Image Preprocessing and Visual Implementation for Fabric Defects Based on OpenCV

Hongrui Xu, Yifan Qu*, Ting Li, Xue Yu

School of Software, Harbin Institute of Information Technology, Harbin, 150000, P. R. China

*1830561918@qq.com

Abstract. Image preprocessing is a critical prerequisite for computer-vision tasks such as fabric-defect detection and target classification, directly affecting the accuracy and efficiency of subsequent models. To solve image noise, uneven illumination and blurred contours commonly observed in fabric-defect inspection scenes, this paper designs a complete preprocessing pipeline with seven core steps and implements real-time visualization based on Python and OpenCV. The pipeline includes grayscale conversion, Gaussian filtering, CLAHE contrast enhancement, adaptive threshold binarization, morphological closing, Canny edge detection and edge-morphology fusion. Experimental results on real fabric defect images show that the proposed pipeline effectively removes noise, corrects illumination deviation and strengthens defect regions contours. The signal-to-noise ratio (SNR) is improved by 32.3% and the target contrast is improved by 52.4%, providing high-quality data support for subsequent fabric-defect detection models such as YOLO.

Keywords: Fabric-defect image preprocessing; OpenCV; CLAHE

1 Introduction

With the rapid development of computer-vision technology[1], applications such as object detection and image classification have been widely deployed in intelligent surveillance, smart cities and industrial inspection. Automatic detection of fabric defects from images is a key technology in intelligent textile manufacturing and quality control. Replacing manual visual inspection with image-based defect detection greatly improves inspection consistency, reduces labour cost and supports the construction of fully automated production lines for the textile industry. As an important branch of intelligent visual surveillance, automatic fabric-defect detection directly determines the practical value of the whole system.

However, fabric images captured on production lines are easily affected by the strong periodic texture of the weave, uneven illumination from line-scan light sources, fibre fluff and slight vibration of the fabric. The captured images therefore commonly suffer from random noise, uneven illumination, low contrast between defects and the surrounding texture, and blurred defect contours. Directly feeding such raw images into

R. Zur et al. (eds.), *Proceedings of the 2026 5th International Conference on Art Design and Digital Technology (ADDT 2026)*, Advances in Computer Science Research 131,
https://doi.org/10.2991/978-94-6239-737-8_66

a defect-classification model causes missed detection of slim broken-yarn defects and false alarms on regular texture patterns.

The core purpose of image preprocessing is to eliminate redundant information and interference in raw images, enhance target features and convert raw images into a standardized form more suitable for subsequent processing[2]. Existing preprocessing algorithms mostly focus on solving a single problem and lack a systematic design for fabric-defect inspection scenes. Moreover, most implementations do not provide visualization functions, making it difficult to intuitively verify algorithm effects, which is unfavourable for debugging, optimization and beginner learning[3]. Aiming at fabric-defect detection, this paper designs a complete visual preprocessing pipeline implemented in Python and OpenCV. It displays intermediate results of each step in real time, which not only facilitates intuitive verification of algorithm effects but also offers clear practical references for beginners. The effectiveness and practicability of the pipeline are verified by experiments on real fabric defect images[4].

2 Principles of Core Image Preprocessing Algorithms

The preprocessing pipeline designed in this paper follows the core logic of "interference removal → feature enhancement" and consists of seven progressive steps. Corresponding algorithms are designed for different image problems to form a closed-loop processing system. The specific principles are as follows.

2.1 Image Reading and Grayscale Conversion

Raw fabric defect images are usually three-channel BGR colour images that contain abundant colour information. However, contour-based detection of defect regions does not heavily rely on colour features, while the additional channels increase computational complexity and may introduce colour interference. Therefore, colour images are first converted into single-channel grayscale images, which retain target contour and grayscale features while reducing the amount of computation and improving processing efficiency[5].

Grayscale conversion uses the cvtColor function in OpenCV, which computes a weighted sum of BGR channel values according to human visual sensitivity to different colours. The formula is:

$$Gray = 0.299 * R + 0.587 * G + 0.114 * B \tag{1}$$

where R, G, B are the red, green and blue channel pixel values (0-255) and Gray is the converted grayscale value (0-255). This weight allocation conforms to human visual characteristics and maximizes the retention of original image details.

