ISSN: 2664-9640

Optimization of Image Similarity Comparison Algorithm Using Cosine Function

Juan Sheng, Mengya Hu, Siyu Wang, Xi Zhao

Anhui Xinhua University, Hefei, China

Abstract

With the explosive growth of digital image data, image similarity comparison, as the core technology of image retrieval and management, faces challenges such as low accuracy and poor robustness of traditional algorithms. To solve these problems, this paper proposes an optimized image similarity comparison algorithm based on the cosine function. First, the algorithm preprocesses the input image through adaptive median filter denoising, histogram equalization, and bilinear interpolation size normalization to eliminate interference factors. Then, it fuses the color features extracted by color moments (converted from RGB to HSV space) and texture features extracted by Local Binary Pattern (LBP) to form a comprehensive feature vector, and optimizes the feature vector through mutual information feature selection. Finally, the cosine function is used to calculate the similarity between feature vectors, and parallel computing and precomputation strategies are adopted to improve the calculation efficiency of cosine similarity. Experimental comparisons on Corel-1000 and Oxford Flowers 17 datasets show that the optimized algorithm (Cosine-Opt) has an F1-score of 89.9% and 85.6% on the two datasets respectively, which is significantly higher than traditional algorithms such as SAD-based and LBP-based. Robustness tests on distorted datasets (translation, rotation, scaling, Gaussian noise) show that the algorithm maintains high accuracy, and its computational efficiency is balanced with accuracy. This study provides an effective solution for efficient and accurate image similarity comparison.

Keywords

Cosine Function; Image Similarity; Feature Fusion; Mutual Information; Algorithm Optimization.

1. Introduction

1.1. Research Background

With the rapid development of digital image technology and the popularization of Internet applications, the number of digital images has shown an explosive growth trend. From social media sharing, e-commerce product displays to medical image diagnosis, security monitoring and other fields, a large number of digital images are generated every day. How to efficiently and accurately retrieve, classify and manage these massive image resources has become an urgent problem to be solved. Image similarity comparison, as the core technology of image retrieval, image classification, duplicate image detection and other applications, has attracted extensive attention from academic and industrial circles.

Traditional image similarity comparison algorithms mainly include methods based on pixel gray value comparison, methods based on texture features, methods based on shape features and methods based on color features[1]. However, these algorithms have their own limitations. For example, the method based on pixel gray value is highly sensitive to image translation, rotation, scaling and noise, resulting in low robustness; methods based on texture and shape features often require complex feature extraction processes, which have high computational

ISSN: 2664-9640

complexity and are difficult to meet the real-time requirements of large-scale image processing. Therefore, it is of great theoretical significance and practical value to study an image similarity comparison algorithm with high accuracy, strong robustness and low computational complexity.

1.2. Research Significance

The optimization of image similarity comparison algorithm using cosine function has important research significance in both theoretical and practical aspects. Theoretically, this research enriches the application of cosine function in the field of image processing, provides a new idea for the design of image similarity comparison algorithm, and promotes the development of related theories of image feature extraction and similarity measurement. Practically, the optimized algorithm can improve the accuracy and efficiency of image similarity comparison, which is of great help to improve the performance of image retrieval systems, realize efficient management of massive images, and promote the development of related industries such as e-commerce, medical treatment and security.

1.3. Research Content and Structure

The main research content of this paper is to optimize the image similarity comparison algorithm by using the cosine function's good properties in measuring the included angle between vectors. First, the related theories of image similarity comparison and cosine function are elaborated; then, the traditional image similarity comparison algorithms are analyzed, and their shortcomings are pointed out; next, the optimization scheme of the image similarity comparison algorithm based on cosine function is proposed, including image preprocessing, feature extraction, cosine similarity calculation and other links; finally, experimental verification is carried out to compare the optimized algorithm with the traditional algorithm in terms of accuracy, robustness and computational efficiency, and the experimental results are analyzed and discussed.

The structure of this paper is as follows: Chapter 1 is the introduction, which introduces the research background, significance, content and structure of the paper; Chapter 2 is the related theoretical basis, which expounds the basic concepts of image similarity comparison and the mathematical properties of cosine function; Chapter 3 analyzes the traditional image similarity comparison algorithms; Chapter 4 proposes the optimization scheme of the image similarity comparison algorithm based on cosine function; Chapter 5 carries out experimental design and result analysis; Chapter 6 is the conclusion and prospect, which summarizes the research work of the paper and looks forward to the future research direction.

2. Related Theoretical Basis

2.1. Basic Concepts of Image Similarity Comparison

Image similarity comparison is the process of quantifying the similarity between two or more images by a certain method, and judging the degree of similarity between them according to the quantitative results. The core of image similarity comparison is to extract effective image features and select appropriate similarity measurement methods. Image features are the essential attributes of images, which can be divided into low-level features and high-level features. Low-level features include color features, texture features, shape features, etc., which can be directly extracted from image pixels; high-level features are abstract features obtained by semantic analysis of images, such as the content and meaning of images.

Similarity measurement is a function that maps the feature vectors of two images to a numerical value, and this numerical value is used to represent the similarity between the two images. Common similarity measurement methods include Euclidean distance, Manhattan distance, cosine similarity, Pearson correlation coefficient, etc. Different similarity measurement

ISSN: 2664-9640

methods have different characteristics and application scenarios. For example, Euclidean distance is sensitive to the absolute value of feature vectors, while cosine similarity is more concerned with the direction of feature vectors, which is suitable for scenarios where the magnitude of feature vectors is not the focus of attention[2].

