16th International Conference on Machine Vision Applications (MVA)
National Olympics Memorial Youth Center, Tokyo, Japan, May 27-31, 2019.

Autoencoder-Based Fabric Defect Detection with Cross-Patch Similarity

Hu Tian, Fei Li
Fujitsu Research & Development Center Co., Ltd., Beijing, China
{tianhu, lifei}@cn.fujitsu.com

Abstract

Fabric quality inspection plays an important role in the textile industry. As an effective approach to learn data representations, autoencoder has been adopted for defect detection. With the basic idea that the defect area cannot be recovered by the model trained on non-defective image patches, the residual is often used as an indication for defect judgement. However, usually the texture (non-defect) area in a defective patch also cannot be well reconstructed, which makes the pixel-wise detection inaccurate. In this paper, by exploring similarities between different patches in the whole test image, a novel autoencoder-based fabric defect detection method is proposed. In order to maintain the texture area in the reconstructed patch, the original encoded latent variable is modified, and the cross-patch similarity is introduced for determining the modification function. The whole algorithm is conducted in an iterative way, and the detection results will become better and better. Experimental results on the benchmark datasets demonstrate the effectiveness of our proposal.

1 Introduction

Fabric defect detection plays an important role in quality control of textile industry. Traditionally, this work is completed by human visual checking. Due to the fact that many defects are small and have low contrast, manual operation is not only time-consuming, but also unreliable. With the development of computer vision, automatic fabric inspection has been widely used. As the same time, fabric texture becomes more and more complex and diverse, which makes the task of quality inspection much more difficult. Therefore, effective and efficient techniques for automatic fabric defect detection are still in need.

Referring to the related surveys [1] [2], the most popular fabric defect detection methods can be generally divided into four categories: statistical, spectral, model-based and image-restoration-based ones. The statistical methods [3] [4] distinguish defects through various features, such as histogram and contrast, extracted from a local window. But they ignore the global image information and are sensitive to noises. The spectral methods [5] [6] [7] highlight the difference between defects and non-defects in frequency domains where Fourier, wavelet and Gabor transformations are typical used tools. However, these methods usually have poor performance for complex fabric images. The model-based methods [8] [9] first utilize probability distribution models such as Gaussian mixture model to represent the normal texture and then treat the areas which do not conform with the model as defects. But it is hard to detect small defects with low contrast. Inspired by the approaches in image de-noising such as sparse representation and low-rank decomposition, the image-restoration-based methods [10] [11] [12] expect to restore the non-defective version for the defective image. However, these methods have high computational complexity and usually generate many false alarms in non-defective regions.

In recent years, deep learning has been widely used, and its great advantages have been demonstrated in many applications including defect detection [13] [14] [15]. Most of these studies use supervised learning, which requires a large amount of defective samples for training. As a type of effective artificial neural network to learn data representations in an unsupervised manner, autoencoder has been adopted for defect detection [16]. It first encodes the input image patch into a latent variable, and then decodes the variable to recover the input. For model training with non-defective patches, the parameters are learned to make the reconstruction error as small as possible. The basic idea for defect detection on the test image is that the non-defective patches can be well reconstructed while the defective ones cannot, and the reconstruction residual can be used as an indicator for pixel-wise detection. However, usually the texture area in a defective patch also cannot be well reconstructed. Therefore, even the defective patches can be exactly selected, the pixel-wise detection results are often not accurate.

In this paper, we present a novel autoencoder-based fabric defect detection method by introducing cross-patch similarity. The reconstruction residual of each test image patch is still adopted for pixel-wise defect detection. To make the results more satisfactory, we try to maintain the texture area in the reconstructed patch. According to the observation that the texture patterns are usually repetitive in the fabric image and most of them are not defective, we modify the obtained latent variable after patch encoding to make the decoded patch from the modified latent variable close to its similar patches. In this way, if an input patch is defective and its similar patches are not, the final reconstructed patch will also be non-defective. That is to say, only the defect area cannot be well reconstructed and thus can be effectively detected. Our proposed algorithm is conducted in an iterative way, and the results of existing autoencoder-based methods can be used as its initial value. By repeatedly updating the modified latent variable and the defect position, the detection results will become more and more accurate.

