

**数字媒体技术实验报告**

题 目 实验1

学 院 计算学部

专 业 软件工程

学 号 120L022109

学 生 李世轩

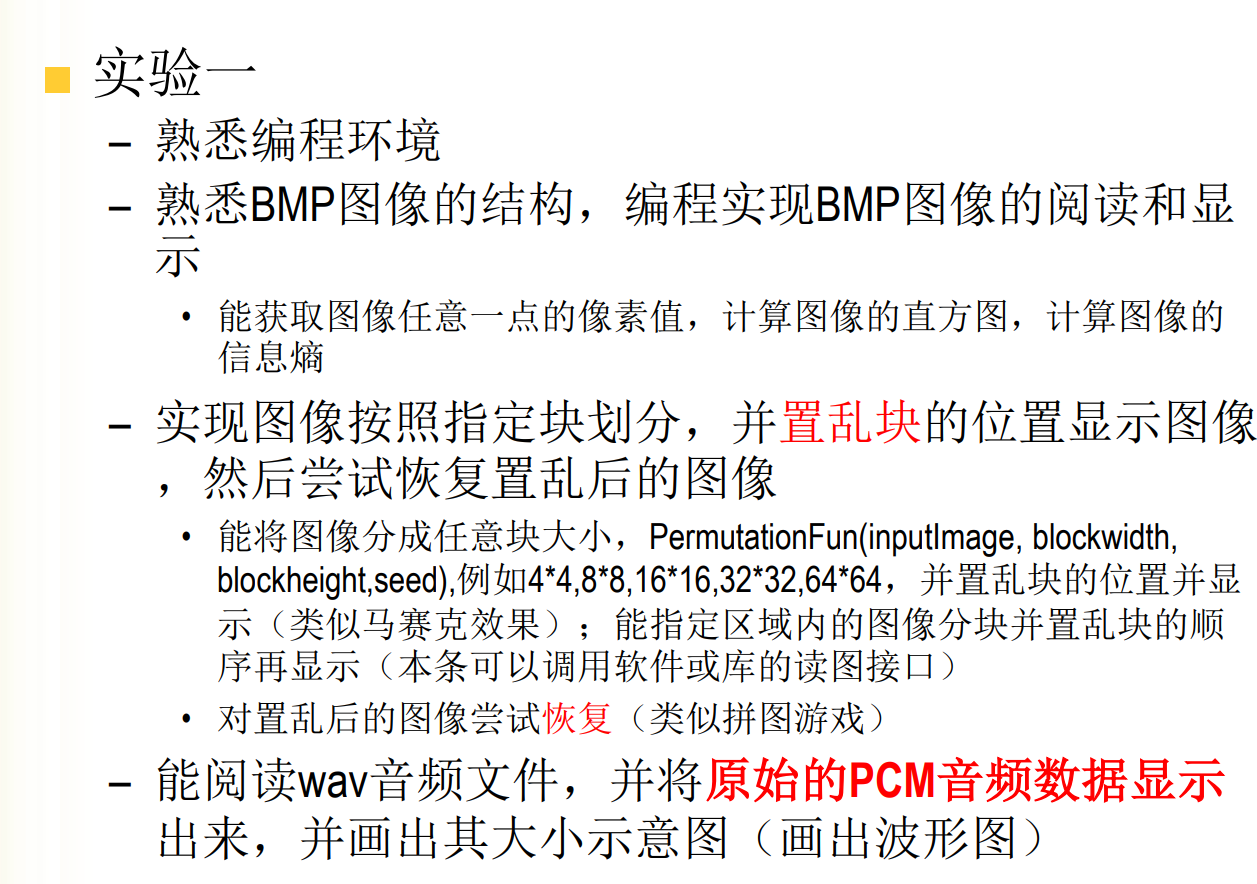
任 课 教 师 刘绍辉

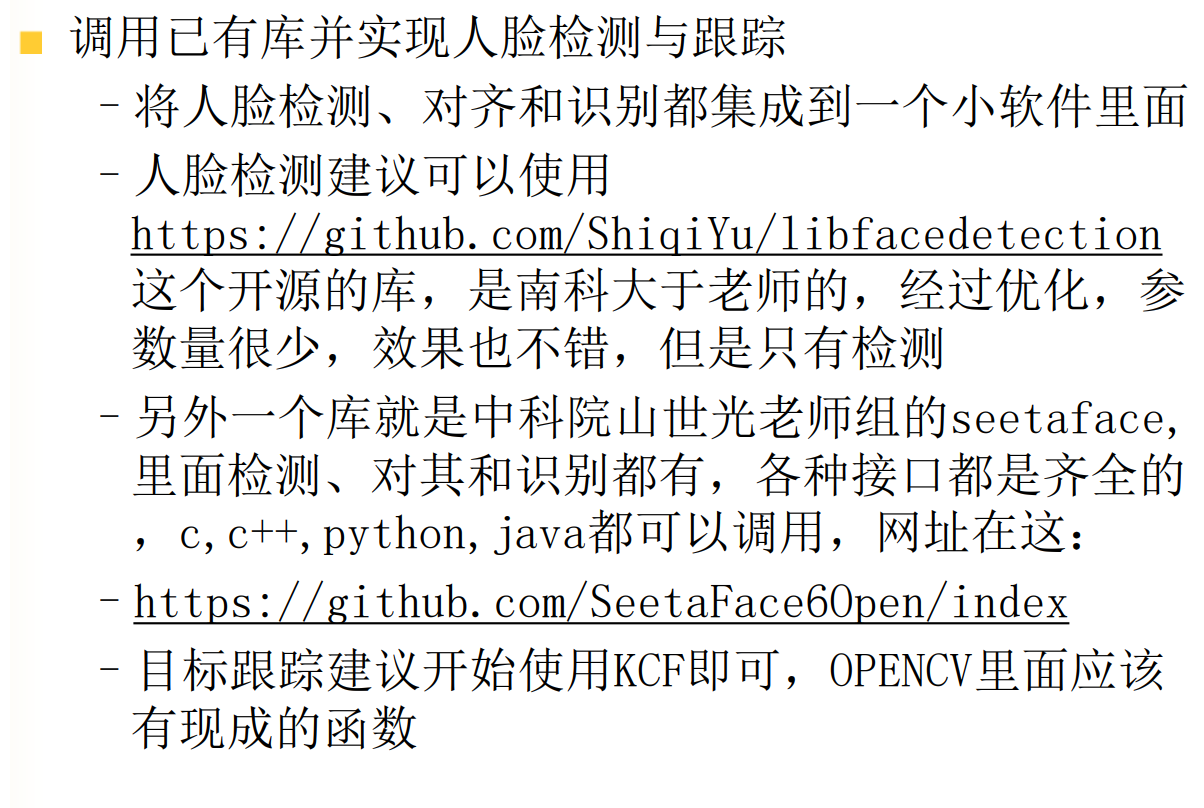
哈尔滨工业大学计算机科学与技术学院

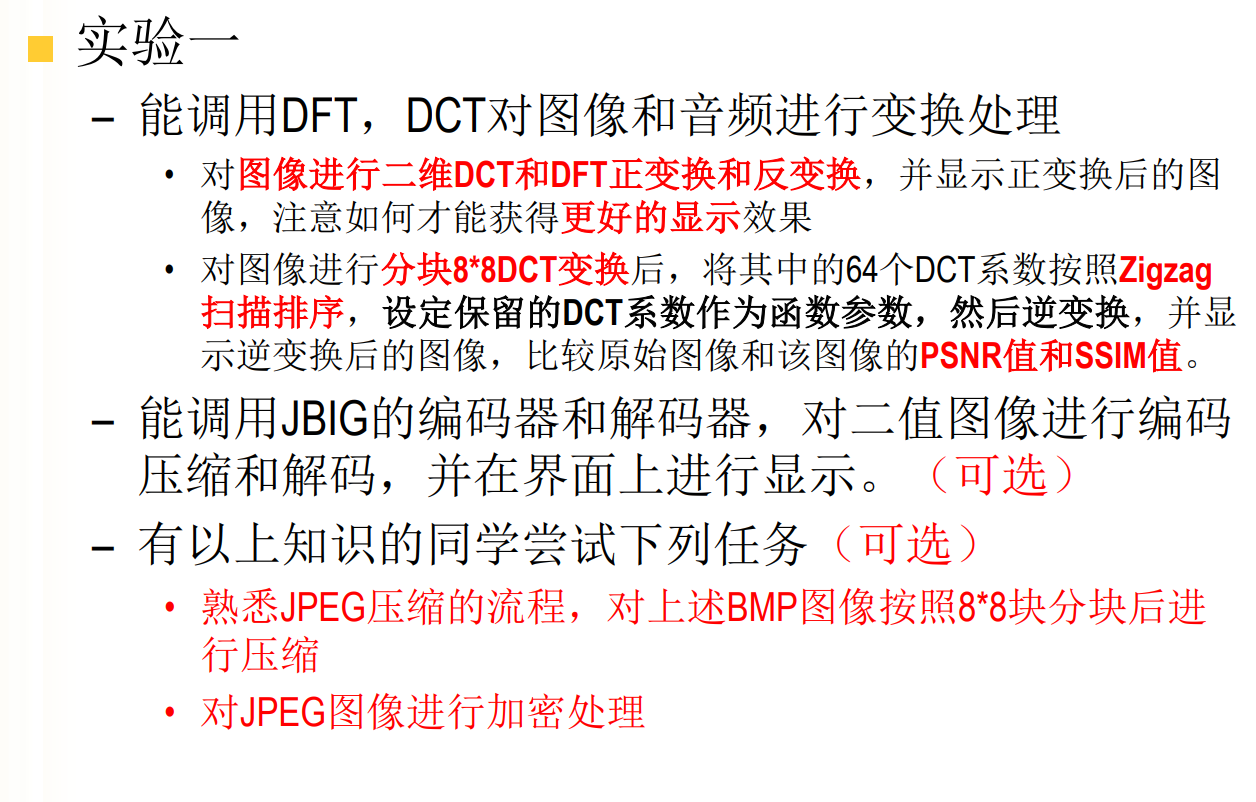
2023.3

**实验1**

1. **实验内容**







1. **实验环境**

操作系统：Windows11专业版 64位

