FCAE subnetwork take an entire image as input without any redundant computation. The high efficiency proves that MS-FCAE meets the real-time inspection demands that are crucial for industrial applications.

B. Influence of the Number of Clusters

In FCAE, the clusters contain texton information in the texture. The cluster number k should be small for simple textures and large for complex textures. The best value is when k equals the number of textons; however, determining the number of textons is subjective. To determine k quickly, its range can be narrowed by performing a background reconstruction verification first and then selecting an appropriate value based on the inspection verification. To illustrate the influence of k intuitively, the data set (g) is used.

1) Background Reconstruction Verification: To further quantify the effect of parameter k, the relationship between the varied k values and the corresponding average background reconstruction residual (ARR) on the scale_1 FCAE subnetwork of MS-FCAE is studied in this experiment.

The experimental results in Fig. 11(a) show the reconstructed background images of a defect-free input at the scale_1 FCAE subnetwork of MS-FCAE and their residual maps using different k values over a large range. The reconstructed background images show that the background images reconstructed at both small k values (such as 10 or 15) and

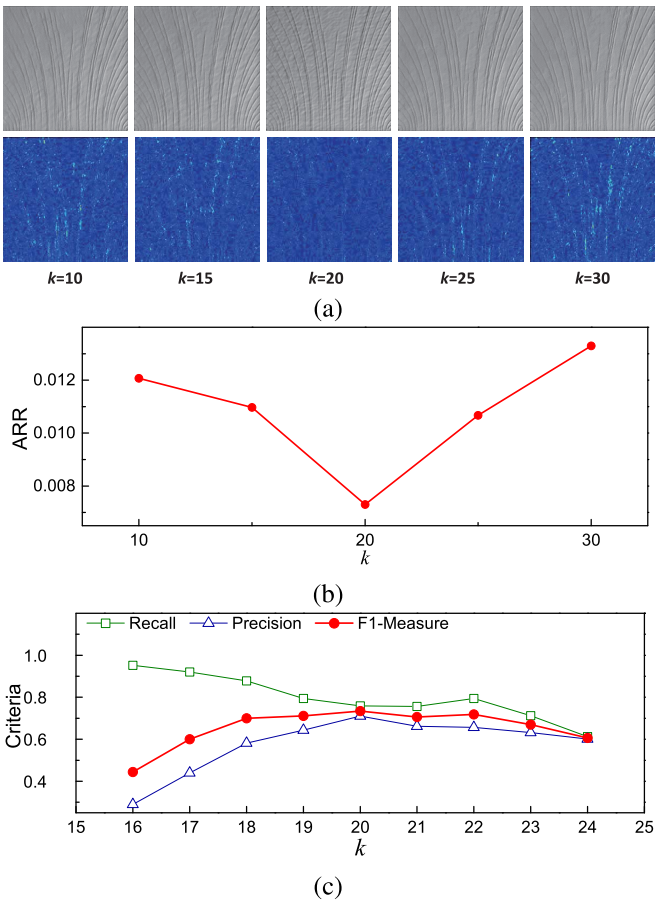


Fig. 11. Influence of the number of clusters k. (a) Influence of the number of clusters k on ARR. (b) Influence of the number of clusters k on Criteria.

at large k values (such as 25 or 30) are corrupted. The images are more refined at a medium k value. Furthermore, the curves of the varying k values are plotted against the corresponding ARR on the scale_1 FCAE subnetwork of MS-FCAE in Fig. 11(b). The curve trend shows that as k increases, the ARR first drops and then rises. This result occurs because a too-small k value causes the clusters to retain too little texture information; this loss of information leads to inferior background reconstruction accuracy. However, both feature vectors and cluster centroids will be dense in a feature space with a too-large k value. During training, these dense distributions cause the soft assignment S to be unstable in every iteration. The instability results in poor optimization and causes convergence difficulties. Thus, MS-FCAE cannot be trained well when k is too large.

The results of the background reconstruction verification experiment show that the number of clusters k should be set to a medium value to guarantee the background reconstruction accuracy. This experiment helped in choosing a proper range for k. For this paper, we determined that the range should be set from 15 to 25.

2) Inspection Verification: To further narrow down the cluster number k value beyond the 15–25 range determined by the background reconstruction verification experiment, we experimented with MS-FCAE models by testing various k values within that range.

TABLE III
DEFECT INSPECTION RESULTS OF MS-FCAE WITH FEATURE CLUSTERING AND WITHOUT FEATURE CLUSTERING

Method \ Metrics	*Recall*	*Precision*	*F1-Measure*
W/ FC	0.74	0.81	0.78
W/O FC	0.42	0.61	0.50

Fig. 11(c) shows the relationship between k and the corresponding *Precision*, *Recall*, and *F1-Measure*. The *Recall* and *Precision* trends show as k increases, the *Recall* drops; however, the *Precision* first increases and then decreases. This result occurs because within a small range, a smaller cluster number causes the features to be more discriminative because there are fewer categories, but it increases the false inspection because the reconstruction results are worse than those when using a larger k value. Thus, a smaller k value will lead to a higher *Recall* but a reduced *Precision*. However, with a too-large k value, the hard defects are reconstructed due to the lack of discriminative features, which causes a lower TP_p value and a reduced *Precision*. As a tradeoff between *Precision* and *Recall*, the *F1-Measure* is utilized. Fig. 11(c) shows the *F1-Measure* trend, which first increases and then decreases as k increases further. Thus, the k value should be selected when the *F1-Measure* reaches its maximum value. For the data set used in this paper, $k = 20$ can be set.