The rest of the paper is organized as follows. Section 2 describes our proposed autoencoder with cross-patch similarity. Our experimental results are presented in Section 3, and it is followed by some conclusions and analysis of future work in Section 4.

2 Autoencoder with Cross-Patch Similarity

Our proposed fabric defect detection method mainly includes a training stage and a test stage. For both training and test images, some preprocessing operations, such as histogram equalization and other gray-level transformations, can be adopted to enhance contrast and reduce influence of illumination [10] [11] [16]. In the following of this section, first the training stage is introduced briefly since it is the same as the existing methods, then the test stage is talked about in detail as our key contribution.

2.1 Model training

The training stage of the existing autoencoder-based fabric defect detection methods can be adopted here. The patches from the available non-defective images are utilized to learn an encoder function $f : \mathcal{X} \to \mathcal{Z}$ and a decoder function $g : \mathcal{Z} \to \mathcal{X}$, where $\mathcal{X}$ and $\mathcal{Z}$ are spaces for image patches and latent variables, respectively. The forms of the functions $f(\mathbf{x}), (\mathbf{x} \in \mathcal{X})$ and $g(\mathbf{z}), (\mathbf{z} \in \mathcal{Z})$ are usually represented by multi-layer neural networks, in which the parameters are optimized by backward propagation. Besides the common idea for minimizing the reconstruction error between $\mathbf{x}$ and $\mathbf{x}' = g(f(\mathbf{x}))$, other generalized techniques such as variational autoencoder and sparse autoencoder can also be adopted. Since the same process is conducted, more details are omitted here.

2.2 Model test

The flowchart for the test stage of our proposed autoencoder with cross-patch similarity is illustrated in Figure 1. At first, the candidate defective patches are selected. Then for each patch, the position of the pixel-wise defect is determined and the latent variable after patch encoding is modified. The two steps are iteratively conducted on the candidate patches for several times. After the process has converged, the final defect detection results can be obtained. Next we will discuss each step in the flowchart one by one.

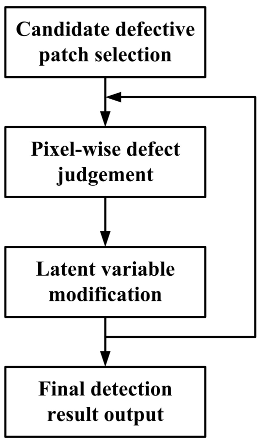
Candidate defective patch selection
Pixel-wise defect judgement
Latent variable modification
Final detection result output

Figure 1. Flowchart for the test stage of our proposed fabric defect detection method.

2.2.1 Candidate defective patch selection

The test image I is divided into non-overlapping patches with the same size as those adopted in the training stage. Each patch $\mathbf{x}$ is sent to the learned autoencoder, the corresponding reconstructed patch $\mathbf{x}' = g(f(\mathbf{x}))$ and the reconstruction residual $\mathbf{r} = \mathbf{x} - \mathbf{x}'$ are calculated. Since the autoencoder has been effectively trained from image patches without defects, the non-defective patches in I should be well reconstructed. Therefore, only the patches corresponding to large reconstruction errors are selected for further analysis. That is to say, the set of the candidate defective patches can be defined by

$$\mathcal{C} = \{ \ \mathbf{x} \mid \ \|\mathbf{x} - g(f(\mathbf{x}))\| > T_1, \ \mathbf{x} \in I \ \} \tag{1}$$

where T_1 is a predefined threshold, and can be determined by the mean and variance of reconstruction errors for the training image patches. Once the candidate set is constructed, the following operations are only conducted on these patches. In this way, it will not only relieve computational burden, but also reduce false alarms.

2.2.2 Pixel-wise defect judgement

For each image patch in the candidate set, the reconstruction residual $\mathbf{r}$ is used as the direct indicator for defect judgement, and the binary detection mask for the pixel (m, n) is determined as

$$\mathbf{M}(m,n) = \begin{cases} 0, & \text{if } \mathbf{r}(m,n) > T_2; \\ 1, & \text{otherwise.} \end{cases} \tag{2}$$

where T_2 is another predefined threshold, and can be decided by the statistics of pixel-wise reconstruction errors in the training stage. The pixels labeled with 0 are judged as defects.