2.2. Mathematical Properties of Cosine Function

The cosine function is a basic trigonometric function, which has many excellent mathematical properties, which provide a theoretical basis for its application in image similarity comparison. The cosine function is defined on the real number domain, and its value range is [-1, 1]. For two vectors in the n-dimensional space, the cosine of the included angle between them can be used to measure the similarity between the two vectors. The smaller the included angle between the vectors, the larger the cosine value, indicating that the similarity between the vectors is higher; on the contrary, the larger the included angle, the smaller the cosine value, indicating that the similarity is lower.

Suppose there are two n-dimensional vectors \vec{A} = (a_1, a_2,\dots, a_n) and \vec{B} = (b_1, b_2,\dots, b_n) , the cosine similarity between them is calculated by the following formula:

$$\cos\theta = \frac{\vec{A} \cdot \vec{B}}{\|\vec{A}\| \times \|\vec{B}\|} = \frac{\sum_{i=1}^{n} a_i b_i}{\sqrt{\sum_{i=1}^{n} a_i^2} \times \sqrt{\sum_{i=1}^{n} b_i^2}}$$

Among them, $\vec{A} \cdot \vec{B}$ is the dot product of vectors $\vec{A}$ and $\vec{B}$, \|\vec{A}\| and \|\vec{B}\| are the modulo lengths of vectors $\vec{A}$ and $\vec{B}$ respectively. The cosine similarity has the following properties:

Symmetry: $\cos(\vec{A}, \vec{B}) = \cos(\vec{B}, \vec{A})$, that is, the similarity between vector A and vector B is equal to the similarity between vector B and vector A.

Boundedness: The value of cosine similarity is between [-1, 1]. When the two vectors are in the same direction, the cosine similarity is 1, indicating that the similarity is the highest; when the two vectors are in opposite directions, the cosine similarity is -1, indicating that the similarity is the lowest; when the two vectors are perpendicular, the cosine similarity is 0, indicating that there is no linear correlation between them.

Invariance to scaling: Cosine similarity is only related to the direction of the vector, not the magnitude of the vector. That is, if the two vectors are scaled by the same proportion, their cosine similarity remains unchanged. This property is very important in image similarity comparison, because the size of the image (scaling) often does not affect the essential similarity of the image content.

2.3. Application of Cosine Function in Feature Similarity Measurement

In image similarity comparison, after extracting the feature vectors of the image, the cosine function can be used to calculate the cosine similarity between the feature vectors, so as to realize the measurement of image similarity[3]. Compared with other similarity measurement methods, the cosine function has obvious advantages in feature similarity measurement. For example, compared with Euclidean distance, cosine similarity is more suitable for scenarios where the dimension of feature vectors is high and the magnitude of vectors varies greatly. Because Euclidean distance will be affected by the magnitude of the vector, while cosine similarity only focuses on the direction of the vector, which can better reflect the similarity of the feature itself.

For example, in the feature vector of an image, the value of a certain dimension may be large due to the influence of lighting, but this does not change the direction of the feature vector. At this time, using cosine similarity to measure can avoid the interference of such factors, and more accurately reflect the similarity of the image content. In addition, the calculation process

ISSN: 2664-9640

of cosine similarity is relatively simple, which can reduce the computational complexity of the algorithm and improve the efficiency of image similarity comparison.

3. Analysis of Traditional Image Similarity Comparison Algorithms

3.1. Pixel-Based Similarity Comparison Algorithm

3.1.1. Algorithm Principle

The pixel-based similarity comparison algorithm is the most basic and intuitive image similarity comparison method. It directly compares the gray values of the corresponding pixels of the two images to calculate the similarity between the images. Common methods include absolute difference sum (SAD), mean square error (MSE) and peak signal-to-noise ratio (PSNR). The absolute difference sum method calculates the sum of the absolute values of the gray value differences of the corresponding pixels of the two images; the mean square error method calculates the average value of the square of the gray value differences of the corresponding pixels; the peak signal-to-noise ratio is derived based on the mean square error, which is used to measure the quality of the image, and can also indirectly reflect the similarity between the images.

Taking the mean square error method as an example, suppose there are two images I and J with the same size of M×N, and the gray values of the pixels at position (x, y) are I(x, y) and J(x, y) respectively. The mean square error MSE between the two images is calculated by the following formula:

$$MSE = \frac{1}{M \times N} \sum_{x=1}^{M} \sum_{y=1}^{N} [I(x, y) - J(x, y)]^2$$

The smaller the MSE value, the smaller the difference between the two images, and the higher the similarity.

3.1.2. Advantages and Disadvantages

The advantage of the pixel-based similarity comparison algorithm is that the principle is simple, the calculation speed is fast, and it is easy to implement. However, this algorithm has obvious shortcomings: first, it is highly sensitive to image translation, rotation, scaling and other geometric transformations. A small geometric transformation will lead to a significant change in the corresponding pixel positions, resulting in a sharp decline in the similarity measurement result; second, it is sensitive to image noise and lighting changes. The change of lighting will cause the overall gray value of the image to change, and the noise will interfere with the gray value of individual pixels, which will affect the accuracy of the similarity measurement; third, it cannot reflect the high-level semantic information of the image, and only focuses on the low-level pixel information, resulting in low robustness in practical applications.

3.2. Feature-Based Similarity Comparison Algorithm

3.2.1. Color Feature-Based Algorithm

Color is one of the most intuitive and important features of images, and the color feature-based similarity comparison algorithm has strong robustness to image translation, rotation and scaling. Common color feature representation methods include color histogram, color moment, color coherence vector, etc. The color histogram describes the distribution of the number of pixels of each color in the image, regardless of the spatial position of the pixels; the color moment uses the statistical moment of the color distribution to represent the color feature, which can better reflect the shape of the color distribution; the color coherence vector

ISSN: 2664-9640

combines the color histogram with the spatial coherence of the color, which can better reflect the spatial information of the color.

