华东师范大学软件工程学院实验报告

姓名：张梓卫　　**学号**：10235101526

实验编号：　　**实验名称**：对 OpenCV 库的初步尝试

一、实验目的

初步接触 Python 中 OpenCV 库的使用，为接下来的 Pytorch 图像识别及学习打下基础，为卷积等内容的深造打下基础。

任务：

1、实现对比度的调整；

2、实现图像融合；

3、对线性滤波相关的功能初步进行使用。

二、实验内容与实验步骤

选择 Python 中的 OpenCV 库，先学习了初步的方法。
优先了解 OpenCV 中的图像处理基础：

opencv 图像存储在 numpy 数组中，存储的是三个灰度图（存储在第三个维度上）
BGR 是存储颜色的顺序，而不是我们常说的 RGB
故，cv2.imshow("Blue", image[:, :, 0/1/2]) # 可以分别显示蓝、绿、红通道
大量的图像算法都是基于灰度图来操作的，灰度图即：

```
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY) # 转换为灰度图
```

三、实验过程与分析

为进行图像处理，优先选用了自己在摄影过程中拍摄的照片，命名为 temp.1 与 temp.2，作为两张实验图片。

先完成最简单的图像融合，这个过程中使用 addWeighted 函数，它的参数分别为两张图片，两张图片的权重，以及亮度。

Python 代码如下：

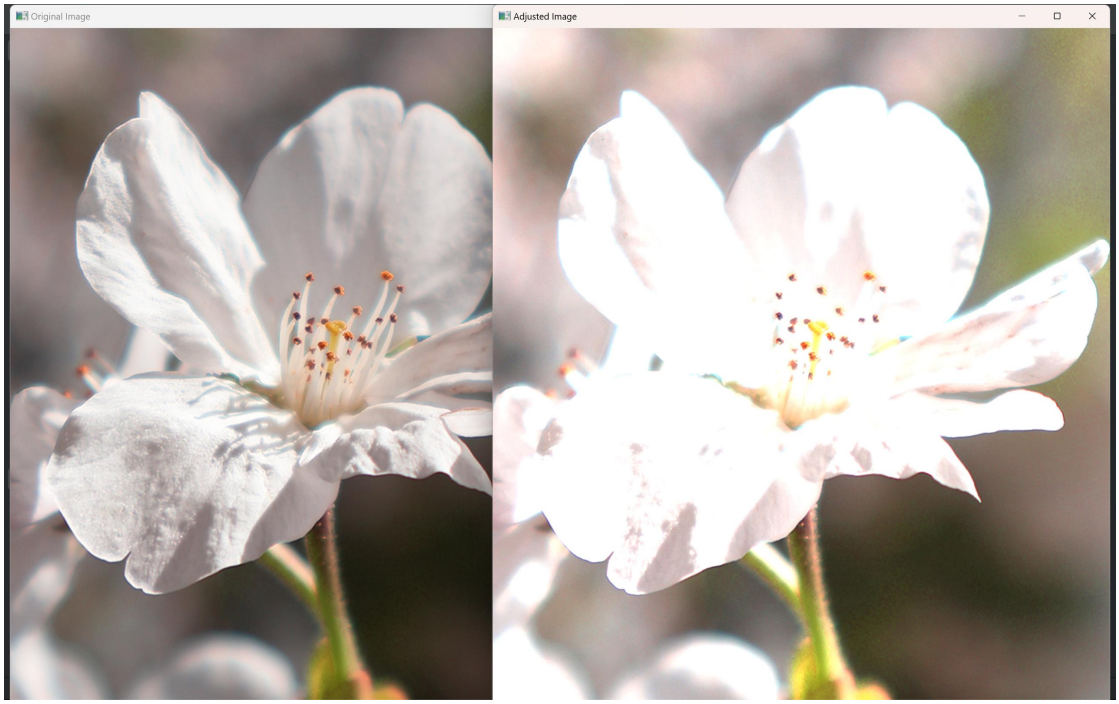
```
import cv2

image1 = cv2.imread("temp.jpg")
image2 = cv2.imread("temp2.jpg")

image1_resized = cv2.resize(image1, (1000, 1000))
image2_resized = cv2.resize(image2, (1000, 1000))

Final = cv2.addWeighted(image1_resized, 0.6, image2_resized, 0.6, 0)
# 图像混合函数，两张图片的尺寸大小必须一致

cv2.imshow("Final", Final)
cv2.waitKey()
cv2.destroyAllWindows()
```

