

华北水利水电大学

North China University of Water Resources and Electric Power

《数字图像处理》 实验报告(1)

院 系： 信息工程学院
专 业： 人工智能
班 级： 2020185
姓 名： 高树林
学 号： 202018526
指 导 教 师： 韩光辉

2022-2023 学年 第一学期

实验一 Python 图像处理基础

一、实验目的

- (1) 熟悉 Python 运行环境及基本语法、图像处理相关工具包安装；
- (2) 掌握应用 OpenCV 的图像读取、显示及保存函数；
- (3) 掌握应用 Matplotlib 的图像绘制、显示及保存函数；
- (4) 掌握绘制图像直方图，并进行直方图均衡化处理。

二、实验内容和要求

代码部分应包含实验目的中对 OpenCV、Matplotlib 工具包的基本功能学习使用的代码。

直方图绘制及直方图均衡化处理可以调用工具包里的函数，但应充分学习掌握、正确使用该函数。

代码中需包含适量注释，说明编程思路。

验证实验 1：在编程环境中输入以下代码(需修改)，应用 OpenCV 的函数读取、显示并保存图像

```
import cv2
lenna = cv2.imread(r".\img\Lenna.png") # 请据实修改路径
print(type(lenna)) # 返回 numpy.ndarray 这个 class
cv2.namedWindow("Lenna",cv2.WINDOW_AUTOSIZE) #请修改窗口名
cv2.imshow("Lena",lenna)
cv2.waitKey(3000) #显示图像的暂停时间设置（单位为毫秒），请修改
cv2.destroyWindow("Lena")
cv2.imwrite('test_imwrite.png',lenna,(cv2.IMWRITE_PNG_COMPRESSION,5))
```

验证实验 2：在编程环境中输入以下代码，应用 Matplotlib 的函数显示及保存图像

```
import cv2
import Matplotlib.pyplot as plt
plt.rcParams['font.family']='[SimHei]' #用来正常显示中文
plt.rcParams['axes.unicode_minus']=False #用来正常显示负号
img_BGR = cv2.imread(r'.\img\iris.jpg') #OpenCV 默认为 BGR 彩色模型
img_RGB = cv2.cvtColor(img_BGR,cv2.COLOR_BGR2RGB) #转换为 RGB 彩色模型
plt.imshow(img_RGB) #Matplotlib 默认为 RGB 彩色模型
plt.show()
```

说明：以上验证实验，实际代码应据实修改；鼓励大家探索更多功能，熟练工具包使用。

三、实验环境(学生填写)

（较详细地说明实验运行环境，包括操作系统、编程 IDE 及版本、编程语言及版本、依赖的第三方类库及版本等内容）

本实验环境分为硬件和软件两大方面叙述。

硬件方面：本实验室基于处理器为 Intel(R) Core(TM) i5-10210U CPU @ 1.60GHz 2.11 GHz，基带 RAM 为 16.0 GB (15.8 GB 可用)，系统类型为 64 位操作系统，基于 x64 的处理器 MagicBook 商务本上运行并得出相应结果和数据的。

软件方面：本实验基于 windows10 家庭中文版，版本号为 21H2，操作系统内部版本为 19044.2130 的 MagicBook 商务本。编程语言选择 python 语言，使用运行时版本号为 11.0.14.1+1-b2043.45 amd64 的 PyCharm 2022.1.1 (Professional Edition)集成开发环境编写代码。其中代码的 python 解释器版本为 Python 3.8.13 (default, Mar 28 2022, 06:59:08) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32。引入 1.0.0.14 版本的 OpenCV-python 库，其中集成的 OpenCV 版本为 4.6.0。引入 Matplotlib3.5.3 版本的库。

四、实验过程(学生填写)

1. OpenCV、Matplotlib 工具包使用

(1)实验方案（设计思路说明）

- 1.实现使用 OpenCV 工具包读取一张图片，并对读取的照片做其他操作。
- 2.查看 1 中读取照片后的数据类型，了解照片经过 OpenCV 库读取之后以什么形式存储在计算机里。
- 3.将 1 中读取的数据转化为图片，并用 OpenCV 工具包和 Matplotlib 工具包显示出来，对比它们的区别。
- 4.将 3 中生成的图片做持久化存储。