C. Influence of the Feature Clustering Module

As described in Section III-C2, the feature clustering module is a crucial part that can improve the reconstruction accuracy of the texture background image. Fig. 7 shows the influence of the feature clustering module qualitatively; to analyze it quantitatively, a comparison between MS-FCAE with and without the feature clustering module is performed using sample Fig. 7(a). The training epochs of MS-FCAE with and without the feature clustering module are the same for fairness. The three metrics *Recall*, *Precision*, and *F1-Measure* are utilized to analyze the inspection results. As indicated in Table III, MS-FCAE with feature clustering module (W/FC) is superior in terms of all three metrics. For example, compared with the MS-FCAE without the feature clustering module (W/O FC), the feature clustering module helps to improve the *Recall* by margins of 0.32, *Precision* by margins of 0.2, and *F1-Measure* by margins of 0.28. These experimental results demonstrate that the feature clustering module is a significant model that can improve the performance of MS-FCAE.

D. Influence of the Segmentation Sensitivity

As described in Section III-D, the segmentation sensitivity ε is a crucial parameter that directly affects the defect inspection performance. However, both the parameter cluster number k and the segmentation sensitivity ε influence the final results. However, while ε can be adjusted in a refined fashion (because it is a decimal), k can be only roughly adjusted (because it is an integer). Therefore, the appropriate k value should be determined first using a fixed segmentation sensitivity; then, ε can be adjusted finely. This experiment is established based on the fixed $k = 20$ setting chosen in Section IV-B.

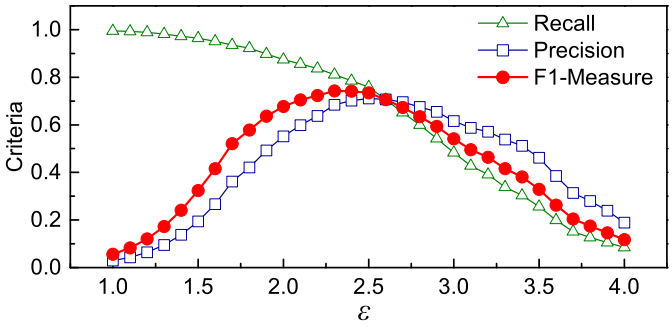


Fig. 12. Influence of segmentation sensitivity ε.

Fig. 12 shows the relationship between ε and the corresponding *Precision*, *Recall*, and *F1-Measure*. Based on the trends of *Precision* and *Recall*, as ε increases, *Recall* decreases, whereas *Precision* increases at first and then decreases. To trade off between recall and precision, ε is determined by using the *F1-Measure* on a validation set. For this data set, the setting $\varepsilon = 2.3$ was chosen.

E. Overall Performance Comparison

To verify the performance of the proposed MS-FCAE method, in this section, the inspection performance of MS-FCAE is compared with those of several existing traditional methods, including LCA [15], PHOT [16], TEXEMS [17], and other unsupervised DNN-based methods, including the CAE for anomaly detection (ACAE) [35], the RCAE [37], and the MSCDAE [5]. All the compared methods are evaluated using the data sets listed in Table I. For each type of texture surface, an MS-FCAE model is trained using 50 defect-free samples of this type of texture surface and is tested on the test set for this type of texture surface. The parameter setup is the same for all types of texture surfaces, as analyzed in Section III-F.

Some of the inspection results of these methods on the eight types of texture samples in the data set are shown in Fig. 13. As shown, in the LCA [15] method (the third row), the periodic textures are usually removed by eliminating the high-frequency parts while retaining the defect (low-frequency parts). The LCA obtains a good performance on simple regular textures, such as samples in Fig. 13(b), (f), and (h). However, it is completely incapable of detecting irregular textures when the frequency domain is complex, such as samples in Fig. 13(c) and (g). The PHOT [16] has no difficulty on images that are irregular but contains patterns that are similarly perceived, such as sample in Fig. 13(g). However, this ability can also be considered as a limitation because large defects can be considered regular subpatterns, such as the defects in Fig. 13(c) and (h). The TEXEMS [17] method models texture using a GMM by extracting color features. It has a good performance on defects with color differences regardless of the regularity or complexity of the texture, such as the defect in Fig. 13(a), (b), (e), and (f).