2.2 Gaussian Filtering for Denoising

Raw fabric defect images are prone to various noises such as Gaussian noise and salt-and-pepper noise during acquisition due to sensor characteristics and environmental

interference. Such noise contaminates image data and severely interferes with the accurate extraction of defect regions contours in subsequent processing, causing deviations in binarization and edge detection. Gaussian filtering is therefore adopted for denoising. The core of the algorithm is to convolve the entire image with a 2D Gaussian kernel; the final grayscale value of each pixel is obtained by Gaussian-weighted averaging of grayscale values of neighbouring pixels. This effectively suppresses random noise while preserving target edges to the greatest extent.

$$G(x,y) = \frac{1}{2\pi\sigma^2} exp\left(-\frac{x^2+y^2}{2\sigma^2}\right) \tag{2}$$

where σ denotes the Gaussian standard deviation, which determines the smoothing strength. A larger σ yields stronger smoothing but weaker edge preservation; a smaller σ reduces smoothing effectiveness and leaves residual noise. Through extensive experimental parameter tuning, a 5×5 kernel with σ=1.0 is selected to balance denoising performance and edge preservation capability, effectively removing mild noise without damaging details.

2.3 CLAHE Contrast Enhancement

Fabric defect images often suffer from uneven illumination, weak defect-background contrast, and blurred details. To address this, CLAHE is used instead of global histogram equalization, which may cause over-exposure, darkness, or detail loss.

CLAHE divides the image into 8 × 8 sub-blocks and performs local histogram equalization. With clipLimit = 2.0, excessive grayscale values are clipped and redistributed to prevent brightness distortion. The enhanced blocks are then merged to improve overall contrast and highlight defect details.

2.4 Adaptive Threshold Binarization

Binarization converts grayscale images into binary images to separate defect foreground from the background for contour extraction. Since uneven illumination can make global methods such as Otsu ineffective, adaptive thresholding is adopted. It calculates local thresholds within a 15 × 15 neighbourhood according to grayscale distribution, improving target-background separation. Inverse binarization makes defect regions white and the background black, while C = 5 helps reduce noise interference.

2.5 Morphological Processing

Binary images may contain fragmented defect regions, residual noise, and irregular edges, leading to segmentation errors. To address this, a closing operation with a 3 × 3 rectangular structuring element is applied. Dilation fills holes and connects broken defect parts, while erosion removes edge burrs and restores clearer contours. This method is suitable for fabric defects because broken-yarn defects are often long and thin, hole defects are roughly rectangular, and small noise blocks caused by surface fluff can be effectively reduced.

2.6 Canny Edge Detection

Edges are an important type of target feature, and the edge contours of defect regions directly determine target shape and recognition accuracy. The Canny edge-detection algorithm features accurate edge localization and strong anti-noise capability, and precisely extracts target edge contours through four steps:

(1) Gaussian filtering for denoising, which cooperates with the previous Gaussian filter to further smooth the image and reduce noise interference in edge detection.

(2) Gradient magnitude and direction calculation using the Sobel operator:

$$M(x,y) = \sqrt{G_x^2 + G_y^2} \tag{3}$$

$$\theta(x,y) = \arctan\frac{G_y}{G_x} \tag{4}$$

(3) Non-maximum suppression: traverse the gradient image and retain only local maxima along gradient directions to refine edges.

(4) Double-threshold screening: a low threshold (30) and a high threshold (100) are set; pixels above the high threshold are strong edges, pixels below the low threshold are non-edges, and pixels in between are kept as edges only if they are connected to strong edges.

2.7 Fusion of Edge and Morphological Results

Morphologically processed images retain complete target contours and internal details but have unclear edges. Canny-edge images accurately extract edges but lack internal details. To further strengthen defect regions contours and improve subsequent recognition accuracy, a bitwise OR operation is performed on the two images to sharpen contours while preserving internal details, forming a complete "contour + detail" feature for stronger support of subsequent fabric-defect detection.

3 Experimental Results and Analysis

3.1 Experimental Environment and Dataset

The experiments were conducted on a Windows 10 platform with an Intel Core i5-10400F CPU, NVIDIA GTX 1650 GPU, Python 3.8, OpenCV 4.5.5, and PyCharm. The dataset included 100 fabric-defect images collected on site and from public datasets, covering different illumination conditions, shooting angles, and backgrounds.

3.2 Experimental Metrics and Methods

Two core metrics are selected to quantitatively evaluate the preprocessing pipeline:

Signal-to-Noise Ratio (SNR): measures noise removal performance. A higher SNR indicates less noise and better image quality.

$$\mathrm{SNR} = 20 \cdot \log_{10}\left(\frac{\mu_{\mathrm{target}}}{\sigma_{\mathrm{noise}}}\right) \tag{5}$$

where μ_{target} is the mean pixel value of the target region and σ_{noise} is the standard deviation of the noise region.

Contrast: measures the difference between target and background. Higher contrast means a clearer target.

$$\mathrm{Contrast} = \frac{I_{\max} - I_{\min}}{I_{\max} + I_{\min}} \tag{6}$$

3.3 Experimental Results

All 100 test images were successfully preprocessed with good visualization at each stage. Quantitative results are shown in Table 1.