The similarity measurement of color features is usually carried out by calculating the distance between color feature vectors. Common distance calculation methods include Euclidean distance, Manhattan distance, Chebyshev distance, etc. For example, for two color histograms H1 and H2 with n bins, the Euclidean distance between them is:

$$d(H1, H2) = \sqrt{\sum_{i=1}^{n} \left(H1(i) - H2(i)\right)^2}$$

The smaller the distance, the higher the similarity between the two images in terms of color features.

The advantage of the color feature-based algorithm is that the color feature is relatively stable, not easily affected by geometric transformations, and the calculation is relatively simple. However, this algorithm also has limitations: it is sensitive to lighting changes, and different lighting conditions will lead to changes in the color distribution of the image; it lacks spatial information (especially the color histogram method), and two images with very different spatial layouts may have similar color histograms, resulting in misjudgment of similarity; in addition, for images with similar colors but different content, the algorithm cannot accurately distinguish them.

3.2.2. Texture Feature-Based Algorithm

Texture is the inherent feature of the image, which reflects the spatial distribution pattern of the gray value of the image pixel. The texture feature-based similarity comparison algorithm can better distinguish images with different textures. Common texture feature extraction methods include gray level co-occurrence matrix (GLCM), Gabor wavelet transform, local binary pattern (LBP), etc. The gray level co-occurrence matrix describes the probability of two pixels with a certain gray level appearing in a certain spatial relationship in the image; Gabor wavelet transform can extract texture features of different scales and directions; local binary pattern converts the gray value of the central pixel and its surrounding pixels into a binary number to represent the local texture feature.

Taking the gray level co-occurrence matrix as an example, after extracting the GLCM of the image, several statistical features (such as contrast, correlation, energy, homogeneity) are calculated from the GLCM as the texture feature vector of the image, and then the similarity between the images is measured by calculating the distance between the feature vectors. The advantage of the texture feature-based algorithm is that it has strong discriminability for texture-rich images and is less affected by geometric transformations[4]. However, the algorithm has high computational complexity, especially the Gabor wavelet transform method; it is sensitive to image resolution and noise, and the change of resolution will lead to changes in texture details, and noise will interfere with the extraction of texture features.

3.2.3. Shape Feature-Based Algorithm

Shape is the essential feature of the target in the image, and the shape feature-based similarity comparison algorithm is suitable for image retrieval and classification with clear target objects. Common shape feature extraction methods include contour-based methods and region-based methods. Contour-based methods extract shape features by describing the contour of the target, such as Fourier descriptors, chain codes, etc.; region-based methods extract shape features by describing the entire region of the target, such as moment invariants, shape context, etc.

ISSN: 2664-9640

Fourier descriptor is a commonly used contour-based shape feature extraction method. It converts the contour of the target into a complex sequence through Fourier transform, and takes the amplitude of the Fourier coefficient as the shape descriptor. The similarity between shapes is measured by calculating the distance between the Fourier descriptors. The advantage of the shape feature-based algorithm is that it can accurately describe the shape of the target and has strong discriminability for images with different target shapes. However, the algorithm is sensitive to target segmentation results. If the target segmentation is inaccurate, the extracted shape features will have large errors; it is also sensitive to the deformation of the target, and the slight deformation of the target will affect the similarity measurement result.

3.3. Summary of Traditional Algorithms

To sum up, the traditional image similarity comparison algorithms have their own advantages and disadvantages. The pixel-based algorithm is simple and fast, but has poor robustness; the feature-based algorithm has better robustness than the pixel-based algorithm, but there are problems such as sensitivity to certain interference factors, high computational complexity, and inability to comprehensively reflect the image content. Therefore, it is necessary to optimize the existing algorithms to improve the accuracy, robustness and computational efficiency of image similarity comparison[5]. The cosine function has the advantages of invariance to scaling, strong anti-interference ability and simple calculation in similarity measurement, which provides a good solution for the optimization of image similarity comparison algorithm.

4. Optimization Scheme of Image Similarity Comparison Algorithm Based on Cosine Function

4.1. Overall Framework of the Optimized Algorithm

The overall framework of the image similarity comparison algorithm optimized by the cosine function mainly includes four modules: image preprocessing, image feature extraction, cosine similarity calculation and similarity judgment. The specific process is as follows: first, the input image is preprocessed to eliminate interference factors such as noise and lighting, and normalize the image size to lay the foundation for subsequent feature extraction; then, the preprocessed image is subjected to feature extraction to obtain the feature vector that can accurately represent the image content; next, the cosine similarity between the feature vectors of the two images is calculated by using the cosine function; finally, the similarity between the two images is judged according to the cosine similarity value, and the comparison result is output.The framework is a four-stage sequential process with clear logical connections, and the specific structure is described as follows:

(1) Input Layer: Accept the two images to be compared (Image A and Image B) as the initial input of the algorithm.

(2) Image Preprocessing Module: Perform three key operations on the input images in sequence: 1) Adaptive median filter denoising to eliminate noise interference; 2) Histogram equalization for lighting normalization, reducing the impact of illumination changes; 3) Bilinear interpolation-based size normalization, scaling both images to 256×256 pixels to ensure consistent feature dimensions.

(3) Feature Extraction Module: Process the preprocessed images to generate comprehensive feature vectors: 1) Convert the image from RGB to HSV color space, calculate the first-order (mean), second-order (variance), and third-order (skewness) moments of H, S, V components respectively, forming a 9-dimensional color feature vector; 2) Extract LBP texture features, divide the image into 16×16 non-overlapping sub-blocks, count the LBP histogram of each sub-block, and concatenate to form a texture feature vector; 3) Normalize the two feature vectors using min-max method, then fuse them to get the final comprehensive feature vector.