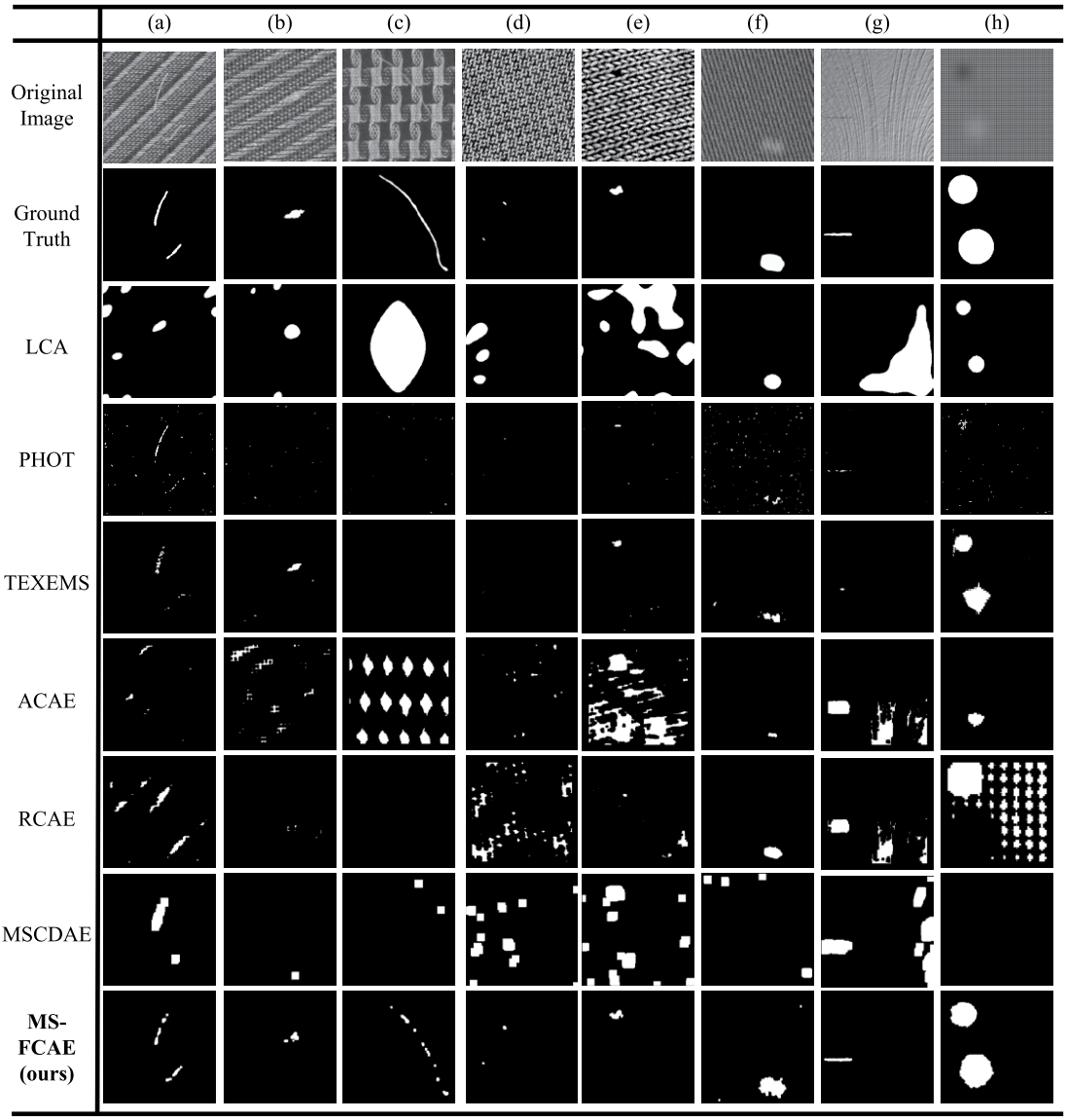


Fig. 13. Examples of the defect inspection performances of the compared methods. (a) Screen sample. (b) Seat sample. (c) Scarf sample. (d) Fabric sample. (e) Knitwear. (f) Cotton sample. (g) Cement sample. (h) TFT-LCD sample.

However, this method also has a limitation for the low-contrast defects in Fig. 13(c), (d), and (g). The ACAE [35] has no constraints on features or processes on input data; thus, it is extremely easy to reconstruct hard defects, which confuses its inspection results on the texture patterns in all the samples except Fig. 13(f). The RCAE [37] method introduces a noise matrix into the ACAE method to make it robust to noise. However, its inspection accuracy is inferior for random texture surface defects because it may consider some image patches as noise during training, such as the samples in Fig. 13(a)–(d). MSCDAE [5] achieves a reasonable *Recall* but relatively low *Precision*. This result occurs because its inspection result on the high-level pyramid is rough due to the low image resolution. All the results of MSCDAE show a square segmentation region. In contrast, the proposed MS-FCAE obtains a good performance on all types of defects and textures. Whether on large defects [samples Fig. 13(a), (c), (g), and (h)] or small defects [samples Fig. 13(b), (d), and (e), complex textures [sample Fig. 13(c)] or irregular textures [sample Fig. 13(g)], MS-FCAE always achieves a good performance. Notably, for Fig. 13(c), all the methods except the proposed MS-FCAE achieve only a poor performance due to the simultaneous existence of complex textures and hard defects. MS-FCAE handles both situations well simultaneously.

The three metrics *Recall*, *Precision*, and *F1-Measure* are utilized to analyze the inspection results quantitatively. The inspection results of all the methods on the eight types of texture samples in the data set are reported in Table IV. The proposed MS-FCAE method is superior in terms of all three metrics: *Recall*, *Precision*, and *F1-Measure*. For example, compared with the most accurate MSCDAE method, the proposed MS-FCAE method improves the *F1-Measure* by margins of 0.168, 0.232, 0.275, 0.275, 0.122, 0.346, 0.134, and 0.204 on the eight types of texture samples.