扩展内容：

1. 将原照片转化为灰度图，并按照实验方案中（3）的方法将其做持久化存储。
2. 将原图片的三个通道分离出来，用 Matplotlib 绘在一张图上，对比他们的差异性。

(2)代码编写（包含适量注释）

```
1. import cv2
2. import matplotlib.pyplot as plt
3. import numpy as np
4.
5.
6. def pltshow(a):
7.     plt.figure("Image") # 图像窗口名称
8.     plt.imshow(a[:, :, ::-1])
9.     plt.axis('on') # 关掉坐标轴为 off
10.    plt.title('image') # 图像题目
11.    plt.savefig('./plt_picture.jpg')
```

```
12.     plt.show()
13.
14.
15. def OpenCVshow(a):
16.     cv2.namedWindow("Gosling's report1", cv2.WINDOW_GUI_EXPANDED) #
    已修改窗口名
17.     cv2.imshow("Lena", a)
18.     cv2.waitKey(50000) # 显示图像的暂停时间设置（单位为毫秒），已修改
19.     cv2.destroyAllWindows("Lenna")
20.     cv2.imwrite('OpenCV_lenna.png', lenna, (cv2.IMWRITE_PNG_COMPRESSI
    ON, 5))
21.
22.
23. def OpenCV_color2gray(path):
24.     img = cv2.imread(path, 0) # 直接以灰度图片读取
25.     OpenCVshow(img)
26.
27.
28. def Plt_color2gray(path):
29.     img = cv2.imread(path, 0)
30.     plt.imshow(img, cmap='gray')
31.     plt.show()
32.
33.
34. def OpenCV_takecolor(a):
35.     blueImg = a[:, :, 0]
36.     greenImg = a[:, :, 1]
37.     redImg = a[:, :, 2]
38.     imgs = np.hstack([blueImg, greenImg, redImg])
39.     cv2.imshow("mutil_pic", imgs)
40.     cv2.waitKey(0) # 显示图像的暂停时间设置（单位为毫秒），请修改
41.
42.
43. def Plt_takecolor(a):
44.     blueImg = a[:, :, 0]
45.     greenImg = a[:, :, 1]
46.     redImg = a[:, :, 2]
47.     plt.figure()
48.     plt.subplot(1, 3, 1)
49.     plt.imshow(blueImg, cmap='gray') # 蓝色通道转化为灰度
50.     plt.subplot(1, 3, 2)
51.     plt.imshow(greenImg, cmap='gray') # 绿色通道转化为灰度
52.     plt.subplot(1, 3, 3)
53.     plt.imshow(redImg, cmap='gray') # 红色通道转化为灰度
```

```

54. plt.show()
55.
56.
57. if __name__ == "__main__":
58.     path = ".\\figures\\img\\Lenna.png"
59.     lenna = cv2.imread(path) # 请据实修改路径
60.     print(type(lenna)) # 返回 numpy.ndarray 这个 class
61.     pltshow(lenna) # 用 Matplotlib 作图显示照片
62.     OpenCVshow(lenna) # 用 OpenCV 作图显示照片
63.     OpenCV_color2gray(path) # 用 OpenCV 做灰度图并保存
64.     Plt_color2gray(path) # 用 Matplotlib 做灰度图并保存
65.     OpenCV_takecolor(lenna) # 用 OpenCV 提取三通道并转为灰度图显示出来
66.     Plt_takecolor(lenna) # 用 Matplotlib 提取三通道并转为灰度图显示出来
    
```

(3)调试（编码实现过程中遇到的错误，及调试解决等内容）

在调试的过程中，出现了很多问题，但是最终还是很幸运的把他们都解决了，并且我还尝试了更多的变换内容，使我感到收获颇丰。下面总结一下我在调试过程中遇到的困难以及解决方案。

