广东工业大学计算机学院

《计算机视觉》

实验报告



|  |  |
| --- | --- |
| 课程名称： | 计算机视觉 |
| 实验名称： | 图像处理：空间域滤波（去模糊、锐化、去噪）、直方图均衡化、色彩空间转换 |
| 指导教师： | 姬玉柱 |
| 学生姓名： | 梁戬希 |
| 学 号： | 3121004914 |
| 实验日期： | 2024-05-24 |
| 实验地点： | 工一319 |
| 实验成绩： |  |

实验报告撰写要求

实验操作是教学过程中理论联系实际的重要环节，而实验报告的撰写又是知识系统化的吸收和升华过程，因此，实验报告应该体现完整性、规范性、正确性、有效性。现将实验报告撰写的有关内容说明如下：

1、 实验报告模板为电子版。

2、 下载统一的实验报告模板，学生自行完成撰写和打印。报告的首页包含本次实验的一般信息：

*  组 号：例如：2-5 表示第二班第5组。
*  实验日期：例如：05-10-06 表示本次实验日期。(年-月-日)……
*  实验编号：例如：No.1 表示第一个实验。
*  实验时间：例如：2学时 表示本次实验所用的时间。

实验报告正文部分，从六个方面（目的、内容、步骤等）反映本次实验的要点、要求以及完成过程等情况。模板已为实验报告正文设定统一格式，学生只需在相应项内填充即可。续页不再需要包含首页中的实验一般信息。

3、 实验报告正文部分具体要求如下：

一、实验目的

本次实验所涉及并要求掌握的知识点。

二、实验环境

实验所使用的设备名称及规格，网络管理工具简介、版本等。

三、实验内容与实验要求

实验内容、原理分析及具体实验要求。

四、实验过程与分析

根据具体实验，记录、整理相应命令、运行结果等，包括截图和文字说明。

详细记录在实验过程中发生的故障和问题，并进行故障分析，说明故障排除的过程及方法。

五、实验结果总结

对实验结果进行分析，完成思考题目，总结实验的心得体会，并提出实验的改进意见。

六、附录

注：本部分仅为展示报告内容，具体报告内容见下一页。

一、实验目的

**要求掌握的知识点：**

**一、线性平滑滤波（均值滤波及加权中值滤波）：**

线性滤波器包含在滤波器邻域内像素的平均值，也称为均值滤波器。

作用：减小图像灰度的“尖锐”变化，减小噪声。

由于图像边缘是由图像灰度尖锐变化引起的， 所以也存在边缘模糊的问题。

灰度等级的快速过渡部分减少。

平滑的最明显的应用是可以减少引起快速过渡的随机噪声。

副作用：边缘模糊。

主要用于减少图像“无关”细节。

kernel size不断增大，细节丢失越多。

**二、统计滤波器（最大、最小和中值滤波，去除椒盐噪声等情况）：**

统计排序滤波器是非线性滤波器；是基于滤波器区域内像素的排序。由排序结果决定的值代替中心像素的值。

中值滤波器主要用途：去除噪声

计算公式：R=mid{zk | k=1, 2, 3,…, n}

最大值滤波器

主要用途：寻找最亮点

计算公式：R=max{zk | k=1, 2, 3,…, n}

最小值滤波器

主要用途：寻找最暗点

计算公式：R=min{zk | k=1, 2, 3,…, n}

中值滤波器的原理：

在模板区域内像素的中间值，作为结果值

计算公式： R=mid{zk | k=1, 2, 3,…, n}

强迫突出的亮点（或暗点）更像它周围的值， 从而消除孤立的亮点（或暗点）。

中值滤波器的特点：

在去除噪音的同时，可以比较好地保留边的锐 度和图像的细节（优于均值滤波器）

能够有效去除脉冲噪声又称椒盐噪声（saltand-pepper noise）：以黑白点叠加在图像上。

最大值滤波器移除暗色像素值，最小值滤波器移除亮色像素值，可分别用于去除“椒噪声”和“盐噪声”。

**三、拉普拉斯锐化滤波：**

拉普拉斯锐化滤波是一种图像处理技术，主要用于增强图像的边缘特征，使图像的细节更加清晰。它基于拉普拉斯算子，其核心思想是通过高通滤波来增强图像中的高频分量（即边缘和细节部分）。

**四、直方图均衡化：**

基本思想：把原始图的直方图变换为均匀分布的形式，这样就增加了像素灰度值的动态 范围，从而达到增强图像整体对比度的效果。

数学表述：找到一种映射/转换函数T(r), 使得 修改（modified）/均衡化后（equalized）的图像其直方图分布趋近平坦的(均匀的)：s = T(r)。