These experimental results demonstrate that MS-FCAE achieves the best performance on the eight textured surfaces

TABLE IV
DEFECT INSPECTION RESULTS OF DIFFERENT METHODS

Criteria, Sample \ Method		LCA	PHOT	TEXEMS	ACAE	RCAE	MSCDAE	**MS-FCAE (ours)**
Recall	(a) screen	0.585	0.076	0.253	0.219	0.578	0.669	**0.740**
	(b) seat	0.732	0.204	0.581	0.377	0.676	0.550	**0.880**
	(c) scarf	0.297	0.072	0.032	0.172	0.163	0.078	**0.330**
	(d) fabric	0.482	0.311	0.438	0.346	0.175	0.312	**0.734**
	(e) knitwear	0.286	0.271	0.253	0.242	0.598	0.568	**0.823**
	(f) cotton	0.663	0.155	0.394	0.103	0.656	0.562	**0.911**
	(g) cement	0.117	0.341	0.142	0.726	0.629	**0.966**	0.811
	(h) TFT-LCD	0.527	0.354	0.572	0.297	0.872	0.565	**0.982**
Precision	(a) screen	0.480	0.168	0.439	0.401	0.648	0.564	**0.810**
	(b) seat	0.191	0.200	0.543	0.053	0.545	0.550	**0.745**
	(c) scarf	0.007	0.082	0.073	0.007	0.196	0.094	**0.390**
	(d) fabric	0.029	0.161	0.210	0.211	0.296	0.220	**0.419**
	(e) knitwear	0.001	0.141	0.147	0.317	0.278	0.311	**0.385**
	(f) cotton	0.436	0.324	0.582	0.531	0.234	0.463	**0.800**
	(g) cement	0.002	0.478	0.265	0.080	0.556	0.444	**0.685**
	(h) TFT-LCD	0.508	0.337	0.567	0.387	0.409	0.702	**0.714**
F1-Measure	(a) screen	0.527	0.105	0.321	0.283	0.611	0.612	**0.780**
	(b) seat	0.304	0.202	0.561	0.093	0.603	0.578	**0.810**
	(c) scarf	0.013	0.077	0.045	0.013	0.178	0.085	**0.360**
	(d) fabric	0.055	0.212	0.284	0.262	0.220	0.258	**0.533**
	(e) knitwear	0.002	0.186	0.186	0.274	0.379	0.402	**0.524**
	(f) cotton	0.526	0.210	0.470	0.172	0.345	0.508	**0.854**
	(g) cement	0.004	0.398	0.185	0.144	0.590	0.608	**0.742**
	(h) TFT-LCD	0.517	0.345	0.569	0.336	0.557	0.626	**0.830**

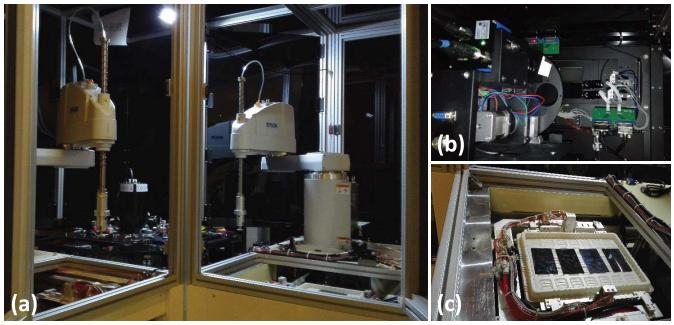


Fig. 14. TFT-LCD defect inspection equipment. (a) Complete equipment, (b) Vision system. (c) Inspected TFT-LCDs.

among all the compared methods. MS-FCAE's improved inspection accuracy is due to several factors. First, the FCAE subnetworks of MS-FCAE can address both regular and irregular textures due to feature representations extracted by the CNNs. Second, the feature clustering module of the FCAE subnetwork improves the discriminability of features between textons and defects. This suppresses the reconstruction of hard defects and thus leads to a higher *Recall*. Third, the multiscale FCAE network helps in addressing defects of various sizes. Fourth, the result fusion module can inspect defects robustly and reduce false detections, resulting in a high *Precision*.

F. Application

To further evaluate the practical performance of MS-FCAE, it is implemented in our AOI equipment and applied to the online inspection of TFT-LCD panel defects, as shown in Fig. 14. The training step is performed off-line to generate a model, which is implemented on a server with 4 Nvidia Titan Xp GPU cards. For our online inspection, the computer employed has 64 GB of RAM, an Intel Xeon E5-2600 v3 processor, and a Nvidia Titan Xp GPU card. The MS-FCAE model is implemented in Python, and parallel computation is performed using CUDA 8.0 and cuDNN v6. The MS-FCAE model is used only for online inference, and the training phase of the MS-FCAE model is conducted off-line on the computer used in the foregoing experiments. A TFT-LCD defect data set (the second TFT-LCD defect data set), including three main types of defects (i.e., spot defects, line defects, and region defects), was collected. The data set contains 2000 defect sample images and 50 defect-free sample images at high resolution (1920×1080 pixels). We utilized the 50 defect-free sample images for training and the 2000 defect sample images for testing and compared its results with the same texture surface defect methods used previously (LCA, PHOT, TEXEMS, ACAE, RCAE, and MSCDAE). In addition, the same parameter settings are used in this experiment as those described previously.