For each candidate defective patch, the reason for its large reconstruction error is that it contains defect area which does not appear in the training samples. However, when we use the encoded latent variable to recover the input patch, it is impossible that the texture area can be maintained while the defect area cannot. In fact, some pixels in the texture area also cannot be well recovered, while some of the reconstructed pixels in the defect area are quite similar with the original ones. Therefore, two problems exist in the pixel-wise detection result based on the initial residual: high false alarms in the texture area and low recall in the defect area. In this paper, we propose the method of latent variable modification to tackle these two problems.

2.2.3 Latent variable modification

We still want to use the patch reconstruction residual as an indicator for determining the pixel-wise detection results, thus the ideal case is that the texture area can be well reconstructed while the defect area cannot. Since it is difficult to directly distinguish texture area based on the candidate patch itself, we try to resort to other non-defective patches in the test image. As we have already learned the latent representation for image patches based on the learned autoencoder, our basic idea is to use a suitable latent variable for non-defective patches as a reference to modify the original latent variable for each candidate defective patch.

For each image patch $\mathbf{x}$ in the candidate set, we first find its K most similar non-defective patches in the whole test image, and denote them as $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_K$. We simply treat the patches that are not in the candidate set $\mathcal{C}$ as non-defective, and the similar patches can be obtained based on some similarity measurement of the pixel values in the image patches. To reduce the negative impact of the defect area in $\mathbf{x}$, the patch $\mathbf{x}$ is masked with $\mathbf{M}$, and it is expected that only the pixels in texture areas are used for similarity calculation.

Let the latent variables for the candidate patch be $\mathbf{z}$ and its similar patches be $\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_K$. To define a reference latent variable to modify $\mathbf{z}$, we calculate the weighted average value $\bar{\mathbf{z}}$ for all the latent variable of the K similar patches as

$$\bar{\mathbf{z}} = \frac{\sum_{k=1}^{K} w_i \mathbf{z}_i}{\sum_{k=1}^{K} w_i} \tag{3}$$

where w_i $(i = 1, 2, \cdots, n)$ are weight coefficients, and determined by the similarity between $\mathbf{x}$ and $\mathbf{x}_i$.

A modification function for the original latent variable $\mathbf{z}$ is introduced and denoted as $d(\mathbf{z})$. The function can either be explicitly defined in a parametric form, or be implicitly defined by a neural network. In our experiments, a fully-connected layer with a weight matrix $\mathbf{W}$ is adopted, then the modification function can be simply written by

$$d(\mathbf{z}) = \mathbf{W}\mathbf{z} \tag{4}$$

The parameter $\mathbf{W}$ is determined by solving an optimization problem, whose cost function is an extension of autoencoder and composed of two items. Like the existing autoencoder-based methods, the first item is about the masked reconstruction error and defined as

$$Q_1 = \| \, (\mathbf{x} - g(\mathbf{W}\mathbf{z})) \circ \mathbf{M} \, \| \tag{5}$$

where $\circ$ is the operation of element-wise multiplication. $\mathbf{M}$ is adopted here as a mask since we just want to reconstruct the texture area in the image patch. Note that the parameters of the decoder function $g(\cdot)$ are fixed. To introduce the constraint about cross-patch similarity, we try to make the distance between the modified latent variable and the reference variable as small as possible, and define the second item as

$$Q_2 = \|\mathbf{W}\mathbf{z} - \bar{\mathbf{z}}\| \tag{6}$$

By combining the two items together, we can get the final cost function as

$$Q = Q_1 + \lambda Q_2 \tag{7}$$

and λ is a combination coefficient.

It should be noted that for the ideal case the repetitive texture patterns in the test image should be exactly the same, but in the practical applications there are some differences due to noises and distortions. Therefore, if we over-emphasize the cross-patch similarity and only take the item Q_2 into consideration for optimization, sometimes we cannot get satisfactory results, especially when the found similar patches are not accurate. Besides, because the texture area in a defective patch cannot be reconstructed well by the trained autoencoder, adding the masked reconstruction item Q_1 to the final cost function can enforce the modified latent variables to better reconstruct the texture area, which can also reduce false alarms.

