Multimedia Technologies

Project Report

1. Members

序号	学号	专业班级	姓名	性别	分工
1	3220103450	计科 2202	姜雨童	女	独立完成

2. Project Introduction

2-1 选题

图像处理是计算机视觉与多媒体技术中的一个基础环节，对于图像的亮度/RGB 颜色通道/直方图的调整与分析有助于我们更好地理解图像的底层特性。

2-2 工作简介

本项目实现一个基于 Python 的 GUI 工具，用于简单的图像处理，包括亮度/对比度调整、RGB 三通道独立调整、直方图均衡化等功能，通过交互界面和可视化直方图来更便捷直观地呈现内容。

项目使用了 PIL、OpenCV、Matplotlib 等库来进行图像处理、直方图绘制等。

2-3 开发环境

开发环境	
操作系统	Windows 11
CPU	12th Gen Intel(R) Core(TM) i5-12500H
内存	16G
开发工具	Visual Studio Code
Python 版本	Python 3.12.3
开源库	Tkinter、PIL（Pillow）、matplotlib、numpy、opencv-python

2-4 系统运行要求

运行环境	
操作系统	Windows/Linux/macOS
CPU	任意
内存	2G

环境依赖	Python 3.x、Pillow 9.x、NumPy 1.x、Matplotlib 3.x、OpenCV 4.x

2-5 运行说明

使用本工具需要确保已安装以下 Python 库：

```
pip install pillow numpy matplotlib opencv-python
```

OR

```
pip install -r requirements.txt
```

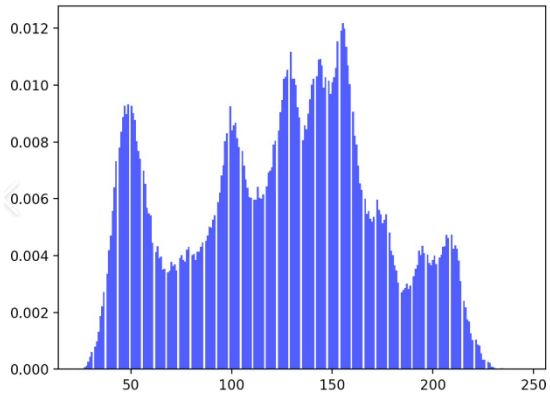
程序运行：

```
python image_processor.py
```

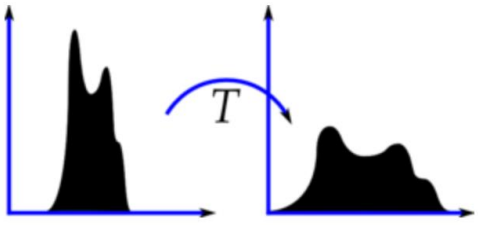
3. Technical Details

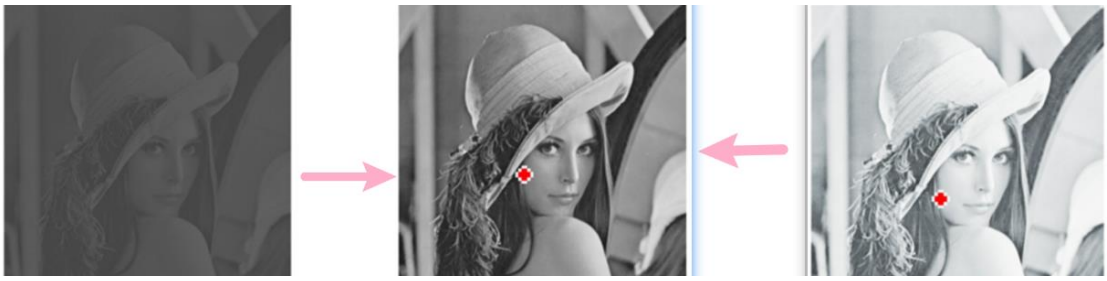
3-1 理论知识

直方图：直方图是用来表现图片中亮度分布的图片，其统计每种灰度级（或是 RGB 模式下对应颜色通道的数值）像素的总数，并以坐标轴纵轴呈现（横轴为灰度级/颜色通道数值）：