Some inspection results of the compared methods are shown in Fig. 15. The original images and the ground-truth images are shown in the first and second columns. The remaining columns show the results of LCA, PHOT, TEXEMS, ACAE, RCAE, MSCDAE, and MS-FCAE. The inspection results show that MS-FCAE both simultaneously and accurately inspects all types of TFT-LCD defects, whereas the other defect inspection methods are suitable only for certain types of defects.

Furthermore, the quantitative average *Recall* and *Precision* of defect inspection on the TFT-LCD defect data set are shown in Fig. 16. The *Recall* values of LCA, PHOT, TEXEMS,

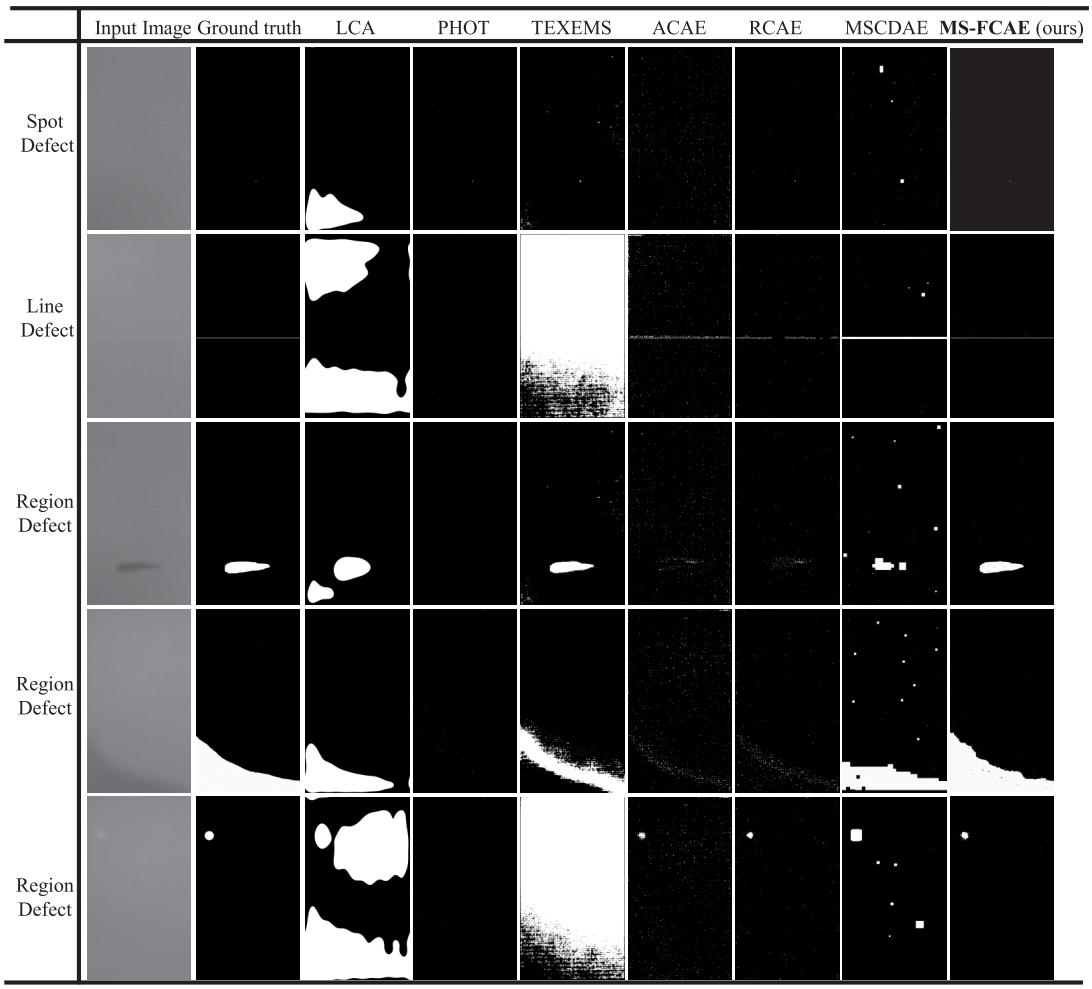


Fig. 15. Defect inspection results obtained for five TFT-LCD images by LCA, PHOT, TEXEMS, ACAE, RCAE, MSCDAE, and the proposed MS-FCAE.

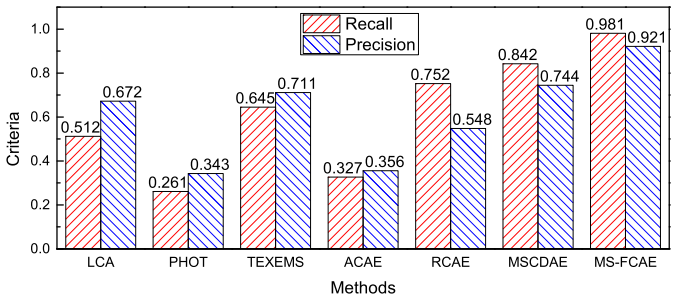


Fig. 16. TFT-LCD defect inspection results of different defect inspection methods for 2000 sample TFT-LCD panel images.