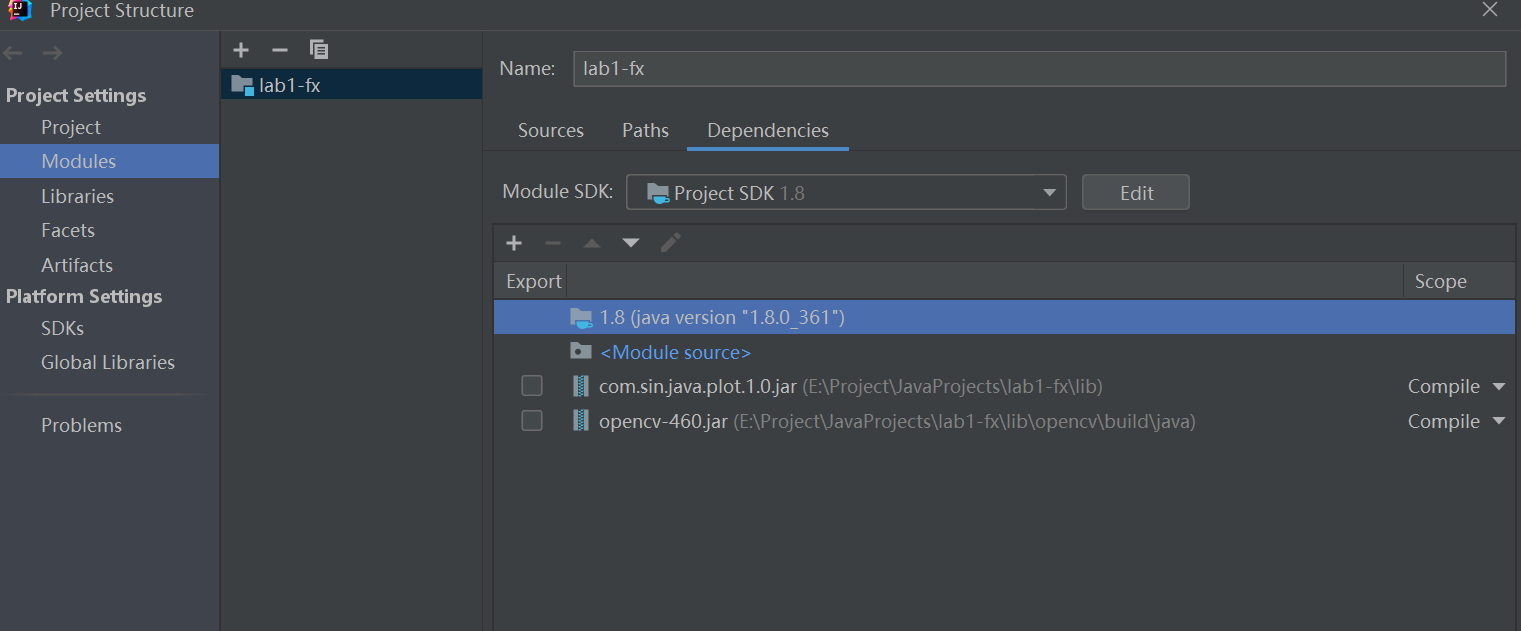
编程语言：java JDK1.8.0\_361

开发环境：IntelliJ IDEA 2022.3.3 (Ultimate Edition)

程序调用库：

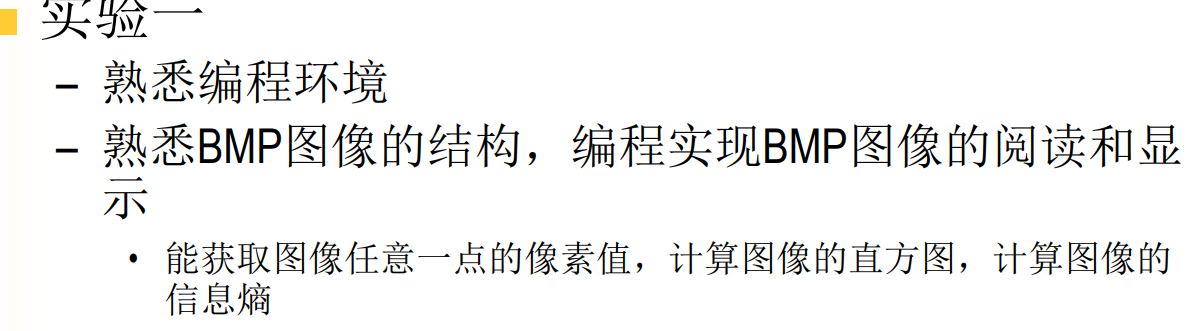
OpenCV 4.6.0

com.sin.java.plot.1.0.jar(一个用于画图的第三方库，基于Java swing开发，地址[北京邮电大学-崔毅东/JavaPlot - 码云 - 开源中国 (gitee.com)](https://gitee.com/bupt-se/JavaPlot/tree/master))