直方图均衡化：将已知灰度（或颜色通道数值，下同）概率密度分布的图像转化成具有均匀灰度概率密度分布的图像。经过直方图均衡化，过暗或过亮的图像能够被转化成明暗对比更鲜明的图像，质量更高且细节层次展现更丰富：





直方图均衡化的实现方法：（记像素值为 i 的像素点个数为 **hist[i]**）

- 计算归一化直方图（将直方图转换为概率分布，如下图中）：

$$p(i) = \frac{hist[i]}{N}, N为图像总像素数$$

像素级 个数

0	9
1	9
2	6
3	5
4	6
5	3
6	3
7	8

统计直方图

像素级 百分比

0	0.18
1	0.18
2	0.12
3	0.10
4	0.12
5	0.06
6	0.06
7	0.16

归一化直方图

像素级 百分比

0	0.18
1	0.37
2	0.49
3	0.59
4	0.71
5	0.78
6	0.84
7	1.00

累计直方图

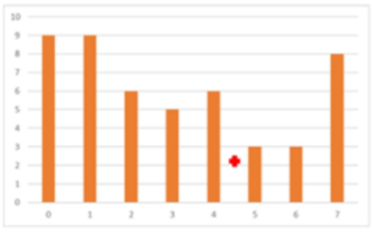
- 计算累计分布函数（CDF），得到每个灰度级的映射级（如上图右）：

$$cdf(i) = \sum_{j=0}^{i} p(j)$$

- 映射到新的灰度级（将上述 CDF 线性拉伸至 0~255 的范围）：

$$map(i) = round(cdf(i) \times 255)$$

- 将原始直方图中纵轴数值替换成映射后的新值 map(i)（如下图）

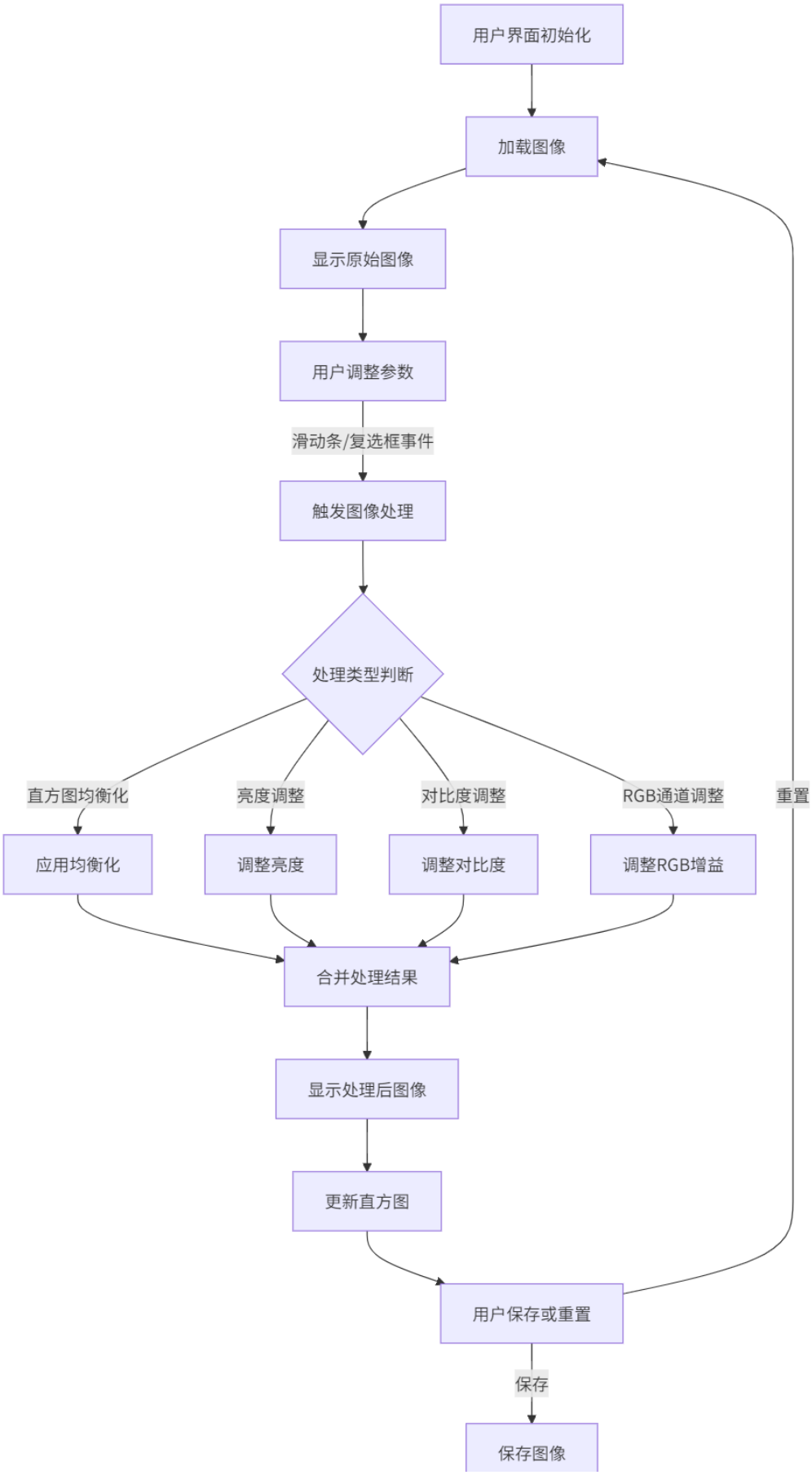


原始直方图

均衡直方图

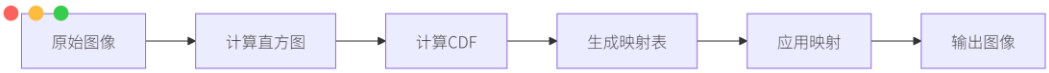
3-2 具体算法

3-2-1 整体流程图


用户界面初始化
加载图像
显示原始图像
用户调整参数
滑动条/复选框事件
触发图像处理
处理类型判断
直方图均衡化
亮度调整
对比度调整
RGB通道调整
应用均衡化
调整亮度
调整对比度
调整RGB增益
合并处理结果
显示处理后图像
更新直方图
用户保存或重置
保存
保存图像
重置

3-2-2 直方图均衡化模块

根据 3-1（理论知识）节的内容，可以得到直方图均衡化的流程：



在代码实现上，可以参考 OpenCV 库的 equalizeHist 函数：

```
void equalizeHist(InputArray src, OutputArray dst) {
    computeHistogram(src, hist);        // 计算直方图
    computeCDF(hist, cdf);              // 计算 CDF
    normalizeCDF(cdf, map);             // 归一化映射表
    applyLUT(src, map, dst);            // 应用查找表
}
```