ACAE, RCAE, MSCDAE, and MS-FCAE are 0.512, 0.261, 0.645, 0.327, 0.752, 0.842, and 0.981, and their *Precision* scores are 0.672, 0.343, 0.711, 0.356, 0.548, 0.744, and 0.921, respectively. Compared with MSCDAE, which performs better than LCA, PHOT, ACAE, and RCAE, the *Recall* value of the proposed MS-FCAE is greater by approximately 0.14, and the *Precision* is approximately 0.18 greater. Clearly, both the *Recall* and *Precision* scores of MS-FCAE are better than those of the other six methods for the second TFT-LCD defect data set.

Table V reports the AIT of these compared methods on the same Intel CPU and GPU. MS-FCAE's time consumption

TABLE V
AIT ON THE SECOND TFT-LCD DEFECT DATA SET (UNIT: ms)

LCA	PHOT	TEXEMS	ACAE	RCAE	MSCDAE	**MS-FCAE**
900	860	52	73054	63425	19321	82

is approximately 82 ms for a 1920 × 1080-pixel image. The computation times of LCA, PHOT, TEXEMS, ACAE, RCAE, and MSCDAE are approximately 900, 860, 52, 73054, 63425, and 19321 ms, respectively. Among these methods, ACAE, RCAE, and MSCDAE are not able to satisfy the requirements of practical applications, while LCA, PHOT, and TEXEMS reach almost the same level as the proposed MS-FCAE. Nevertheless, their *Recall* scores are too low, which limits their practical applications. The proposed MS-FCAE meets industrial real-time inspection demands.

In summary, all the experimental results demonstrate that the proposed MS-FCAE method achieves not only the best inspection accuracy but also the lowest time consumption among all the compared methods and can meet online industrial inspection requirements. The feature extraction network and the multiscale FCAE network process the whole input image at one time, which improves the efficiency

of MS-FCAE. The multiscale FCAE network reconstructs multiple texture background images at different scale levels and utilizes the feature clustering module to enhance the discriminability of the encoded features to improve the background reconstruction accuracy. This approach improves the inspection accuracy for various types of texture defects.

V. Conclusion

In this paper, we introduce an unsupervised MS-FCAE method that is suitable for simultaneous inspection of various types of texture surface defects while requiring only a small number of defect-free texture samples for training; this method does not require any defect samples. The proposed MS-FCAE utilizes multiple FCAE subnetworks at different scale levels to reconstruct texture background images. The residual images can be obtained by subtracting these texture background images from the input image; then, these images are fused into one defect image. To achieve high time efficiency, each FCAE subnetwork utilizes a fully convolutional neural network to extract the original feature maps directly from the input image. In addition, each FCAE subnetwork uses feature clustering to improve the discriminability of the encoded feature maps. Extensive experimental results on several texture surface inspection data sets demonstrated that MS-FCAE achieves the state-of-the-art inspection accuracy with high efficiency. Moreover, this method significantly improves the inspection accuracy for hard defects. Furthermore, MS-FCAE can be easily integrated into practical online inspection applications by using the GPU-based parallel processing strategy. In the future, we will apply MS-FCAE to defect inspection for more texture surfaces. In addition, since the features of shallow layers in the feature extraction network are not of high-level semantics, we will further improve its performance by improving the semantic representation of the shallow layers.

References

[1] O. Silvén, M. Niskanen, and H. Kauppinen, "Wood inspection with non-supervised clustering," *Mach. Vis. Appl.*, vol. 13, nos. 5–6, pp. 275–285, 2003.

[2] B. Mallik-Goswami and A. K. Datta, "Detecting defects in fabric with laser-based morphological image processing," *Textile Res. J.*, vol. 70, no. 9, pp. 758–762, 2000.

[3] Z. W. Wang, M. Zhou, G. G. Slabaugh, J. Zhai, and T. Fang, "Automatic detection of bridge deck condition from ground penetrating radar images," *IEEE Trans. Autom. Sci. Eng.*, vol. 8, no. 3, pp. 633–640, Jul. 2011.

[4] H. Yang, K. Song, S. Mei, and Z. Yin, "An accurate Mura defect vision inspection method using outlier-prejudging-based image background construction and region-gradient-based level set," *IEEE Trans. Autom. Sci. Eng.*, vol. 15, no. 4, pp. 1704–1721, Oct. 2018.

[5] S. Mei, H. Yang, and Z. Yin, "An unsupervised-learning-based approach for automated defect inspection on textured surfaces," *IEEE Trans. Instrum. Meas.*, vol. 67, no. 8, pp. 1266–1277, Jun. 2018.

[6] B.-K. Kwon, J.-S. Won, and D.-J. Kang, "Fast defect detection for various types of surfaces using random forest with VOV features," *Int. J. Precis. Eng. Manuf.*, vol. 16, no. 5, pp. 965–970, May 2015.

[7] H. Yang, S. Zheng, J. Lu, and Z. Yin, "Polygon-invariant generalized Hough transform for high-speed vision-based positioning," *IEEE Trans. Autom. Sci. Eng.*, vol. 13, no. 3, pp. 1367–1384, Jul. 2016.

[8] B. Zhang, H. Yang, and Z. Yin, "A region-based normalized cross correlation algorithm for the vision-based positioning of elongated IC chips," *IEEE Trans. Semicond. Manuf.*, vol. 28, no. 3, pp. 345–352, Aug. 2015.

[9] B. Zhang, H. Yang, and Z. Yin, "A spatial-constraint-based feature point matching algorithm for the positioning of multiple IC instances," *IEEE Trans. Semicond. Manuf.*, vol. 29, no. 2, pp. 137–144, May 2016.