问题 1：OpenCV 报错，报错的截图如下图 1 所示。

```

[ WARN:0@0.014] global D:\a\opencv-python\opencv-python\opencv\modules\imgcodecs\src\loadsave.cpp (239) cv
<class 'NoneType'>
Traceback (most recent call last):
  File "D:/2020185_and_10208/Kernel_lessons/数字图像处理/report1_1.py", line 5, in <module>
    cv2.imshow("Lena",lenna)
cv2.error: OpenCV(4.6.0) D:\a\opencv-python\opencv-python\opencv\modules\highgui\src\window.cpp:967: error
    
```

图 1 报错 1

原因解析：报这个错误的原因是因为前面引用照片路径的时候，路径含有中文，中文在编译的时候会乱码，因此不会找到对应的图片或者找到的文件不能被 OpenCV 执行和操作，因此报错。此外路径中的单斜杠如果后面跟的数据与它能组成转义字符，也不能被正确的识别为正常的路径。

解决方案：将文件路径的中文改为英文并且把“/”改为“/”。

```

import cv2
lenna = cv2.imread(".\\figures\\img\\Lenna.png") # 请据实修改路径
print(type(lenna)) # 返回numpy.ndarray 这个 class
    
```

图 2 报错 1 解决方案

问题 2：使用 Matplotlib 工具包显示图片色彩差异很大，对比图如下图所示。



图 3 问题 2 的差异对比（左边为 OpenCV 输出，右边为 Matplotlib 输出）

原因解析:造成这种差异的原因是 OpenCV 读取的图片是 BGR 格式的，而 Matplotlib 只能输出的是 RGB 格式的数据，如果不是按照 RGB 顺序的数据经 Matplotlib 输出会有色差。

解决方案:虽然相对于将 OpenCV 读取的数据修改为 RGB 格式的数据，用 PIL 库直接来读取图片为 RGB 数据更为方便，但是为了让实验不引入那么多工具包，可以如下图 4 所示用切片的形式使 OpenCV 读取的数据的结构发生改变，之后用 Matplotlib 工具包处理该数据。

```
def pltshow(a):
    plt.figure("Image") # 图像窗口名称
    plt.imshow(a[:, :, ::-1])
```

