Multi-Scale LBP and GMM-Based Texture Recognition and Reproduction for Building Decoration Materials

Na Li[1,2]
[1]Henan Technical College of Construction, Zhengzhou 450064, China
[2]Kyrgyz National University Named After Jusup Balasagyn, Bishkek 720033, Kyrgyz RepublicEmail: lina__edu@163.com

Keywords: texture recognition, multi-scale local binary mode, gaussian mixture model, texture reproduction, decorative material

Received: May 9, 2025

With the increasing demand for material texture recognition and reproduction in the architectural decoration industry, traditional methods relying on manual annotation are inefficient, and existing technologies are difficult to meet the requirements of complex texture processing. To accurately identify the texture and high-quality reproduction of building decoration materials, a texture recognition method based on multi-scale Local Binary Pattern algorithm combined with Gaussian mixture model clustering is proposed, and a complete model including image preprocessing, feature extraction, and texture reproduction modules is constructed. The experimental results on the Dresden Texture Dataset dataset show that the proposed method performs well in texture feature extraction, with a maximum accuracy of 96.84% for tile texture recognition and an average intersection to union ratio (MIoU) of 0.97. The practical application results show that the proposed method has a fast texture recognition speed and less memory consumption. The average time and size of the generated CSV files are 2.93 s and 8.75 KB, respectively, while the average time and size of the CSV files generated by the 3D Gaussian speckle model are 4.42 s and 21.41 KB, respectively. Meanwhile, the comprehensive score of texture reproduction performance is 8.93 points, followed by 8.27 points for 3D Gaussian speckle. From this, the method proposed by the research can quickly identify and reproduce the complex textures of building decoration materials with high fidelity. This study provides a new technological solution for digital texture recognition and production control of building decoration materials, promoting the intelligent and high-quality development of the building decoration industry.

Povzetek: Za prepoznavanje tekstur gradbenih dekorativnih materialov je razvita večmerilna metoda (MLBP) v kombinaciji z Gaussovim mešanim modelom (GMM), ki omogoča zajemanje večnivojskih značilk in njihovo pretvorbo v strojno berljive datoteke (CSV) za proizvodnjo. Model združuje predobdelavo slik, večmerilno ekstrakcijo in samodejno klastriranje, kar omogoča hitro, zanesljivo in realistično reprodukcijo kompleksnih tekstur gradbenih materialov z majhno porabo virov.

1 Introduction

In recent times, the construction and decoration industry has ushered in opportunities for rapid advancement driven by both technological innovation and market demand [1]. With people's pursuit of high-quality architectural decoration, the recognition and reproduction technology of material texture has become a key link in the development of the industry [2]. Traditional texture recognition methods rely on manual annotation, which is difficult to meet the efficiency and accuracy requirements of the modern architectural decoration industry. Meanwhile, the texture reproduction technology for building decoration materials has undergone a continuous process of advancement. Transitioning from early methods such as screen printing and rubber-roller printing to modern techniques like inkjet printing and dry granule technology, there has been a remarkable enhancement in both the realism and diversity of texture reproduction [3]. However, in the face of complex and ever-changing texture requirements, existing technologies still have limitations [4]. Many scholars have conducted extensive research to efficiently identify texture patterns. Liu et al. proposed a new approach grounded on deep neural network transfer learning to address the limitations of traditional texture analysis methods in complex texture recognition. The experiment outcomes indicated that the performance of this method was superior to Gray-Level Co-Occurrence Matrix (GLCM) and Local Binary Pattern (LBP) [5]. Kamijyo et al. proposed a crystal texture calculation method based on deep neural networks to reduce the computational workload in numerical crystallographic texture optimization. The experiment outcomes indicated that this calculation method could determine the optimal volume fraction for selecting texture components [6]. Shukla et al. proposed a material recognition method based on Convolutional Neural Networks (CNN) to address the issue of insufficient

accuracy in traditional methods for material quality identification. The experiment outcomes indicated that the CNN classifier outperformed other deep learning classifiers in recognition accuracy [7]. Bai et al. proposed a high-precision texture recognition real-time artificial sensing system based on a single ion electron sliding sensor to solve the problem of insufficient texture recognition response in modern robots during interaction. The experiment outcomes indicated that the system had an accuracy of 98.9% at random sliding rates and could accurately distinguish fine surface features [8].

In terms of texture generation, Siddiqui et al. proposed a Texturify algorithm based on Generative Adversarial Networks (GAN) to solve the problem of texture prediction caused by the scarcity of 3D texture data. The experiment outcomes indicated that the Texturify algorithm could generate high-quality textures directly on the surface of 3D objects through layered 4-RoSy parameterization and facial convolution operators, with excellent performance [9]. Cao et al. proposed TexFusion, a new method for synthesizing textures for 3D geometry using only large-scale text guided image diffusion models, to address the inefficiency and instability issues of traditional methods in 3D texture synthesis. The experiment outcomes indicated that TexFusion could generate diverse and globally coherent textures [10]. Gao et al. proposed a Generations Explicit Textured 3D (GET3D) model that could directly generate rich geometric details and high fidelity textures to address the shortcomings of traditional 3D generative models in texture support. The experiment outcomes indicated that the GET3D model could generate high-quality 3D texture meshes [11]. The summary of related work is shown in Table 1.

Table 1: Related work table

Texture Recognition Methods	Methodology	Dataset / Evaluation Context	Performance Metrics	Key Findings
Liu et al. [5]	Deep Neural Network Transfer Learning	Not Specified	Accuracy (CA)	Superior to GLCM & LBP
Kamijyo et al. [6]	Deep Neural Network-based Calculation	Numerical crystal texture data	Optimal volume fraction	Determines optimal texture component fractions
Shukla et al. [7]	CNN Classifier	Material-specific dataset	Recognition Accuracy (CA)	Outperformed other DL classifiers
Bai et al. [8]	Ion Electron Sliding Sensor System	Surface feature dataset (random sliding)	Accuracy (98.9%)	High-precision distinction of fine surface features
Siddiqui et al. [9]	Texturify (GAN + 4-RoSy parameterization)	3D texture datasets	Qualitative/Quantitative quality metrics	Direct high-quality 3D surface texture generation
Cao et al. [10]	TexFusion (Text-guided image diffusion)	3D geometry datasets	Diversity, global coherence	Generated diverse & globally coherent textures
Gao et al. [11]	GET3D (Explicit Textured 3D Generation)	3D shape datasets	Fidelity, geometric detail	High-quality 3D textured mesh output
Proposed Method (MLBP)	MLBP + GMM Clustering	Dresden Texture Dataset (100+ images per type)	CA: 96.84% (Tile), MIoU: 0.97	Highest CA/MIoU; Minimal CSV size (8.75KB); Repro. Score: 8.93

However, existing methods have many shortcomings in texture analysis and generation. Traditional texture analysis methods, such as LBP algorithm, have insufficient accuracy in complex texture recognition. Although deep neural network-based methods have superior performance, they rely on large-scale data and computing resources. In terms of 3D texture generation, existing methods either lack geometric details or are limited in texture support. In addition, some methods are sensitive to noise and the sampling method can easily lead to aliasing effects. A GMM-MLBP model, which is based on a multi-scale Local Binary Pattern (MLBP) algorithm, is put forward with the aim of efficiently and precisely capturing texture images and replicating them onto the textures of building decoration materials. This research makes an innovative move by integrating the Gaussian Mixture Model (GMM) for texture recognition, thereby enhancing the accuracy and generalization capability of

texture feature extraction. An image preprocessing module is introduced to reduce the consumption of computing resources. A texture reproduction module is introduced to generate CSV files, and the precise control of the production machine for building decoration materials is achieved.