输出结果如下所示：

随后，选用第一张图片，进行图片对比度线性处理。使用 Python 代码实现简单的线性变换：

```python
import cv2
import numpy as np

def adjust_contrast(image, alpha, beta):
    adjusted_image = cv2.convertScaleAbs(image, alpha=alpha, beta=beta)
    return adjusted_image

image = cv2.imread('temp.jpg')

# 调整对比度和亮度
alpha = 1.5  # 对比度增益因子
beta = 5   # 亮度增益
adjusted_image = adjust_contrast(image, alpha, beta)

cv2.imshow('Adjusted Image', adjusted_image)
cv2.waitKey()
cv2.destroyAllWindows()
```

Original Image And Adjusted Image :



注意，这种变化不是伽马校正，伽马校正是一种非线性的变化，非线性转换目的主要是为了优化存储空间与带宽，传递函数能更好地帮我们利用编码空间。

目前普遍使用的 SRGB 颜色空间标准，传递函数 Gamma 值为 2.2。使用 Gamma 校正的原因有：

(1) 人眼在计算机上对于暗部细节观察多，而亮部细节观察少；
(2) 计算机问题，早期性能不行，Gamma ~ 2.2 情况下，可以节约资源存储亮部，更多资源存储暗部。

查阅资料后，得出相关信息：

亮度和对比度调整：

亮度和对比度调整为点运算，常用的点运算理论公式如下：

$$g(x) = \alpha f(x) + b$$

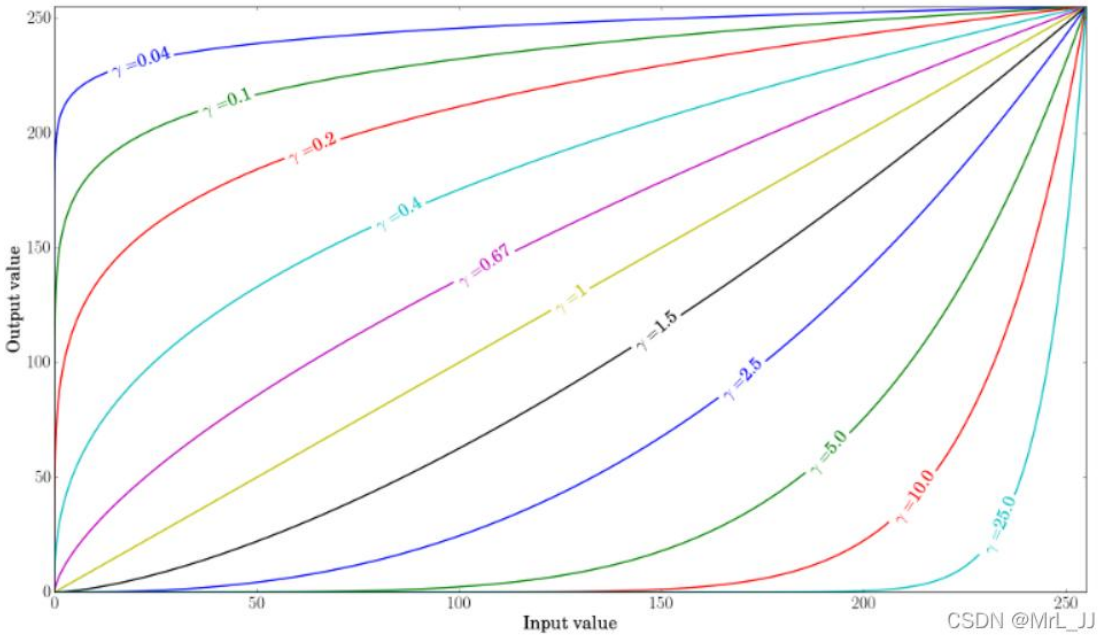
参数 α >0 和 b 通常称为增益参数和偏置参数；有时这些参数被称为分别控制对比度和亮度。f(x)想象成源图像像素，而 g(x)是输出图像像素。