2.2.4 Iterative refinement

After the optimal parameter $\mathbf{W}$ is calculated, the original latent variable $\mathbf{z}$ is modified as $\mathbf{W}\mathbf{z}$, and the reconstructed image patch is updated as $\mathbf{x}' = g(\mathbf{W}\mathbf{z})$. Then the reconstruction residual and the pixel-wise detection mask should also be re-calculated. Based on the new mask $\mathbf{M}$, we can find new similar patches for each candidate defective patch, update the corresponding weighted average latent variable $\bar{\mathbf{z}}$, and re-optimize the parameter $\mathbf{W}$. This process can be repeated for several times until it converges, and the final obtained defects will be more accurate than the initial results.

3 Experimental Results

3.1 Evaluation datasets

In order to verify the effectiveness of the proposed algorithm, two public fabric datasets are used in our experiments: TILDA dataset [17] and patterned fabric dataset [18]. Our experiments are performed separately on the two datasets. The TILDA dataset contains the most common types of fabric defects. The size of the fabric image is 768×512. We select 51 non-defective images for training and 100 defective images for test. Since there is no pixel-level ground truth, we only show the visual results of defect detection to evaluate our method.

The patterned fabric dataset is provided by Industrial Automation Research Laboratory from the University of Hong Kong. It consists of 256×256 fabric images belonging to three kinds of patterns: dot-, star- and box-patterned fabric. Since the ground truth defect masks are available, both quantitative and qualitative evaluations are performed on this dataset. As similar conclusions can be made for the three kinds of patterns, only the experimental results on the star-patterned fabrics are illustrated in this paper. There are altogether 25 defect-free images and 25 defective images in the dataset, and the defects are classified as five types: broken end, hole, netting multiple, thick bar and thin bar.

3.2 Implementation details

For the autoencoder model, we adopt a Conv-Deconv architecture. The encoder part consists of several convolutional layers with kernel size 4×4 and stride size 2×2. The following decoder part consists of several deconvolutional layers with kernel size 4×4 and stride size 2×2. The dimension of the latent variable for an image patch is set to 10. We randomly sample image patches from non-defective images to train the autoencoder. Different patch sizes are used on the two datasets: 32×32 for TILDA dataset and 16×16 for patterned fabric dataset. We implement the model using MXNet [19] framework and adopt Adam optimizer with initial learning rate 0.001 to train this model for 300,000 iterations.

For parameter setting, the combination coefficient λ is set to 1. The patch threshold T_1 is set to 13 and the pixel threshold T_2 is set to 40, which are determined according to experimental validation. The number of similar non-defective patches K is set to 5. And the refinement process is iterated for 2 times.

3.3 Results on the TILDA dataset

We first evaluate the performance of candidate defective patch selection. Some results are given in Figure 2. From Figure 2(a) and Figure 2(b), we can obviously see that the reconstructed images have large discrepancy in the defect areas compared with the raw images. This manifests that the trained autoencoder can be used to well reconstruct non-defective patches. Thus it is reasonable to select the candidate defective patches by the reconstruction errors. Figure 2(c) shows the selected candidate defective patches by our method. Most of the defect areas can be covered by these patches. Hence it is an efficient way that the refinement of pixel-level defect detection is just performed on these candidate patches.