2 Method and materials

2.1 Improvement of LBP Algorithm and Material Texture Recognition

Texture recognition is a crucial step in the production process of building decoration materials, which helps in the manufacturing of various materials such as imitation natural stone and imitation natural wood [12]. Traditional texture recognition methods rely on manual annotation, which is not only time-consuming and labor-intensive, but also susceptible to subjective factors, resulting in low production efficiency and accuracy. As computer vision technology advances, texture recognition algorithms based on image processing have gradually become an effective means to solve this problem [13]. Among them, the LBP algorithm is widely used for texture feature extraction due to its advantages of simple calculation, insensitivity to lighting and rotation [14]. However, traditional LBP algorithms have some limitations, especially when dealing with complex textures. Due to their fixed 3×3 neighborhood window, it is difficult to capture large-scale texture features, resulting in incomplete or discontinuous texture segmentation results [15]. To overcome this deficiency, an MLBP algorithm is proposed in the study. MLBP increases the receptive field of feature extraction by introducing multiple sub-scale operators, enabling the simultaneous capture of texture features at different scales. The MLBP algorithm structure is in Figure 1.

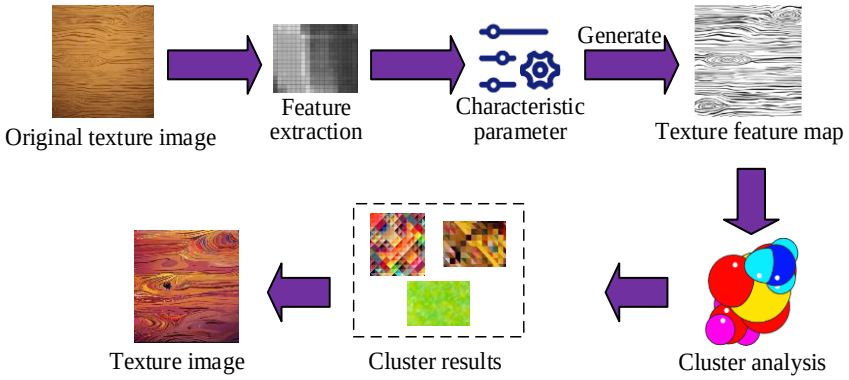


Figure 1: MLBP structure diagram

As shown in Figure 1, MLBP captures texture information by dividing the image into multiple local regions of different scales and applying multiple sub-operators within each region. These sub-operators are distributed in different spatial positions and can simultaneously extract large-scale textures and small-scale details in the image. The calculation formula for each sub-scale operator is in equation (1) [16].

$$MLBP_{S_i}(x_c) = \sum_{p=1}^{P_i} s(I_p - I_c) \times 2^{p-1} \quad (1)$$

In equation (1), $MLBP_{S_i}$ represents the output of the i th sub scale operator. x_c represents the central pixel point. p_i represents the number of sampling points in the i th sub-scale operator. I_p is the grayscale value of the p th sampling point. I_c is the grayscale value of the central pixel. $s()$ is the threshold function, as shown in equation (2) [17].

$$s(x) = \begin{cases} 1, & x \geq 0 \\ 0, & \text{otherwise} \end{cases} \quad (2)$$

MLBP enhances feature extraction capability by combining multiple sub scale operators. Each sub-scale operator is responsible for processing specific texture features, while operators of different scales generate more comprehensive texture descriptions through fusion. The fusion formula is in equation (3).

$$MLBP(x_c) = [MLBP_{S_1}(x_c), MLBP_{S_2}(x_c), \cdots, MLBP_{S_n}(x_c)], \quad i = 1, 2 \ldots n \quad (3)$$

In equation (3), $MLBP(x_c)$ denotes the final multi-scale texture feature vector. n represents the total number of sub scale operators. The sub-scale sizes are 3×3, 5×5, and 7×7, corresponding to neighborhood radii R of 2, 3, and 5, respectively. The number of sampling points P is 8, 10, and 16, respectively. This allows for dynamic adjustment of parameters based on texture complexity to achieve effective feature extraction. This multi-scale design enables MLBP to adapt to images with diverse texture styles, significantly improving the generalization ability of feature extraction. In texture segmentation, the features extracted by MLBP are combined with GMM for clustering. The structure of GMM is in Figure 2.

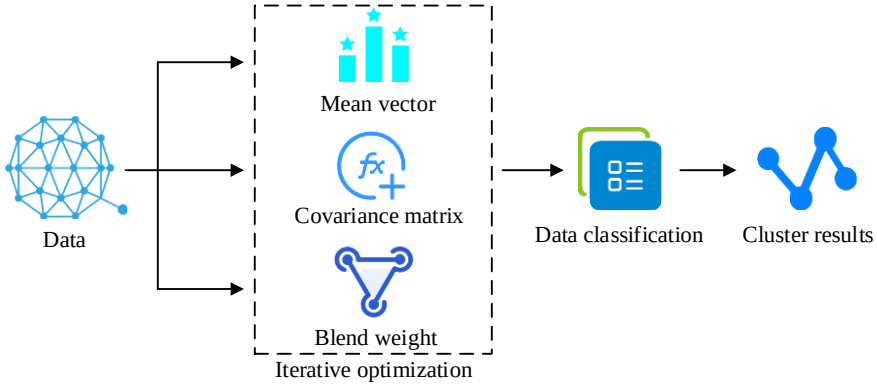


Figure 2: GMM structure diagram

As shown in Figure 2, the GMM structure is based on a linear combination of multiple Gaussian distributions (GDs), with each GD representing a cluster or category. The core of GMM lies in approximating complex data distributions through a mixture of multiple GDs, thereby achieving flexible modeling and clustering of data. In GMM, the probability of each data point (pixel) being assigned to multiple clusters is determined by the parameters of the GD, including the mean vector, covariance matrix, and mixture weights. The formula for calculating the probability density of pixels is in equation (4).

$$p_{GMM}(x) = \sum_{k=1}^{K} \pi_k \mathrm{N}(x|\ \mu_k, \mathrm{S}_k) \tag{4}$$

In equation (4), $p_{GMM}(x)$ represents the probability density of pixel x. π_k is the mixture weight of the k th GD. $\mathrm{N}(x|\mu_k, \mathrm{S}_k)$ is the probability density function of GD, where μ_k represents the mean and S_k represents the covariance matrix. The mixed weights represent the relative importance of each GD, and the sum of all mixed weights is 1. The parameters of each GD are iteratively optimized using the Expectation Maximization (EM) algorithm to maximize the likelihood function of the data. The clustering formula of GMM is in equation (5).