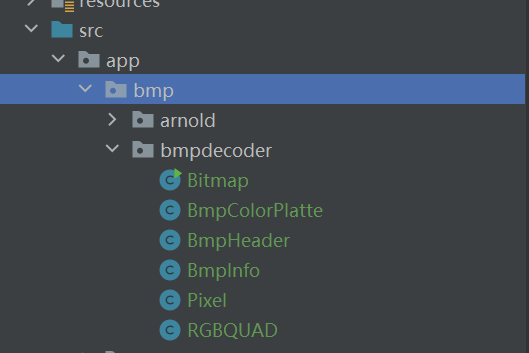


1. **实验步骤及结果分析**

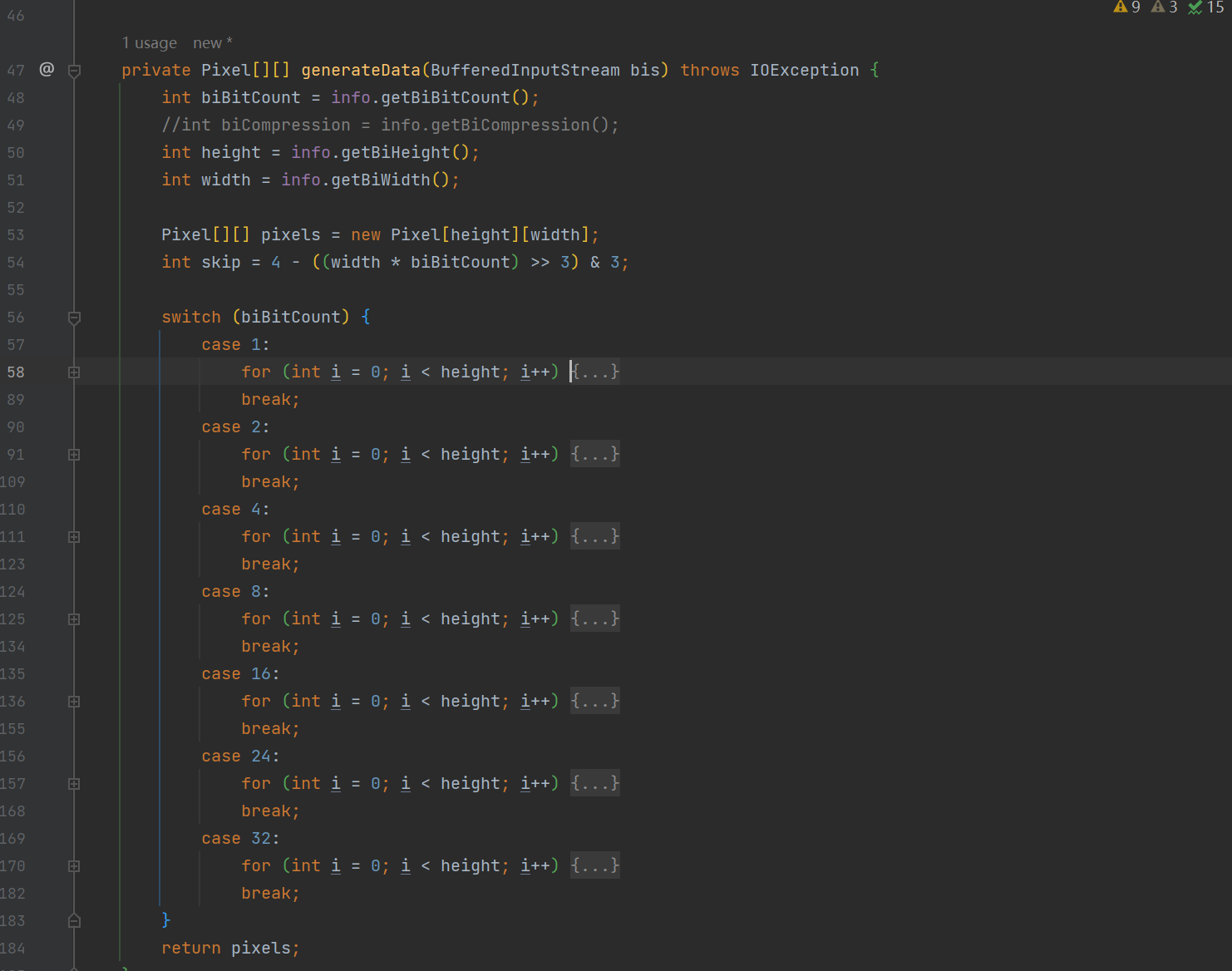
接下来根据不同的小题来分析：



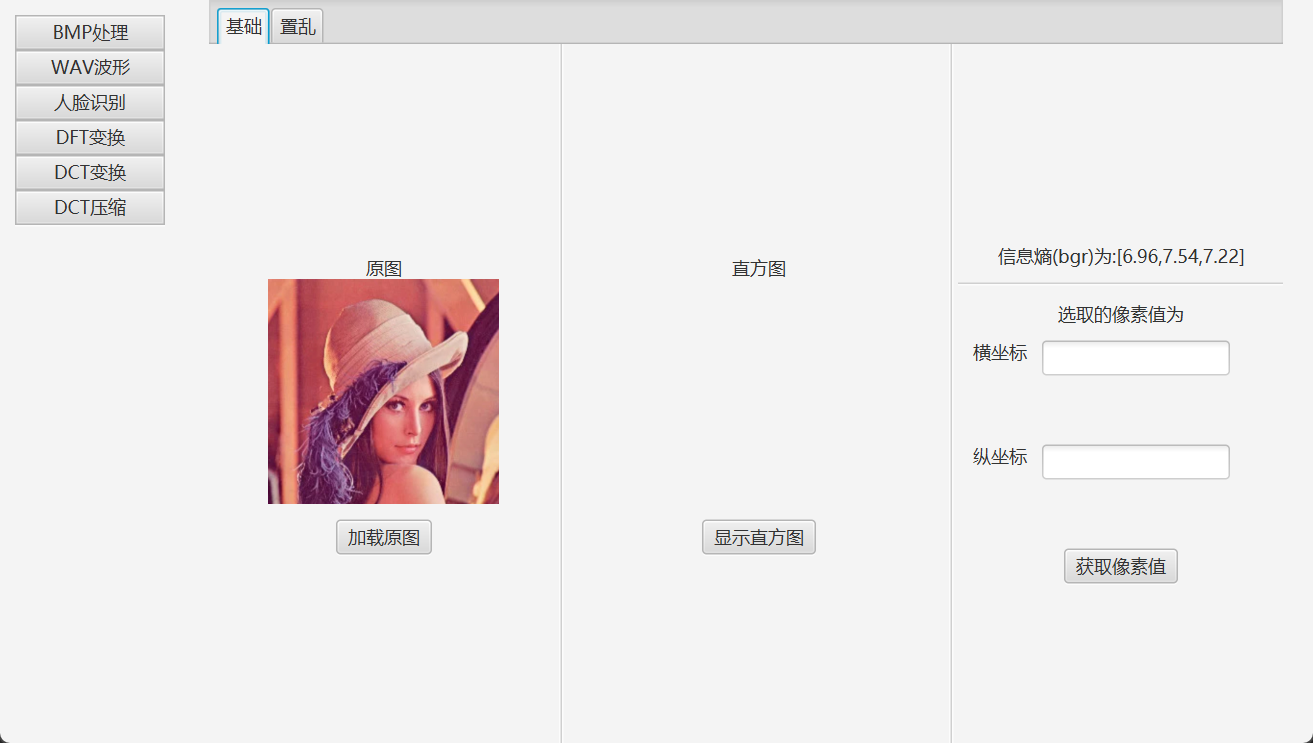
首先需要根据BMP图像的结构来实现Bmp的解码器，这里主要参考博客：BMP格式详解（[BMP格式详解\_bmp头部信息\_Valineliu的博客-CSDN博客](https://blog.csdn.net/u012877472/article/details/50272771)）。根据其结构，构建了一系列相关的类。



其中对于数据的解析功能主要是根据读取到的头部信息针对不同的biBitCount，进行不同的动作。详细内容见源代码



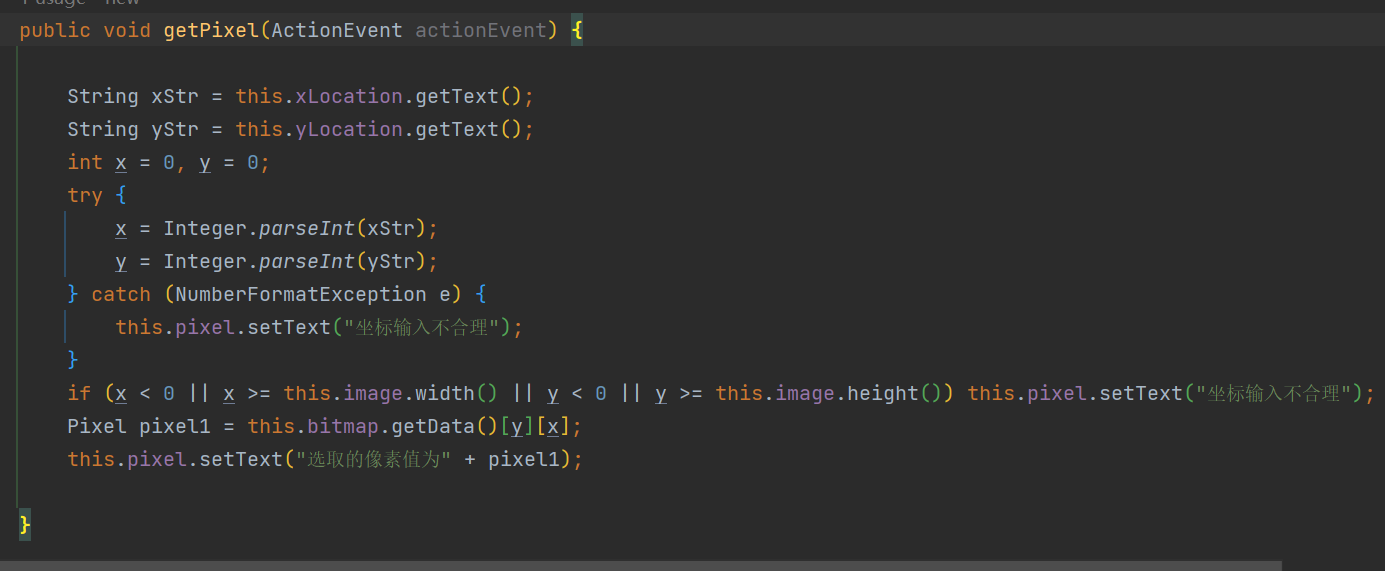
接下来是图像的显示部分，将读入的信息转化成OpenCv的Mat类，并利用javafx的ImageView进行显示。下图是展示文件选择及显示。