由此可见，变换后变量s在其定义域内的概率密度是均匀分布的；

因此，用r的累积分布函数作为变换函数，可产生一幅灰度级分布具有均匀概率密度的图像；

其结果扩展了像素取值的动态范围；

数学角度：直方图均衡化处理是以累积分布函数变换法为基础的直方图修正法。

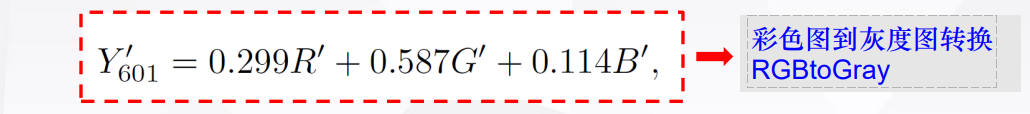
**五、YUV色彩空间变换：**

早期电视视频信号传输标准

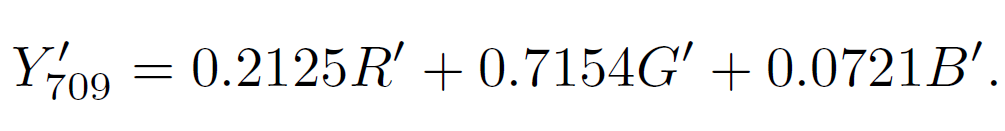
视频传输最早的YIQ，来自北美NTSC制式

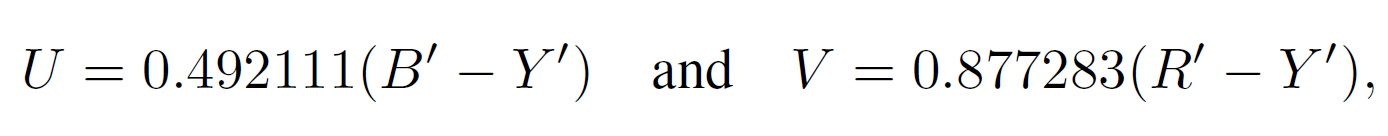
YUV标准由欧洲为PAL制式指定

（PAL）RGB（gamma矫正后）→ YUV：



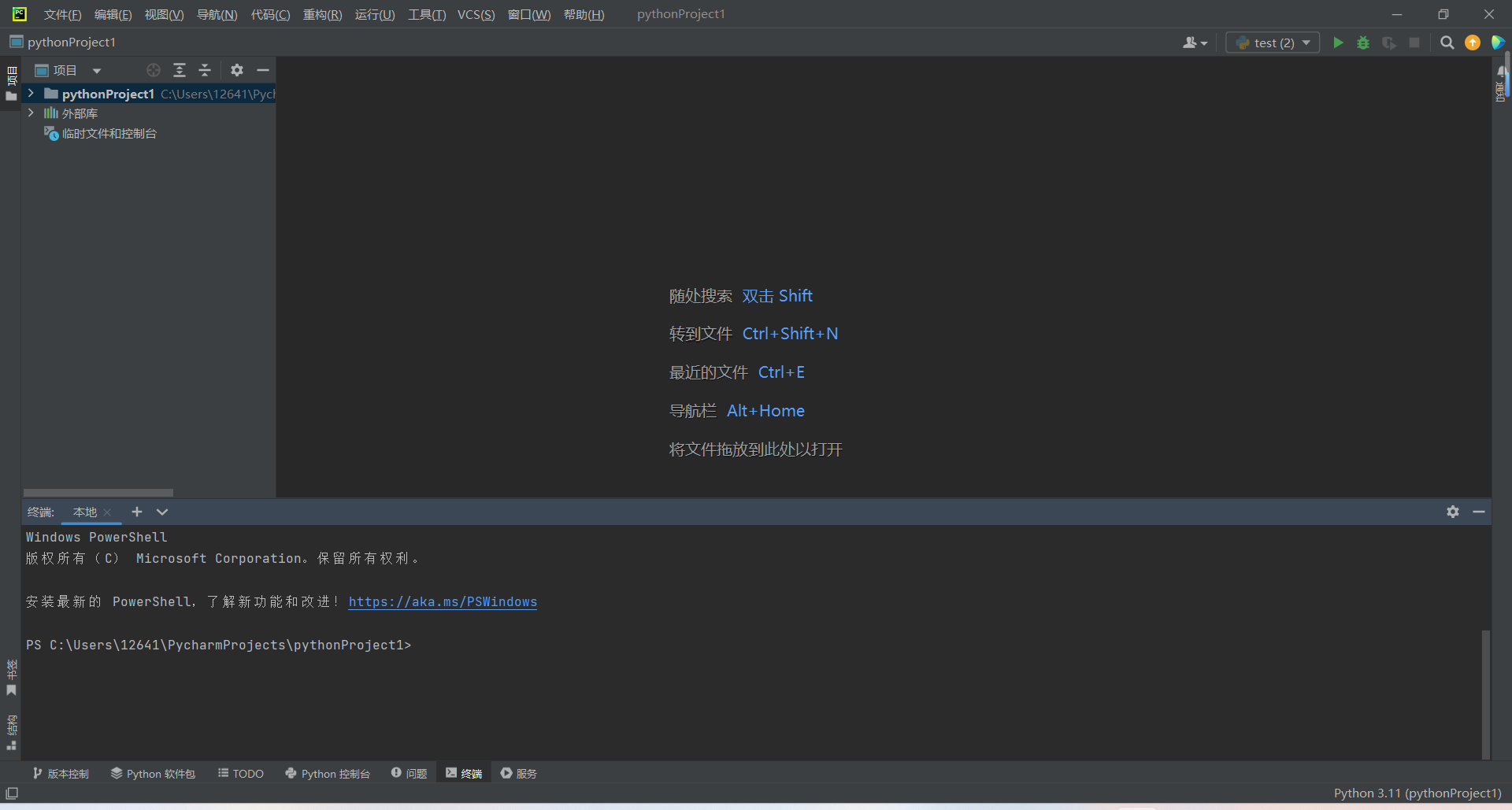
（HDTV）RGB（gamma矫正后）→YUV：

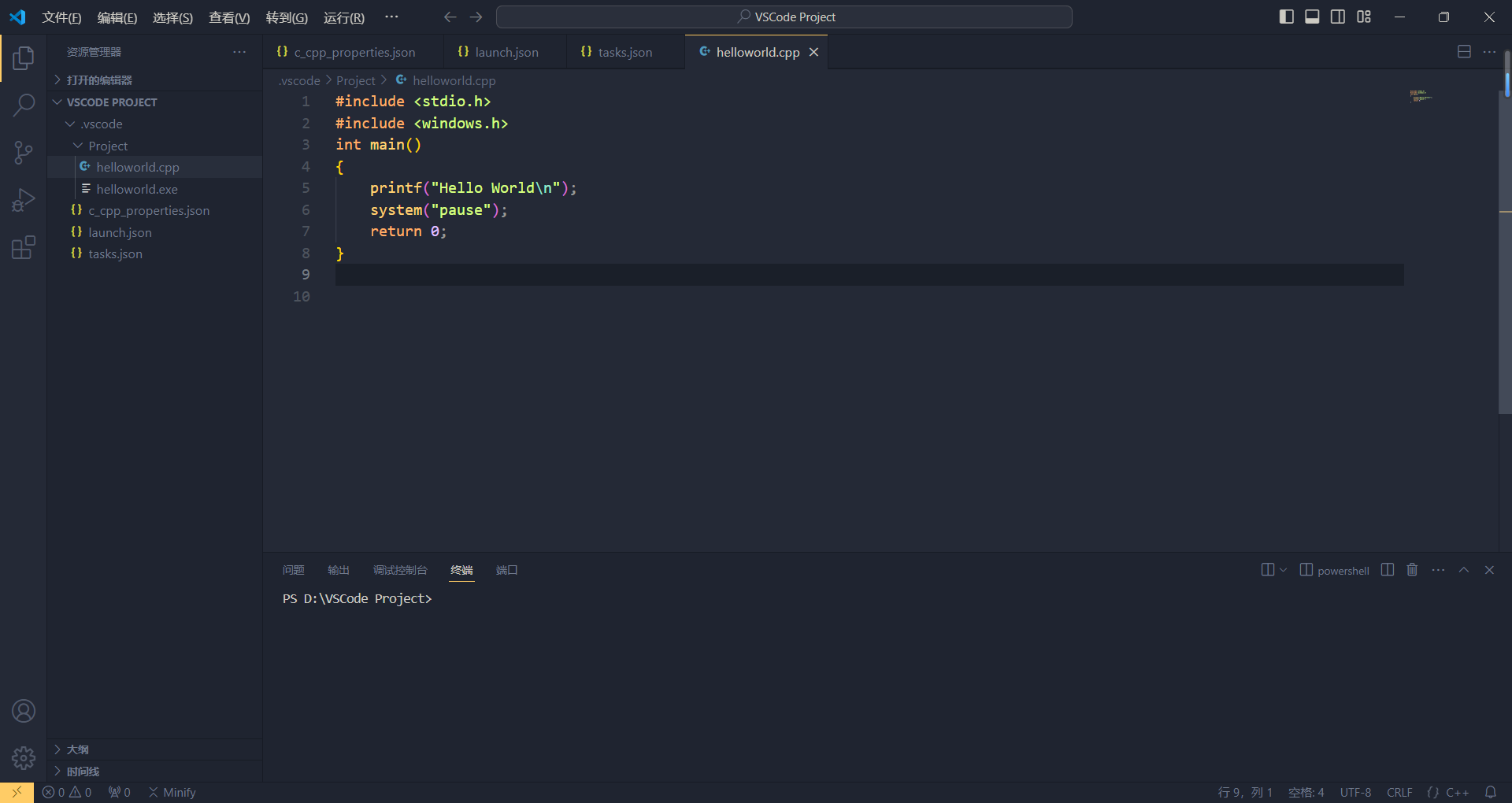




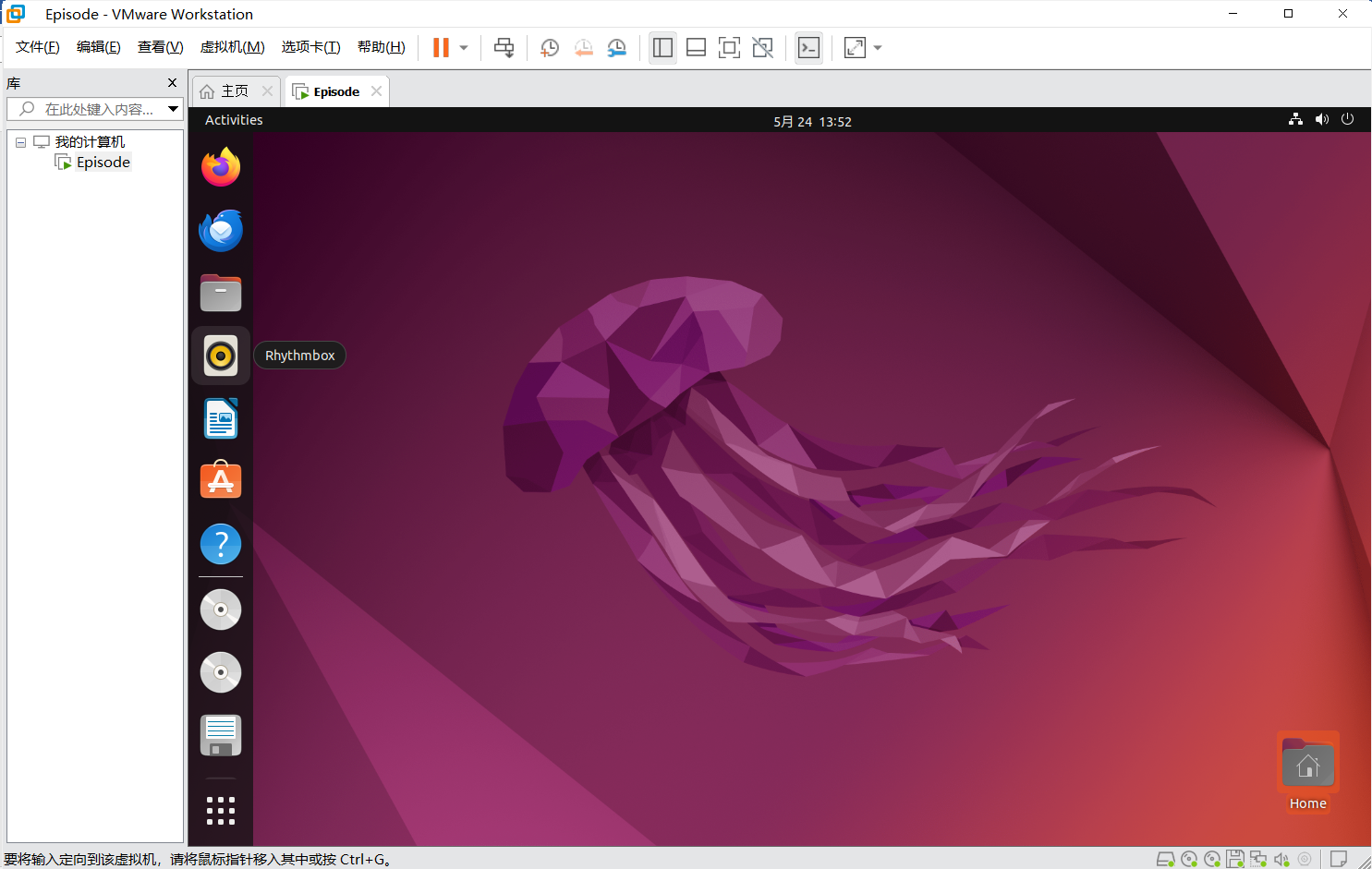
二、实验环境

PyCharm Community Edition:

VScode:



Ubuntu:

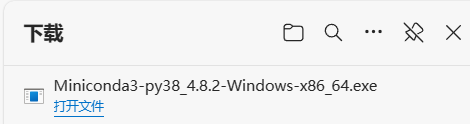


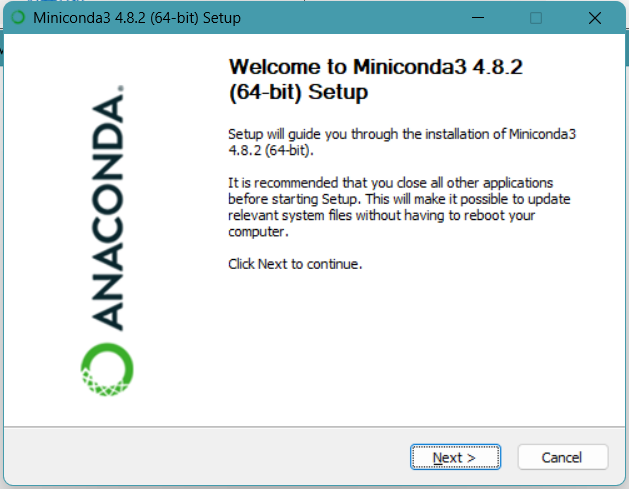
先前已安装好！

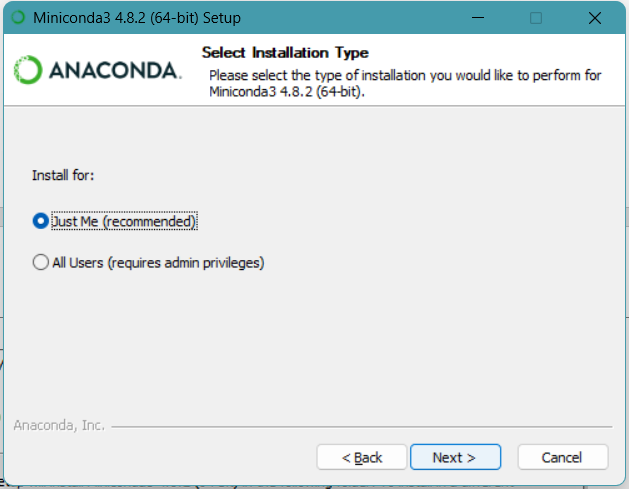
本次实验还需要安装Miniconda：

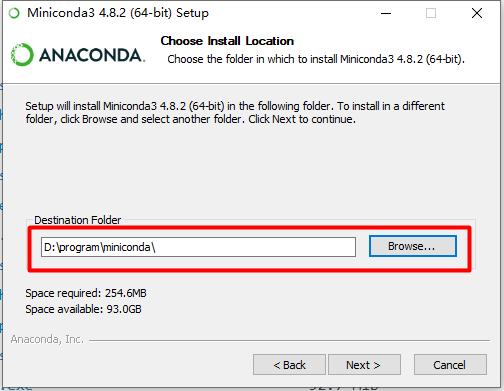
1、在清华源下载安装包并完成miniconda的安装：

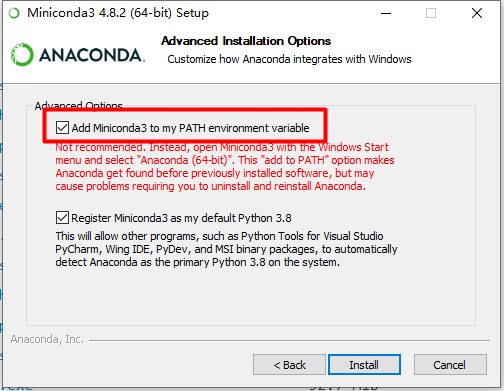


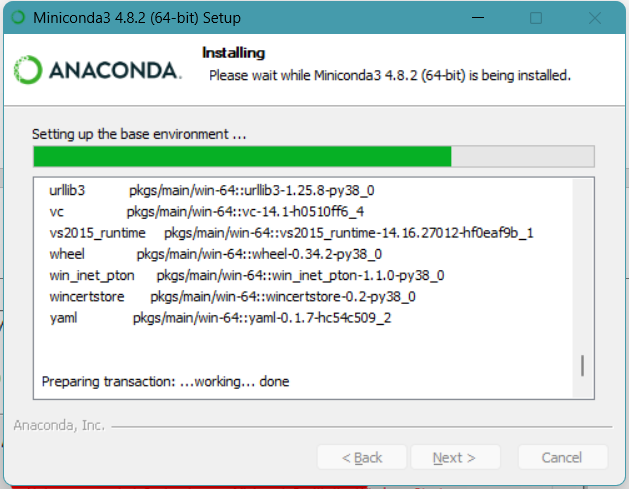


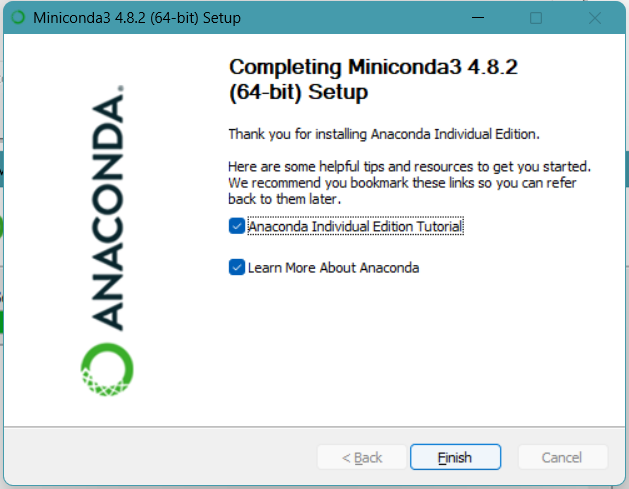






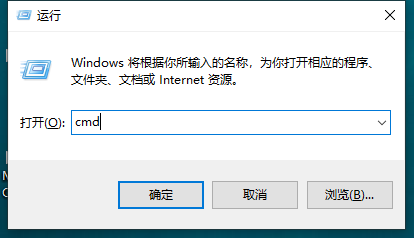






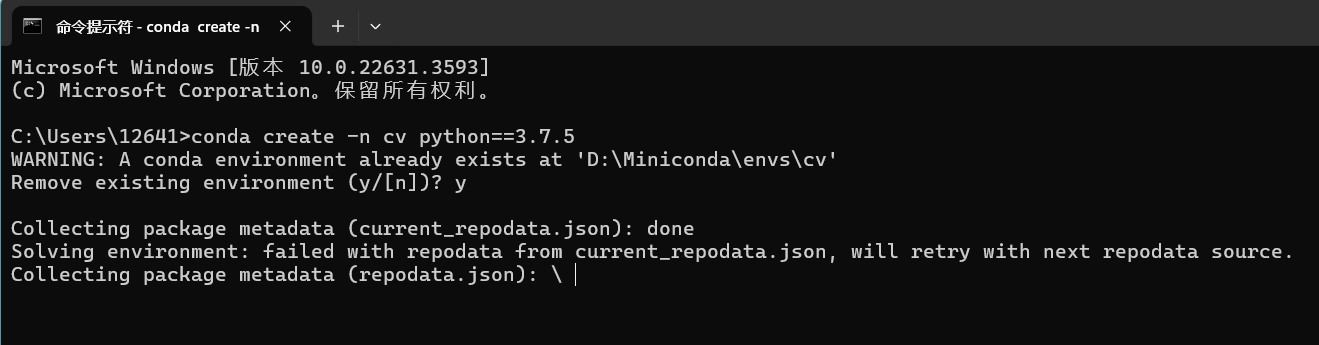
到此miniconda已安装完成！

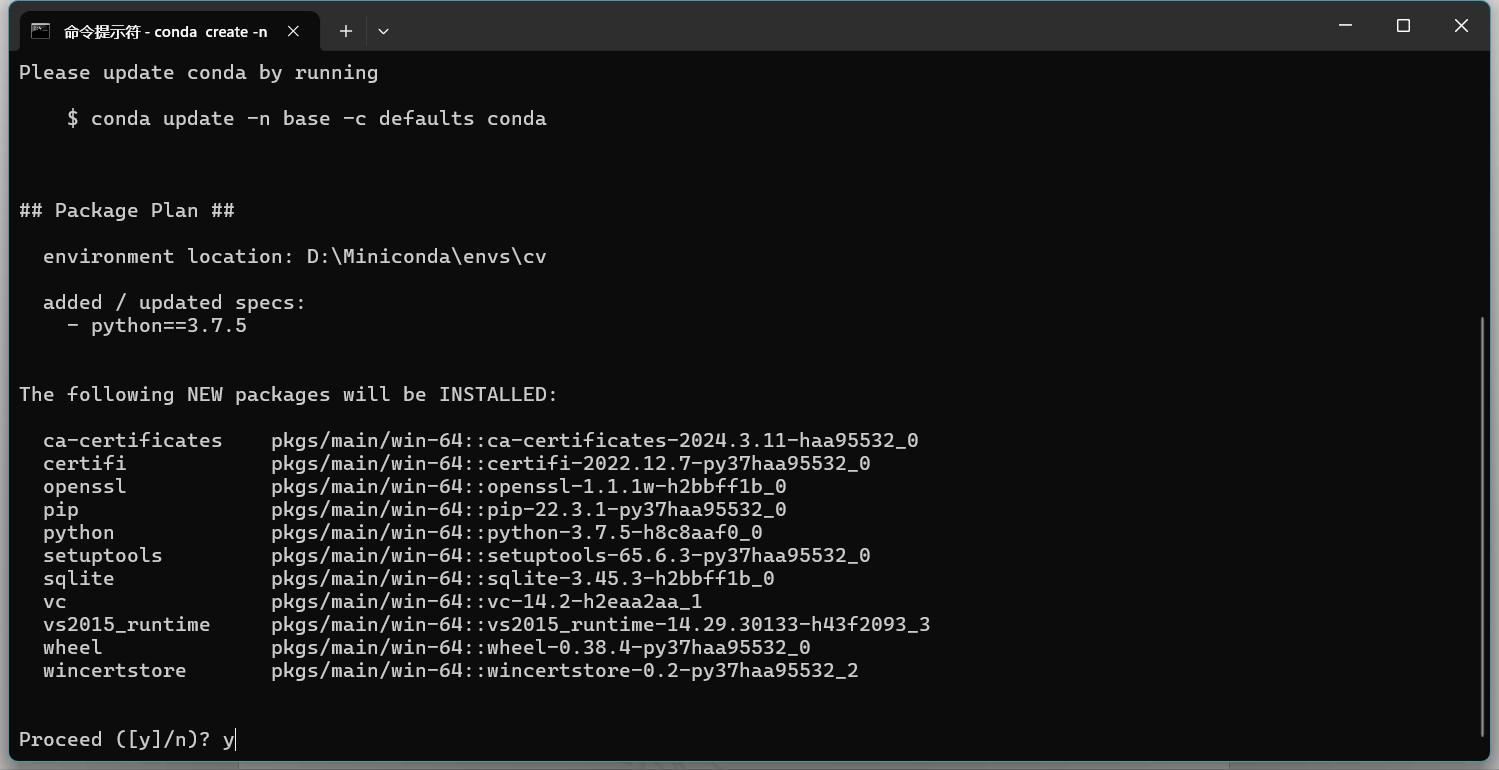
2、打开命令行窗口cmd：



输入以下命令创建虚拟环境，Python 版本 为 3.7.5，创建过程需要输入 y 确认

>> conda create -n cv python==3.7.5

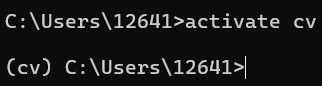






虚拟环境创建成功后输入对应名称即可进入对应虚拟环境

>> activate cv



3、pip换源：