$$\gamma_k(x) = \frac{\pi_k \times \mathrm{N}(x|\mu_k, \mathrm{S}_k)}{\sum_{j=1}^{K} \pi_j \times \mathrm{N}(x|\mu_j, \mathrm{S}_j)} \tag{5}$$

In equation (5), $\gamma_k(x)$ represents the probability that pixel x belongs to the k th cluster. The calculation method for updating mixed weights is in equation (6).

$$\pi_k = \frac{1}{N} \sum_{n=1}^{N} \gamma_x(x) \tag{6}$$

In equation (6), N represents the image size. The calculation method for updating the mean is in equation (7).

$$\mu_k = \frac{\sum_{n=1}^{N} \gamma_k(x) x_n}{\sum_{n=1}^{N} \gamma_k(x)} \tag{7}$$

The calculation method for updating the covariance matrix is in equation (8).

$$\mathrm{S}_k = \frac{\sum_{n=1}^{N} \gamma_k(x)(x_n - \mu_k)(x_n - \mu_k)^T}{\sum_{n=1}^{N} \gamma_k(x)} \tag{8}$$

In equation (8), T represents the transpose matrix. In texture description, MLBP features represent the texture information of an image by calculating feature histograms. The calculation process of feature histogram is in Figure 3.

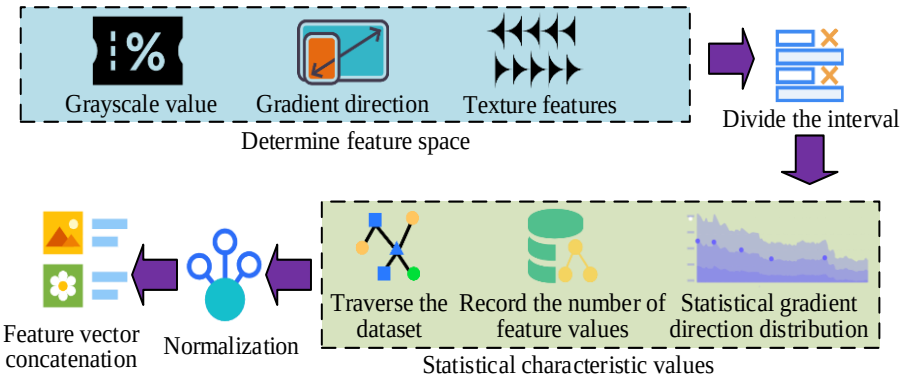


Figure 3: Calculation process of feature histogram

In Figure 3, in the calculation process of the feature histogram, the range of feature values needs to be determined based on the feature type (such as grayscale value, gradient direction, texture feature, etc.) first. Next, the range of eigenvalues is divided into several equally wide intervals, called "boxes". Taking the grayscale histogram as an example, [0, 255] can be divided into 256 boxes, each with a width of 1. Then, the process traverses the dataset (such as every pixel in the image or every point in the point cloud), counts which box each feature value falls into, and records the number of feature values in each box. For example, when calculating a grayscale histogram, it is necessary to count the number of times each grayscale value appears. To eliminate the influence of data size or lighting conditions, it is necessary to normalize the histogram by dividing the number of feature values in each box by the total number of feature values to obtain the normalized histogram. The normalization process is in equation (9).

$$H_{norm}(i) = \frac{H(i)}{\sum_{j=1}^{N} H(j)} \tag{9}$$

In equation (9), $H(i)$ and $H(j)$ represent the i th and j th components of the histogram, respectively. Finally, if the feature histogram is locally computed, all local histograms are concatenated into a high-dimensional feature vector as the final feature description. For each scale, the calculation formula for the feature histogram is in equation (10).

$$H_{S_i}(k_t) = \sum_{x \in X} \delta(MLBP_{S_i}(x) = k_t) \tag{10}$$

In equation (10), $H_{S_i}(k)$ represents the feature histogram. s_i represents different scales. X represents the collection of all pixels in the image. $\delta()$ represents the indicator function, which takes a value of 1 when the condition is met and 0 otherwise. k_t represents the index of feature values. In texture matching or classification tasks, the study uses weighted chi square distance to calculate the similarity between two feature histograms, as shown in equation (11).

$$D(H_1, H_2) = \sum_{i=1}^{N} \frac{(H_{1,i} - H_{2,i})^2}{H_{1,i} + H_{2,i}} \tag{11}$$

In equation (11), $D(H_1, H_2)$ represents the weighted chi square distance between feature histograms H_1 and H_2. Furthermore, the crux of the MLBP algorithm resides in the methodology it employs to generate feature vectors. MLBP concatenates texture features extracted at varying scales, thereby constructing a comprehensive feature vector. This all-encompassing vector is more adept at encapsulating the texture information inherent in the image, ultimately yielding texture image data of the material.

2.2 Identification and reproduction model of texture in building decoration materials

The study effectively breaks through the limitations of traditional LBP algorithm in extracting complex texture features by introducing multi-scale sub operators, enabling it to capture both large-scale textures and small-scale details simultaneously. After feature extraction, GMM is used for clustering, and EM algorithm is used to optimize parameters, thereby achieving flexible modeling and accurate classification of textures. Meanwhile, by calculating feature histograms and using weighted chi square distances, the performance of texture matching and classification is further improved, providing a solid technical foundation for efficient recognition and reproduction of textures in building decoration materials. This research aims to build a lightweight, interpretable texture recognition and reproduction system for architectural materials with minimal computational overhead. To reproduce the texture of building decoration materials, a GMM-MLBP model for recognizing and reproducing the texture of building decoration materials is constructed based on the above technology. To reduce computational complexity and improve the functionality of the GMM-MLBP model, an image preprocessing module and a texture reproduction module are introduced into the GMM-MLBP model. The structure of the GMM-MLBP model is in Figure 4.

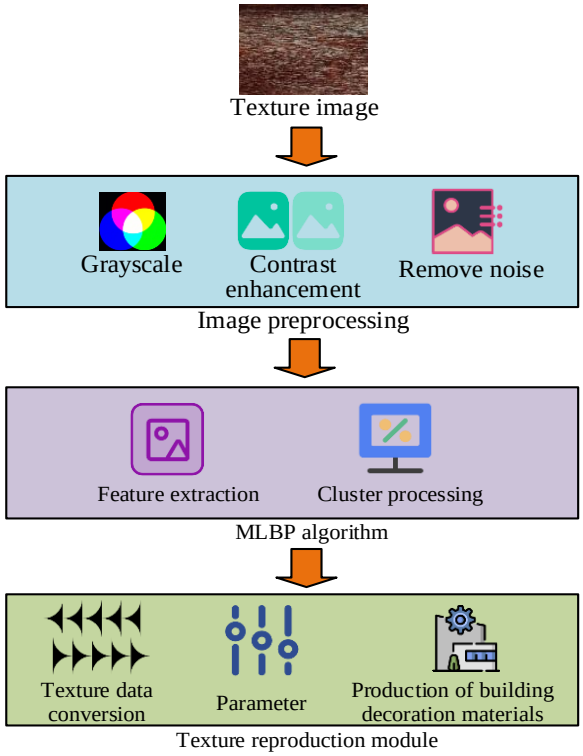


Figure 4: Structure of GMM-MLBP model

As shown in Figure 4, the GMM-MLBP model mainly consists of three major structures: image preprocessing module, MLBP algorithm, and texture reproduction module. The process of generating textures using the GMM-MLBP model is shown in Figure 5.