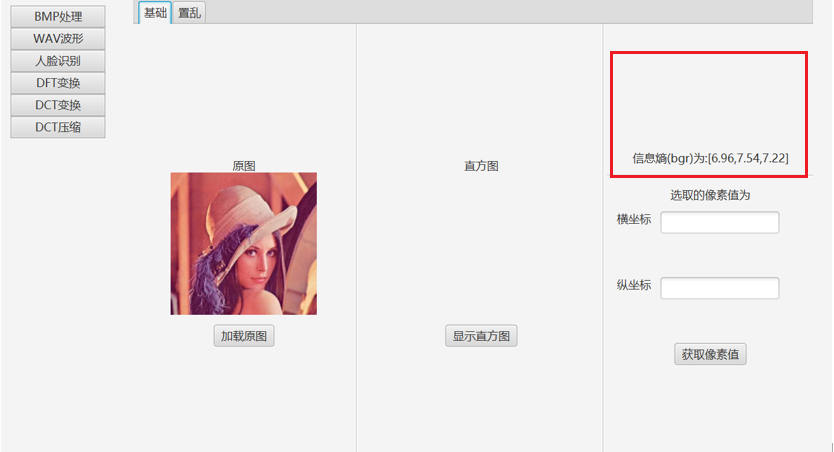
其中逻辑部分见下图



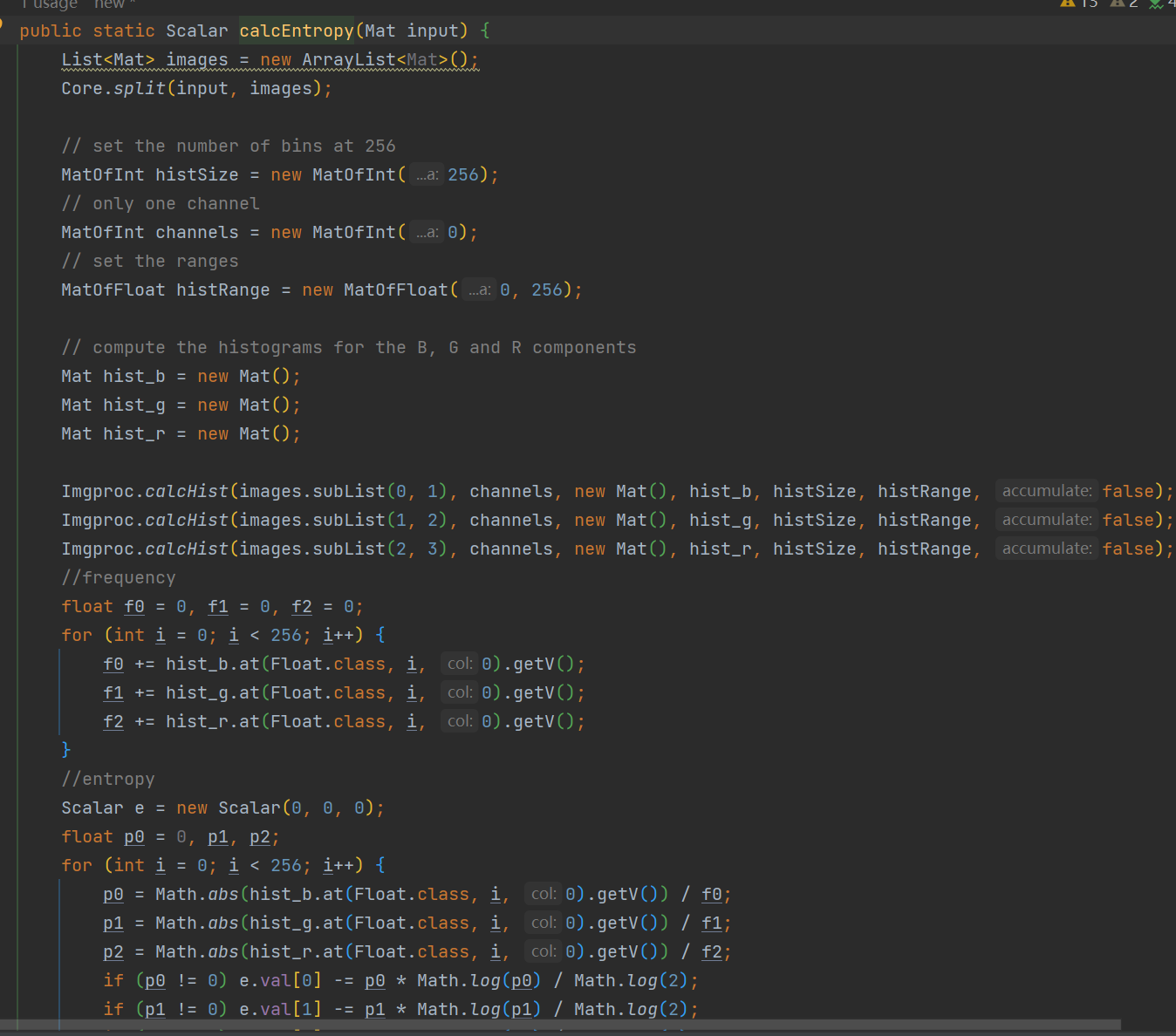
当图像信息已经被读入后，获取像素值也就相对容易了。



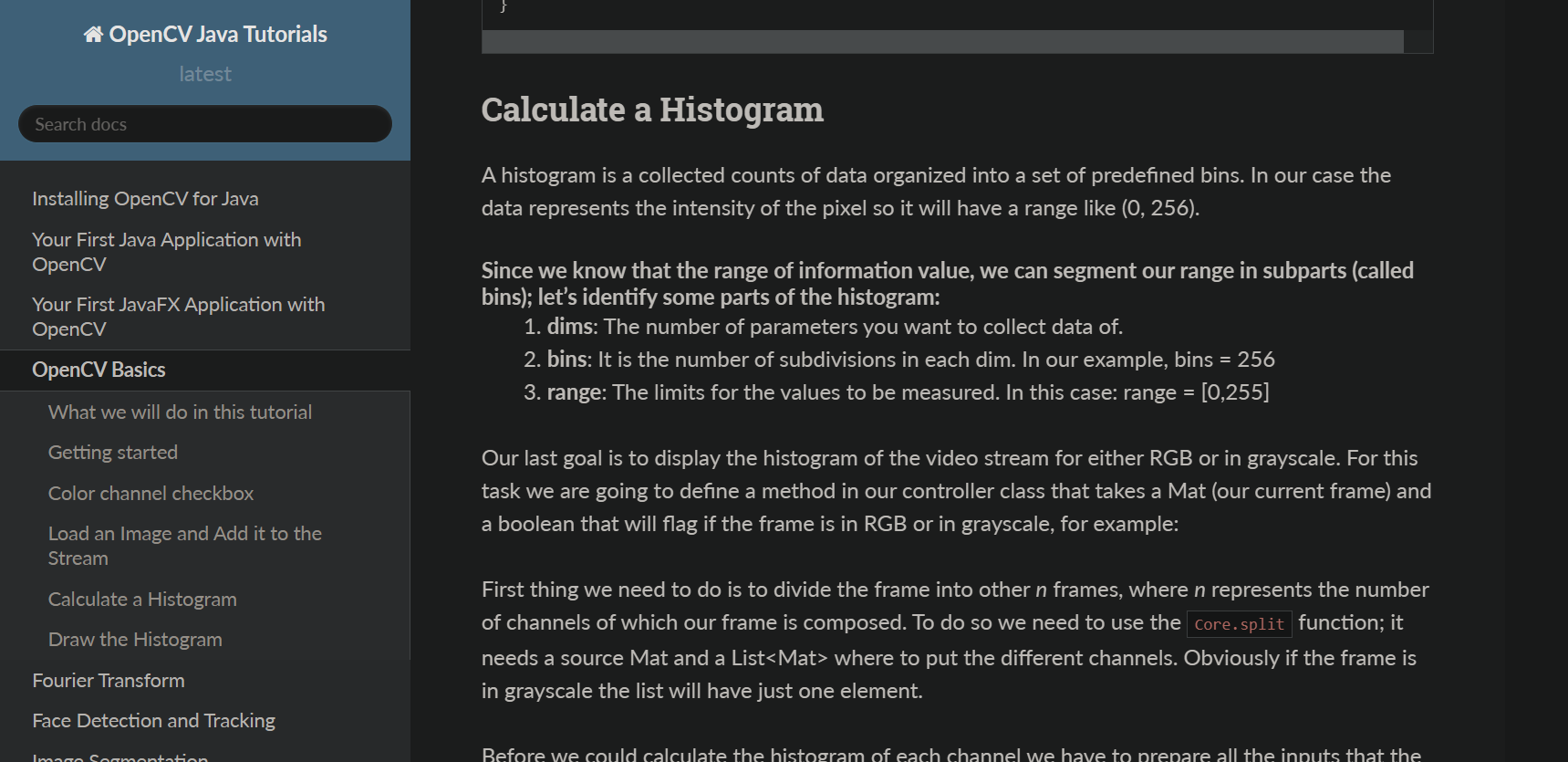
关于图像信息熵，这里直接在加载图片完成后显示在了界面的右上角，这里分rgb三个通道进行显示。如下图



信息熵的计算算法如图，主要参考博客：【学习日记】OpenCV-C++ 计算图像的熵值（[【学习日记】OpenCV-C++ 计算图像的熵值\_RainyMacondo的博客-CSDN博客](https://blog.csdn.net/weixin_44211644/article/details/128007758)）。

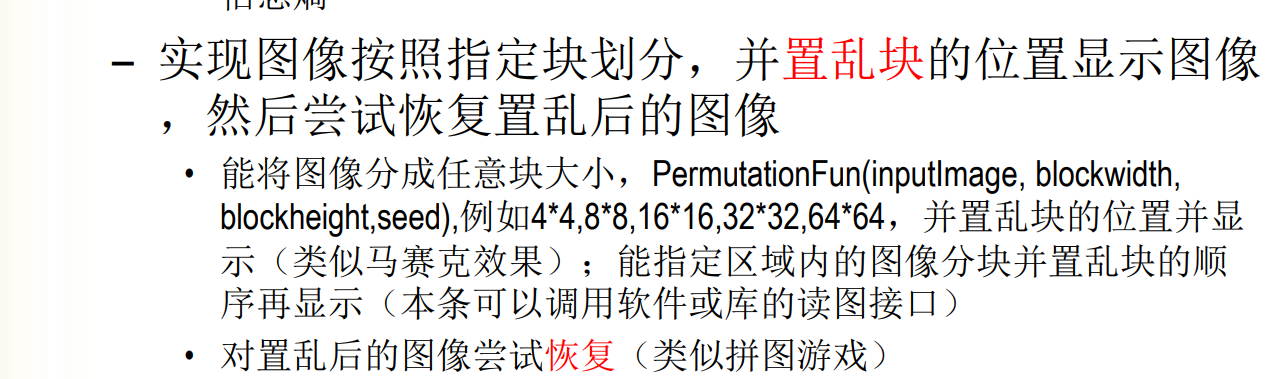


关于图像直方图的显示及计算，这里参考了opencv的官方教程：[OpenCV Basics — OpenCV Java Tutorials 1.0 documentation (opencv-java-tutorials.readthedocs.io)](https://opencv-java-tutorials.readthedocs.io/en/latest/04-opencv-basics.html)，这里不再赘述，只展示成果。