Python 可以通过pip和conda两种方式来安装包，但是两者所安装的包并不完全兼容，在实际使用过程中建议只选择一种方式来安装包，本实验使用的是pip，但是由于pip的官方源在国外，直连速度较慢，因此需要换为国内的镜像源，于是我们通过修改pip的配置文件来永久性地使用清华镜像源。具体步骤如下：

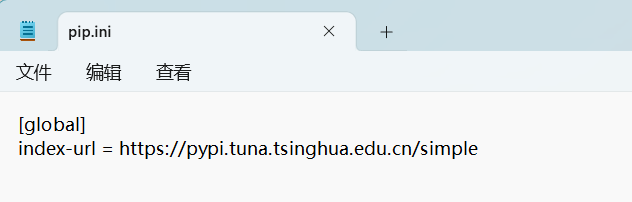
1. 创建或编辑配置文件：

对于Windows用户，配置文件路径是C:\Users\12641\AppData\Roaming\pip\pip.ini

1. 在配置文件中添加以下内容：

[global]

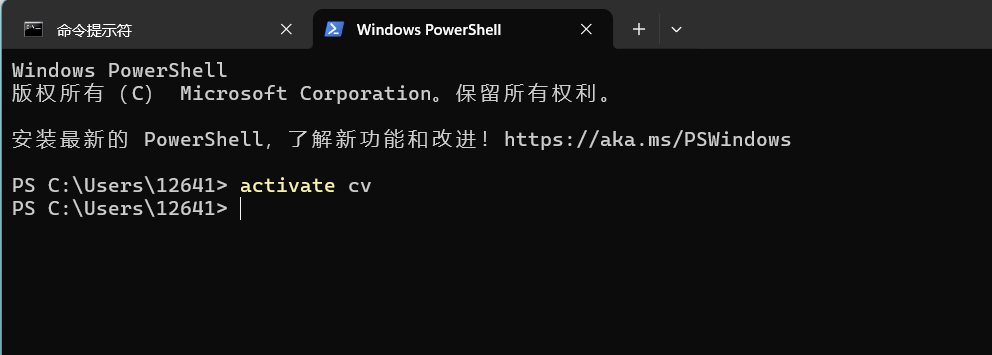
index-url = <https://pypi.tuna.tsinghua.edu.cn/simple>



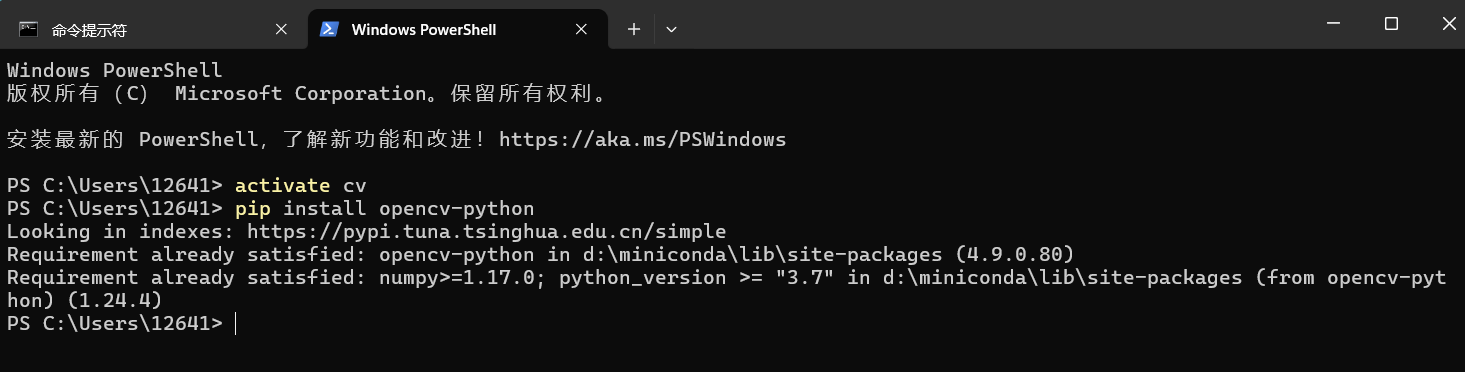
1. 这将把全局的包索引地址设置为清华镜像源。

4、新建一个命令行窗口，输入以下命令激活 cv 安装虚拟环境。

(Windows)：

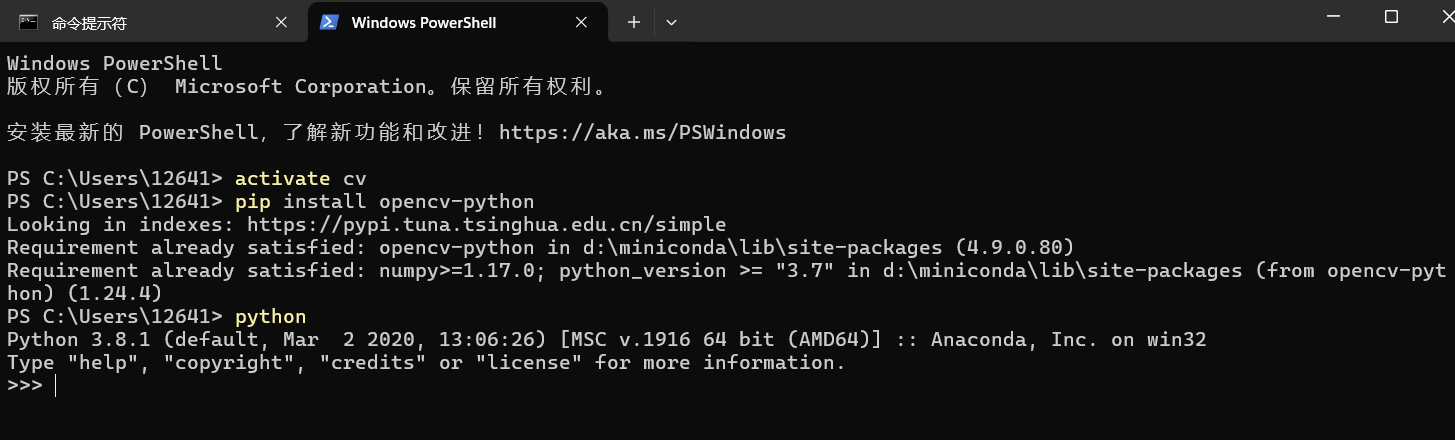


成功激活环境后，利用 pip 命令安装最新版本的opencv库



安装完毕后，可在命令行验证安装是否成功 ：

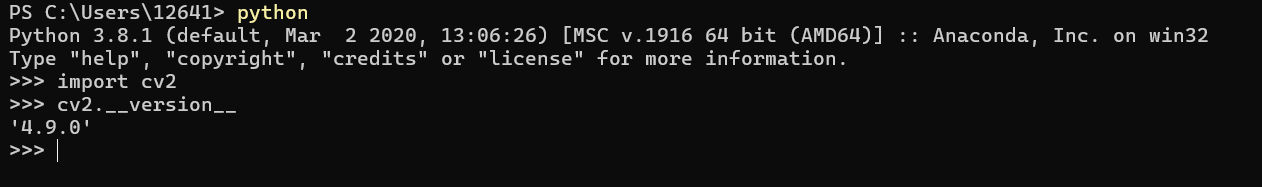
>> python



>>> import cv2

>>> cv2.\_\_version\_\_

安装正常并显示：'4.9.0'