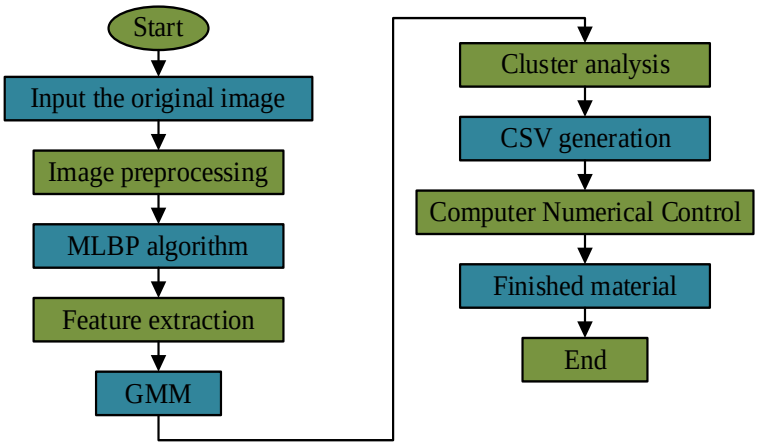


Figure 5: The process of generating textures using the GMM-MLBP model

The input texture image first goes through an image preprocessing module, which is responsible for grayscale, contrast enhancement, and denoising of the image, reducing data dimensionality and preserving texture information. The structure of the image preprocessing module is in Figure 6.

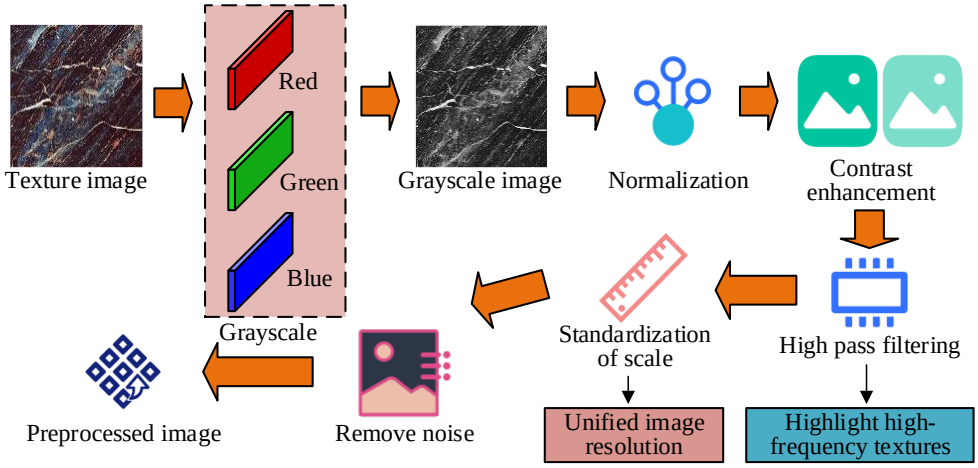


Figure 6: Structure of image preprocessing module

As shown in Figure 6, the image preprocessing module includes steps such as grayscale conversion, normalization, contrast enhancement, and denoising. Firstly, grayscale conversion converts color images into grayscale images, reducing data dimensions through weighted averaging while preserving the basic structural information of the image, in order to simplify the processing and reduce computational complexity. The formula for calculating grayscale values is in equation (12).

$$I_{gray} = 0.299 \times R_h + 0.587 \times G_h + 0.114 \times B_h \quad (12)$$

In equation (12), I_{gray} represents the grayscale value of the image. R_h, G_h, and B_h represent the red, green, and blue channel values of the image respectively. Next, in order to eliminate the influence of lighting conditions on texture, it is necessary to normalize the image. The normalization formula is in equation (13).

$$I_{norm} = \frac{I_{gray} - \min(I_{gray})}{\max(I_{gray}) - \min(I_{gray})} \quad (13)$$

In equation (13), I_{norm} represents the normalized image. Normalization operation shrinks pixel values to a uniform range of [0, 1]. Contrast enhancement can highlight texture features, and histogram equalization is used in research. Histogram equalization enhances the contrast of an image by adjusting its grayscale distribution to approach a uniform distribution. To highlight texture details, it is necessary to remove low-frequency background information from the image. The high pass filtering method used in the study is Laplace filtering, and the calculation formula for high pass filtering is in equation (14).

$$I_{filtered} = I_{norm} * H \quad (14)$$

In equation (14), $I_{filtered}$ represents the image after high pass filtering. * represents convolution operation. If the scale of the texture image is inconsistent, scale standardization is required. The research scales images to a uniform resolution through interpolation methods. The denoising step uses median filtering and Gaussian filtering to remove noise from the image and improve image

quality. The median filtering denoising calculation is in equation (14).

$$I_{\text{denoised}} = median(I_{\text{filtered}}) \quad (15)$$

In equation (15), I_{denoised} represents the denoised image. $median$ represents median filtering. The Gaussian filtering denoising calculation is in equation (16).

$$I_{\text{denoised}} = I_{\text{filtered}} * G_{\sigma} \quad (16)$$

In equation (16), G_{σ} represents Gaussian kernel and σ represents standard deviation. Finally, obtain the preprocessed image. The preprocessed image enters the MLBP algorithm module for extracting texture features. The texture reproduction module converts the extracted texture features into control files suitable for production. This module adjusts the texture channel map based on the parameters of the fabric machine, such as the number and distribution of fabric ports, and generates a CSV format control file. These files control the material production machine to set corresponding textures during the production process through a matrix composed of 0 and 1, and finally generate building decoration materials with texture features. The texture feature processing flow of the texture reproduction module is in Figure 7.