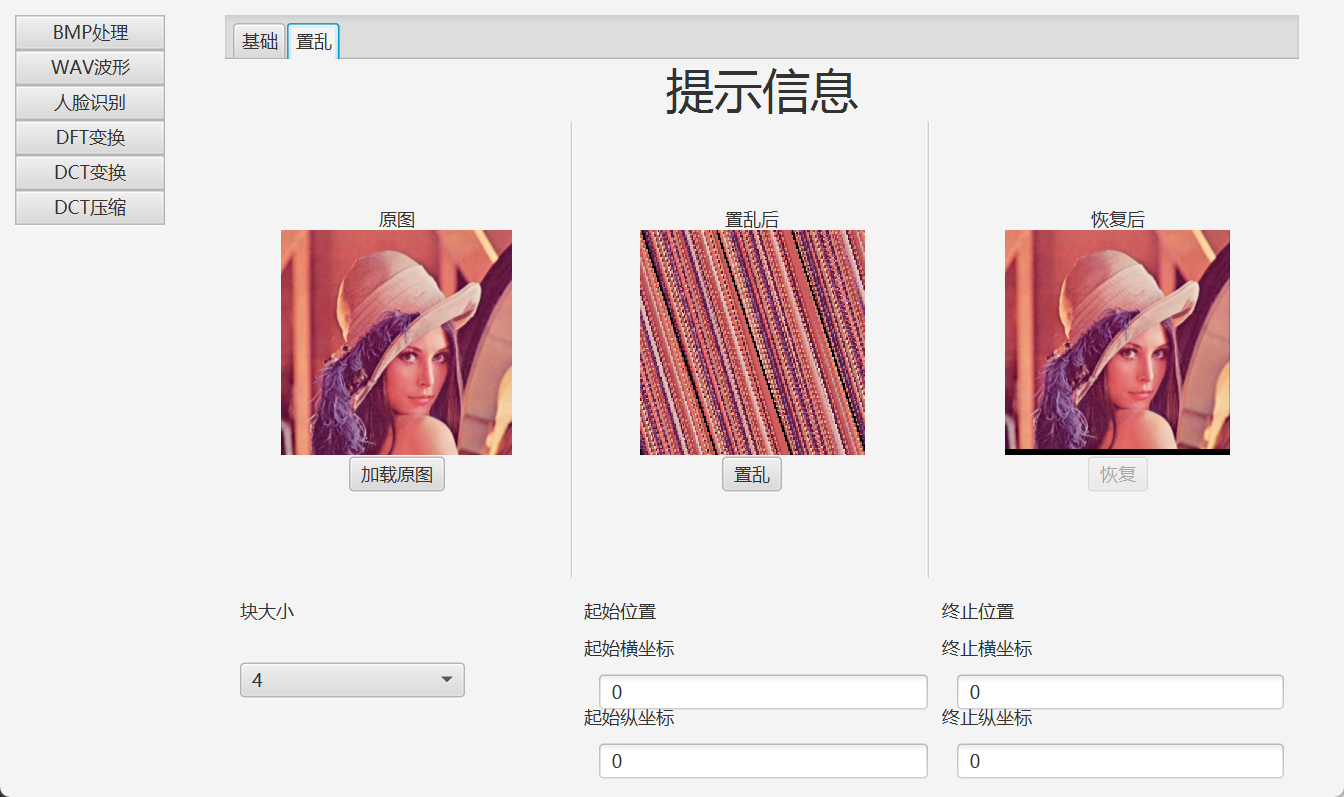
接下来展示下一部分。



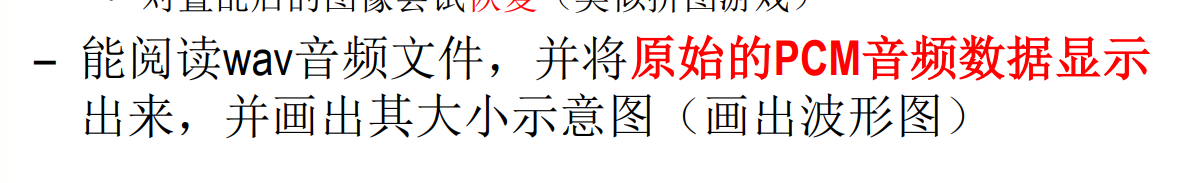
图像分块置乱及恢复使用Arnold算法，主要参考博客：图像分块与置乱(Arnold）（[图像分块与置乱(Arnold）\_图像块置乱\_谷粱幽梦的博客-CSDN博客](https://blog.csdn.net/weixin_44367406/article/details/116024279)）



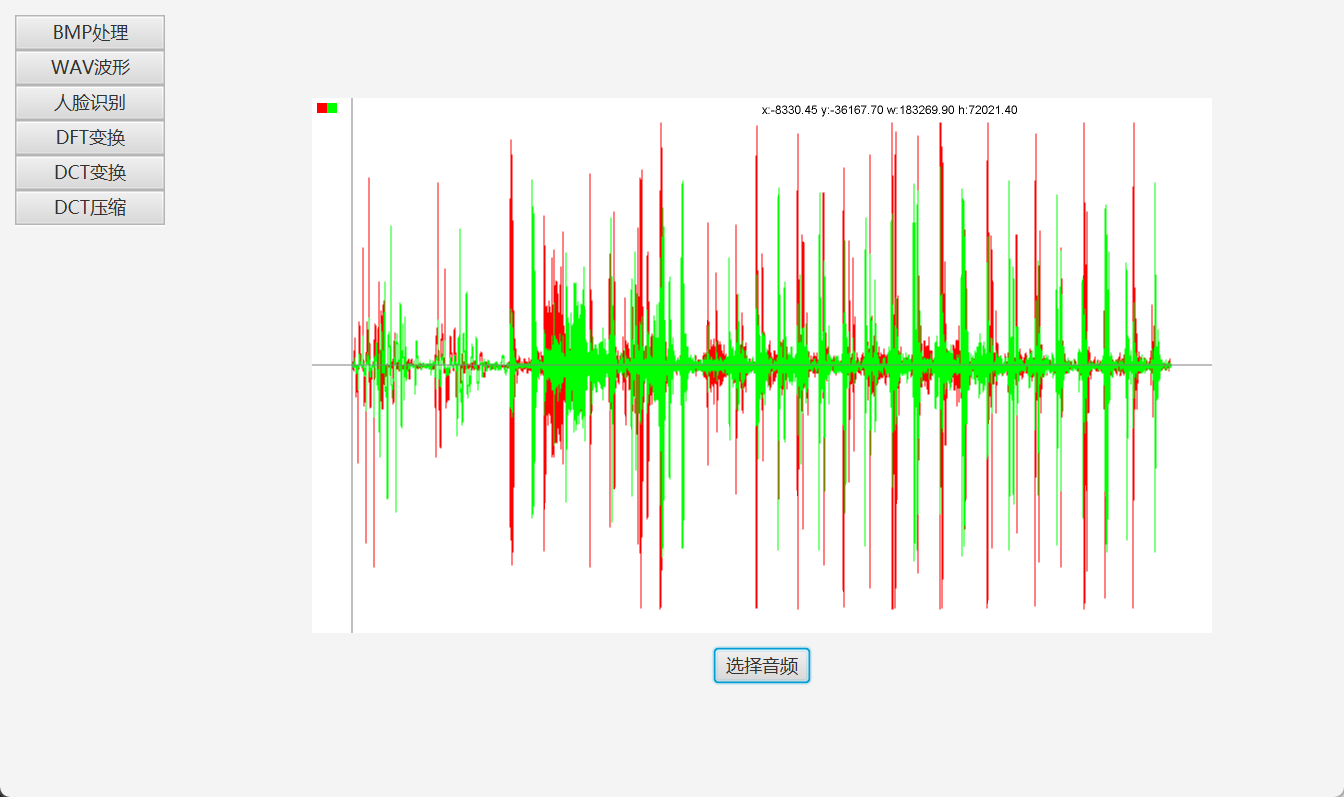
这里展示分块逻辑，当起始坐标与终止坐标的差值不足一个分块大小时，会对全图进行分块置乱，当横坐标差值与纵坐标差值（对块大小求余并向下取整后）不一致时，会按照较小的一方对坐标进行重置，这是因为Arnold本身的限制。