图 4 问题 2 解决方案

(4)实验结果（实验运行结果截图等体现实验结果的内容）

结果 1: 如下图 5 所示为用 OpenCV 读取并打开照片的结果。

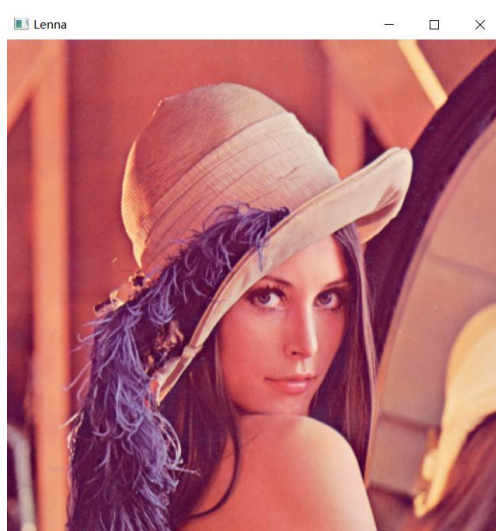


图 5 使用 OpenCV 打开图片

结果 2: 如下图 6 所示为利用 Matplotlib 读取并打开照片的结果。

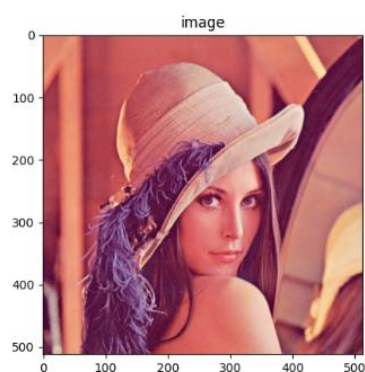


图 6 使用 Matplotlib 打开图片

结果 3: 如下图 7 所示为利用 OpenCV 读取并打开图片灰度图的结果。

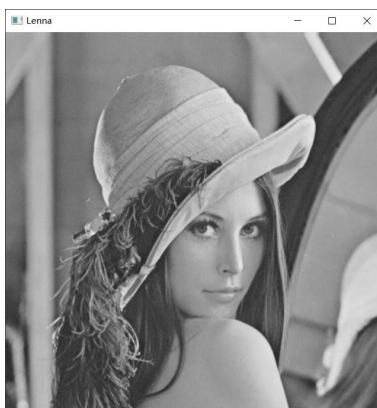


图 7 利用 OpenCV 读取并显示图片灰度图

结果 4: 如下图 8 所示为利用 Matplotlib 读取并打开图片灰度图的结果。

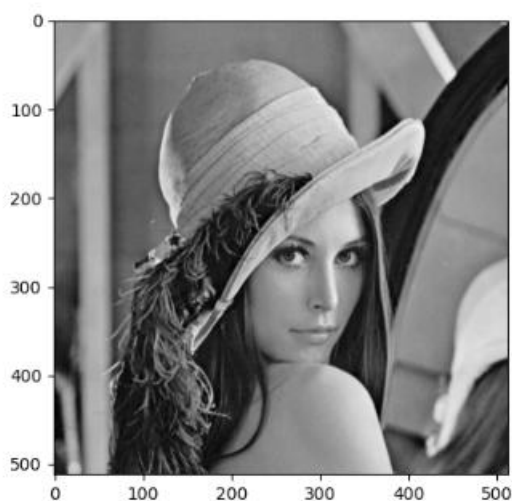


图 8 利用 Matplotlib 读取并显示图片灰度图

结果 5: 如下图 9 所示为利用 OpenCV 读取图片，并显示三个通道的灰度图。



图9 利用 OpenCV 读取图片蓝（左）、绿（中）、红（右）三通道的灰度图

结果 6: 如下图 10 所示为利用 Matplotlib 读取图片，并显示三个通道的灰度图。

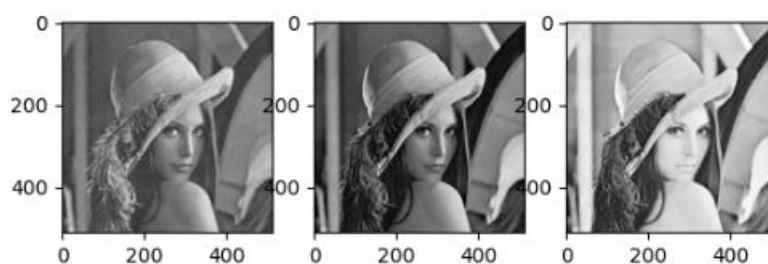


图 10 利用 Matplotlib 读取图片蓝（左）、绿（中）、红（右）三通道的灰度图

(5)分析与总结（与本次实验相关的分析与总结）

本次实验主要是对数字图像处理的两大库——OpenCV 和 Matplotlib 库的基本操作的训练。通过本次实验，我深刻认识到了这两个工具包在读取图片时获取的数据是不一样的，前者获取的是 BGR 格式的三维 numpy 数组，后者获取的是 RGB 格式的三维 numpy 数组。其次，我深刻理解了 OpenCV 和 Matplotlib 两个工具包的可视化方法，和在一个页面同时输出多张照片的方法。OpenCV 的方法是把要显示的图片放在一个列表里面，然后显示列表就能将所有添加到列表的元素全部显示出来了。而在 Matplotlib 工具包里面，则是新建画布，这个画布可以容纳多少照片可以自己定义，之后可以进行显示照片的操作，但是同时需要进行的是要规定需要显示的照片的位置，用 `plt.subplot()` 函数来控制。此外，他们保存图片的方式也不一样，OpenCV 是通过从 `cv.imwrite()` 函数来保存的，而 Matplotlib 工具包中一般是使用 `plt.savefig()` 来保存图片的。