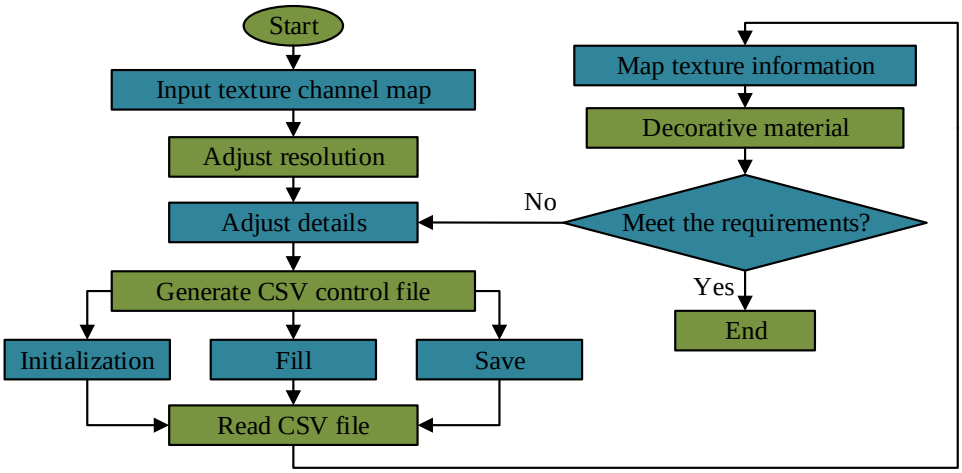


Figure 7: Texture feature processing flowchart of texture reproduction module

As shown in Figure 7, the texture reproduction module remaps the extracted texture features to the image, generating a texture distribution map or CSV control file that meets production requirements. This process is based on multiple texture channel maps output by the texture segmentation module, each channel map representing a specific texture type with binary information (1 represents belonging, 0 represents not belonging). These channel diagrams are converted into CSV files using a specific algorithm, where 1 and 0 represent the opening and closing of the pigment ports, respectively, to control the production machine to reproduce the texture. When converting, the system adjusts the resolution of the texture channel map based on machine parameters such as the number and distribution of pigment ports to ensure the integrity of texture features. Subsequently, building decoration materials with the corresponding textures are generated based on the CSV file. In the event that the generated results do not meet the specified requirements, the system modifies the relevant parameters and creates a new CSV file. This iterative process continues until the desired outcome is achieved. The Pseudo Code of GMM-MLBP model is in Figure 8.

```
# ===== GMM-MLBP MODEL PSEUDOCODE =====
function GMM_MLBP_MODEL(input_image, machine_params):
    # --- Image Preprocessing ---
    preprocessed_img = PREPROCESS(input_image,
                                  target_size=(1024,1024),
                                  grayscale_weights=[0.299, 0.587, 0.114],
                                  denoise_methods=['median','gaussian'])

    # --- Multi-Scale LBP Feature Extraction ---
    mlbp_features = []
    for scale in SCALES:  # SCALES = [(3×3, R=2, P=8), (5×5, R=3, P=10), (7×7, R=5, P=16)]
        scale_features = COMPUTE_MLBP(preprocessed_img,
                                      radius=scale.R,
                                      points=scale.P)
        hist = BUILD_HISTOGRAM(scale_features, bins=256)
        mlbp_features.append(hist)

    combined_features = CONCATENATE(mlbp_features)

    # --- GMM Clustering ---
    gmm = INIT_GMM(K=5)  # K typically 3-10
    texture_maps = gmm.FIT_PREDICT(combined_features,
                                   max_iter=100,
                                   covariance_type='diag')

    # --- Texture Reproduction ---
    control_matrix = ADJUST_RESOLUTION(texture_maps,
                                       target_res=machine_params.resolution)

    csv_data = GENERATE_CSV(control_matrix,
                            threshold=0.5,
                            format='binary')

    return csv_data  # Average size: 8.75KB

# Key Parameters (from paper):
# - SCALES: Multi-scale operators (3×3,5×5,7×7)
# - RADIUS (R): [2, 3, 5]
# - POINTS (P): [8, 10, 16]
# - GMM CLUSTERS (K): 3-10 (dynamically adjusted)
# - HISTOGRAM BINS: 256
# - MAX EM ITERATIONS: 100
```

Figure 8: Pseudo code of GMM-MLBP model

3 Result

3.1 Performance analysis of MLBP algorithm

To confirm the capability of the MLBP algorithm in texture feature extraction, a high-performance experimental platform was established, and the Multi-Scale Feature Fusion CNN (MSFF) and the Spatial Continuity and Gray Diversity (SCGD) algorithm considering spatial continuity and grayscale diversity were compared as comparative algorithms [18, 19]. The parameters and algorithm parameters of the experimental platform are shown in Table 2.

Table 2: Experimental platform parameters and algorithm parameters table

Experimental platform parameters	
CPU	Intel Core i5 12400F
GPU	NVIDIA RTX 3060 Ti
Storage	1TB SSD
Memory	DDR4 3200MHz 32GB
Operating system	Windows 10 Professional
Programming language	Python 3.7
Algorithm parameters	
Feature extraction method	Based on MLBP
Feature vector dimension	Dynamically adjusted

Sub-scale sizes	3×3, 5×5, 7×7
Neighborhood radius (R)	2, 3, 5 (adjusted according to texture complexity)
Number of sampling points (P)	8, 10, 16 (matched with neighborhood radius)
Feature histogram window (w)	10, 15, 30 pixels (adjusted according to texture type)
Clustering method	GMM
Number of clusters	Dynamically adjusted (typically 3-10)

The dataset used in the study was the publicly available Dresden Texture Dataset, which includes multiple texture types (such as marble, wood, tiles, etc.), with approximately 100 images for each texture type, and image resolutions ranging from 1024×1024 to 4096×4096. The ratio of training set to testing set was 8:2. The performance indicators of the algorithm were Classification Accuracy (CA) and Mean Intersection over Union (MIoU). The range of CA values was from 0 to 100%, representing the proportion of correctly classified samples to the total sample size. The MIoU value range was from 0 to 1, used to evaluate the degree of overlap between predicted results and real labels in segmentation tasks. For different types of texture images, the average test results of CA and MIoU for each algorithm in the Dresden Texture Dataset are shown in Figure 9.

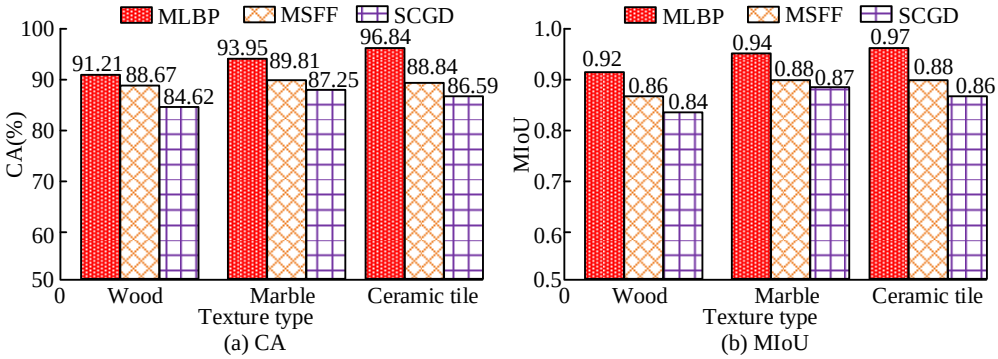


Figure 9: The average values of CA and MIoU for feature extraction of different types of texture images by various algorithms

According to Figure 9 (a), in terms of wood texture, the CA of MLBP algorithm, MSFF algorithm, and SCGD algorithm were 91.21% (± 0.35%), 88.67% (± 0.42%), and 84.62% (± 0.51%), respectively, indicating that the MLBP algorithm had a higher accuracy in texture recognition. The CA of the MLBP algorithm for recognizing marble and tile textures were 93.95% (± 0.28%) and 96.84% (± 0.19%), respectively. This was because tile images were often more regular, resulting in higher recognition accuracy. According to Figure 9 (b), the texture segmentation performance of MLBP algorithm was also superior to MSFF algorithm and SCGD algorithm. Taking ceramic tile texture as an example, the MIoU of MLBP algorithm, MSFF algorithm, and SCGD algorithm were 0.97 (± 0.001), 0.88 (± 0.03), and 0.86 (± 0.02%), respectively. The MLBP algorithm could accurately recognize and segment texture images. The algorithm was tested 100 times on the Dresden Texture Dataset for image texture recognition. During the testing process, the computational resource consumption of each cycle is in Figure 10.