至此环境配置完成！

三、实验内容与实验要求

**一、线性平滑滤波（均值滤波及加权中值滤波）：**

线性滤波器包含在滤波器邻域内像素的平均值，也称为均值滤波器。

作用：减小图像灰度的“尖锐”变化，减小噪声。

由于图像边缘是由图像灰度尖锐变化引起的， 所以也存在边缘模糊的问题。

灰度等级的快速过渡部分减少。

平滑的最明显的应用是可以减少引起快速过渡的随机噪声。

副作用：边缘模糊。

主要用于减少图像“无关”细节。

kernel size不断增大，细节丢失越多。

**二、统计滤波器（最大、最小和中值滤波，去除椒盐噪声等情况）：**

统计排序滤波器是非线性滤波器；是基于滤波器区域内像素的排序。由排序结果决定的值代替中心像素的值。

中值滤波器主要用途：去除噪声

计算公式：R=mid{zk | k=1, 2, 3,…, n}

最大值滤波器

主要用途：寻找最亮点

计算公式：R=max{zk | k=1, 2, 3,…, n}

最小值滤波器

主要用途：寻找最暗点

计算公式：R=min{zk | k=1, 2, 3,…, n}

中值滤波器的原理：

在模板区域内像素的中间值，作为结果值

计算公式： R=mid{zk | k=1, 2, 3,…, n}

强迫突出的亮点（或暗点）更像它周围的值， 从而消除孤立的亮点（或暗点）。

中值滤波器的特点：

在去除噪音的同时，可以比较好地保留边的锐 度和图像的细节（优于均值滤波器）

能够有效去除脉冲噪声又称椒盐噪声（saltand-pepper noise）：以黑白点叠加在图像上。

最大值滤波器移除暗色像素值，最小值滤波器移除亮色像素值，可分别用于去除“椒噪声”和“盐噪声”。

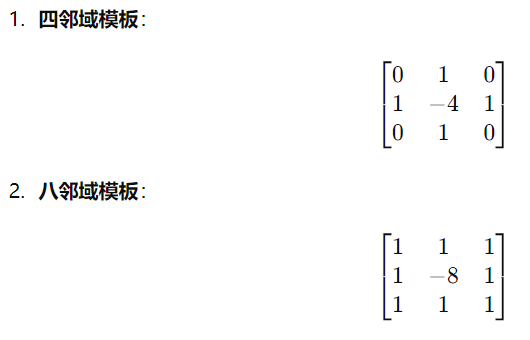
**三、拉普拉斯锐化滤波：**

拉普拉斯锐化滤波是一种图像处理技术，主要用于增强图像的边缘特征，使图像的细节更加清晰。它基于拉普拉斯算子，其核心思想是通过高通滤波来增强图像中的高频分量（即边缘和细节部分）。

拉普拉斯算子是一种二阶导数算子，用于检测图像中的边缘。它计算的是图像灰度值的二阶导数。对于二维图像，拉普拉斯算子定义为：



在离散情况下，常用的拉普拉斯模板包括以下几种形式：

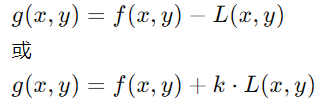


拉普拉斯锐化滤波的步骤如下：

计算图像的拉普拉斯变换：将图像与拉普拉斯算子模板进行卷积，得到拉普拉斯变换图像。



图像锐化：将原始图像与拉普拉斯变换图像相加，得到锐化后的图像。



其中 𝑘 是一个调整系数，通常取 𝑘=1。

通过上述步骤，图像的边缘得到了增强，而平滑区域的变化不大，从而达到了锐化的效果。

**四、直方图均衡化：**

基本思想：把原始图的直方图变换为均匀分布的形式，这样就增加了像素灰度值的动态 范围，从而达到增强图像整体对比度的效果。

数学表述：找到一种映射/转换函数T(r), 使得 修改（modified）/均衡化后（equalized）的图像其直方图分布趋近平坦的(均匀的)：s = T(r)。

s = T(r), 0 ≤ r ≤ 1

T(r)满足下列两个条件：

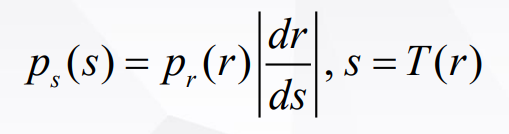
1）T(r)在区间0 ≤ r ≤ 1中，为单值且单调递增；

2）当0 ≤ r ≤ 1时，0 ≤ T(r) ≤ 1。

条件(1)：保证原图各灰度级在变换后仍保持从黑到白（或从白到黑）的排列次序；

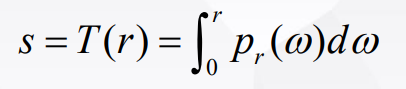
条件(2)：保证变换前后灰度值动态范围一致性。

设 pr (r) 和 ps (s) 分别表示随机变量r和s的概率密度分布。pr (r)和T(r)已知，且T -1 (s)满足条件 1，则ps (s) 能利用一个简单的公式获得，即：



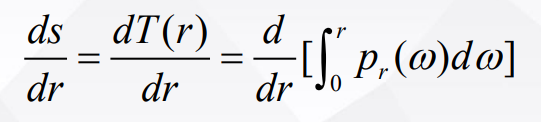
因此，s的概率密度函数（PDF）由输入图像的灰度等级的PDF以及选择的变换函数决定。

选择下面的公式作为变换函数：

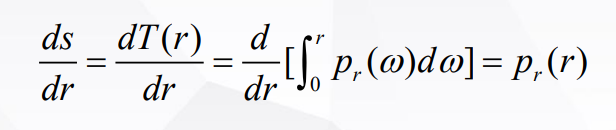


该变换满足上述两个条件： 1）T(r)在区间0 ≤ r ≤ 1中，为单值且单调递增；

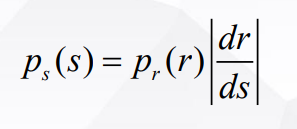
2）当0≤r≤1时，0 ≤ T(r) ≤ 1。



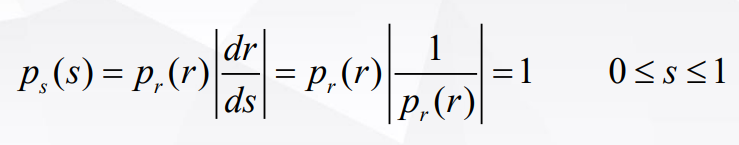
根据莱布尼茨定理，关于上限的定积分的导数 就是该上限的积分值，所以有:



而：



带入得：



由此可见，变换后变量s在其定义域内的概率密度是均匀分布的；

因此，用r的累积分布函数作为变换函数，可产生一幅灰度级分布具有均匀概率密度的图像；

其结果扩展了像素取值的动态范围；

数学角度：直方图均衡化处理是以累积分布函数变换法为基础的直方图修正法。

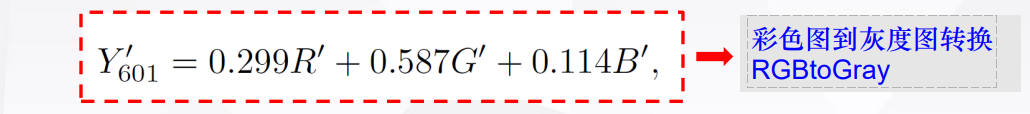
**五、YUV色彩空间变换：**

早期电视视频信号传输标准

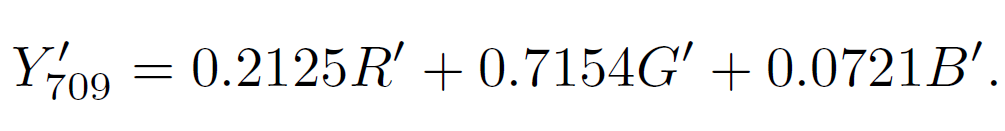
视频传输最早的YIQ，来自北美NTSC制式

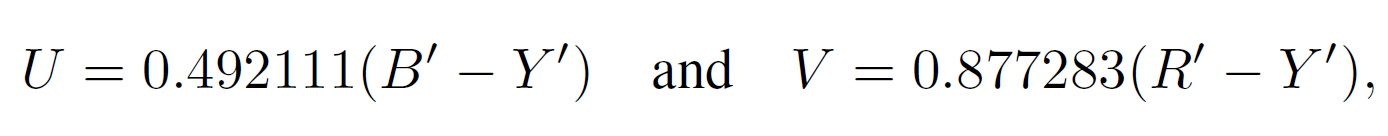
YUV标准由欧洲为PAL制式指定

（PAL）RGB（gamma矫正后）→ YUV：



（HDTV）RGB（gamma矫正后）→YUV：





四、实验过程与分析

*实验内容、原理分析及具体实验要求。根据具体实验，记录、整理相应命令、运行结果等，包括截图和文字说明。可详细记录在实验过程中发生的问题，并进行分析，说明排除的过程及方法。（可多页）*