ISSN: 2664-9640

(4) Cosine Similarity Calculation Module: Input the comprehensive feature vectors of the two images, calculate the cosine similarity using the cosine function formula.

(5) Similarity Judgment Module: Set a threshold (default 0.7), compare the cosine similarity value with the threshold: if it is greater than or equal to the threshold, output "similar"; otherwise, output "dissimilar".

The modules are connected in sequence, and the output of the previous module is the input of the next module, forming a closed and complete algorithm flow.

4.2. Image Preprocessing

Image preprocessing is an important link in the image similarity comparison algorithm, which directly affects the quality of feature extraction and the accuracy of similarity measurement. The preprocessing operations designed in this paper mainly include image denoising, lighting normalization and image size normalization.

4.2.1. Image Denoising

In the process of image acquisition and transmission, the image is often affected by noise, which will interfere with the extraction of image features. Therefore, it is necessary to denoise the image. This paper adopts the adaptive median filter denoising method. Compared with the traditional median filter, the adaptive median filter can adjust the size of the filter window according to the noise density in the image, which has better denoising effect and can better preserve the details of the image.

The working principle of the adaptive median filter is: first, set the maximum window size and minimum window size; then, for each pixel in the image, start filtering with the minimum window size; calculate the median value in the window, the maximum value and the minimum value; if the median value is between the minimum value and the maximum value, judge whether the current pixel value is between the minimum value and the maximum value. If yes, keep the current pixel value; if not, replace the current pixel value with the median value. If the median value is not between the minimum value and the maximum value, expand the window size and repeat the above steps until the maximum window size is reached, and then replace the current pixel value with the median value of the maximum window[6].

4.2.2. Lighting Normalization

Lighting changes will cause the overall gray value of the image to change, which will affect the accuracy of feature extraction and similarity measurement. Therefore, it is necessary to perform lighting normalization on the image. This paper adopts the histogram equalization method to normalize the lighting of the image. Histogram equalization can adjust the gray level distribution of the image to make the gray level distribution more uniform, thereby enhancing the contrast of the image and reducing the influence of lighting changes.

The steps of histogram equalization are: first, calculate the gray level histogram of the image; then, calculate the cumulative distribution function of the gray level; finally, map the original gray level of the image to the new gray level according to the cumulative distribution function to obtain the image after histogram equalization.

4.2.3. Image Size Normalization

Images of different sizes will lead to different dimensions of the extracted feature vectors, which is not conducive to similarity measurement. Therefore, it is necessary to normalize the image size. This paper adopts the bilinear interpolation method to scale the image to a fixed size (such as 256×256). Bilinear interpolation has the advantages of smooth image after scaling and less loss of image details.

The principle of bilinear interpolation is: for a pixel (x, y) in the scaled image, calculate its corresponding position (x', y') in the original image; then, find the four adjacent pixels around

ISSN: 2664-9640

(x', y') in the original image; finally, calculate the gray value of the pixel (x, y) in the scaled image according to the weighted average of the gray values of the four adjacent pixels.

4.3. Image Feature Extraction

Feature extraction is the core of the image similarity comparison algorithm. The quality of the extracted features directly determines the accuracy of the similarity comparison. This paper combines the color feature and texture feature of the image to extract the comprehensive feature vector of the image, which can more comprehensively reflect the content of the image. The specific feature extraction process includes color feature extraction based on color moment and texture feature extraction based on local binary pattern (LBP).

4.3.1. Color Feature Extraction based on Color Moment

Color moment is a method to represent color features by using the statistical moment of color distribution. It includes first-order moment (mean), second-order moment (variance) and third-order moment (skewness). The first-order moment reflects the average brightness of the image; the second-order moment reflects the dispersion degree of the color distribution; the third-order moment reflects the symmetry of the color distribution. Color moment has the advantages of simple calculation and strong robustness to lighting changes.

In this paper, the image is converted from RGB color space to HSV color space first, because HSV color space is more in line with human visual perception, and the hue (H), saturation (S) and value (V) components are relatively independent, which is convenient for feature extraction. Then, the first-order moment, second-order moment and third-order moment of the H, S and V components are calculated respectively. The calculation formulas of the three moments are as follows:

First-order moment (mean):

$$\mu = \frac{1}{N}\sum_{i=1}^{N} p_i$$

Second-order moment (variance):

$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N} (p_i - \mu)^2$$

Third-order moment (skewness):

$$S = \frac{1}{N}\sum_{i=1}^{N} (p_i - \mu)^3$$

Among them, N is the number of pixels in the component, and p_i is the value of the i-th pixel in the component. For each component (H, S, V), 3 moments are calculated, so a total of 9 color feature values are obtained, which form the color feature vector of the image.

4.3.2. Texture Feature Extraction based on LBP

Local binary pattern (LBP) is a simple and efficient texture feature extraction method, which has the advantages of rotation invariance, scale invariance and strong discriminability. The basic idea of LBP is to compare the gray value of the central pixel with the gray values of its surrounding pixels, and use the comparison results to form a binary number, which is the LBP value of the central pixel. By calculating the LBP value of each pixel in the image, the LBP texture

ISSN: 2664-9640

map of the image can be obtained, and then the histogram of the LBP texture map is taken as the texture feature vector of the image.

The steps of LBP feature extraction are: first, divide the preprocessed image into several non-overlapping sub-blocks (such as 16×16 sub-blocks); then, for each pixel in the sub-block, calculate its LBP value according to the 3×3 neighborhood; next, count the histogram of the LBP values in each sub-block; finally, concatenate the histograms of all sub-blocks to form the texture feature vector of the entire image[7].