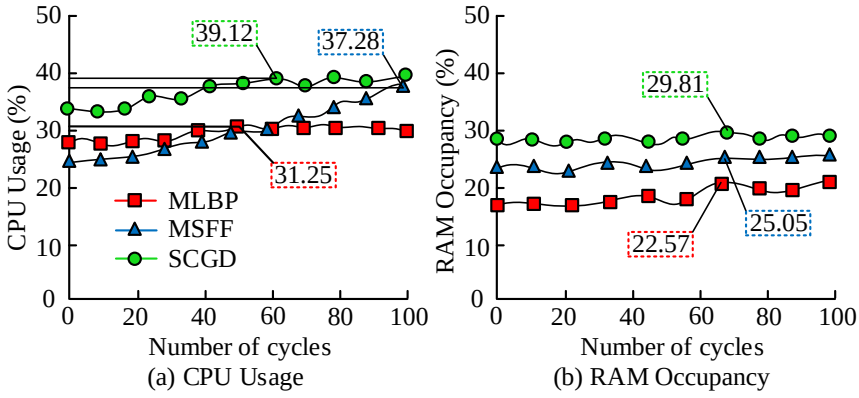


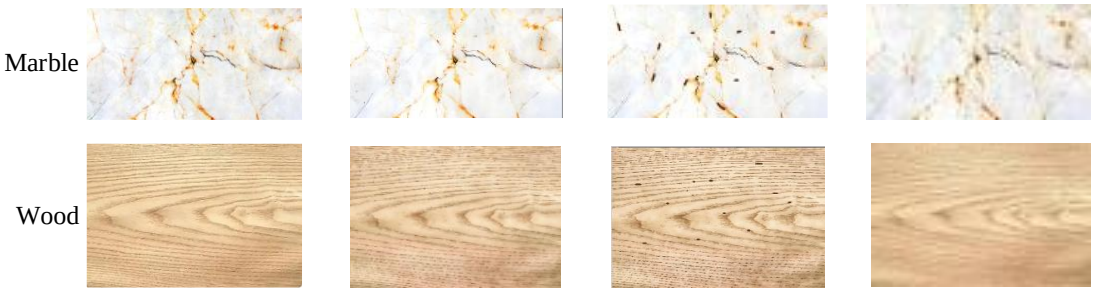
Figure 10: Computing resource consumption during image texture recognition process

According to Figure 10 (a), during the entire cycle testing process, the maximum CPU utilization of MLBP algorithm, MSFF algorithm, and SCGD algorithm were 31.25%, 37.28%, and 39.12%, respectively, indicating that MLBP algorithm had lower computational resource requirements than other algorithms. According to Figure 10 (b), the maximum RAM occupancy rates of MLBP algorithm, MSFF algorithm, and SCGD algorithm were

22.57%, 25.05%, and 29.81%, respectively. Due to the combination of MLBP algorithm and GMM for clustering analysis, its computational resources and RAM usage were lower.

3.2 Performance analysis of GMM-MLBP model

In the previous section, a performance analysis was conducted on the MLBP algorithm, which showed good performance, accurate recognition and segmentation of texture images, and low computational resource consumption. To verify the performance of the GMM-MLBP model, practical applications were conducted in a building decoration material production company, using 3D Gaussian Speckle (3DGS) and Two Branch Convolutional Network (TBCNN) as comparison models [20, 21]. The CNC woodworking carving machine model was STM6090, manufactured by STYLECNC. It supports 2D, 2.5D, and 3D processing and is suitable for various materials such as hardwood, medium density fiberboard, plywood, etc. It can achieve complex texture carving and is suitable for customized texture processing of building decoration materials. The input parameters of the machine were controlled through CSV files generated by the model. The actual performance of GMM-MLBP model, MSFF algorithm, and SCGD algorithm in reproducing is in Figure 11.



(a) Original image (b) GMM-MLBP (c) 3DGS (d) TBCNN

Figure 11: Reproduction of wood and marble texture actual effect

As illustrated in Figure 11, the GMM-MLBP model demonstrated remarkable capability in precisely extracting wood texture features. By enhancing the texture details, it enabled high-fidelity reproduction of authentic wood textures. This made the model highly applicable in the production of building decoration materials. Although the 3DGS model could accurately reproduce wood texture, there were too many impurity points in the generated texture. The wood texture generated by the TBCNN model was of poor quality and blurry. Due to the denoising optimization of the original image by the GMM-MLBP model, it could reproduce better wood texture. The reproduction effect of marble texture was similar to that of wood texture. In practical applications, the time and memory consumption (CSV file size) of each model in generating textures are shown in Figure 12.

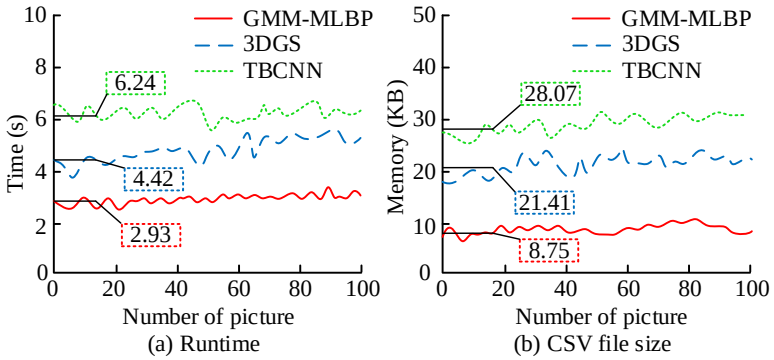


Figure 12: Time and memory consumption of texture generation by various models

As shown in Figure 12 (a), due to the preprocessing module of the GMM-MLBP model, the time variation during texture generation was stable, with an average time of 2.93 s, which was 1.49 s faster than the 3DGS model (4.42 s). This time advantage is crucial in actual production, as it can significantly improve production efficiency, reduce waiting time, and accelerate the entire workflow. In Figure 12 (b), the average sizes of CSV files generated by the GMM-MLBP model, 3DGS model, and TBCNN model were 8.75 KB, 21.41 KB, and 28.07 KB, respectively. Due to the removal of redundant data in texture images by the GMM-MLBP model, the generated CSV files were smaller and could occupy less memory. To obtain a more comprehensive analysis of the texture reproduction performance of the GMM-MLBP model, the study also sought professional personnel from the decoration material production company to score the wood texture, marble texture, and tile texture reproduced by different models. The scoring results were based on 10 professional evaluators in the field of building decoration

materials. The scoring criteria included Detail Realism, Color Reproduction, and Overall Aesthetics, with a maximum score of 10 points for each item. The agreement between raters was evaluated by calculating the mean and standard deviation, and the standard deviation of each indicator was less than 0.6, indicating a high degree of consistency among raters. The final rating results are shown in Table 3.