伽玛校正：

Gamma 校正可以通过在输入值和映射的输出值之间进行非线性变换来校正图像的亮度。理论公式如下：

$$O = \left(\frac{I}{255}\right)^{r} \times 255$$

由于这种关系是非线性的，所以效果对所有的像素都是不一样的，并取决于它们的原始值。I 为图像原始灰度值，O 为最终计算输出灰度值。



CSDN @MrL_JJ

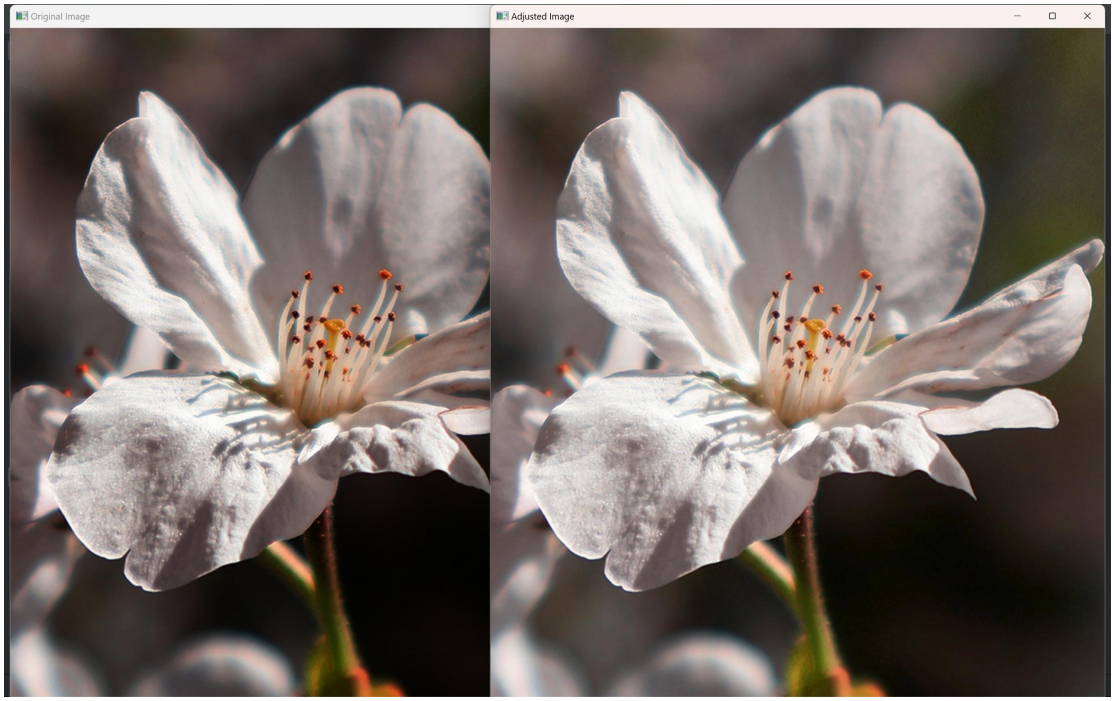
Python 使用 numpy 库、OpenCV 库实现如下：

```
import cv2
import numpy as np

def adjust_gamma(image, gamma=1.0):
    # 通过构建伽马校正查找表进行伽马校正
    inv_gamma = 1.0 / gamma
    # 使用 numpy 的 arange 函数创建了一个查找表
    # 生成从 0 到 255 的数字数组（RGB）
    # 数组中的每个值 i 都通过除以 255.0 进行归一化
    # 然后乘以 inv_gamma 的幂，再缩放回 0 ~ 255 的范围
    # 得到的数组转换为 8 位 无符号整数数组
```

```
    table = np.array([((i / 255.0) ** inv_gamma) * 255
        for i in np.arange(0, 256)]).astype("uint8")

    # 最后，使用 OpenCV 的 LUT 函数将查找表应用到图像
    # 根据查找表映射图像的像素值进行伽马校正
    return cv2.LUT(image, table)

# 读取图像
image = cv2.imread('temp.jpg')

# 设置伽马值（可根据需要调整）
gamma = 1.5

# 进行伽马校正
adjusted_image = adjust_gamma(image, gamma=gamma)

# 显示原始图像和伽马校正后的图像
cv2.imshow('Original Image', image)
cv2.imshow('Adjusted Image', adjusted_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

输出结果如下所示：

左图：Original Image，右图：Adjusted Image，其中 Gamma = 1.35.



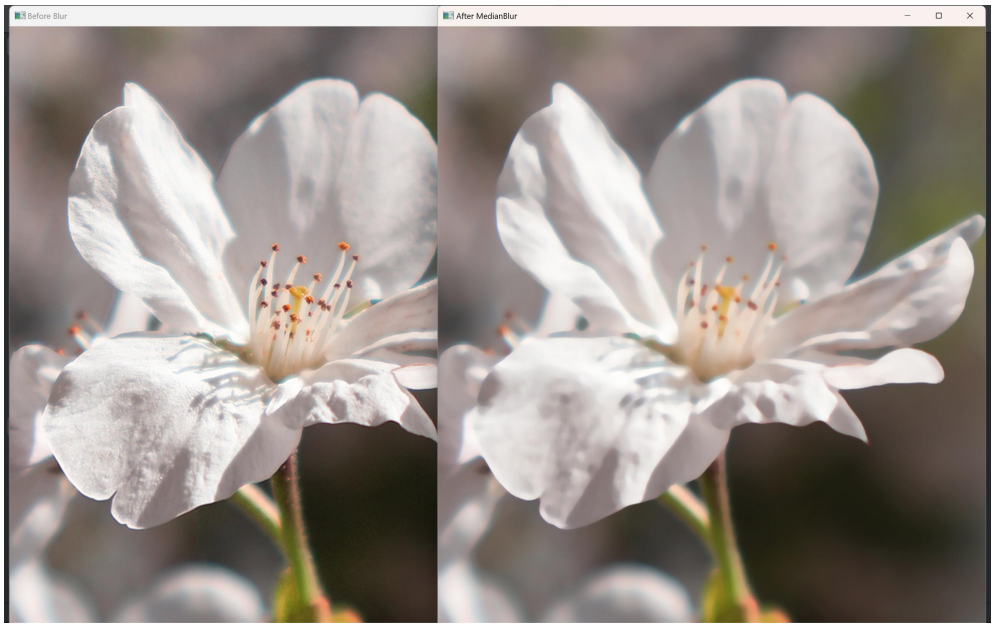
下一步测试高斯滤波、均值滤波：

Python 代码如下所示：

