JAVA如何实现《计算图片相似度》？

1. 使用像素比较

通过比较两张图片的像素值来判断相似度。这种方法简单直接，但可能不够精确，尤其对于压缩或缩放后的图片。

2. 使用特征提取和比较

更精确的方法是提取图片的特征（如SIFT、SURF等），然后比较这些特征来判断图片的相似度。这种方法对于旋转、缩放和光照变化等具有较好的鲁棒性。

3. 使用深度学习

深度学习模型（如卷积神经网络CNN）可以提取图片的高层次特征，并通过比较这些特征来判断图片的相似度。这种方法通常需要大量的训练数据和计算资源，但可以实现较高的相似度判断精度。

第三方库

A. OpenCV

OpenCV是一个开源的计算机视觉库，提供了丰富的图像处理和分析功能。你可以使用OpenCV来提取图片的特征，并比较这些特征来判断图片的相似度。

B. JavaCV

JavaCV是OpenCV的Java接口，它允许你在Java应用程序中使用OpenCV的功能。

C. Deeplearning4j

Deeplearning4j是一个用于Java和Scala的深度学习库。你可以使用它来训练一个卷积神经网络模型，用于判断图片的相似度。

D. Apache Commons Imaging

Apache Commons Imaging是一个用于处理图像的库，虽然它本身并不直接提供相似度比较的功能，但你可以使用它来读取和处理图像数据，然后结合其他方法（如像素比较或特征提取）来实现相似度比较。

注意事项

* 在选择方法和库时，需要考虑你的具体需求，如精度要求、计算资源、处理速度等。
* 对于大规模的图片相似度比较任务，可能需要使用更高效的算法和并行计算技术来提高处理速度。
* 在使用深度学习模型时，需要准备足够的训练数据和进行模型调优以获得较好的性能。

抉择

综上所述，我找到了一种在计算上相对简单和高效的办法，虽然不如基于深度学习的方法在复杂场景下的准确性高，但仅仅用于常规的比较两张图的相似度，是足够了的。其原理就是结合上面的1、2点的像素比较和特征提取的结合。它使用像素级别的信息来提取简单特征，并通过比较这些特征来判断图像的相似度。

介绍

将图像转换为灰度后通过汉明距离计算图片相似度的原理及可行性分析如下：

原理：

1. 图像转换为灰度：这一步是为了简化图像的色彩信息，将彩色图像转化为灰度图像。灰度图像中，每个像素只有一个亮度值，通常范围在0（黑色）到255（白色）之间，这样可以去除色彩对相似度判断的影响。
2. 计算指纹：在灰度图像的基础上，进一步将图像缩小到固定尺寸（例如8x8像素），然后计算所有像素的灰度平均值。每个像素的灰度值与平均值进行比较，大于或等于平均值记为1，小于平均值记为0。这样，整个图像就被转换

为一个固定长度的二进制数（在这个例子中是64位），即图像的“指纹”。
3. 汉明距离计算：汉明距离是两个等长字符串在相同位置上不同字符的个数。在图像相似度判断中，可以计算两个图像指纹之间的汉明距离。如果汉明距离较小，说明两个图像的指纹在大部分位置上都是相同的，即两个图像相似；反之，如果汉明距离较大，则说明两个图像在很多位置上都不相同，即两个图像不相似。

可行性：

这种方法在理论上是可行的，并且在实际应用中也有一定效果。通过将图像转换为灰度并计算指纹，可以去除色彩和细节信息对相似度判断的影响，只保留图像的基本结构和明暗信息。汉明距离计算简单快速，适用于大规模图像相似度比较任务。

然而，需要注意的是，这种方法可能对于某些复杂的图像相似度判断任务不够准确。例如，如果两张图像的内容非常相似但位置或角度有所差异，那么它们的指纹可能会有很大的不同，导致判断为不相似。此外，如果图像的细节信息对于相似度判断非常重要，那么将图像转换为灰度并缩小尺寸可能会丢失这些信息，影响判断的准确性。

因此，在选择是否使用这种方法时，需要根据具体的应用场景和需求进行权衡。如果需要快速判断图像的大致相似度，并且可以接受一定的误差，那么这种方法是一个不错的选择。如果需要更精确的相似度判断，可能需要考虑使用更复杂的图像处理和分析方法。