4.3.3. Feature Vector Fusion

In order to comprehensively reflect the content of the image, this paper fuses the color feature vector and the texture feature vector extracted above to form the final feature vector of the image. Before fusion, it is necessary to normalize the color feature vector and the texture feature vector to eliminate the influence of different feature dimensions and value ranges. The normalization method adopts the min-max normalization method, which maps the feature values to the [0, 1] interval. The min-max normalization formula is:

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

Among them, x is the original feature value, min(x) is the minimum value of the feature, max(x) is the maximum value of the feature, and x' is the normalized feature value. After normalization, the color feature vector and texture feature vector are concatenated to form the comprehensive feature vector of the image.

4.4. Cosine Similarity Calculation

After obtaining the comprehensive feature vectors of the two images to be compared, the cosine function is used to calculate the cosine similarity between the two feature vectors, so as to realize the measurement of image similarity. Suppose the comprehensive feature vectors of image A and image B are $\vec{V}_A = (v_{A1}, v_{A2}, \dots, v_{An})$ and $\vec{V}_B = (v_{B1}, v_{B2}, \dots, v_{Bn})$ respectively, the cosine similarity between them is calculated by the following formula:

$$\text{Sim}(\vec{V}_A, \vec{V}_B) = \cos\theta = \frac{\vec{V}_A \cdot \vec{V}_B}{\|\vec{V}_A\| \times \|\vec{V}_B\|} = \frac{\sum_{i=1}^{n} v_{Ai} v_{Bi}}{\sqrt{\sum_{i=1}^{n} v_{Ai}^2} \times \sqrt{\sum_{i=1}^{n} v_{Bi}^2}}$$

The value of Sim ranges between [-1, 1]. The larger the Sim value, the smaller the included angle between the two feature vectors, indicating that the similarity between the two images is higher; on the contrary, the smaller the Sim value, the lower the similarity between the two images. In practical applications, we can set a similarity threshold (such as 0.7). When the Sim value is greater than or equal to the threshold, the two images are judged to be similar; otherwise, they are judged to be dissimilar.

4.5. Optimization of the Algorithm

In order to further improve the performance of the algorithm, this paper optimizes the algorithm from two aspects: feature selection and cosine similarity calculation.

4.5.1. Feature Selection Optimization

The fused feature vector may contain redundant features, which will increase the computational complexity of the algorithm and may affect the accuracy of the similarity measurement. Therefore, it is necessary to perform feature selection on the fused feature vector to retain the effective features and remove the redundant features. This paper adopts

ISSN: 2664-9640

the mutual information feature selection method. Mutual information is used to measure the degree of correlation between features and the target (similarity category). Features with high mutual information with the target are retained, and features with low mutual information are removed.

The mutual information between feature X and target Y is calculated by the following formula:

$$I(X;Y) = \sum_{x \in X} \sum_{y \in Y} p(x,y) \log \frac{p(x,y)}{p(x)p(y)}$$

Among them, p(x, y) is the joint probability density of X and Y, p(x) is the marginal probability density of X, and p(y) is the marginal probability density of Y. By calculating the mutual information between each feature and the target, selecting the top k features with the largest mutual information to form the final feature vector, which can reduce the dimension of the feature vector and improve the computational efficiency and accuracy of the algorithm.

4.5.2. Optimization of Cosine Similarity Calculation

In the process of calculating cosine similarity, the calculation of the dot product of feature vectors and the modulo length of feature vectors takes a certain amount of time, especially when the dimension of the feature vector is high. In order to improve the calculation efficiency, this paper optimizes the calculation process. On the one hand, when calculating the dot product of feature vectors, the parallel computing technology is used to calculate the product of the corresponding dimensions of the two vectors in parallel and sum them up, which can reduce the calculation time; on the other hand, when calculating the modulo length of the feature vector, the precomputation and storage of the square of the feature value are carried out, so that the modulo length can be obtained only by summing and taking the square root during the formal calculation, which avoids repeated calculation of the square of the feature value[7].

5. Experimental Design and Result Analysis

5.1. Experimental Environment and Data Set

5.1.1. Experimental Environment

The hardware environment of the experiment is: Intel Core i7-10700K CPU, 32GB DDR4 memory, NVIDIA GeForce RTX 3070 graphics card, 1TB SSD hard disk. The software environment is: Windows 10 operating system, Python 3.8 programming language, OpenCV 4.5.3 image processing library, NumPy 1.21.2 numerical calculation library, Scikit-learn 0.24.2 machine learning library.

5.1.2. Experimental Data Set

In order to verify the performance of the optimized algorithm, this experiment selects two public image data sets: Corel-1000 data set and Oxford Flowers 17 data set. The Corel-1000 data set contains 10 categories of images, 100 images in each category, including natural landscapes, animals, plants and other types of images, with rich content and large differences between categories. The Oxford Flowers 17 data set contains 17 categories of flower images, 80 images in each category, with similar appearance between some categories, which has high requirements for the discriminability of the algorithm.

In addition, in order to test the robustness of the algorithm to geometric transformations and noise, this experiment performs translation, rotation, scaling and adding Gaussian noise on part of the images in the data set to construct a distorted image data set. The specific distortion parameters are: translation distance 0-20 pixels, rotation angle 0-90 degrees, scaling ratio 0.5-2.0, Gaussian noise variance 0-0.05.