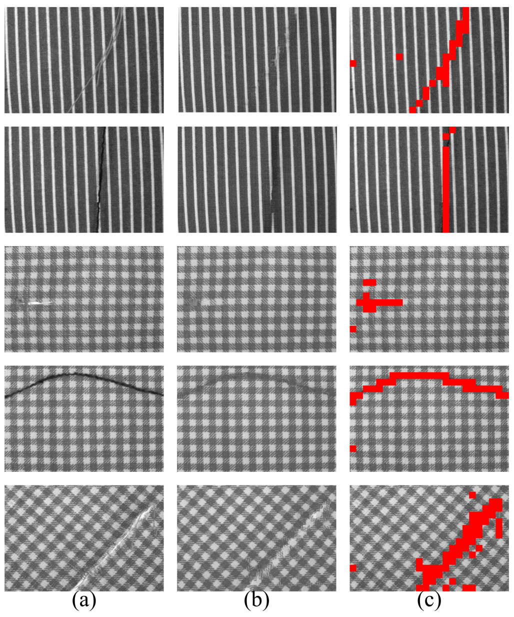

Figure 2. Results of candidate defective patch selection. (a) Raw defective images. (b) Reconstructed images by the learned autoencoder. (c) Candidate defective patches labeled in red.

Figure 3 presents the detection results of comparative experiments on various types of defects. Figure 3(a) shows the original defective fabric images. The results in Figure 3(b) are obtained by the method that the pixels in the candidate defective patches with large reconstruction errors are directly selected as defects according to the learned autoencoder (AE method). We can see that some pixels in texture areas are taken as defects while some pixels of real defects cannot be detected, which are both caused by inaccurate reconstruction. From the results of our method in Figure 3(d), it is noted that less false alarms appear in the texture area and also more pixels in the defect area are detected.

To validate the effect of cross-patch similarity, we remove the item of Q_2 by setting λ to 0 in Eq. (7). This means that we only optimize the reconstruction item Q_1 and the results are illustrated in Figure 3(c). Comparing them with Figure 3(d), our method with cross-patch similarity can find more defects (see the regions in yellow ellipses) even at the regions with low contrast (e.g., the second and the fifth rows in Figure 3). In addition, we can find less false alarms in the texture area in Figure 3(c) compared with Figure 3(b). This verifies the effect of the reconstruction item Q_1 that the texture area in a defective patch can be better

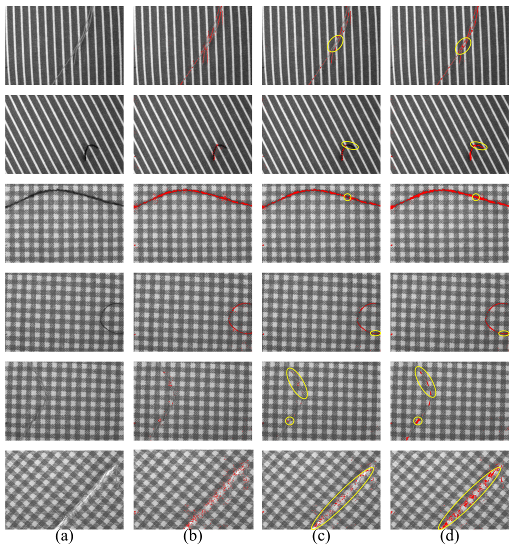

Figure 3. Examples of detection on TILDA dataset. (a) Raw defective images. (b) Results obtained by the reconstruction residual of the autoencoder. (c) Results of our method without cross-patch similarity. (d) Results of our method with cross-patch similarity.

reconstructed to some extent.

3.4 Results on the patterned fabric dataset

For quantitative evaluation, the following three measurement metrics are used: accuracy, recall and precision, and they are defined as

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{8}$$

$$Recall = \frac{TP}{TP + FN} \tag{9}$$

$$Precision = \frac{TP}{TP + FP} \tag{10}$$

where true positive TP and false negative FN refer to the numbers of defective pixels which are detected as defective and non-defective, true negative TN and false positive FP refer to the numbers of non-defective pixels which are detected as non-defective and defective.

We compare our proposal with the AE method and the method based on image decomposition (ID method) in [10]. Table 1 lists the results under the three metrics in which the best ones are shown in bold. First of all, compared with the AE method, our method obviously has better performance on all the five types of defects. This again verifies the effect of the latent variable modification. Our method shows higher precision but lower recall than the ID method on most defect types. This is because our method mainly focuses on correct detection for defective pixels and false alarm reduction for non-defects, while the ID method tends to output a large and complete area for each defect. We argue that the improvement for detection precision is more difficult and more important in practical applications. Some examples of detection results for each type of defects are given in Figure 4. From the results, we can see that most of the defects are well detected by our method and the generated masks are very similar to the ground truth. Moreover, less false detection appears in our detection results, which reveals that our detection method is very robust to many kinds of defects and thus has high detection precision. On the contrary, the ID method often generates too many false alarms.