这里直接调用该函数进行图像的直方图均衡化处理：

```
def apply_histogram_equalization(self):
    img_array = np.array(self.processed_image)
    if len(img_array.shape) == 3:  # 彩色图像
        img_yuv = cv2.cvtColor(img_array, cv2.COLOR_RGB2YUV)
        img_yuv[:,:,0] = cv2.equalizeHist(img_yuv[:,:,0])
        img_array = cv2.cvtColor(img_yuv, cv2.COLOR_YUV2RGB)
    else:  # 灰度图像
        img_array = cv2.equalizeHist(img_array)
    self.processed_image = Image.fromarray(img_array)
```

3-3 技术细节

3-3-1 关键库与函数

本项目基于 Python 实现，使用了如下开源库：

库/模块	核心函数/类	功能
PIL.Image	**Image.open(), Image.save()**	图像加载与保存
PIL.ImageEnhance	**Brightness(), Contrast()**	亮度与对比度调整
OpenCV	**cvtColor(), equalizeHist()**	颜色空间转换、直方图均衡化

库/模块	核心函数/类	功能
Matplotlib	**FigureCanvasTkAgg**	在 Tkinter 中嵌入动态直方图
numpy	**array(), ravel()**	图像数据转换为数组、展平像素值

3-3-2 自定义函数

1. **apply_processing()**

该函数用于整合所有的图像处理步骤：首先是直方图均衡化，然后是对亮度和对比度进行调整，最后是对 RGB 三个颜色通道的调整。

代码整体逻辑如下：

```
def apply_processing(self):
        self.processed_image = self.original_image.copy()
        if self.histogram_enabled.get():
            self.apply_histogram_equalization()
        self.apply_brightness_contrast()
        self.apply_color_adjustment()
        self.display_image(self.processed_image, self.processed_label)
self.update_histogram()
```