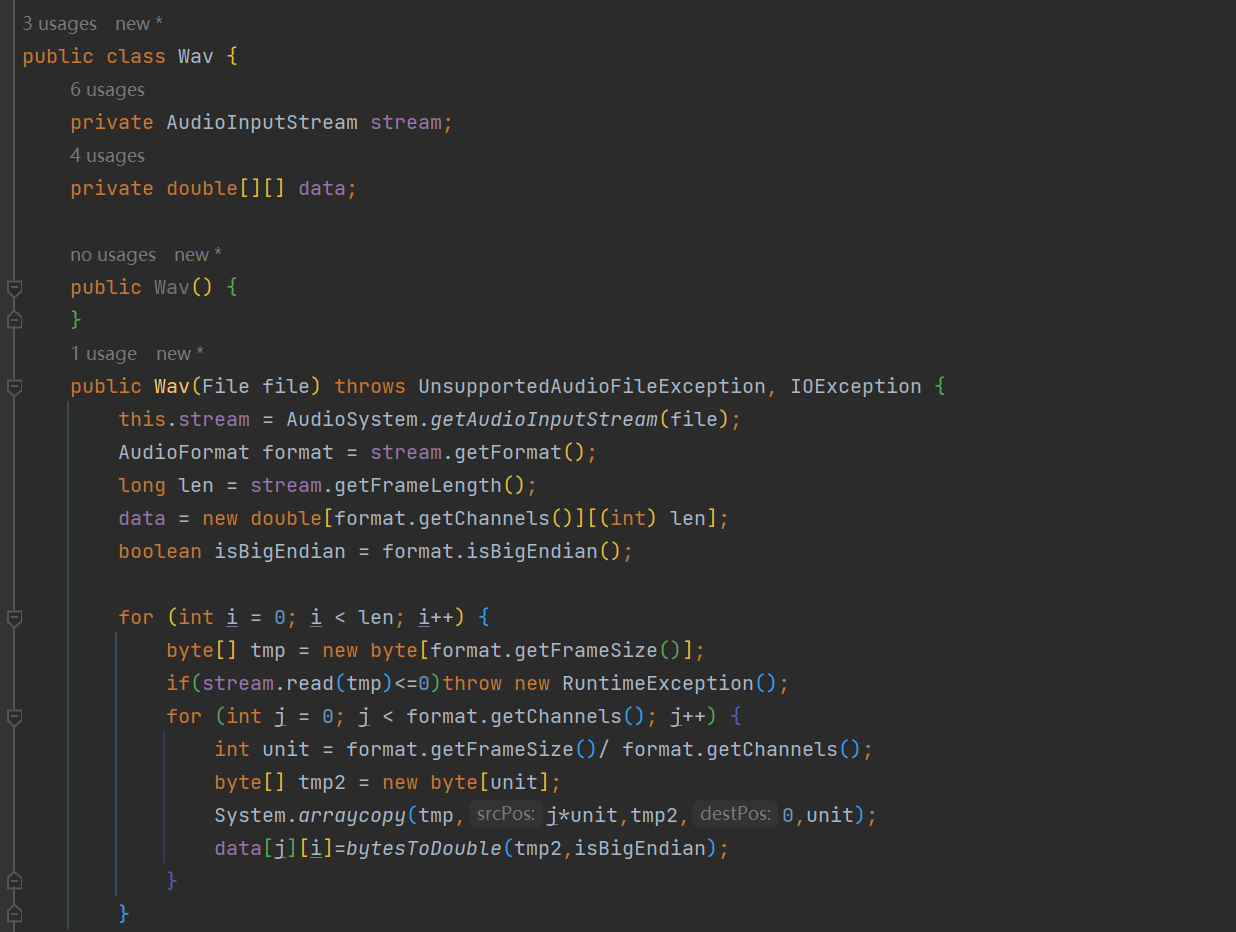
接下来展示下一部分。



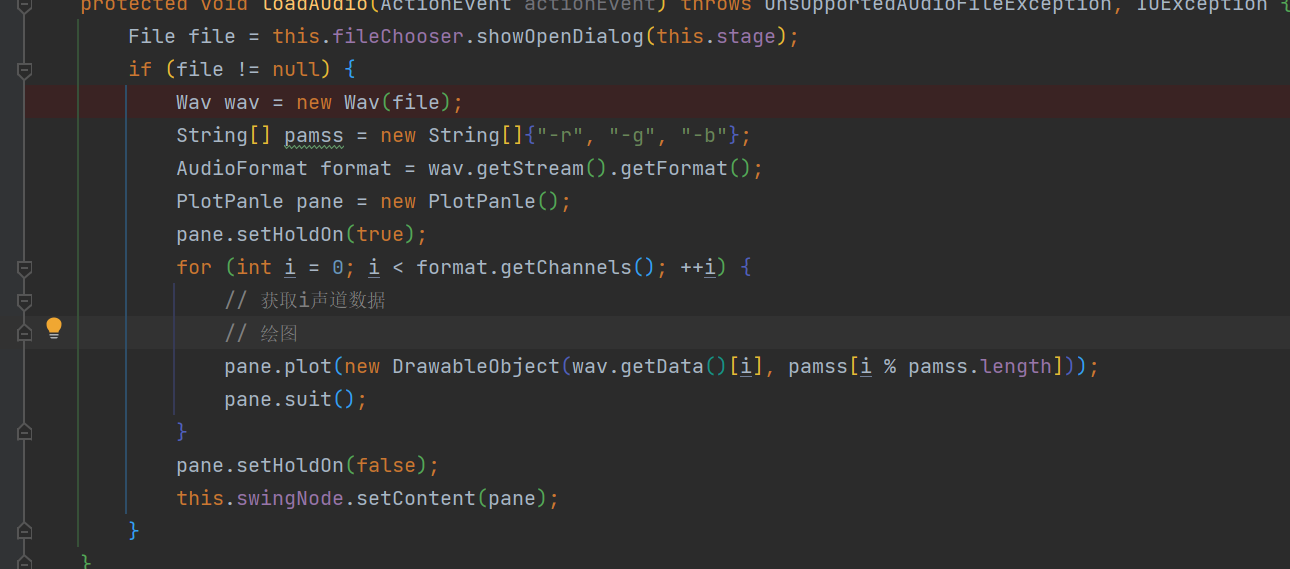
这里先展示成果



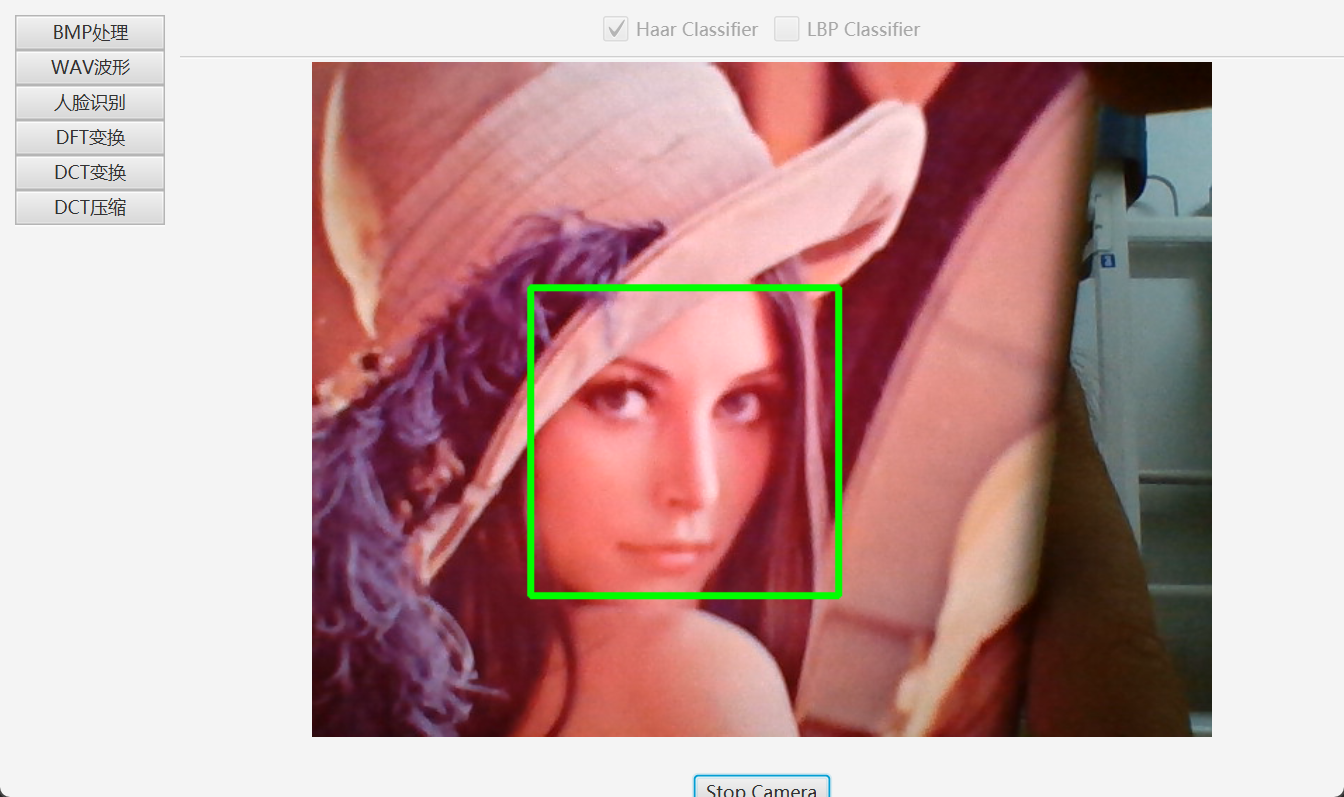
接下来介绍其中的一些实现，首先是对wav文件的解码，这里使用jdk中的音频输入流AudioInputStream，并使用根据其中的信息读出wav文件中的pcm信息。这里需要注意的是audioInputStream的读入是以frame为单位的，所以当读入缓冲区大小小于frameSize时，将无法读入，当音频有多个声道时，应当将读入的frame相应的分成多部分。