通过本次实验，我对图像处理的方法和技巧都有了很高层次的掌握，也能开始对像素点进行相应的操作，为在计算机视觉方向上的路打下了坚实的基础！

2. 绘制图像直方图及直方图均衡化处理

(1)实验方案（设计思路说明）

1. 使用 `plt.hist()`函数是 `matplotlib` 工具包中的内置函数，能够将原图直接转化为其对应的直方图。用它完成直方图的绘制。

2. 使用 `OpenCV` 工具包中的 `cv2.equalizeHist()`函数将原直方图均衡化，该函数的返回值是一个 `numpy` 数组类型，其含义为 `RGB` 类型的三通道图片像素。经过 `plt.imshow()`函数处理返回的是一张图，通过 `plt.show()`可以将照片显示出来。

3.最后的结果展示在同一张图片上，包含原图像、原图像的直方图、原图像的均衡直方图、均衡化后的图像或其灰度图，做一个对比。

(2)代码编写（包含适量注释）

```
1. import cv2
2. from matplotlib import pyplot as plt
3. from pylab import mpl
4. # 设置显示中文字体
5. mpl.rcParams["font.sans-serif"] = ["SimHei"]
6. # 设置正常显示符号
7. mpl.rcParams["axes.unicode_minus"] = False
8. plt.subplots_adjust(left=None, bottom=None, right=None, top=None, wspace=0.5, hspace=0.5) # 设置子图左右和上下间隔，一般为0.5为佳
9.
10. plt.subplot(241) # 三个参数分别代表的是行、列、序号，序号从左往右从上到下增加，从1开始
11. plt.imshow(cv2.imread('.\\figures\\img\\lenna.png')[ :, :, :-1]) # 用OpenCV读取图片，并将其转化为RGB格式，转化完成后用matplotlib展示出来
12. plt.xlabel('lenna 原图',fontsize=10) # 设置横坐标名称和字体
13.
14. plt.subplot(242)
15. img = cv2.imread('.\\figures\\img\\lenna.png', 0)
16. plt.hist(img.ravel(), 255, [0, 256]) # plt.hist()是matplotlib内置画图函数，可以直接统计并绘制直方图
17. plt.title("lenna 的直方图", fontsize=10)
18.
19. plt.subplot(243)
20. plt.hist(cv2.imread('.\\figures\\img\\lenna.png', cv2.IMREAD_GRAYSCALE).ravel(), 256) # 转为均衡图并显示
21. plt.title('lenna 均衡化直方图', fontsize=10)
22.
23. plt.subplot(244)
24. plt.imshow(cv2.equalizeHist(cv2.imread('.\\figures\\img\\lenna.png', cv2.IMREAD_GRAYSCALE)), 'gray')
25. plt.xlabel('lenna 均衡灰度图',fontsize=10)
26.
```