2. **display_image()**

该函数用于动态调整图像大小（包括处理前图像和处理后图像，图像高度限制在不超过窗口高度的三分之一）并将其显示在 Tkinter 界面。

代码整体逻辑如下：

```
    def display_image(self, image, label):
        # 保持宽高比，计算缩略图尺寸
        display_size = (400, min(400, max_height))
        display_image = image.copy()
        display_image.thumbnail(display_size,
Image.Resampling.LANCZOS)

        photo = ImageTk.PhotoImage(display_image)
```

3. **update_histogram()**

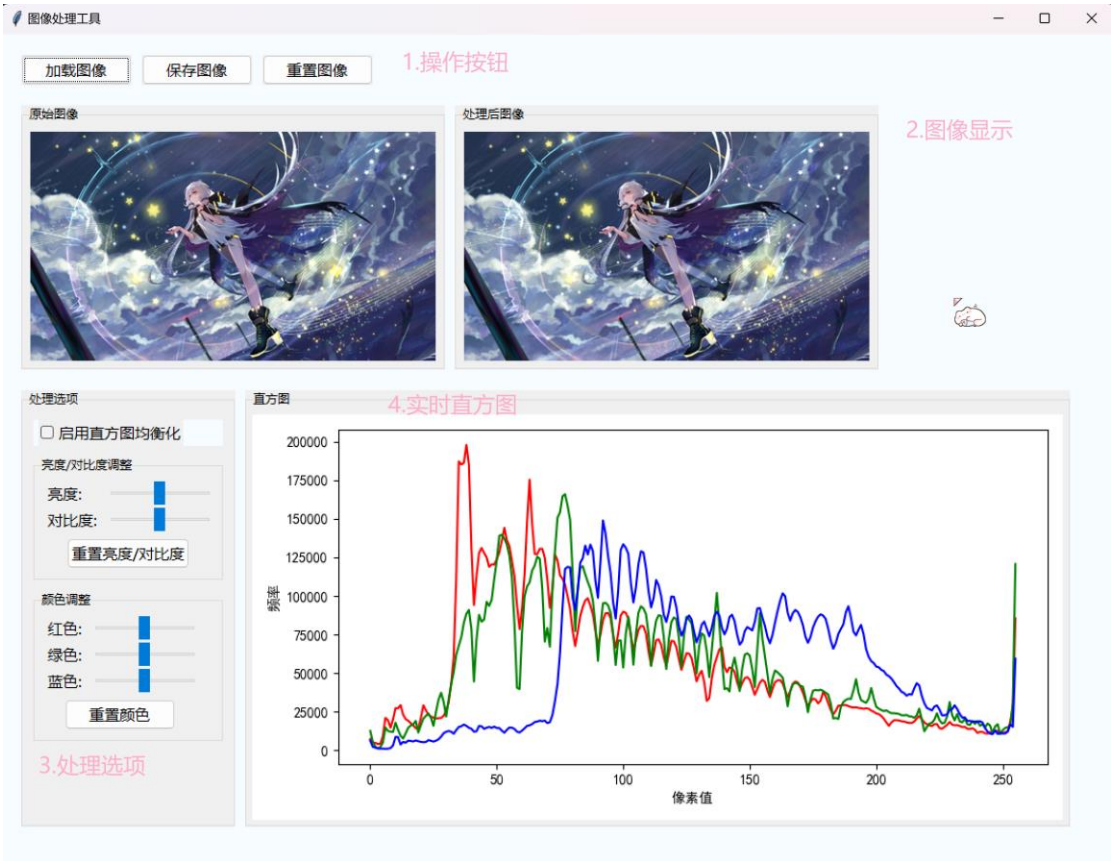
该函数调用 Matplotlib 库进行实时直方图的绘制，具体来说本部分根据图像类型（RGB/灰度）分别计算三通道或单通道直方图。