[10] H. Yang, S. Mei, K. Song, B. Tao, and Z. Yin, "Transfer-learning-based online Mura defect classification," *IEEE Trans. Semicond. Manuf.*, vol. 31, no. 1, pp. 116–123, Feb. 2018.

[11] S. Mei, H. Yang, and Z. Yin, "Unsupervised-learning-based feature-level fusion method for mura defect recognition," *IEEE Trans. Semicond. Manuf.*, vol. 30, no. 1, pp. 105–113, Feb. 2017.

[12] X. Xie, "A review of recent advances in surface defect detection using texture analysis techniques," *Electron. Lett. Comput. Vis. Image Anal.*, vol. 7, no. 3, pp. 1–22, 2008.

[13] H.-F. Ng, "Automatic thresholding for defect detection," *Pattern Recognit. Lett.*, vol. 27, no. 14, pp. 1644–1649, 2006.

[14] T. Vujasinovic *et al.*, "Gray-level co-occurrence matrix texture analysis of breast tumor images in prognosis of distant metastasis risk," *Microsc. Microanal.*, vol. 21, no. 3, pp. 646–654, 2015.

[15] D.-M. Tsai and T.-Y. Huang, "Automated surface inspection for statistical textures," *Image Vis. Comput.*, vol. 21, no. 4, pp. 307–323, 2003.

[16] D. Aiger and H. Talbot, "The phase only transform for unsupervised surface defect detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2010, pp. 295–302.

[17] X. Xie and M. Mirmehdi, "TEXEMS: Texture exemplars for defect detection on random textured surfaces," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 8, pp. 1454–1464, Aug. 2007.

[18] J. Zhou, D. Semenovich, A. Sowmya, and J. Wang, "Dictionary learning framework for fabric defect detection," *J. Textile Inst.*, vol. 105, no. 3, pp. 223–234, 2014.

[19] S. Mehta, A. P. Azad, S. A. Chemmengath, V. Raykar, and S. Kalyanaraman. (2017). "DeepSolarEye: Power loss prediction and weakly supervised soiling localization via fully convolutional networks for solar panels." [Online]. Available: https://arxiv.org/abs/1710.03811

[20] Z. Yu, X. Wu, and X. Gu, "Fully convolutional networks for surface defect inspection in industrial environment," in *Proc. Int. Conf. Comput. Vis. Syst.*, 2017, pp. 417–426.

[21] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Proc. Int. Conf. Med. Image Comput. Comput.-Assisted Intervent*. Cham, Switzerland: Springer, 2015, pp. 234–241.

[22] Z. Zhou, M. M. R. Siddiquee, N. Tajbakhsh, and J. Liang. (2018). "UNet++: A nested U-net architecture for medical image segmentation." [Online]. Available: https://arxiv.org/abs/1807.10165

[23] R. Caruana, "Multitask learning," *Mach. Learn.*, vol. 28, no. 1, pp. 41–75, 1998.

[24] R. Ren, T. Hung, and K. C. Tan, "A generic deep-learning-based approach for automated surface inspection," *IEEE Trans. Cybern.*, vol. 48, no. 3, pp. 929–940, Mar. 2018.

[25] K. Gopalakrishnan, S. K. Khaitan, A. Choudhary, and A. Agrawal, "Deep convolutional neural networks with transfer learning for computer vision-based data-driven pavement distress detection," *Construction Building Mater.*, vol. 157, pp. 322–330, Dec. 2017.

[26] S. Kim, W. Kim, Y.-K. Noh, and C. P. Frank, "Transfer learning for automated optical inspection," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, May 2017, pp. 2517–2524.

[27] Y. Li, W. Zhao, and J. Pan, "Deformable patterned fabric defect detection with fisher criterion-based deep learning," *IEEE Trans. Autom. Sci. Eng.*, vol. 14, no. 2, pp. 1256–1264, Apr. 2017.

[28] M. Kang, K. Ji, X. Leng, X. Xing, and H. Zhou, "Synthetic aperture radar target recognition with feature fusion based on a stacked autoencoder," *Sensors*, vol. 17, no. 1, p. 192, 2017.

[29] F. Gu, F. Flórez-Revuelta, D. Monekosso, and F. Remagnino, "Marginalised stacked denoising autoencoders for robust representation of real-time multi-view action recognition," *Sensors*, vol. 15, no. 7, pp. 17209–17231, 2015.

[30] P. He, P. Jia, S. Qiao, and S. Duan, "Self-taught learning based on sparse autoencoder for e-nose in wound infection detection," *Sensors*, vol. 17, no. 10, p. 2279, 2017.

[31] S. Malek, F. Melgani, M. L. Mekhalfi, and Y. Bazi, "Real-time indoor scene description for the visually impaired using autoencoder fusion strategies with visible cameras," *Sensors*, vol. 17, no. 11, p. 2641, 2017.

[32] Z. Zhu, X. Wang, S. Bai, C. Yao, and X. Bai, "Deep learning representation using autoencoder for 3D shape retrieval," *Neurocomputing*, vol. 204, pp. 41–50, Sep. 2016.