Table 3: Texture reproduction performance evaluation

Texture type	Evaluation indicator	GMM-MLBP	3DGS	TBCNN
Wood texture	Detail realism	9.20	8.50	7.80
	Color reproduction	9.00	8.30	7.50
	Overall aesthetics	9.10	8.40	7.70
	Composite score	9.10	8.40	7.67
	Ranking	1	2	3
Marble texture	Detail realism	8.90	8.20	7.60
	Color reproduction	8.80	8.10	7.40
	Overall aesthetics	8.70	8.10	7.50
	Composite score	8.80	8.10	7.50
	Ranking	1	2	3
Tile texture	Detail realism	9.00	8.40	7.90
	Color reproduction	8.90	8.30	7.70
	Overall aesthetics	8.80	8.20	7.80
	Composite score	8.90	8.30	7.80
	Ranking	1	2	3
Overall evaluation	Composite score	8.93	8.27	7.66
	Ranking	1	2	3
Average value	SSIM	0.97	0.92	0.86
	PSNR (dB)	32.75	28.40	25.18

According to Table 3, the GMM-MLBP model performed the best in texture reproduction performance. In the comprehensive evaluation of three texture types (wood texture, marble texture, and tile texture), the GMM-MLBP model achieved the highest scores of 9.10, 8.80, and 8.90, respectively, with an overall comprehensive score of 8.93, ranking first. This indicated that the GMM-MLBP model performed well in the three evaluation indicators of detail realism, color reproduction, and overall aesthetics, and could highly restore the details and colors of real textures. At the same time, it also had a more natural and aesthetically pleasing visual effect. In contrast, the second-ranked 3DGS model received comprehensive scores of 8.40, 8.10, and 8.30 for wood texture, marble texture, and tile texture, respectively, with an overall comprehensive score of 8.27. Although 3DGS performed well in terms of details and color expression, it was slightly inferior to the GMM-MLBP model in terms of overall aesthetics, especially in the reproduction of tile textures, where there was a significant gap in color reproduction and aesthetic ratings compared to the GMM-MLBP model. In summary, the GMM-MLBP model demonstrated excellent performance in texture reproduction tasks, being able to more accurately restore the details and colors of real textures, and also had advantages in visual effects. This result indicated that the GMM-MLBP model performed well in handling textures of different complexity and types, providing a reliable solution for texture recognition and reproduction in the building decoration materials industry.

4 Discussion

The texture recognition and reproduction model based on MLBP algorithm proposed in the study demonstrated significant advantages in the application of building decoration materials. Compared to deep learning methods based on CNN and GAN, the MLBP algorithm performed outstandingly in terms of efficiency and resource consumption. Although existing CNN methods (such as Liu and Aldrich [5] and Shukla et al. [7]) achieved high accuracy in specific tasks (such as Bai et al. [8]'s 98.9%), they relied on large-scale training data and GPU computing power, which limited their deployment in production environments. Similarly, GAN driven texture generation techniques (such as Texturify [9] and GET3D [11]) could synthesize high-quality visual output, but had huge computational costs (such as generating 21.41 KB CSV files compared to the 3DGS model). The MLBP algorithm proposed in the study achieved a ceramic tile texture classification accuracy (CA) of 96.84% and an average intersection to union ratio (MIoU) of 0.97 on the standard dataset Dresden Texture Dataset, while only requiring up to 22.57% of RAM usage. This efficiency stemmed from its manually designed feature extraction mechanism, which avoided the dual dependence of deep learning models on data and hardware.

The excellent performance of the GMM-MLBP model on regularized textures such as tiles was attributed to its multi-scale operator architecture. Ceramic tile textures typically exhibited significant spatial periodicity and geometric regularity (such as grid like arrangement), which was highly consistent with the multi-scale local feature extraction of the GMM-MLBP model. By integrating sub operators such as 3×3, 5×5, and 7×7, the model synchronously captured micro details (such as brick joints) and macro structures (such as repeating units), while GMM clustering effectively segmented homogeneous regions. In contrast, irregular textures such as weathered wood or marble with veins posed a challenge to fixed scale operators due to their strong randomness. However, the GMM-MLBP model still outperformed the comparison algorithms MSFF and SCGD on wood (CA 91.21%, MIoU 0.91) and marble (CA 93.95%, MIoU 0.93), verifying its generalization ability.

However, the GMM-MLBP model had limitations when dealing with highly complex or irregular textures. Its fixed scale sub-operators were difficult to adapt to textures lacking dominant structural frequencies (such as fractal granite or organic surfaces), resulting in a decrease in feature discriminative power. The parameterization assumption of GMM was difficult to accurately characterize nonlinear mixed regions in texture boundaries with gradient transitions (such as gradient ceramics), which could lead to segmentation errors. Although the preprocessing module had denoising capabilities, extreme noise or occlusion was still more likely to affect performance than CNN based methods due to its lack of learning driven robustness. These limitations point to the need to introduce adaptive scale selection mechanisms or integrate lightweight CNN modules in the future to enhance adaptability to chaotic textures.

In terms of real-time performance and production scalability, the GMM-MLBP model demonstrated clear advantages. The average processing speed of 2.93 seconds per image and the pure CPU execution mode met the sub 5-second delay requirements of industrial online quality inspection. The average CSV control file generated was only 8.75 KB, significantly reducing storage and transmission overhead (compared to 3DGS's 21.41 KB). The GMM-MLBP model was implemented in Python, making it easy to integrate into factory PLC and MES systems without the need for dedicated GPU support, which was significantly better than high computing power generation models such as GET3D [11]. In the future, it can be further optimized to adapt to ARM architecture embedded devices and expand edge scene applications. From an industrial practice perspective, the GMM-MLBP model framework bridges the gap between academic research and production demand. Its ability to directly output machine-readable control files (CSV) enables closed-loop control from texture design to production parameters. This is in sharp contrast to existing GAN methods [9-11] that only focus on visual synthesis and are detached from the production chain.

The GMM-MLBP model has unique advantages in texture feature extraction. Compared with deep learning models such as ResNet and MobileNet, the GMM-MLBP model has a relatively simple structure and low computational resource requirements. ResNet relies on deep residual learning and a large number of convolutional layers to extract complex features, while MobileNet achieves efficient computation through depthwise separable convolution, making it suitable for mobile devices. The GMM-MLBP model focuses on texture details, extracts features using multi-scale local binary patterns, and combines Gaussian mixture model clustering to achieve texture recognition, which is more suitable for scenes with obvious texture features. For building decoration material manufacturers, the GMM-MLBP model provides a practical solution that strikes a balance between accuracy, cost, and practicality.