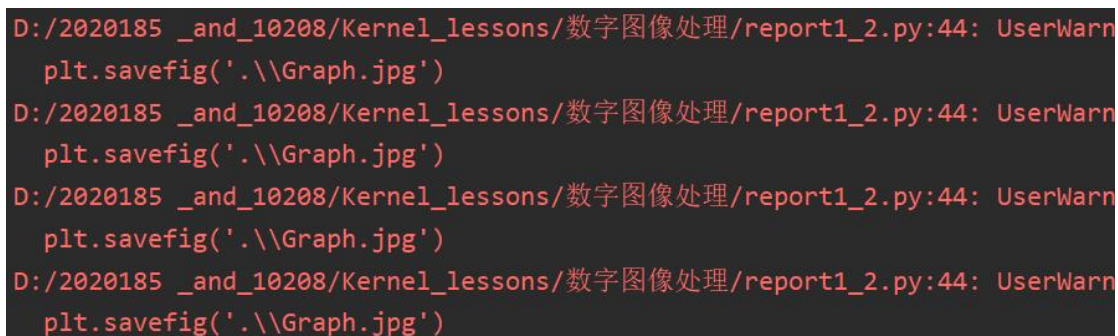
```

27. plt.subplot(245)
28. plt.imshow(cv2.imread('.\\figures\\img\\iris.jpg')[:, :, ::-1]) # 用
    OpenCV 读取图片, 并将其转化为 RGB 格式, 转化完成后用 matplotlib 展示出来
29. plt.xlabel('iris 原图', fontsize=10) # 设置横坐标名称和字体
30.
31. plt.subplot(246)
32. img1 = cv2.imread('.\\figures\\img\\iris.jpg', 0)
33. plt.hist(img1.ravel(), 255, [0, 256])
34. plt.title("iris 的直方图", fontsize=10)
35.
36. plt.subplot(247)
37. plt.hist(cv2.imread('.\\figures\\img\\iris.jpg', cv2.IMREAD_GRAYSCALE)
    .ravel(), 256)
38. plt.title('iris 的均衡化直方图', fontsize=10)
39.
40. plt.subplot(248)
41. plt.imshow(cv2.equalizeHist(cv2.imread('.\\figures\\img\\iris.jpg', c
    v2.IMREAD_GRAYSCALE)), 'gray')
42. plt.xlabel('iris 均衡灰度图', fontsize=10)
43.
44. plt.savefig('.\\Graph.jpg')
45. plt.show()

```

(3)调试（编码实现过程中遇到的错误，及调试解决等内容）

问题 1：在调试代码的过程中，我遇到以下警告，报错截图如图 11 所示。



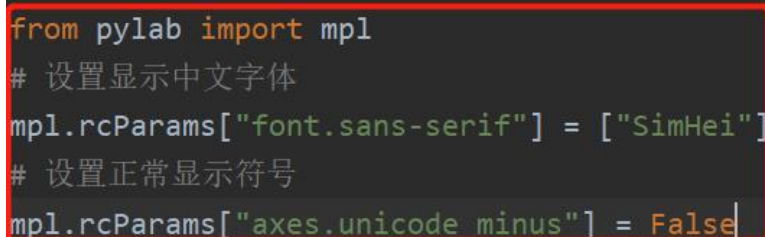
```

D:/2020185 _and_10208/Kernel_lessons/数字图像处理/report1_2.py:44: UserWarn
plt.savefig('.\\Graph.jpg')
D:/2020185 _and_10208/Kernel_lessons/数字图像处理/report1_2.py:44: UserWarn
plt.savefig('.\\Graph.jpg')
D:/2020185 _and_10208/Kernel_lessons/数字图像处理/report1_2.py:44: UserWarn
plt.savefig('.\\Graph.jpg')
D:/2020185 _and_10208/Kernel_lessons/数字图像处理/report1_2.py:44: UserWarn
plt.savefig('.\\Graph.jpg')

```

图 11 问题 1 警告截图

问题分析：经过分析，我明白这是因为我没有规定字体（第 2 页代码给有相应的代码段）于是我将代码段粘贴到我的代码中，问题得以解决。下图 12 为解决成功的解决方法。



```

from pylab import mpl
# 设置显示中文字体
mpl.rcParams["font.sans-serif"] = ["SimHei"]
# 设置正常显示符号
mpl.rcParams["axes.unicode_minus"] = False