其整体代码逻辑如下：

```
def update_histogram(self):
        self.histogram_figure.clear()
        img_array = np.array(self.processed_image)
        if len(img_array.shape) == 3:    # 彩色图像
            for i, color in enumerate(('r', 'g', 'b')):
                hist = cv2.calcHist([img_array], [i], None, [256], [0, 256])
                plt.plot(hist, color=color)
        else:    # 灰度图像
            plt.hist(img_array.ravel(), 256, [0, 256])
    self.histogram_canvas.draw()
```

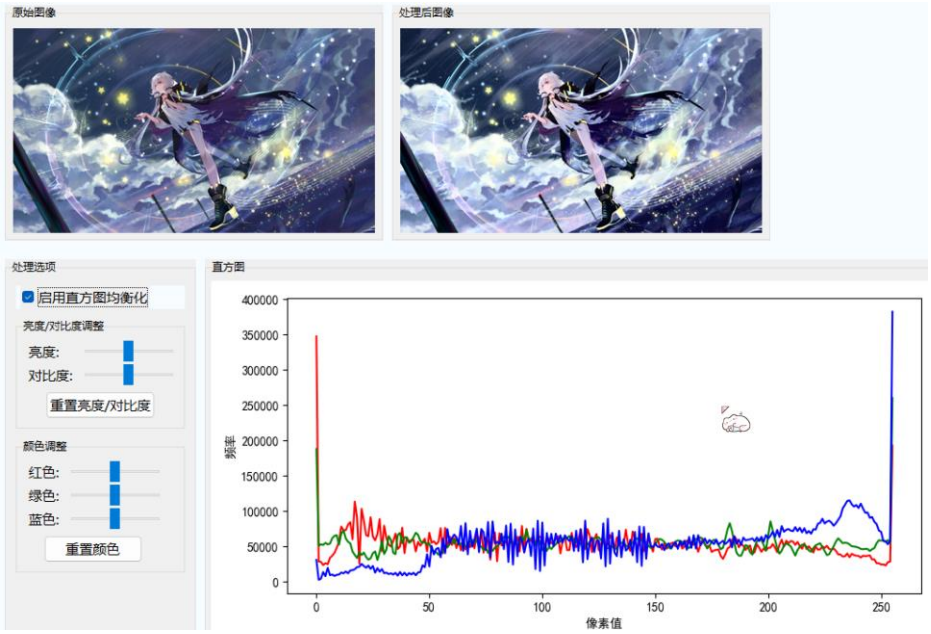
4. Experiment Results

4-1 用户界面

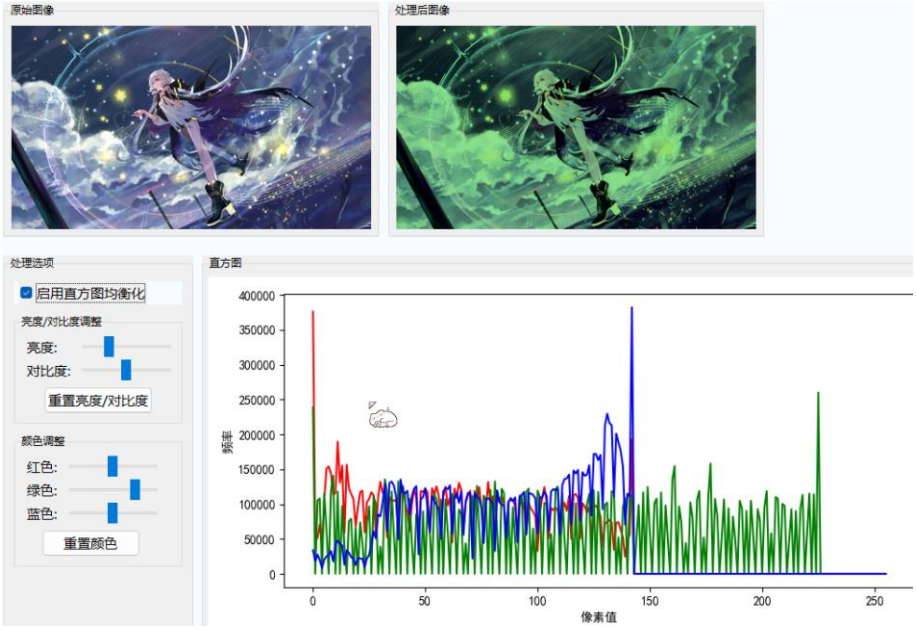


4-2 操作说明

- 点击【加载图片】按钮进行图片的加载，支持 bmp、png 和 jpg 格式
- 点击【启用直方图均衡化】前的选择框进行处理：



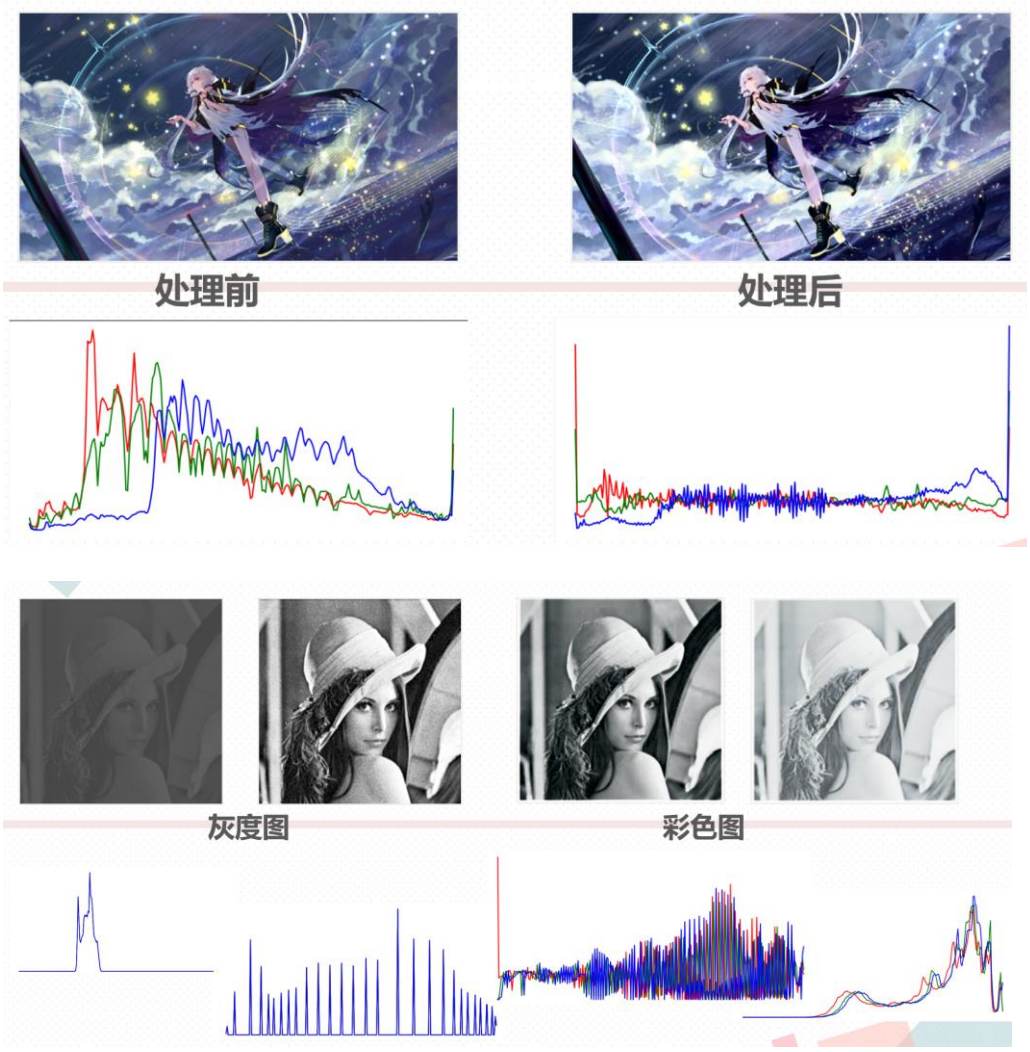
- 拖动【滑块条】对亮度/对比度/颜色通道进行修改：



- 点击【重置图像】按钮对所有操作进行重置/点击单独的重置按钮可以对细分项（亮度对比度/颜色通道）进行重置
- 点击【保存图像】按钮对图像进行保存（可选格式同上）：