[33] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

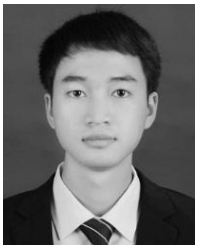
[34] J. Masci, U. Meier, D. Cireşan, and J. Schmidhuber, "Stacked convolutional auto-encoders for hierarchical feature extraction," in *Proc. Int. Conf. Artif. Neural Netw.*, 2011, pp. 52–59.
[35] M. Ke, C. Lin, and Q. Huang, "Anomaly detection of Logo images in the mobile phone using convolutional autoencoder," in *Proc. IEEE Int. Conf. Syst. Inform.*, Nov. 2017, pp. 1163–1168.
[36] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proc. Int. Conf. Mach. Learn.*, 2008, pp. 1096–1103.
[37] R. Chalapathy, A. K. Menon, and S. Chawla, "Robust, deep and inductive anomaly detection," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discovery Databases*, 2017, pp. 36–51.
[38] R. E. Bellman, *Adaptive Control Processes: A Guided Tour*. Princeton, NJ, USA: Princeton Univ. Press, 2015.
[39] J. Donahue *et al.*, "DeCAF: A deep convolutional activation feature for generic visual recognition," in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. 647–655.
[40] M.-N. Wu, C.-C. Lin, and C.-C. Chang, "Brain tumor detection using color-based K-means clustering segmentation," in *Proc. IEEE Int. Conf. Intell. Inf. Hiding Multimedia Signal Process.*, Nov. 2007, pp. 245–250.
[41] R. Sibson, "SLINK: An optimally efficient algorithm for the single-link cluster method," *Comput. J.*, vol. 16, no. 1, pp. 30–34, 1973.
[42] E. Forgy, "Cluster analysis of multivariate data: Efficiency versus interpretability of classification," *Biometrics*, vol. 21, no. 3, pp. 768–769, 1965.
[43] G. McLachlan and D. Peel, *Finite Mixture Models*. Hoboken, NJ, USA: Wiley, 2004.
[44] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *J. Roy. Statist. Soc.*, vol. 39, no. 1, pp. 1–38, 1977.
[45] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. KDD*, vol. 96, 1996, pp. 226–231.
[46] J. Xie, R. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 478–487.
[47] S.-C. Zhu, C.-E. Guo, Y. Wang, and Z. Xu, "What are textons?" *Int. J. Comput. Vis.*, vol. 62, nos. 1–2, pp. 121–143, 2005.
[48] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, pp. 2579–2605, Nov. 2008.
[49] R. Kohavi *et al.*, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *Proc. Int. Joint Conf. AI*, Aug. 1995, vol. 14, no. 2, pp. 1137–1145.
[50] G. Kylberg, "Kylberg texture dataset version 1.0," Centre Image Anal., Swedish Univ. Agricult. Sci., Uppsala, Sweden, Tech. Rep., 2011.
[51] G. Kylberg and I. Sintorn. (2013). *Kylberg Sintorn Rotation Dataset*. [Online]. Available: http://www.cb.uu.se/~gustaf/KylbergSintornRotation/
[52] P. Mallikarjuna, A. T. Targhi, M. Fritz, E. Hayman, B. Caputo, and J. O. Eklundh, "The KTH-TIPS2 database," Comput. Vis. Active Perception Lab., Stockholm, Sweden, Tech. Rep., 2006.
[53] M. Wieler and T. Hahn. (Jun. 25, 2017). *Weakly Supervised Learning for Industrial Optical Inspection*. [Online]. Available: https://hci.iwr.uni-heidelberg.de/node/3616

Hua Yang (M'13) received the B.S. and M.S. degrees from the Huazhong University of Science and Technology, Wuhan, China, in 2006 and 2008, respectively, and the Ph.D. degree from Hiroshima University, Higashihiroshima, Japan, in 2011.

He was with Hiroshima University as a Research Associate from 2011 to 2012 and subsequently as an Assistant Professor. He is currently an Associate Professor with the School of Mechanical Science and Engineering, Huazhong University of Science and Technology. His current research interests include high-speed vision and its applications (object recognition, detection and tracking, particle image velocity, and dynamic visual inspections).

Yifan Chen received the B.S. degree in mechanical design, manufacturing, and automation from the Huazhong University of Science and Technology, Wuhan, China, in 2016, where he is currently pursuing the M.S. degree with the State Key Laboratory of Digital Manufacturing Equipment and Technology.

His current research interests include image segmentation, object detection, and deep learning.



Kaiyou Song received the B.S. degree in mechanical design, manufacturing, and automation from Wuhan University, Wuhan, China, in 2015. He is currently pursuing the Ph.D. degree with the State Key Laboratory of Digital Manufacturing Equipment and Technology, Huazhong University of Science and Technology, Wuhan.

His current research interests include image segmentation, object detection, and deep learning.

Zhouping Yin (M'08) received the B.S., M.S., and Ph.D. degrees in mechanical engineering from the Huazhong University of Science and Technology, Wuhan, China, in 1994, 1996, and 2000, respectively.

Since 2005, he has been the Vice Head of the State Key Laboratory of Digital Manufacturing Equipment and Technology, Huazhong University of Science and Technology, where he is currently a Professor with the School of Mechanical Science and Engineering. He leads a research group and conducts research on electronic manufacturing equipment and technology, including flexible electronics and machine vision.