5 Conclusion

To precisely discern a diverse array of textures and replicate building decoration materials with authentic-looking textures, an efficient technology for the texture recognition and reproduction of building decoration materials, which is grounded in the MLBP algorithm, has been successfully developed. A corresponding GMM-MLBP model was proposed by introducing preprocessing modules, multi-scale sub operators, GMM, and texture reproduction modules. The MLBP algorithm performed well in texture feature extraction, with a maximum CA of 96.84% and MIoU of 0.97. The highest CA and MIoU of the MSFF algorithm were 89.81 and 0.88, respectively. The MLBP algorithm performed significantly better than other compared algorithms. In practical applications, the GMM-MLBP model could generate high-quality texture control files through image preprocessing and texture feature optimization. The average size of the generated CSV file was only 8.75 KB, which was significantly better than the 21.41 KB of the 3DGS model, thus reducing memory usage. In the comprehensive evaluation of texture reproduction performance, the GMM-MLBP model achieved the highest scores in detail realism, color reproduction, and overall aesthetics for wood, marble, and tile textures, with an overall comprehensive score of 8.93, significantly better than the 3DGS model's 8.27. This indicated that the GMM-MLBP model had significant advantages in the field of texture reproduction for building decoration materials and could provide efficient and high-quality technical support for related industries. However, changes in lighting conditions can affect the grayscale value and contrast of the image, thereby affecting the effectiveness of the GMM-MLBP model in extracting and recognizing texture features; The parameters of the model need to be manually adjusted and lack an adaptive optimization mechanism. Future work will introduce adaptive lighting compensation algorithms to reduce the impact of lighting changes on images, combined with adaptive optimization algorithms in deep learning to automatically adjust model parameters and improve the model's adaptability.

References

[1] Hwang S W, Lee T, Kim H, Chung H, Choi J G, Yeo H. Classification of wood knots using artificial neural networks with texture and local feature-based image descriptors. Holzforschung, 2022, 76(1): 1-13. https://doi.org/10.1515/hf-2021-0051

[2] Xu X. Multimedia VR image improvement and simulation analysis based on visual VR restructuring algorithm. Informatica, 2024, 48(1): 107-118. https://doi.org/10.31449/inf.v48i1.5368

[3] Wimmer G, Schraml R, Hofbauer H, Petutschnigg A, Uhl A. Robustness of texture-based roundwood tracking. European Journal of Wood and Wood Products, 2023, 81(3): 669-683. https://doi.org/10.1007/s00107-022-01913-4

[4] Abdul Hamid L B, Mohd Khairuddin A S, Khairuddin U, Rosli N R, Mokhtar N. Texture image classification using improved image enhancement and adaptive SVM. Signal, Image and Video Processing, 2022, 16(6): 1587-1594. https://doi.org/10.1007/s11760-021-02113-y

[5] Liu X, Aldrich C. Deep learning approaches to image texture analysis in material processing. Metals, 2022, 12(2): 355-377. https://doi.org/10.3390/met12020355

[6] Kamijyo R, Ishii A, Coppieters S, Yamanaka A. Bayesian texture optimization using deep neural network-based numerical material test. International Journal of Mechanical Sciences, 2022, 223(1): 107285-107301. https://doi.org/10.1016/j.ijmecsci.2022.107285

[7] Shukla A, Kalnoor G, Kumar A, Yuvaraj N, Manikandan R, Ramkumar M. Improved recognition rate of different material category using convolutional neural networks. Materials Today: Proceedings, 2023, 81(1): 947-950. https://doi.org/10.1016/j.matpr.2021.04.307

[8] Bai N, Xue Y, Chen S, Shi L, Shi J, Zhang Y, Guo C F. A robotic sensory system with high spatiotemporal resolution for texture recognition. Nature Communications, 2023, 14(1): 7121-7143. https://doi.org/10.1038/s41467-023-42722-4

[9] Siddiqui Y, Thies J, Ma F, Shan Q, Nieß ner M, Dai A. Texturify: Generating textures on 3d shape surfaces. European Conference on Computer Vision. Cham: Springer Nature Switzerland, 2022, 13663(1): 72-88. https://doi.org/10.1007/978-3-031-20062-5_5

[10] Cao T, Kreis K, Fidler S, Sharp N, Yin K. Texfusion: Synthesizing 3d textures with text-guided image diffusion models. Proceedings of the IEEE/CVF International Conference on Computer Vision. 2023, 1(1): 4169-4181. https://doi.org/10.1109/ICCV51070.2023.00385

[11] Gao J, Shen T, Wang Z, Chen W, Yin K, Li D, Fidler S. Get3d: A generative model of high quality 3d textured shapes learned from images. Advances in Neural Information Processing Systems, 2022, 35(1): 31841-31854. https://doi.org/ 10.48550/arXiv.2209.11163

[12] Li D, Li X, Wang B. Texture direction recognition of wooden beams and columns based on improved meta-learning. Signal, Image and Video Processing, 2023, 17(8): 4447-4454. https://doi.org/10.1007/s11760-023-02678-w

[13] Chen Y. Indoor environment 3D space design based on 3D modeling and image processing. Informatica, 2025, 49(9): 103-112. https://doi.org/10.31449/inf.v49i9.5498

[14] Vu H N, Nguyen M H, Pham C. Masked face recognition with convolutional neural networks and local binary patterns. Applied Intelligence, 2022, 52(5): 5497-5512. https://doi.org/10.1007/s10489-021-02728-1

[15] Prema C E, Suresh S, Krishnan M N, Leema N. A novel efficient video smoke detection algorithm using co-occurrence of local binary pattern variants. Fire Technology, 2022, 58(5): 3139-3165. https://doi.org/10.1007/s10694-022-01306-2

[16] Zhang Z, Wang M. Multi-feature fusion partitioned local binary pattern method for finger vein recognition. Signal, Image and Video Processing, 2022, 16(4): 1091-1099. https://doi.org/10.1007/s11760-021-02058-2

[17] Angizi S, Morsali M, Tabrizchi S, Roohi A. A near-sensor processing accelerator for approximate local binary pattern networks. IEEE Transactions on Emerging Topics in Computing, 2023, 12(1): 73-83. https://doi.org/10.1109/TETC.2023.3285493

[18] Gong G, Wang X, Zhang J, Shang X, Pan Z, Li Z, Zhang J. MSFF: A multi-scale feature fusion convolutional neural network for hyperspectral image classification. Electronics, 2025, 14(4): 797-824. https://doi.org/10.3390/electronics14040797

[19] Wei H, Jia K, Wang Q, Ji F, Cao B, Qi J, Yan X. A texture feature extraction method considering spatial continuity and gray diversity. International Journal of Applied Earth Observation and Geoinformation, 2024, 130(1): 103896-103922. https://doi.org/10.1016/j.jag.2024.103896

[20] Ress V, Meyer J, Zhang W, Skuddis D, Soergel U, Haala N. 3D Gaussian splatting aided localization for large and complex indoor-environments. arXiv preprint arXiv. 2025, 2502(1):13803-1311. https://doi.org/10.48550/arXiv.2502.13803

[21] Pan X, Yu Z, Yang Z. A deep learning multimodal fusion framework for wood species identification using near-infrared spectroscopy GADF and RGB image. Holzforschung, 2023, 77(11-12): 816-827. https://doi.org/10.1515/hf-2023-0062