Table 1. Comparison before and after preprocessing.

Metric	Before	After	Improvement
SNR (dB)	23.5	31.1	32.3%
Contrast	0.24	0.51	52.4%

The results show that SNR increases from 23.5 dB to 31.1 dB, with a 32.3% improvement, indicating that Gaussian filtering and morphological processing effectively suppress noise and clean the background. Contrast rises from 0.24 to 0.51, showing that CLAHE and adaptive binarization enhance defect-background differences and make target features clearer. These improvements confirm the pipeline's effectiveness in noise reduction and feature enhancement for fabric-defect detection.

3.4 Experimental Result Visualization

Fig. 1 presents the visualized representation of the experimental results after overall processing, with (a) the original image, (b) the grayscale image, (c) the Gaussian-filtered image, (d) the contrast-enhanced image, (e) the binarized image, (f) the morphological processing result, (g) the edge-detection result, and (h) the combined image after edge and morphological fusion.

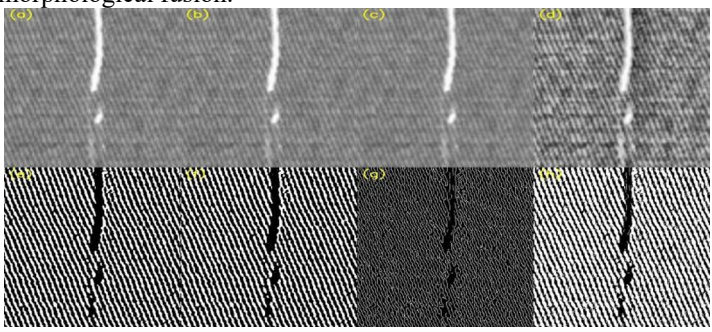

Fig. 1. Visualized representation of overall experimental results after processing.

The proposed image-preprocessing pipeline is targeted, visualizable, practical, efficient, and stable. It addresses common fabric-defect inspection problems, including noise, uneven illumination, and blurred contours. Real-time visualization supports intuitive debugging, while the well-commented OpenCV-based code is easy to run and compatible with mainstream detection models such as YOLO. With processing time under 0.5 s per image, it meets real-time batch-processing requirements and achieves stable SNR and contrast improvements across different fabric images.

However, performance may decline on patterned fabrics with strong colour blocks, where periodic textures interfere with binarization. Very thin broken-yarn defects on dark fabrics may also retain slight residual noise. Future work will focus on improving robustness under these challenging conditions.

4 Conclusion and Future Work

This study proposes a visualizable OpenCV-based preprocessing workflow for fabric-defect detection, including grayscale conversion, Gaussian denoising, CLAHE enhancement, adaptive thresholding, morphological restoration, Canny edge detection, and edge-morphology fusion. Stage-by-stage visualization supports algorithm debugging and learning.

Experiments show that the workflow effectively reduces noise, corrects illumination bias, and enhances defect contours, improving SNR by 32.3% and contrast by 52.4%. With processing time under 0.5 s per image, it meets real-time inspection requirements.

Future work will focus on Gabor-based texture suppression, FFT-based anomaly enhancement, and integration with lightweight models such as YOLOv8-n or PaDiM for intelligent textile production-line inspection.

References

1. Li, S., Gao, L., Wang, J., et al.: Information-theoretic graph fusion with vision-language-action model for policy reasoning and dual robotic control. Information Fusion, 104193 (2026). https://doi.org/10.1016/j.inffus.2026.104193.
2. Li, Changfeng, and Wenqin Tong. "A Visual Acuity Assessment System Based on Static Gesture Recognition and Naive Bayes Classifier." International Journal of Information Technologies and Systems Approach (IJITSA) 17.1 (2024): 1-23. DOI: 10.4018/IJITSA.345926
3. Zhou, S., Chen, R., An, Z. et al. Application of large language models to quantum state simulation. Sci. China Phys. Mech. Astron. 68, 240313 (2025). https://doi.org/10.1007/s11433-024-2598-y
4. K. Yang et al., "Robust Multi-Agent Collaborative Perception via Spatio-Temporal Awareness," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 35, no. 6, pp. 5644-5658, June 2025, doi: 10.1109/TCSVT.2025.3528980.
5. Ma, Z., Li, Y., Huang, M. et al. Automated real-time detection of surface defects in manufacturing processes of aluminum alloy strip using a lightweight network architecture. J Intell Manuf 34, 2431–2447 (2023). https://doi.org/10.1007/s10845-022-01930-3