1. **空间域滤波实验：**

首先将代码填写完整，然后测试线性平滑滤波（均值滤波及加权中值滤波）、统计滤波器（最大、最小和中值滤波，去除 椒盐噪声等情况）、拉普拉斯锐化滤波的效果：

1、我们选取图片1\_smooth.jpg作为平滑低通滤波的测试，运行得到图片1\_enhanced.jpg，

代码中的PSNR 值是图像处理中一个重要的指标，用于评估图像处理后的质量。在这个例子中，PSNR 值为36.55，表示原始图像和经过平滑低通滤波处理后的图像之间的峰值信噪比。PSNR 值越高，表示两幅图像之间的差异越小，即处理后的图像与原始图像之间的失真程度越低。PSNR 值为36.55说明处理后的图像相对于原始图像来说，保留了较高的信息量，因为它们之间的峰值信噪比很高。这意味着平滑低通滤波有效地降低了图像中的噪声，并且在这个滤波过程中保留了原始图像的细节和特征。



图1：1\_smooth.jpg



图2：1\_enhanced.jpg

2、将平滑后的图像1\_enhanced.jpg进行锐化高通滤波，得到图片1\_sharpened.jpg，且可以看到PSNR (original vs. sharpened):为32.16，这个数据说明经过锐化高通滤波后的图像相对于原始图像来说，存在一定程度的信息损失，因为它们之间的峰值信噪与平滑低通滤波相较之下比较低。这意味着锐化高通滤波增强了图像的边缘和细节，但也引入了一些额外的噪声或者细微的失真。



图3：1\_sharpened.jpg

3、利用均值、中值、最大值、最小值对椒盐噪声图像进行去噪并查看结果

此处我们选取2.jpg作为处理对象，因为它具有椒盐噪声，适合进行去噪处理，首先我们使用getGrayImg 函数来将彩色图像转换为灰度图像，然后我们用均值、中值、最大值、最小值对椒盐噪声图像2.jpg进行去噪，得到结果：

PSNR (noisy vs. denoised mean): 12.522393185672465

PSNR (noisy vs. denoised median): 11.963458726949199

PSNR (noisy vs. denoised max): 7.1707556911529045

PSNR (noisy vs. denoised min): 5.86131039265061

因此我们可以发现，均值、中值滤波的PSNR值较高，说明均值滤波器和中值滤波器在这次处理中的去噪效果相对较好，虽然仍有一定的噪声残留，但相比于其他两种滤波器保留了更多的图像细节，但PSNR 值并不总是能完全反映去噪效果，从视觉上看，中值滤波器在去除噪音的同时，可以比较好地保留边的锐度和图像的细节（优于均值滤波器，均值滤波器容易使图像模糊，这从下图中也可以看出）。

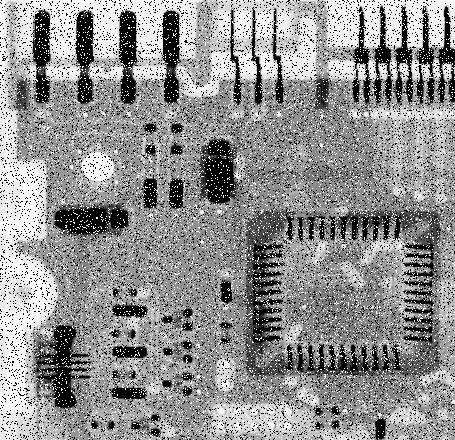


图4：2.jpg



图5：2\_denoised\_max.jpg



图6：2\_denoised\_mean.jpg

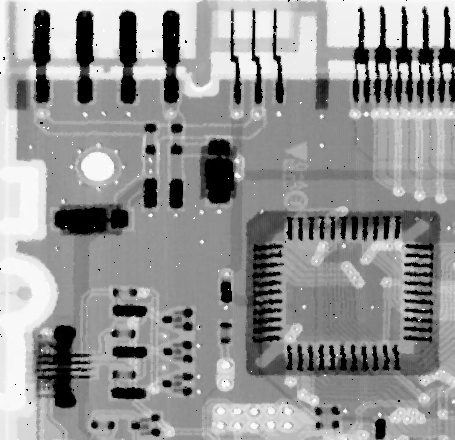


图7：2\_denoised\_median.jpg

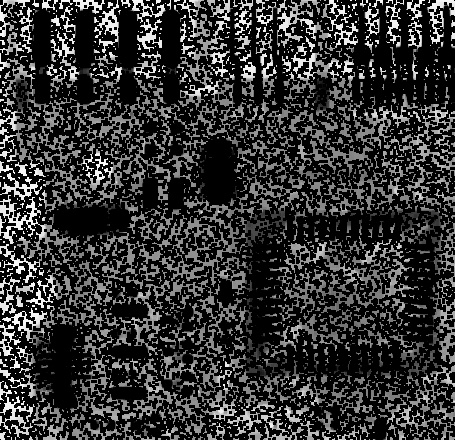


图8：2\_denoised\_min.jpg

而且：从图5、8可以看出，最大值滤波器可以去除椒噪声（所以剩下了盐噪声，图片白色较明显，使图像看起来更亮），最小值滤波器可以去除盐噪声（所以剩下了椒噪声，图片黑色较明显，使图像看起来更暗），因此可以说明：最大值滤波器移除暗色像素值，最小值滤波器移除亮色像素值。