代码

```
package com.pec.mall.core.util;
import org.apache.commons.codec.binary.Hex;
import javax.imageio.ImageIO;
import java.awt.*;
import java.awt.color.ColorSpace;
import java.awt.image.BufferedImage;
import java.awt.image.ColorConvertOp;
import java.io.File;
import java.io.IOException;
import java.nio.charset.StandardCharsets;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
```

```
public class ImageUtil {

    public static double hammingDistanceSimilarity(File imageFile1, File
file2){
        int[] pixels1 = new int[]{};
        int[] pixels2 = new int[]{};
        try {
            pixels1 = getImgFinger(imageFile1);
            pixels2 = getImgFinger(file2);
        } catch (Exception e){

        }
        // 获取两个图的汉明距离（假设另一个图也已经按上面步骤得到灰度比
较数组）
        int hammingDistance = getHammingDistance(pixels1, pixels2);
        // 通过汉明距离计算相似度，取值范围 [0.0, 1.0]
        double similarity = calSimilarity(hammingDistance)*100;
        return similarity;
    }


    private static int[] getImgFinger(File imageFile) throws IOException {
        Image image = ImageIO.read(imageFile);
        // 转换至灰度
        image = toGrayscale(image);
        // 缩小成32x32的缩略图
        image = scale(image);
        // 获取灰度像素数组
        int[] pixels1 = getPixels(image);
        // 获取平均灰度颜色
        int averageColor = getAverageOfPixelArray(pixels1);
        // 获取灰度像素的比较数组（即图像指纹序列）
        pixels1 = getPixelDeviateWeightsArray(pixels1, averageColor);
        return pixels1;
    }

    // 将任意Image类型图像转换为BufferedImage类型，方便后续操作
    public static BufferedImage convertToBufferedFrom(Image srcImage)
{
        BufferedImage bufferedImage = new
BufferedImage(srcImage.getWidth(null),
                srcImage.getHeight(null), BufferedImage.TYPE_INT_ARGB);
        Graphics2D g = bufferedImage.createGraphics();
        g.drawImage(srcImage, null, null);
        g.dispose();
        return bufferedImage;
    }
```

```
    // 转换至灰度图
    public static BufferedImage toGrayscale(Image image) {
        BufferedImage sourceBuffered = convertToBufferedFrom(image);
        ColorSpace cs = ColorSpace.getInstance(ColorSpace.CS_GRAY);
        ColorConvertOp op = new ColorConvertOp(cs, null);
        BufferedImage grayBuffered = op.filter(sourceBuffered, null);
        return grayBuffered;
    }

    // 缩放至32x32像素缩略图
    public static Image scale(Image image) {
        image = image.getScaledInstance(32, 32, Image.SCALE_SMOOTH);
        return image;
    }

    // 获取像素数组
    public static int[] getPixels(Image image) {
        int width = image.getWidth(null);
        int height = image.getHeight(null);
        int[] pixels = convertToBufferedFrom(image).getRGB(0, 0, width,
height,
                null, 0, width);
        return pixels;
    }

    // 获取灰度图的平均像素颜色值
    public static int getAverageOfPixelArray(int[] pixels) {
        Color color;
        long sumRed = 0;
        for (int i = 0; i < pixels.length; i++) {
            color = new Color(pixels[i], true);
            sumRed += color.getRed();
        }
        int averageRed = (int) (sumRed / pixels.length);
        return averageRed;
    }

    // 获取灰度图的像素比较数组（平均值的离差）
    public static int[] getPixelDeviateWeightsArray(int[] pixels,final int
averageColor) {
        Color color;
        int[] dest = new int[pixels.length];
        for (int i = 0; i < pixels.length; i++) {
            color = new Color(pixels[i], true);
            dest[i] = color.getRed() - averageColor > 0 ? 1 : 0;
        }
        return dest;
    }
```

```
    // 获取两个缩略图的平均像素比较数组的汉明距离（距离越大差异越大）
    public static int getHammingDistance(int[] a, int[] b) {
        int sum = 0;
        for (int i = 0; i < a.length; i++) {
            sum += a[i] == b[i] ? 0 : 1;
        }
        return sum;
    }

    // 通过汉明距离计算相似度
    public static double calSimilarity(int hammingDistance){
        int length = 32*32;
        double similarity = (length - hammingDistance) / (double) length;

        // 使用指数曲线调整相似度结果
        similarity = java.lang.Math.pow(similarity, 2);
        return similarity;

    }
}
```

测试

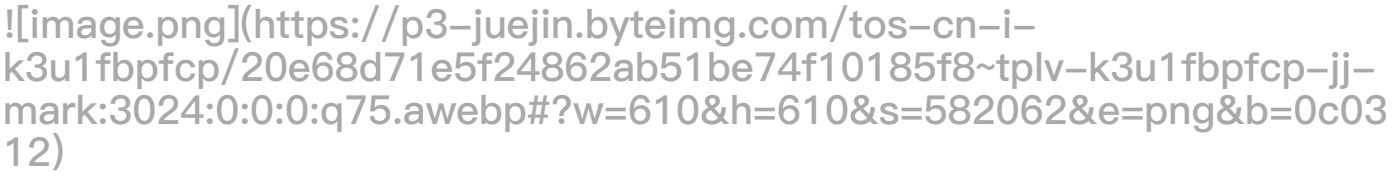
图一：

图二：

图三：

k3u1fbpfcp/cc5bccb733f3404399aed93ab3f535a3~tplv-k3u1fbpfcp-jj-mark:3024:0:0:0:q75.awebp#?w=500&h=500&s=310087&e=png&b=97b5c9)

图四：



结果：

```
File file1 = FileUtil.writeBytes(HttpUtil.downloadBytes("https://avatar3.pddpic.com/a/Q0NVTkdwcHo1STY4cXRNWWRvRFZQVHF2MkpYRUpINWlDQT09djA0-1650468196?imageMogr2/thumbnail/100x"), "image1.jpg");
File file2 = FileUtil.writeBytes(HttpUtil.downloadBytes("https://wx.qlogo.cn/mmhead/ver_1/uO9KDkCtJWHkheqGuqxw1h92yGdgd4kmHnl77z1oxa9j0rLvb8NX35bDcpzn5guraX9ia8Pt1z1C0sOu2sTkicF9D9Krf7As0oxhLR9FetX0M/0"), "image2.jpg");

// 获取图像
double similarity = getSimilarity(file1, file2);
System.out.println(similarity);

// 图一图二相似度：98.63748550415039
// 图一图三相似度：72.01776504516602
// 图一图四相似度：32.860660552978516
```

耗时：
0.261秒

原文链接: https://juejin.cn/post/7355826327357669402