```

图 12 问题 1 解决方案截图

问题 2: matplotlib 画图时的子图之间，安放太挤导致重叠的部分字体和数据无法辨认，问题截图如图 13 所示。

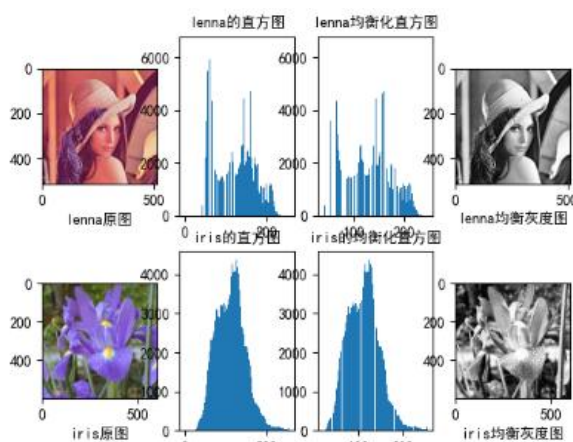


图 13 matplotlib 画布上的子图太挤

问题解析: 没有设置子图与子图之间的距离，因此当一个子图里面的数据过长或者标签过长时导致重合，此时只需要设置子图之间的左右距和上下距。该指标是通过 `subplots_adjust()` 函数来控制的。其中的参数 `wspace` 和 `hspace` 一般取 0.5 即可。问题解决的截图如图 14 所示。

```
plt.subplots_adjust(left=None,
                    bottom=None,
                    right=None,
                    top=None,
                    wspace=0.5,
                    hspace=0.5) # 设置子图左右和上下间隔，一般为0.5为佳
```