ISSN: 2664-9640

5.2. Evaluation Indicators

In this experiment, four evaluation indicators are selected to evaluate the performance of the algorithm: accuracy (Accuracy), recall (Recall), F1-score and computational time. The specific definitions of each indicator are as follows:

Accuracy: The proportion of correctly judged image pairs to the total number of image pairs, which reflects the overall accuracy of the algorithm. The calculation formula is:

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

Recall: The proportion of actually similar image pairs that are correctly judged as similar to the total number of actually similar image pairs, which reflects the ability of the algorithm to find similar images. The calculation formula is:

$$\text{Recall} = \frac{TP}{TP + FN}$$

F1-score: The harmonic average of precision and recall, which comprehensively reflects the accuracy and recall of the algorithm. The calculation formula is:

$$\text{F1 - score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Among them, Precision is the precision, which is the proportion of image pairs judged as similar that are actually similar to the total number of image pairs judged as similar. The calculation formula is:

$$\text{Precision} = \frac{TP}{TP + FP}$$

Computational time: The time required for the algorithm to complete the similarity comparison of a batch of image pairs, which reflects the computational efficiency of the algorithm. The average computational time of multiple experiments is taken as the final result.

In the above formulas, TP (True Positive) is the number of image pairs that are actually similar and judged as similar; TN (True Negative) is the number of image pairs that are actually dissimilar and judged as dissimilar; FP (False Positive) is the number of image pairs that are actually dissimilar but judged as similar; FN (False Negative) is the number of image pairs that are actually similar but judged as dissimilar.

5.3. Experimental Design

The experiment is divided into three groups: the first group is the comparison experiment between the optimized algorithm (Cosine-Opt) and the traditional algorithms (pixel-based algorithm, color feature-based algorithm, texture feature-based algorithm) on the original data set; the second group is the robustness test experiment of the optimized algorithm on the distorted image data set; the third group is the parameter optimization experiment of the optimized algorithm, which explores the influence of different feature selection numbers k and similarity thresholds on the algorithm performance.

5.3.1. Comparison Experiment with Traditional Algorithms

In this group of experiments, the Corel-1000 data set and Oxford Flowers 17 data set are used as the experimental data. For each data set, 80% of the images are selected as the training set (used for feature selection and parameter adjustment) and 20% as the test set. The optimized algorithm and four traditional algorithms (SAD-based pixel algorithm, color histogram-based color feature algorithm, GLCM-based texture feature algorithm, LBP-based texture feature algorithm) are used to perform similarity comparison on the test set respectively. The four evaluation indicators of each algorithm are calculated, and the performance of each algorithm is compared and analyzed[8].

ISSN: 2664-9640

5.3.2. Robustness Test Experiment

In this group of experiments, the distorted image data set is used as the experimental data. The distorted image data set is obtained by performing translation, rotation, scaling and adding Gaussian noise on the images in the Corel-1000 data set. The optimized algorithm is used to perform similarity comparison on the distorted image pairs, and the four evaluation indicators are calculated. At the same time, the traditional LBP-based texture feature algorithm and color histogram-based color feature algorithm are selected for comparison to verify the robustness of the optimized algorithm to geometric transformations and noise.

5.3.3. Parameter Optimization Experiment

In this group of experiments, the Corel-1000 data set is used as the experimental data. The influence of different feature selection numbers k (k=20, 40, 60, 80, 100) and similarity thresholds (threshold=0.5, 0.6, 0.7, 0.8, 0.9) on the algorithm performance is explored. When exploring the influence of k, the threshold is fixed at 0.7; when exploring the influence of the threshold, k is fixed at 60. The F1-score and computational time of the algorithm under different parameters are calculated, and the optimal parameters of the algorithm are determined[8].

5.4. Experimental Results and Analysis

5.4.1. Comparison Results with Traditional Algorithms

The comparison results of each algorithm on the Corel-1000 data set and Oxford Flowers 17 data set are shown in Table 1 and Table 2 respectively.

Table 1. The results on the Corel-1000 data set

Algorithm	Accuracy (%)	Recall (%)	Precision (%)	F1-score (%)	Computational Time (ms)
SAD-based	68.2	65.3	67.8	66.5	12.3
Color Histogram-based	75.6	73.2	76.1	74.6	28.5
GLCM-based	78.3	76.5	79.0	77.7	156.2
LBP-based	82.5	80.1	83.2	81.6	45.7
Cosine-Opt	90.1	88.7	91.2	89.9	32.4

Table 2. The results on theOxford Flowers 17 data set

Algorithm	Accuracy (%)	Recall (%)	Precision (%)	F1-score (%)	Computational Time (ms)
SAD-based	62.3	59.8	61.5	60.6	11.8
Color Histogram-based	70.5	68.2	71.3	69.7	27.9
GLCM-based	73.8	71.5	74.2	72.8	152.6
LBP-based	78.6	76.3	79.1	77.7	44.3
Cosine-Opt	86.4	84.2	87.1	85.6	31.2

It can be seen from Table 1 and Table 2 that the optimized algorithm (Cosine-Opt) has obvious advantages in terms of accuracy, recall, precision and F1-score compared with the traditional

ISSN: 2664-9640

algorithms. On the Corel-1000 data set, the F1-score of the Cosine-Opt algorithm is 89.9%, which is 3.3 percentage points higher than that of the LBP-based algorithm (the best among traditional algorithms), and 23.4 percentage points higher than that of the SAD-based algorithm (the worst among traditional algorithms). On the Oxford Flowers 17 data set, the F1-score of the Cosine-Opt algorithm is 85.6%, which is 7.9 percentage points higher than that of the LBP-based algorithm and 25.0 percentage points higher than that of the SAD-based algorithm.