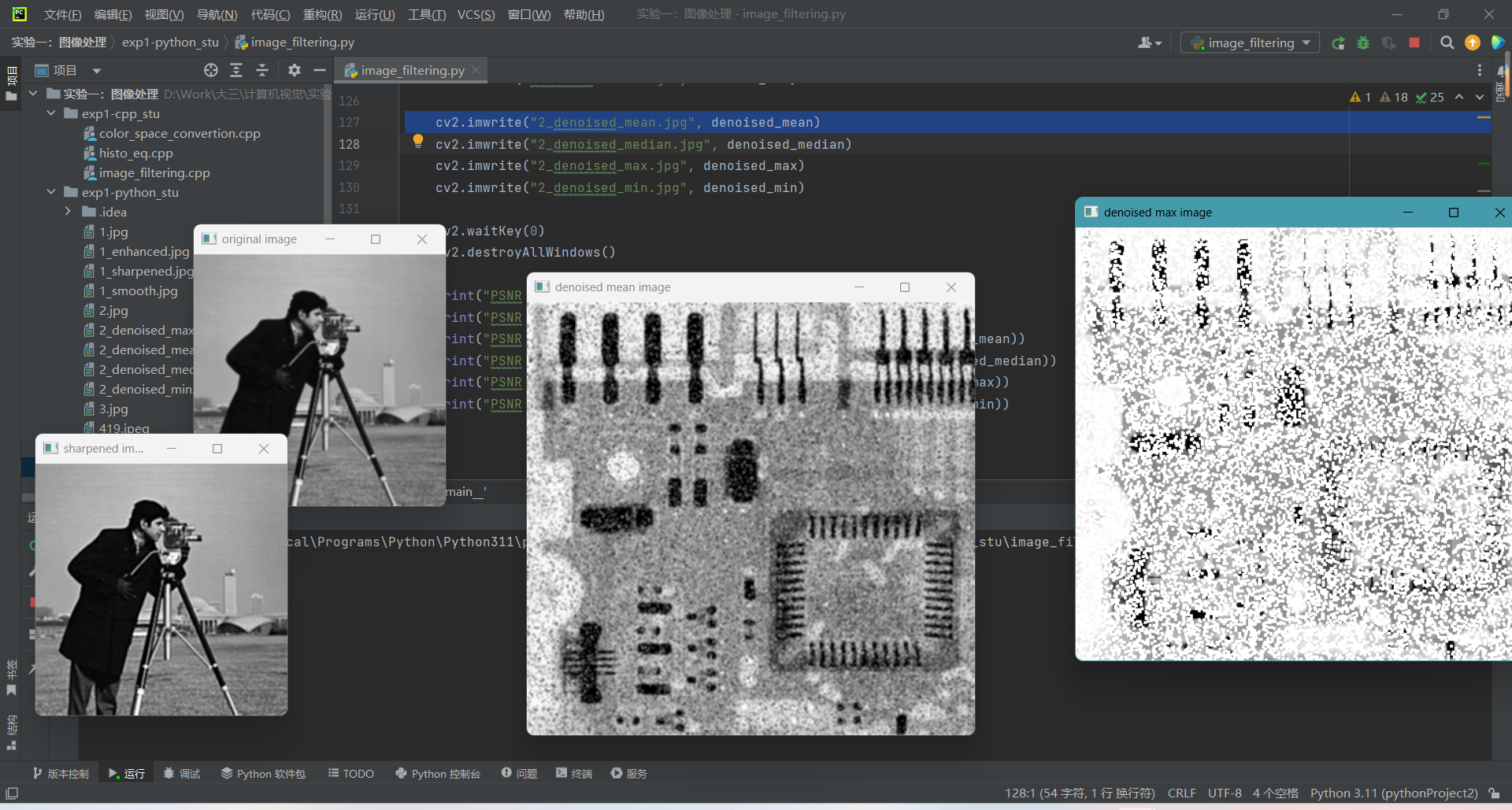


图9：运行过程1

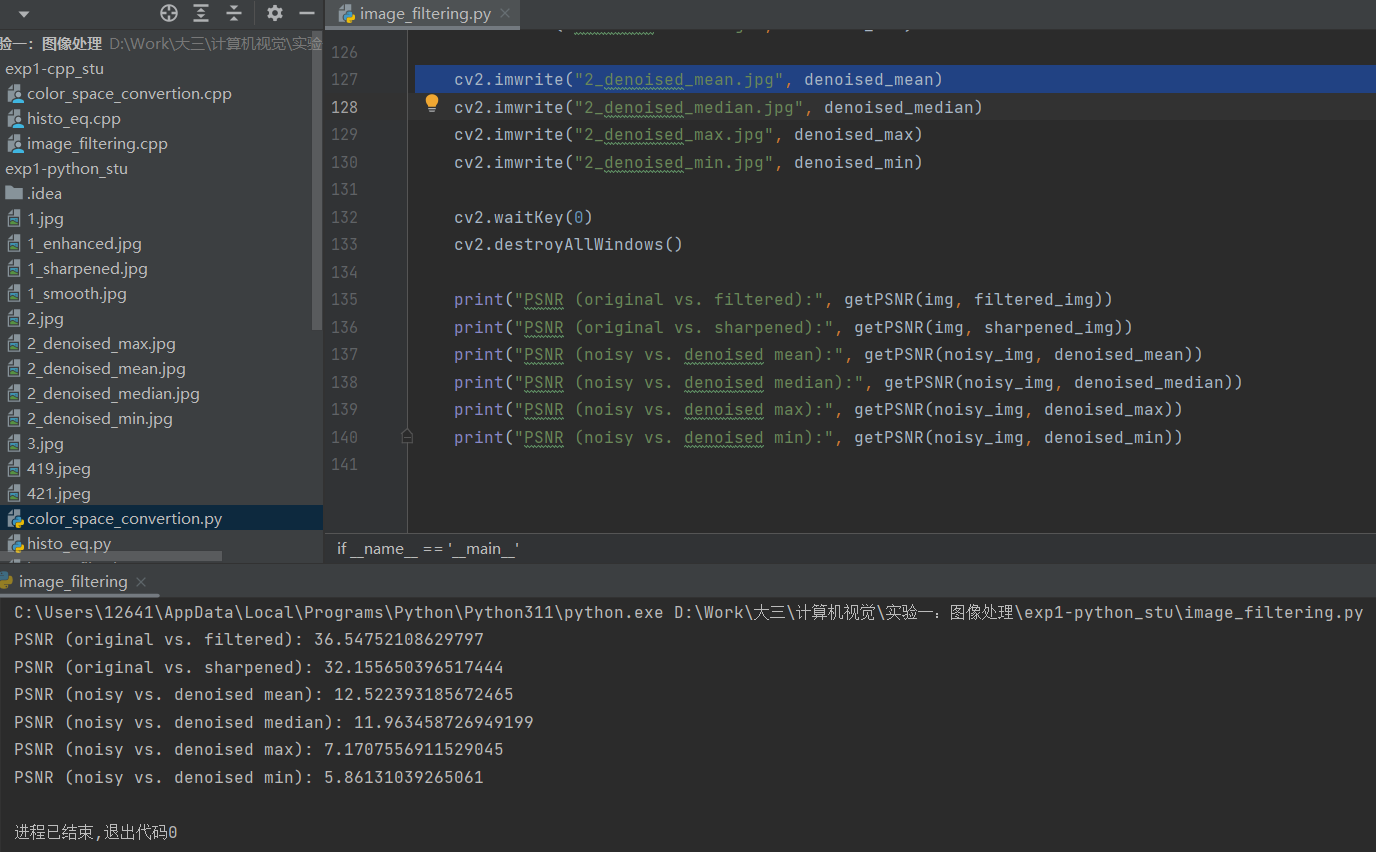


图10：运行过程2

1. **直方图均衡化实验：**

首先将代码填写完整，然后运行代码进行直方图均衡化实验：

1、获取Lena.jpg的灰度图像，然后使用get\_histogram函数计算灰度图像的直方图及其累积概率分布，生成新的像素值映射表，使用image\_equalization函数生成累积概率分布映射表 **S** 对灰度图像进行直方图均衡化，将原图像中的每个像素值替换为映射表中的对应值，从而实现图像的均衡化，从而得到处理后的图像LenaRGBLow1\_enhanced.jpg，这使得原图像的对比度得到了有效增强，尤其是在原始图像对比度较低的情况下，这种增强效果更加明显。



图11：Lena.jpg



图12：LenaRGBLow1\_enhanced.jpg

2、运行代码得到PSNR值 4.17，表示经过直方图均衡化处理后的图像与原始图像之间有显著差异，这是因为直方图均衡化可以显著增强图像的对比度，使得暗部和亮部的细节更加明显；可以使得视觉效果改善，图像更加清晰和锐利。

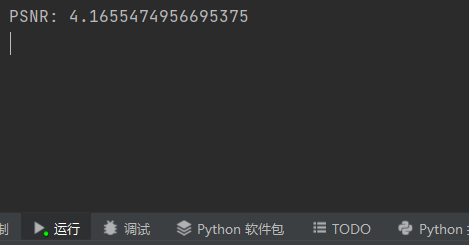


图13：PSNR运行结果

3、在运行后得到原始灰度图，以及经过直方图均衡化之后的图像以及它们所对应的直方图。

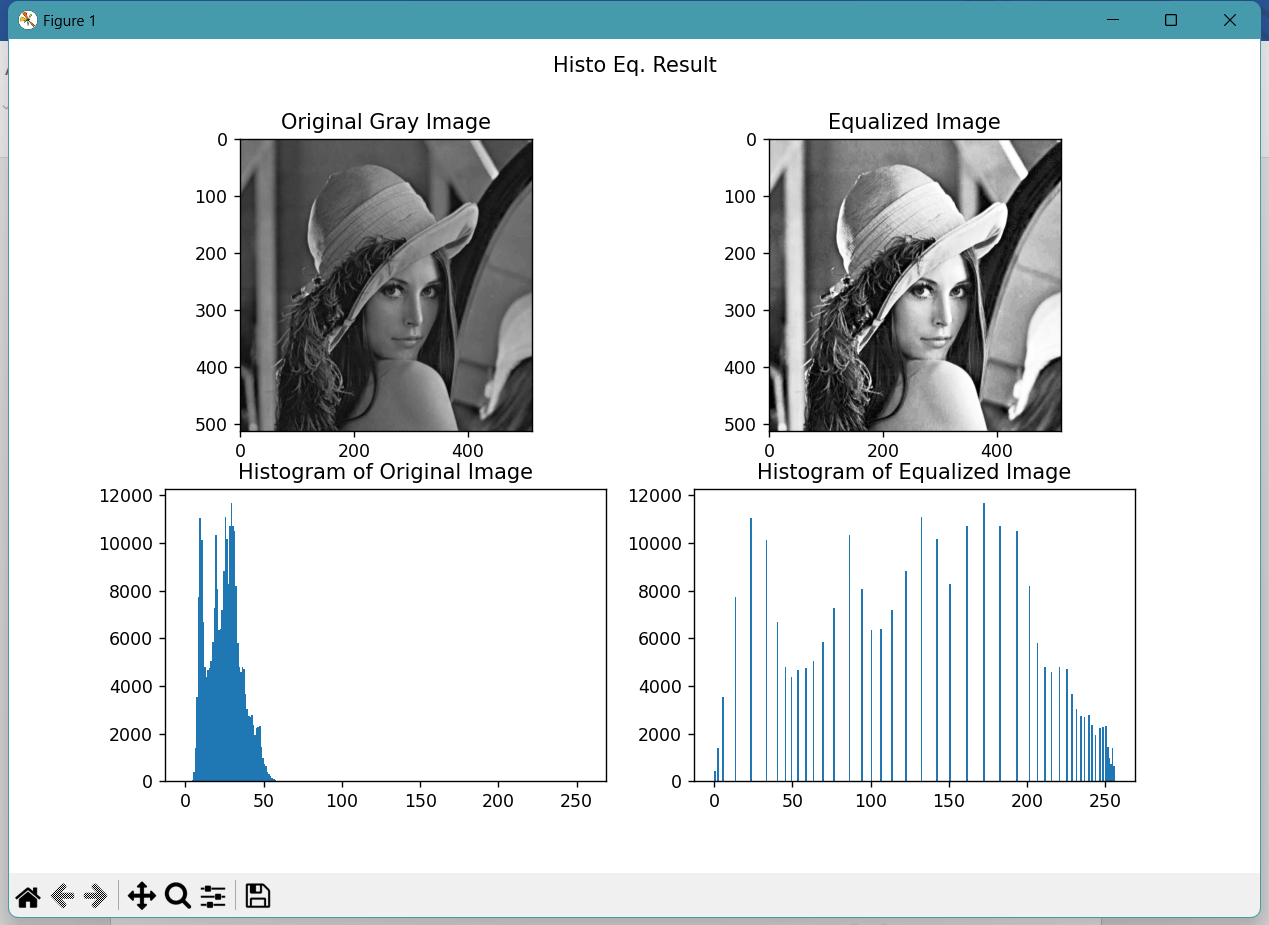


图14：图像及其直方图表示

**A、第一个子图（左上角）**：显示原始灰度图像。

**B、第二个子图（右上角）**：显示经过直方图均衡化处理后的图像，即 Equalized Image。

**C、第三个子图（左下角）**：显示原始图像的灰度直方图，展示了原始图像中各灰度级的像素分布情况。

**D、第四个子图（右下角）**：显示经过直方图均衡化处理后的图像的灰度直方图，展示了均衡化后图像中各灰度级的像素分布情况。

通过这些子图，我们可以直观地比较原始图像和均衡化后的图像之间的差异，以及它们的灰度分布情况。一般来说，直方图均衡化会使得图像的灰度分布更加均匀，对比度增强，因此你可能会看到均衡化后图像的直方图在整个灰度范围内更加平均分布，而原始图像的直方图可能会偏向某些灰度值区间。

