基于 JAVA 平台的图像傅里叶变换与离散余弦变换程序的设计与实现

付 豪，ZY1601119

摘 要：本文基于 Java 平台实现了图像的傅里叶变换与余弦变换程序的设计与编程。该程序具有打开一副图像，进行傅里叶变换与余弦变换的功能。本文重点描述了傅里叶变换与余弦变换的原理与源码设计。

1 程序设计与实现

1.1 整体架构设计

本文利用Java进行了图像处理程序的实现。目的在于实现打开一副图像，进行直方图均衡，将灰度线性变换，将灰度拉伸。程序目录结构如图1所示。其中程序源码各类功能如下：

Complex：复数类；

DiscreteCosTransformer：进行离散余弦变换；

FFT：进行一维傅里叶变换；

FourierTransformer：进行二维傅里叶变换；

ImageProcessing：提供用户界面，进行输入输出；

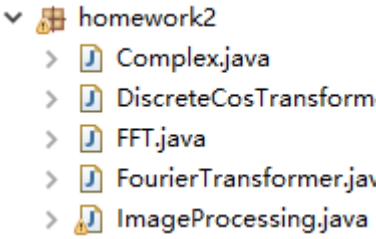

图 1 程序的目录结构

1.2 图片读取原理

本程序由文件系统读取图片的方法readFile()如图2（a）所示，利用的库如图2（b）所示。

(a)
```
vate static void readFile() throws IOException {
    String filePath;
    System.out.println("请输入文件路径inFile:");
    Scanner in = new Scanner(System.in);
    filePath = in.nextLine();
    File input = new File(filePath);
    image = ImageIO.read(input);
}
```

(b)
```
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;
import java.util.Scanner;
```

图 2 图片读取的实现

（a）读取图片的源代码实现；（b）读取图片利用的库

如图2（a）所示，本程序图片读取的实现主要利用了Java的ImageIO的read方法，其首先会从当前已注册的ImageReader中自动选择一个去解码输入的File，之后会返回形成的BufferedImage类型的对象。该输入的File包装在ImageInputStream中。

1.2 傅里叶变换原理

本程序的傅里叶变换主要由FourierTransformer类、Complex类、FFT类实现。FourierTransformer类为主调类，Complex类为复数类，FFT类实现一维快速傅里叶变换。

FourierTransformer类在读入图像后，首先计算进行傅里叶变换的宽度和高度（2的整数次方），分配内存，进行初始化；之后依次在Y方向、X方向调用FFT类进行一维快速傅里叶变换；再后将图像看做二维函数，图像灰度值为函数在相应XY处的函数值，对其进行二维快速傅里叶变换，得到一个复数矩阵，将此矩阵水平循环移动半宽，垂直循环移动半高；最后返回傅里叶变换后图像。其中，使用的FFT类中进行一维快速傅里叶变换的源码如图3所示。

```
public static Complex[] fft(Complex[] x) {
    int N = x.length;
    if (N == 1) {
        return new Complex[] { x[0] };
    Complex[] even = new Complex[N / 2];
    for (int k = 0; k < N / 2; k++) {
        even[k] = x[2 * k];
    }
    Complex[] q = fft(even);
    Complex[] odd = even;
    for (int k = 0; k < N / 2; k++) {
        odd[k] = x[2 * k + 1];
    }
    Complex[] r = fft(odd);
    Complex[] y = new Complex[N];
    for (int k = 0; k < N / 2; k++) {
        double kth = -2 * k * Math.PI / N;
        Complex wk = new Complex(Math.cos(kth), Math.sin(kth));
        y[k] = q[k].plus(wk.times(r[k]));
        y[k + N / 2] = q[k].minus(wk.times(r[k]));
    }
    return y;
}
```

图 3 一维快速傅里叶变换的源码实现

如图3所示，二维傅里叶变换实现的公式如下式：

$$F(k_1,k_2)=\frac{1}{N}\sum_{n_1=0}^{N-1}\sum_{n_2=0}^{N-1}f(n_1,n_2)e^{-j\frac{2\pi}{N}(k_1n_1+k_2n_2)}$$

1.3 离散余弦变换原理

本程序的离散余弦变换主要由DiscreteCosTransformer类实现。

FourierTransformer类在读入图像后首先利用coefficient()方法求取离散余弦变换系数矩阵；之后利用transposingMatrix()方法对系数矩阵进行转置；再后利用matrixMultiply()方法先将系数矩阵与原图像矩阵相乘再将结果与转置后的系数矩阵