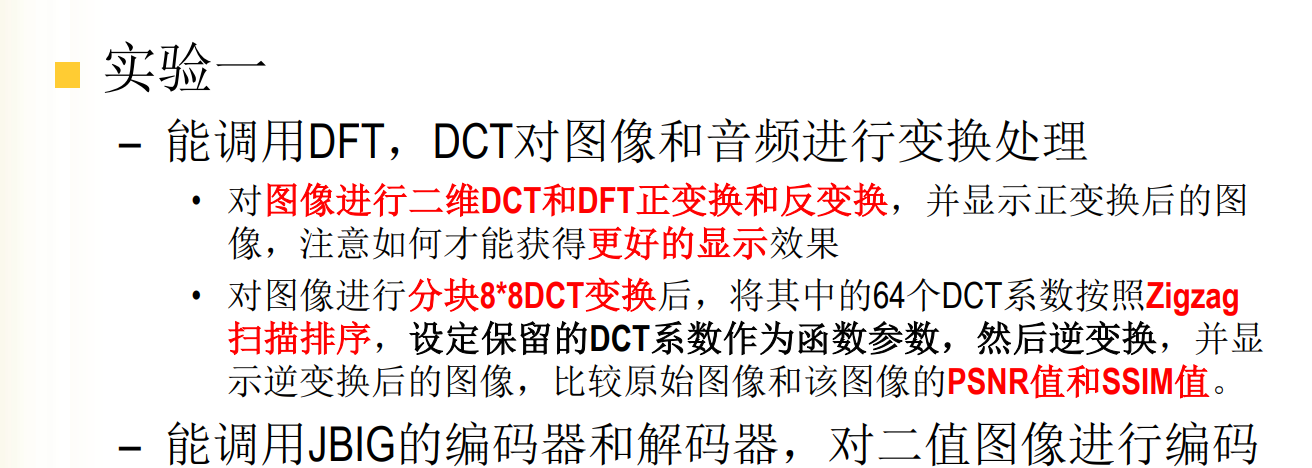
至于画图部分使用com.sin.java.plot.1.0.jar中的一些工具类。



接下来展示人脸识别与追踪部分。这里主要使用opencv中已经训练好的模型，参考[Face Detection and Tracking — OpenCV Java Tutorials 1.0 documentation (opencv-java-tutorials.readthedocs.io)](https://opencv-java-tutorials.readthedocs.io/en/latest/06-face-detection-and-tracking.html)，这里不再对其进行赘述。

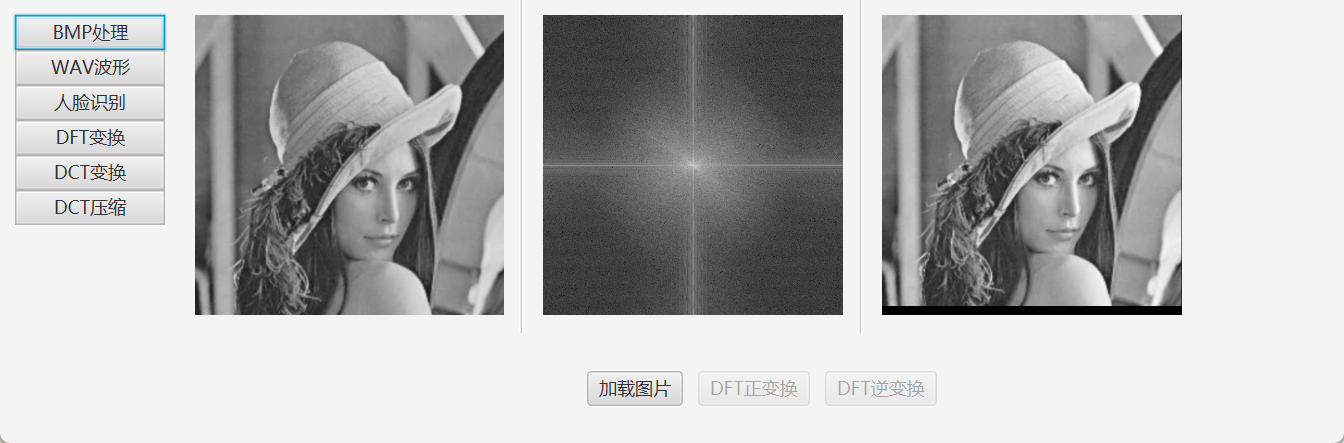


接下来进行下一部分的展示



首先是对图片进行DFT和DCT变换与反变换。

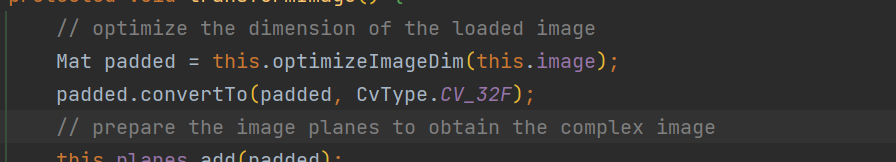
先展示结果如下：

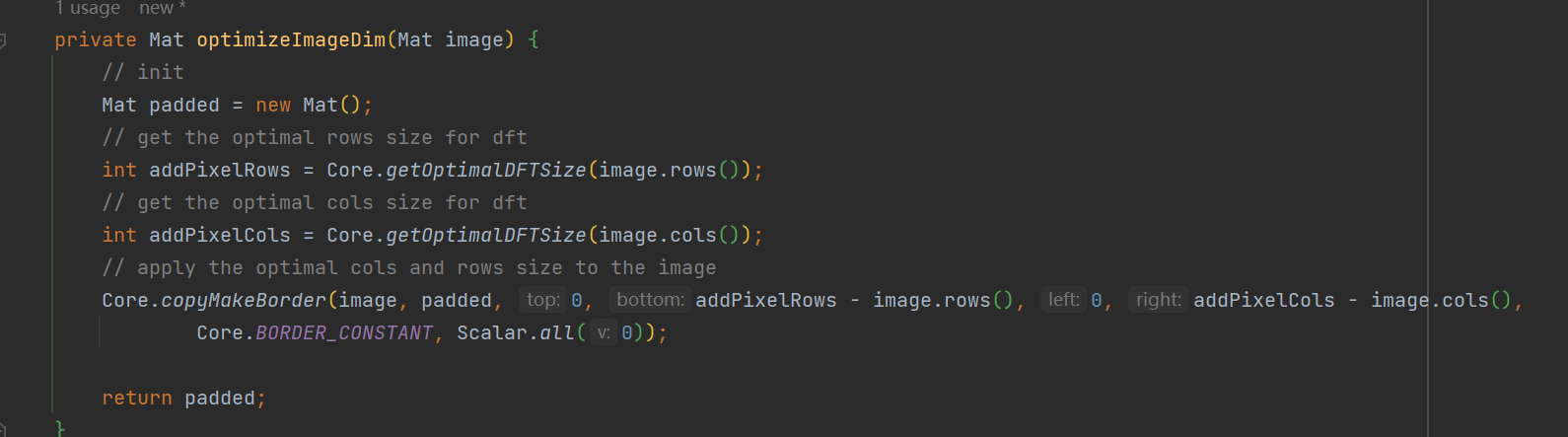




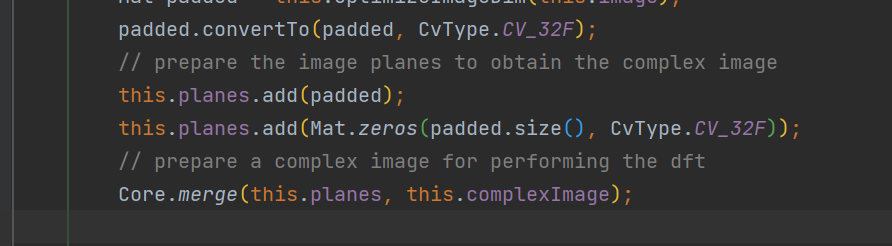
这里DFT变化同样参考opencv官网教程[Fourier Transform — OpenCV Java Tutorials 1.0 documentation (opencv-java-tutorials.readthedocs.io)](https://opencv-java-tutorials.readthedocs.io/en/latest/05-fourier-transform.html)。