Table 1. Quantitative results for each defect type on the star-patterned dataset.

Defect	Method	*Accuracy*	*Recall*	*Precision*
Broken End	ID	97.58	75.62	17.64
	AE	98.92	35.43	25.24
	Ours	**99.25**	**76.81**	**40.45**
Hole	ID	97.16	**72.34**	33.29
	AE	99.16	35.69	28.28
	Ours	**99.25**	70.90	**40.28**
Netting Multiple	ID	**98.75**	**82.19**	56.60
	AE	98.18	33.53	45.26
	Ours	98.71	60.84	**62.48**
Thick Bar	ID	**99.30**	**94.45**	**79.93**
	AE	96.45	15.62	35.64
	Ours	97.02	45.48	55.97
Thin Bar	ID	94.98	**81.59**	13.86
	AE	98.40	31.86	25.87
	Ours	**99.02**	74.26	**50.23**

4 Conclusions and Future Work

In this paper, with the basic idea to maintain the texture area in the reconstructed defective patch, a novel autoencoder with cross-patch similarity is proposed for fabric defect detection. To make use of the repetitive texture patterns in the test image, the similar non-defective patches for each candidate defective patch are found, and their corresponding latent variables are weighted combined to be a reference for modifying the original latent variable of the candidate patch. Based on an iterative update process, the output for each defective patch decoded by the modified latent variable will be non-defective, thus the defect area in the patch can be effectively detected from the reconstruction residual.

For further work, we will try to improve the method by introducing global information of the defect. In our proposal, whether a pixel belongs to a defect area totally depends on its reconstruction result, and the relationship between defective pixels is ignored. Next we will focus on the structure description for the whole defect, and try to get better detection results based on combination of both the pixel-wise reconstruction and the overall characteristic of the defect.

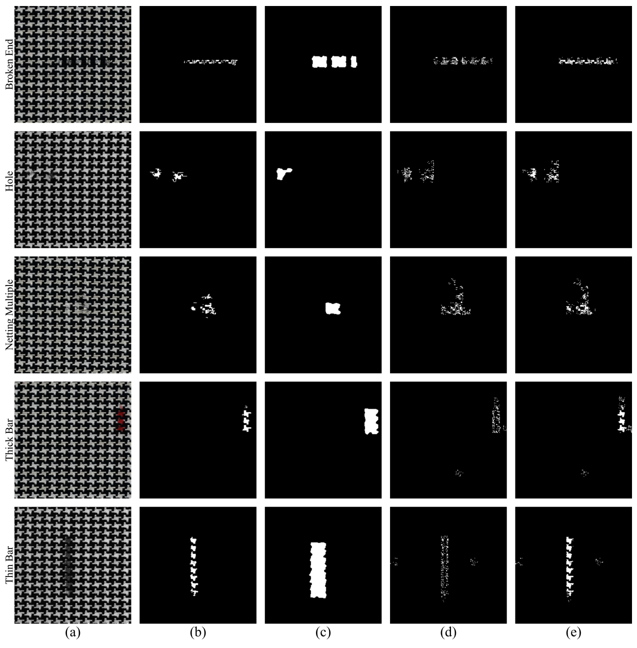

Figure 4. Examples of detection results on star-patterned dataset. (a) Raw defective images. (b) Ground truth. (c) Results of ID method. (d) Results of AE method. (e) Results of our method.

References

[1] Henry Y. T. Ngan, Grantham K. H. Pang, and Nelson H. C. Yung, "Automated fabric defect detection – A review," *Image and Vision Computing*, vol. 29, no. 7, pp. 442–458, 2011.

[2] Kazım Hanbay, Muhammed Fatih Talu, and Ömer Faruk Özgüven, "Fabric defect detection systems and methods – A systematic literature review," *Optik*, vol. 127, no. 24, pp. 11960–11973, 2016.