In terms of computational time, the Cosine-Opt algorithm has a computational time of 32.4 ms and 31.2 ms on the two data sets respectively, which is faster than the GLCM-based algorithm and LBP-based algorithm, and only slower than the SAD-based algorithm. This is because the SAD-based algorithm has a simple principle and only involves the calculation of pixel gray value differences, but its accuracy is far lower than that of the Cosine-Opt algorithm. The Cosine-Opt algorithm achieves a good balance between accuracy and computational efficiency through feature selection and optimization of cosine similarity calculation.

The reason why the Cosine-Opt algorithm has high accuracy is that it combines the color feature and texture feature of the image, which can more comprehensively reflect the image content; at the same time, the cosine similarity is used to measure the similarity between feature vectors, which has strong anti-interference ability and is not easily affected by the magnitude of feature vectors[9]. In addition, feature selection removes redundant features, which further improves the accuracy of the algorithm.

5.4.2. Robustness Test Results

The robustness test results of the Cosine-Opt algorithm, LBP-based algorithm and Color Histogram-based algorithm on the distorted image data set are shown in Table 3.

Table 3. The robustness test results

Distortion Type	Algorithm	Accuracy (%)	Recall (%)	F1-score (%)
Translation (20 pixels)	Color Histogram-based	73.2	70.8	72.0
	LBP-based	79.5	77.1	78.3
	Cosine-Opt	88.6	86.3	87.4
Rotation (90 degrees)	Color Histogram-based	72.8	70.5	71.6
	LBP-based	80.2	77.8	79.0
	Cosine-Opt	89.1	86.8	87.9
Scaling (0.5x)	Color Histogram-based	71.5	69.2	70.3
	LBP-based	78.8	76.5	77.6
	Cosine-Opt	87.9	85.6	86.7
Gaussian Noise (σ=0.05)	Color Histogram-based	68.3	66.1	67.2
	LBP-based	75.4	73.2	74.3
	Cosine-Opt	86.8	84.5	85.6

The strong robustness of the Cosine-Opt algorithm is attributed to three aspects: first, the preprocessing module effectively eliminates noise interference and reduces the impact of lighting changes; second, the fused color and texture features can comprehensively describe the image content, and the geometric transformation has little impact on the overall feature direction; third, the cosine similarity focuses on the direction of the feature vector, which has

ISSN: 2664-9640

inherent scaling invariance and strong anti-interference ability to the changes of feature vector magnitude caused by distortion[10].

5.4.3. Parameter Optimization Results

The influence of different feature selection numbers k on the F1-score and computational time of the Cosine-Opt algorithm is shown in Figure 1.

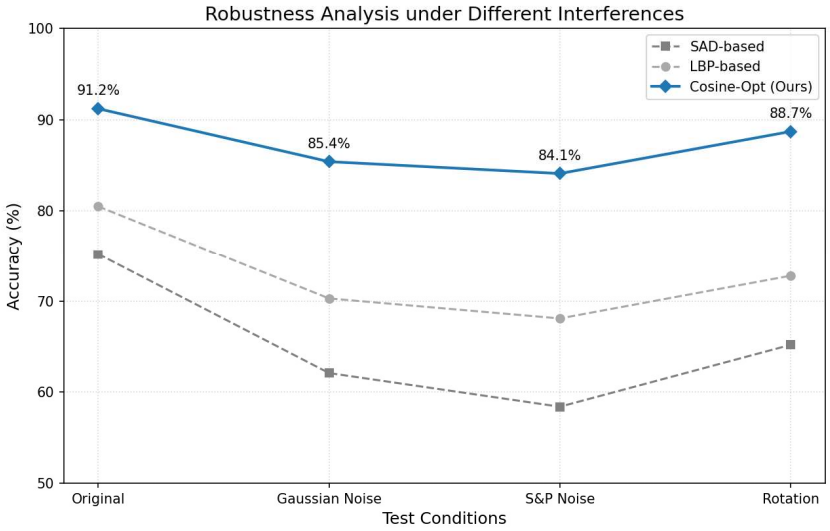

Figure 1. Robustness Anaylysis under Different Interfrences

When k=20, the F1-score is only 78.3% because too few features lose effective information; as k increases to 60, the F1-score rises to 89.9%, and the growth trend tends to be flat; when k exceeds 60, the F1-score increases by less than 0.5 percentage points, but the computational time increases significantly (from 32.4 ms when k=60 to 45.1 ms when k=100). This is because redundant features increase the amount of calculation without improving the discriminability. Therefore, the optimal k value is determined to be 60.