在这部分中，首先对图片进行了扩展，使得dft在其上可以有更好的效果。

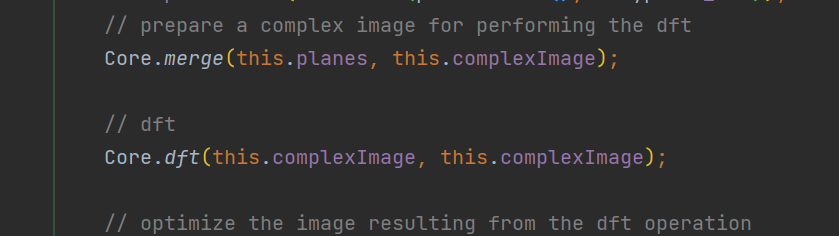




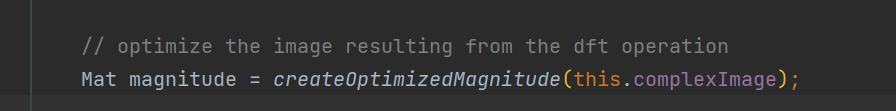
然后DFT的结果很复杂，因此我们必须为复杂值和实际值腾出位置。我们通常至少以浮点格式存储这些。因此，我们将输入图像转换为此类型，并使用另一个通道对其进行扩展以保存复杂值。

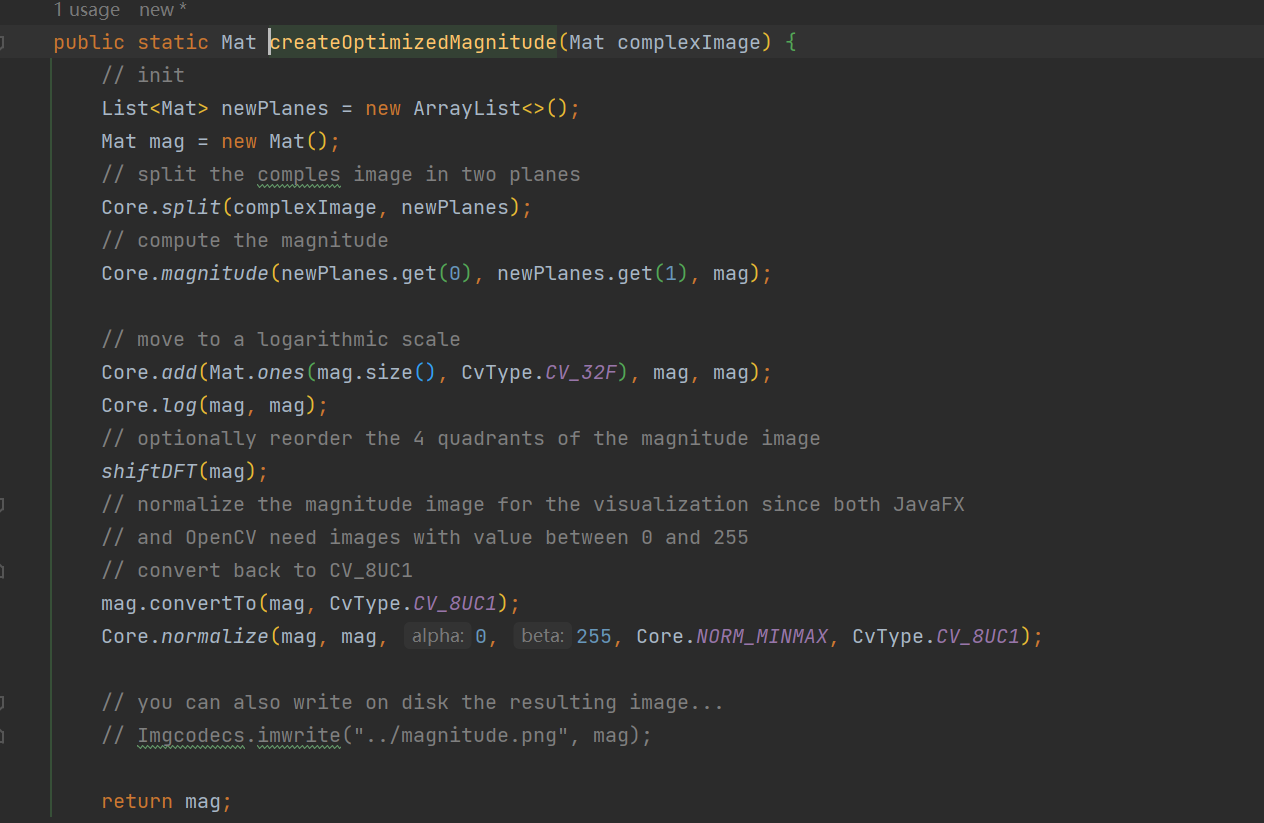


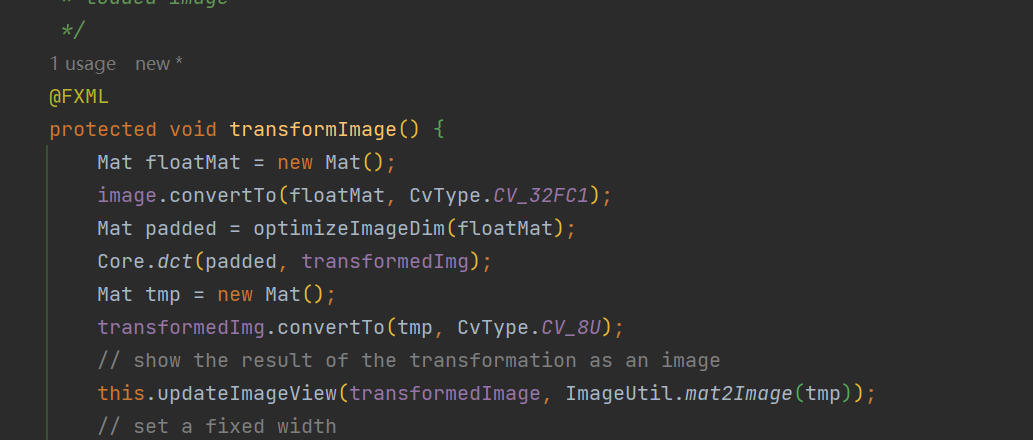
现在我们可以应用 DFT，然后从复杂图像中获取实部和虚部：



不幸的是，傅里叶系数的动态范围太大，无法显示在屏幕上。要使用灰度值进行可视化，我们可以将线性刻度转换为对数刻度：



关于DCT部分，我们没有这么多的工作要做，只要将读入的灰度图转化为CV\_32F再进行dct即可。



关于DCT变换压缩部分，（对图像进行分块8\*8DCT变换后，将其中的64个DCT系数按照Zigzag 扫描排序，设定保留的DCT系数作为函数参数，然后逆变换，并显示逆变换后的图像，比较原始图像和该图像的PSNR值和SSIM值。）

首先需要将图片长宽扩展到8的倍数，然后将其分块，对所有的块进行DCT变换，再对所有块zigzag排序，这里还进行了量化与反量化，之后进行图像重建即可，具体见源码。



然后是对PSNR和SSIM的计算，这里参考[OpenCV: Similarity check (PNSR and SSIM) on the GPU](https://docs.opencv.org/4.3.0/dd/d3d/tutorial_gpu_basics_similarity.html)，原代码采用c++实现，这里使用Java进行重写。具体细节不再赘述。

1. **结论**

完成了一个界面化，可视化的，基本符合实验要求的小的java应用。

1. **参考文献**

[1] [北京邮电大学-崔毅东/JavaPlot - 码云 - 开源中国 (gitee.com)](https://gitee.com/bupt-se/JavaPlot/tree/master)

[2] [BMP格式详解\_bmp头部信息\_Valineliu的博客-CSDN博客](https://blog.csdn.net/u012877472/article/details/50272771)

[3] [【学习日记】OpenCV-C++ 计算图像的熵值\_RainyMacondo的博客-CSDN博客](https://blog.csdn.net/weixin_44211644/article/details/128007758)

[4] [OpenCV Basics — OpenCV Java Tutorials 1.0 documentation (opencv-java-tutorials.readthedocs.io)](https://opencv-java-tutorials.readthedocs.io/en/latest/04-opencv-basics.html)

[5] （[图像分块与置乱(Arnold）\_图像块置乱\_谷粱幽梦的博客-CSDN博客](https://blog.csdn.net/weixin_44367406/article/details/116024279)

[6] [Face Detection and Tracking — OpenCV Java Tutorials 1.0 documentation (opencv-java-tutorials.readthedocs.io)](https://opencv-java-tutorials.readthedocs.io/en/latest/06-face-detection-and-tracking.html)

[7] [Fourier Transform — OpenCV Java Tutorials 1.0 documentation (opencv-java-tutorials.readthedocs.io)](https://opencv-java-tutorials.readthedocs.io/en/latest/05-fourier-transform.html)。

[8] [OpenCV: Similarity check (PNSR and SSIM) on the GPU](https://docs.opencv.org/4.3.0/dd/d3d/tutorial_gpu_basics_similarity.html)，