相乘获得离散余弦变换后的结果矩阵；最后返回离散余弦变换后的图像。其源码如图4所示。

```
private void DCT() throws InterruptedException {
    int n = iw;
    PixelGrabber pg = new PixelGrabber(im, 0, 0, iw, ih, pixels, 0, iw);
    pg.grabPixels();

    double[][] iMatrix = new double[n][n];
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            iMatrix[i][j] = (double) (pixels[i * n + j]);
        }
    }
    double[][] quotient = coefficient(n); // 求系数矩阵
    double[][] quotientT = transposingMatrix(quotient, n); // 转置系数矩阵

    double[][] temp = new double[n][n];
    temp = matrixMultiply(quotient, iMatrix, n);
    iMatrix = matrixMultiply(temp, quotientT, n);


    int newpix[] = new int[n * n];
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            newpix[i * n + j] = (int) iMatrix[i][j];
        }
    }
    imageAuth = new BufferedImage(n, n, BufferedImage.TYPE_BYTE_GRAY);
    ColorModel colorModel = imageAuth.getColorModel();
    WritableRaster raster = colorModel.createCompatibleWritableRaster(n, n);
    raster.setPixels(0, 0, n, n, newpix);
    imageAuth.setData(raster);
}
```

图4 离散余弦变换的源码实现

如图4所示，离散余弦变换实现的公式如下式：

$$F(u,v) = \frac{2}{N}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y)\cos[\frac{(2x+1)}{2N}u\pi]\cos[\frac{(2y+1)}{2N}v\pi]$$

2 实验结果

2.1 傅里叶变换结果展示

傅里叶变换前后图像如图 5 所示。

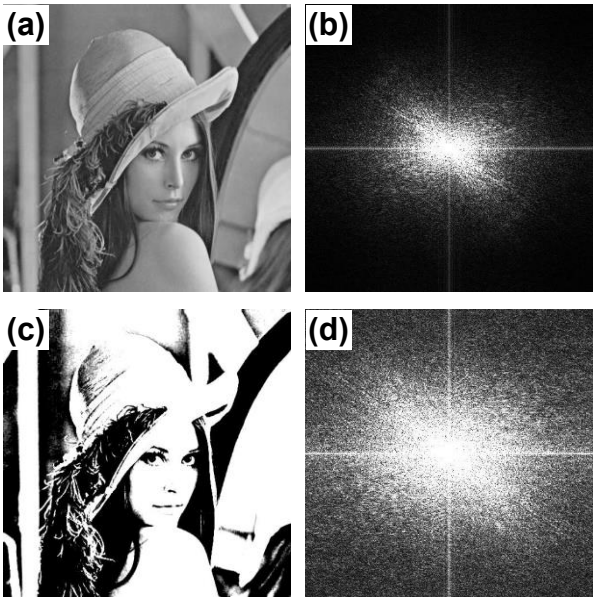

图 5 傅里叶变换结果展示

（a）原图像 1；（b）处理后图像 1；（c）原图像 2；（d）处理后图像 2；

2.2 离散余弦变换结果展示

离散余弦变换前后图像及直方图如图 6 所示。

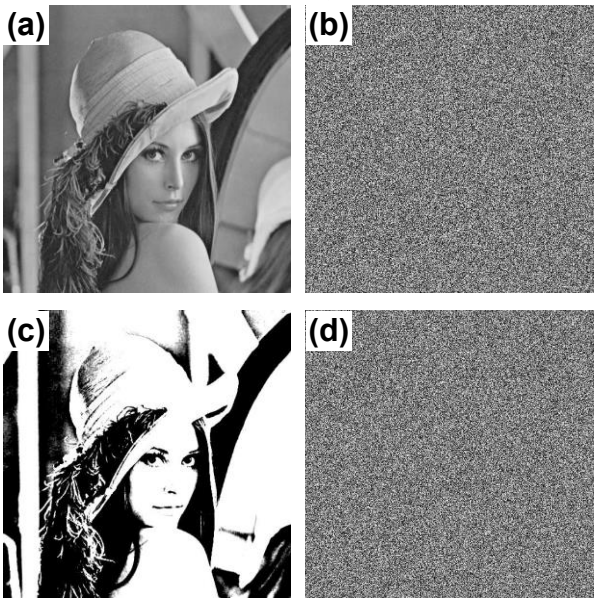

图 6 离散余弦变换结果展示

（a）原图像 1；（b）处理后图像 1；（c）原图像 2；（d）处理后图像 2；

3 结 论

（1）本文基于 Java 平台实现了图像傅里叶变换与离散余弦变换程序的设计与编程，该程序具有打开一副图像，进行傅里叶变换与余弦变换的功能。

（2）本文重点描述了图像傅里叶变换与离散余弦变换的原理与源码设计。