```
import cv2
```

```python
image = cv2.imread("temp.jpg",) # 读取图片

Gauss = cv2.GaussianBlur(image ,(5,5), 0) # 均值滤波（图片，内核大小）
# 这里 sigmaX = 0，代表这个值由内核决定
Median = cv2.medianBlur(image, 5) # 中值滤波（图片，内核大小）

cv2.imshow("After GaussBlur", Gauss)
cv2.imshow("Before Blur", image)
cv2.imshow("After MedianBlur", Median)
cv2.waitKey() # 等待按键，为了让图片不一闪而过
```

中值滤波前后对比如下所示：



高斯滤波前后对比如下所示：

四、实验结果总结

在本次作业中，充分感受到了 OpenCV 库的强大，批量化处理图片带来的震撼，以及对计算机成像有了更深的原理理解。

完成了任务：

1、对比度调整

2、线性滤波

3、图像融合

非常喜欢刚接触一个库时，不断地去查对应的方法、参数的感觉，像是真正的学习且学以致用的过程。因为我们的课程里尚未学习 Python 的语法，于是就自己马上过了一遍，几乎是从零开始的，让我对应用领域的编程越来越感兴趣。

五、附录（源代码）

在学习过程中，顺带了解了实时读取人脸识别这种 OpenCV 最广为人知的算法，于是通过学习造了 Template Recognize：

```
import cv2
import numpy as np

image = cv2.imread("temp.jpg", cv2.IMREAD_GRAYSCALE) # 读取图片（以灰度图的形式读取）
cv2.imshow("Image", image) # 显示图片

print("The Shape is:", image.shape) # 显示图片的尺寸

template = image[310:350,380:405] # 第一个值是高（Y 的值），第二个值是宽

cv2.imshow("Template", template) # 显示模板
match = cv2.matchTemplate(image, template, cv2.TM_CCOEFF_NORMED)
# TM_CCOEFF_NORMED：相关系数匹配法，防止因为光照导致的结果不准确

locations = np.where(match >= 0.4) # 匹配系数 >= 0.9
high, weight = template.shape

for point in zip(*locations[::-1]):
    cv2.rectangle(image, point, (point[0] + 5, point[1] + 5), 255, 1)
  # 画矩形框：图片，左上角坐标，右下角坐标，颜色，线宽

cv2.imshow("Match", image)
cv2.waitKey()
```