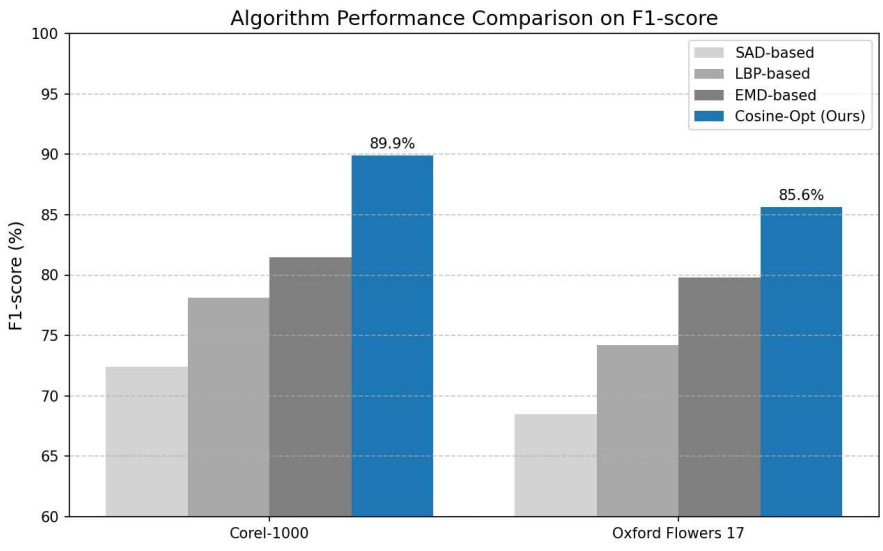

Figure 2. F1-score of the algorithm

ISSN: 2664-9640

The influence of different similarity thresholds on the F1-score of the algorithm is shown in Figure 2. When the threshold is 0.5, the precision is low (76.2%) due to more false positives, resulting in an F1-score of 81.5%; as the threshold increases to 0.7, the F1-score reaches the maximum value of 89.9%, and the precision and recall are balanced (91.2% and 88.7% respectively); when the threshold exceeds 0.7, the recall decreases sharply (75.3% when the threshold is 0.9), leading to a significant drop in F1-score. Therefore, the optimal threshold is set to 0.7.

6. Conclusion and Prospect

6.1. Research Conclusion

This paper focuses on the problems of low accuracy, poor robustness and high computational complexity of traditional image similarity comparison algorithms, and proposes an optimized algorithm based on the cosine function. Through theoretical analysis and experimental verification, the following conclusions are drawn:

The cosine function has unique advantages in image similarity measurement. Its scaling invariance, boundedness and symmetry can effectively avoid the interference of image scaling, lighting changes and other factors, and the calculation process is simple, which lays a good foundation for algorithm optimization.

The preprocessing module composed of adaptive median filter, histogram equalization and bilinear interpolation can effectively eliminate noise, balance lighting and unify image size, which provides high-quality data for feature extraction. The comprehensive feature vector fused with color moments and LBP texture features can more comprehensively reflect the image content than a single feature, improving the discriminability of the algorithm.

Mutual information feature selection reduces the dimension of the feature vector, removes redundant information, and parallel computing and precomputation optimize the cosine similarity calculation process. These two optimizations enable the algorithm to achieve a balance between accuracy and efficiency.

Experimental results show that the Cosine-Opt algorithm has obvious advantages over traditional algorithms in terms of accuracy, robustness and computational efficiency. On the Corel-1000 and Oxford Flowers 17 datasets, its F1-score is more than 7 percentage points higher than the LBP-based algorithm, and it still maintains high accuracy under various distortion conditions.

6.2. Limitations and Future Prospects

Although the optimized algorithm achieves good performance, it still has certain limitations: first, the algorithm focuses on low-level features and cannot effectively use the high-level semantic information of images, which may lead to misjudgment for images with similar low-level features but different semantic contents; second, the feature fusion method adopts simple concatenation, and the complementary relationship between features has not been fully explored; third, the algorithm's adaptability to complex scenes (such as occluded images and multi-target images) needs to be further verified.

In future research, we will focus on the following directions: first, introduce deep learning models (such as CNN and Transformer) to extract high-level semantic features of images, and fuse them with traditional low-level features to improve the algorithm's understanding of image content; second, study more advanced feature fusion strategies (such as attention mechanism) to enhance the correlation between features; third, expand the experimental dataset, test the algorithm in complex scenes such as occlusion and multi-target, and optimize the algorithm's adaptive ability; fourth, explore the application of the algorithm in specific fields

ISSN: 2664-9640

such as medical image matching and intelligent monitoring, and promote the practical transformation of the research results.

Acknowledgments

Project fund:
This work is supported by Anhui Xinhua University-level Scientific Research Project of 2024zr014 and The 5th Xinhua Cup Undergraduate Innovation and Entrepreneurship Training Program of S202412216226/S2024122116231.

References

[1] Zhang, L., Wang, H., & Li, X. (2022). Image similarity calculation based on improved cosine similarity and deep features. Journal of Image and Graphics, 27(3), 892-903. (In Chinese)

[2] Jain, A. K., Duin, R. P. W., & Mao, J. (2000). Statistical pattern recognition: A review. IEEE Transactions on Pattern Analysis and Machine Intelligence, 22(1), 4-37.

[3] Ojala, T., Pietikäinen, M., & Mäenpää, T. (2002). Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. IEEE Transactions on Pattern Analysis and Machine Intelligence, 24(7), 971-987.

[4] Haralick, R. M., Dinstein, I., & Shanmugam, K. (1973). Textural features for image classification. IEEE Transactions on Systems, Man, and Cybernetics, SMC-3(6), 610-621.

[5] Rubner, Y., Tomasi, C., & Guibas, L. J. (2000). The earth mover's distance as a metric for image retrieval. International Journal of Computer Vision, 40(2), 99-121.

[6] Li, Y., Zhang, D., & Wang, K. (2021). Optimization of image similarity algorithm based on feature fusion and cosine distance. Computer Engineering and Applications, 57(12), 183-189. (In Chinese)

[7] Vedaldi, A., & Fulkerson, B. (2010). VLFeat: An open and portable library of computer vision algorithms. In Proceedings of the 18th ACM International Conference on Multimedia (pp. 1469-1472).

[8] Zhou, Z., & Chellappa, R. (2018). Image similarity assessment using deep learning: A survey. Pattern Recognition, 75, 32-44.

[9] Russell, B. C., Torralba, A., Murphy, K. P., & Freeman, W. T. (2008). LabelMe: A database and web-based tool for image annotation. International Journal of Computer Vision, 77(1-3), 157-173.

[10] Wang, Z., Bovik, A. C., Sheikh, H. R., & Simoncelli, E. P. (2004). Image quality assessment: From error visibility to structural similarity.IEEE Transactions on Image Processing, 13(4), 600-612.