[3] M. Alper Selver, Vural Avşar, and Hakan Özdemir, "Textural fabric defect detection using statistical texture transformations and gradient search," *The Journal of The Textile Institute*, vol. 105, no. 9, pp. 998–1007, 2014.

[4] Shengqi Guan, Ting Zhao, and Hongyu Shi, "Image segmentation of fabric defect based on object rarity feature," *Journal of Textile Research*, 2015.

[5] Kaustubh Sakhare, Akshada Kulkarni, Mansi Kumbhakarn, and Nachiket Kare, "Spectral and spatial domain approach for fabric defect detection and classification," in *Proceedings of the International Conference on Industrial Instrumentation and Control*, 2015, pp. 640–644.

[6] P. Vidhya Lakshmi and A. Senthil Kumar, "An feed forward model for fabric defect detection using region based texture similarity matrix and gabor filters," *Asian Journal of Research in Social Sciences and Humanities*, vol. 6, no. 6special, pp. 302–312, 2016.

[7] Henry Y. T. Ngan, Grantham K. H. Pang, S. P. Yung, and Michael K. Ng, "Wavelet based methods on patterned fabric defect detection," *Pattern Recognition*, vol. 38, no. 4, pp. 559–576, 2005.

[8] Renzhong Li, Huanhuan Zhang, Junfeng Jing, and Pengfei Li, "Fabric defect detection based on Gaussian mixture models of EM algorithm," *Computer Engineering and Applications*, 2014.

[9] Pengfei Li, Huanhuan Zhang, Junfeng Jing, Renzhong Li, and Juan Zhao, "Fabric defect detection based on multi-scale wavelet transform and Gaussian mixture model method," *The Journal of The Textile Institute*, vol. 106, no. 6, pp. 587–592, 2015.

[10] Michael K. Ng, Henry Y. T. Ngan, Xiaoming Yuan, and Wenxing Zhang, "Patterned fabric inspection and visualization by the method of image decomposition," *IEEE Transactions on Automation Science and Engineering*, vol. 11, no. 3, pp. 943–947, 2014.

[11] Le Tong, Wai Keung Wong, and Chun Kit Kwong, "Fabric defect detection for apparel industry: A nonlocal sparse representation approach," *IEEE Access*, vol. 5, pp. 5947–5964, 2017.

[12] Chunlei Li, Guangshuai Gao, Zhoufeng Liu, Di Huang, Sheng Liu, and Miao Yu, "Defect detection for patterned fabric images based on GHOG and low-rank decomposition," *arXiv preprint arXiv:1702.05555*, 2017.

[13] Junfeng Jing, Amei Dong, Pengfei Li, and Kaibing Zhang, "Yarn-dyed fabric defect classification based on convolutional neural network," *Optical Engineering*, vol. 56, no. 9, 2017.

[14] Yundong Li, Weigang Zhao, and Jiahao Pan, "Deformable patterned fabric defect detection with fisher criterion-based deep learning," *IEEE Transactions on Automation Science and Engineering*, vol. 14, no. 2, pp. 1256–1264, 2017.

[15] Domen Racki, Dejan Tomazevic, and Danijel Skocaj, "A compact convolutional neural network for textured surface anomaly detection," in *Proceedings of the IEEE Winter Conference on Applications of Computer Vision*, 2018, pp. 1331–1339.

[16] Shuang Mei, Yudan Wang, and Guojun Wen, "Automatic fabric defect detection with a multi-scale convolutional denoising autoencoder network model," *Sensors*, vol. 18, no. 4, pp. 1064, 2018.

[17] The working group of DFG, "TILDA textile texture database," Website, `https://lmb.informatik.uni-freiburg.de/resources/datasets/tilda.en.html`.

[18] Henry Y. T. Ngan, "Patterned fabric database," Website, `https://ytngan.wordpress.com/codes/`.

[19] Tianqi Chen, Mu Li, Yutian Li, Min Lin, Naiyan Wang, Minjie Wang, Tianjun Xiao, Bing Xu, Chiyuan Zhang, and Zheng Zhang, "MXNet: A flexible and efficient machine learning library for heterogeneous distributed systems," *arXiv preprint arXiv:1512.01274*, 2015.