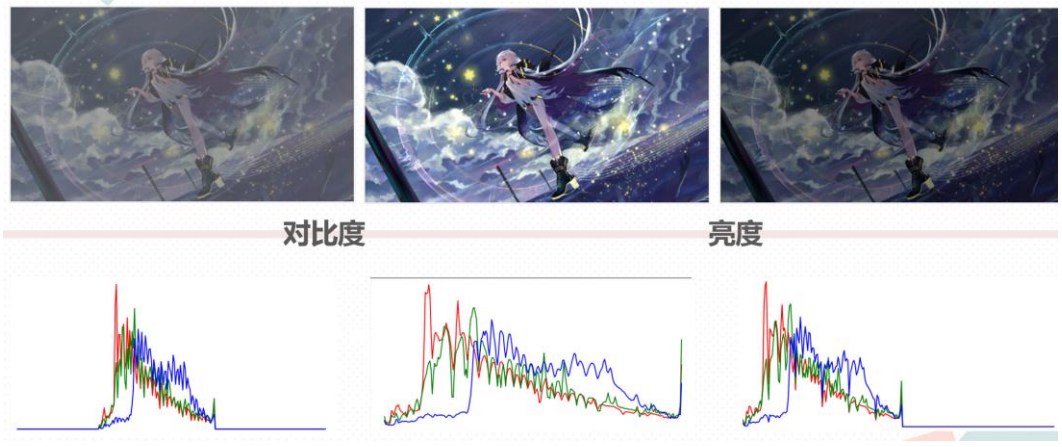
4-3 运行结果与分析

1. 图像直方图均衡化：

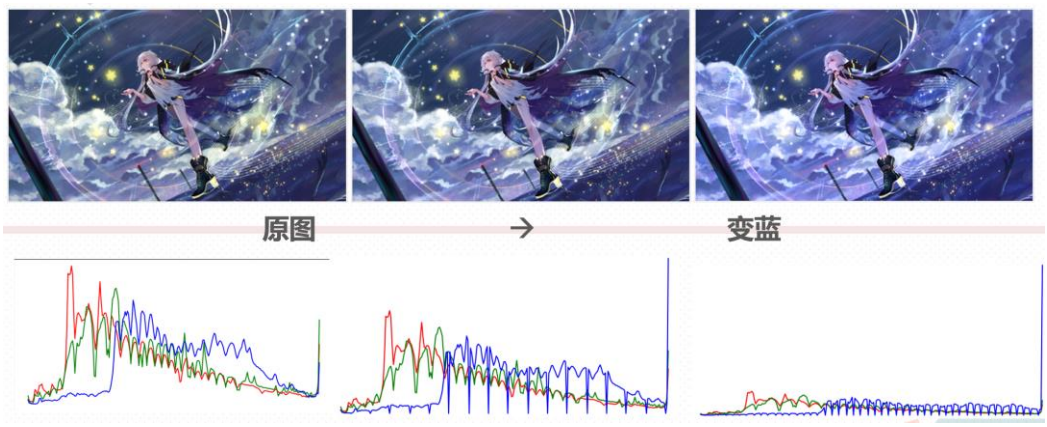


（备注：中间两张是经过处理后的图片）

2. 亮度/对比度



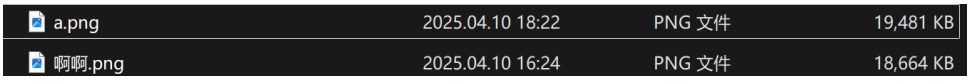
3. 颜色通道（这里以调整蓝色通道为例）



4. 结果分析

   根据上述结果可以看到，程序能较好地实现直方图均衡化的目的，对亮度/对比度，或是颜色通道的信息调整也能实时反映在直方图上。

   另外放大生成的图片/查看图片像素信息可以得知程序并不影响被处理图片的大小和清晰度：



   总体而言本次项目达到了实验的目的，也让我对图像的底层特性有了更好的理解。

References:

1. 真正搞懂图像直方图-CSDN 博客
2. 数字图像处理(17)：直方图均衡化处理-CSDN 博客
3. contrast-image · PyPI