图 14 问题 2 解决的截图

(4)实验结果（实验运行结果截图等体现实验结果的内容）

结果 1: 图片通过 Matplotlib 工具包中的 `hist` 函数处理得到的直方图如下图 15 所示。

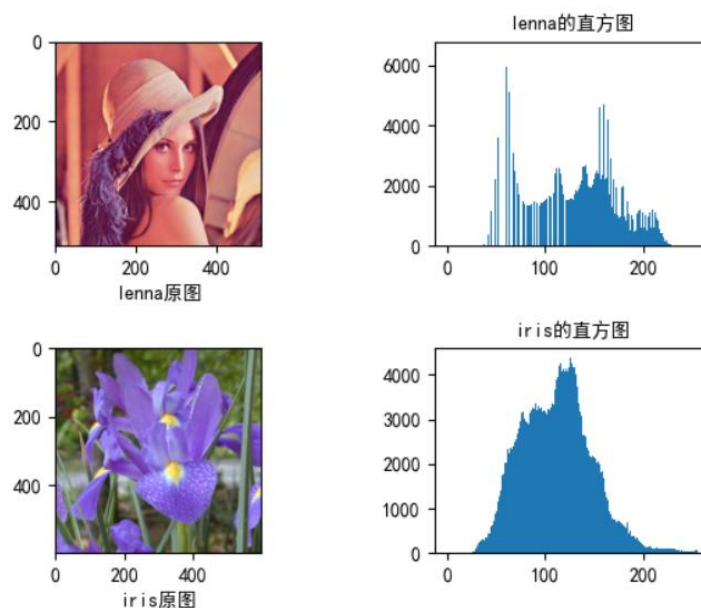


图 15 hist 函数处理得到的直方图

结果 2: 利用 OpenCV 的内置函数 `equalizeHist()` 处理直方图得到均衡直方图如下图 16 所示。

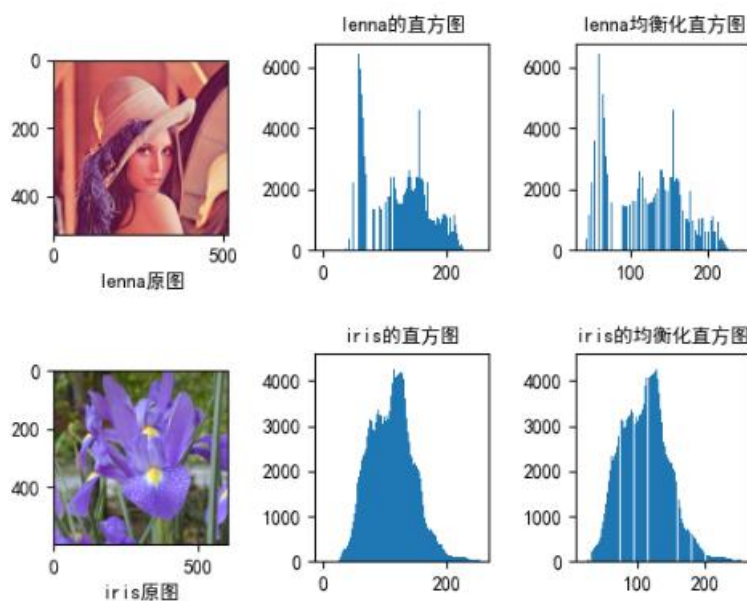


图 16 `equalizeHist()` 处理得到均衡直方图

最终结果: 将最后的结果展示在同一张图片上，包含原图像、原图像的直方图、原图像的均衡直方图、均衡化后的图像或其灰度图的结果如下图所示。

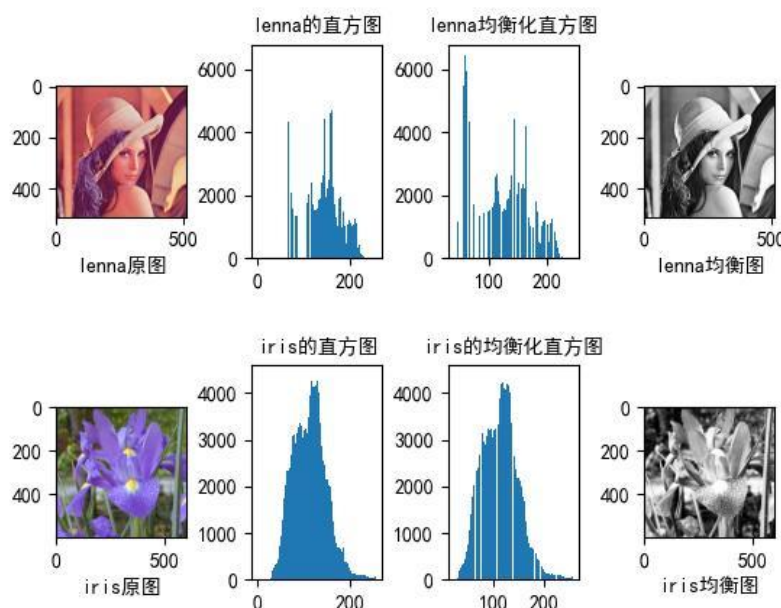


图 17 使用 Matplotlib 工具包做出统计并绘制的图

(5)分析与总结（与本次实验相关的分析与总结）

本次实验主要是图片读取，读取通过 OpenCV 的 `imread()` 函数进行读取，读取的格式为 BGR 格式。Matplotlib 中有一个直接能将该数据转化为直方图的函数 `Hish()`，之后通过 OpenCV 库的 `equalhish()` 函数将 `Hish()` 处理得到的图均衡化。整个过程中难点在于 RGB 和 BGR 格式的相互转化。

通过本次实验，我掌握了图像转化成其所对应的直方图的方法。深刻了解到原始图像由于其灰度分布可能集中在较窄的区间，造成图像不够清晰，因此通过改变图像的直方图进而能够改变图像的像素分布。直方图均衡化在增强动态范围偏小的图像的对比度上有着很普遍的应用。这在生活中的一些场景里也有着非常重要的作用，比如警方在侦办犯罪活动的时候监控的内容由于对比度不高，难以看清嫌疑人的某些特征，可以用到此技术。此外，在医疗检测方面也能运用到该技术从而达到监测某些肉眼难以观测的生物细菌或者肿瘤。此外，在本次实验中，我掌握了一门技能——作图。这份实验报告完成下来，我做了不下 20 个图，对作图的操作越发炉火纯青，在以后的实验报告中我就可以用作图的方式来展现我的数据，展现我的成果，而不是用那些干瘪的文字来堆砌。我会在以后的每一门、每一个实验报告中尽可能地运用到在数字图像处理课程学到的知识并与当前课程相结合，做出更令人满意的报告。我也仍会在毕业论文和后来发的期刊论文或者会议上尽可能的用本次学到的使用 Matplotlib 工具包结合实验数据特征对数据进行可视化。总之，在这次实验中学习的知识和技术和操作都为我在研究计算机视觉领域上拉开了一个非常不错的序幕。希望以数字图像处理为起点，在计算机视觉领域不断前行，能成为何凯明，李飞飞那样的科研巨人！