1. **YUV色彩空间变换实验：**

首先将代码填写完整，然后运行代码进行YUV色彩空间变换实验：

1、本部分主要对色彩空间变换进行实验 。编辑 color\_space\_conversion.py 文件，实现 RGB 到 YUV 空间的转换，并通过设置 lightness\_en 增强亮度分量 Y，并转换回 RGB 空间查看增强效果。

2、调用RGB2YUV\_enhance函数将"Lena.jpg"对应的RGB图像转换为 YUV 色彩空间，并增强亮度分量 Y，同时输出Y、U、V图像到屏幕中。Y、U、V 图像是分别表示图像亮度（Luminance）和色度（Chrominance）的三个通道图像。在彩色图像中，Y 通道代表亮度信息，而 U 和 V 通道则代表颜色信息。

Y（亮度）通道：包含图像的灰度信息，表示图像的明暗度。Y 通道在灰度图像中与原始图像相同，而在彩色图像中则包含了彩色信息的亮度成分。

U、V（色度）通道：包含了图像的色彩信息。U 通道表示蓝色色度（Blue Chrominance），V 通道表示红色色度（Red Chrominance）。这两个通道描述了图像中颜色的差异和饱和度，允许我们区分不同颜色。

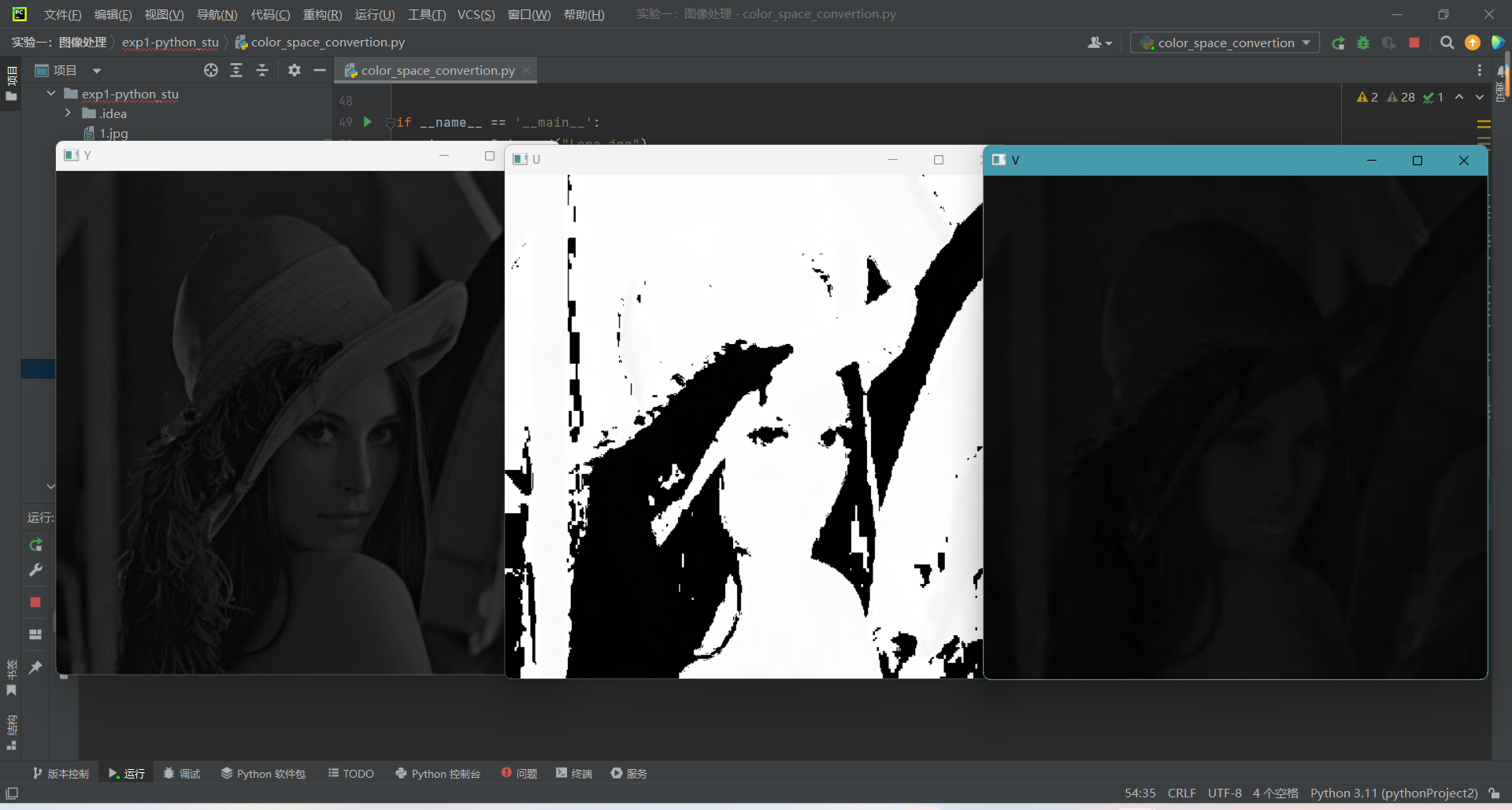


图15：Y、U、V图像

3、代码接着运行，输出Enhanced Light即enhanced\_image.jpg图像，该图像是是通过增强图像的亮度通道 Y 而得到的图像。在 RGB 到 YUV 空间转换后，我们增强了 Y 通道，然后将其转换回 RGB 空间，生成了增强亮度的图像。因此，enhanced\_image.jpg 图像显示了原始图像中更明亮的部分。



图16：enhanced\_image.jpg

通过此实验我们可以分析出颜色空间转换后进行图像增强的原理：

1. **颜色空间转换**：将 RGB 图像转换为 YUV（亮度、色度）空间。Y 通道代表图像的亮度信息，而 U 和 V 通道代表图像的色度信息。
2. **增强亮度**：在 YUV 空间中，我们增强了亮度通道 Y。通过缩放 Y 通道的值，可以增加或减少图像的整体亮度。在这里，我们通过乘以一个增强系数来增强 Y 通道。
3. **转换回 RGB 空间**：增强了 Y 通道后，我们将 YUV 图像转换回 RGB 空间。这样做可以保留图像的色彩信息，并在图像增强后生成最终的增强图像。

这种图像增强的原理是通过调整图像的亮度通道来改变图像的整体亮度，从而使图像看起来更明亮或更暗。

五、实验结果总结

*对实验结果进行分析，完成思考题目，总结实验的心得体会，并提出实验的****改进意见****。（可多页）*

六、附录

**1、空间域滤波实验完整代码：**

import numpy as np  
import cv2  
import math  
import os  
  
# average smoothing kernel  
averageKernel = np.array([[1 / 9, 1 / 9, 1 / 9],  
 [1 / 9, 1 / 9, 1 / 9],  
 [1 / 9, 1 / 9, 1 / 9]]).astype(np.float32)  
  
# gaussian smoothing kernel  
weightedAverageKernel = np.array([[1 / 16, 2 / 16, 1 / 16],  
 [2 / 16, 4 / 16, 2 / 16],  
 [1 / 16, 2 / 16, 1 / 16]]).astype(np.float32)  
  
# sharpen kernel  
laplacianKernel = np.array([[0.0, -1.0, 0.0],  
 [-1.0, 5.0, -1.0],  
 [0.0, -1.0, 0.0]]).astype(np.float32)  
  
  
def getGrayImg(img):  
 gray = np.zeros((img.shape[0], img.shape[1]), np.uint8)  
 timg = img.astype(np.float32)  
 for i in range(timg.shape[0]):  
 for j in range(timg.shape[1]):  
 # R\*0.299 + G\*0.587 + B\*0.114  
 gray\_intensity = timg[i][j][0] \* 0.114 + timg[i][j][1] \* 0.587 + timg[i][j][2] \* 0.299  
 gray[i][j] = np.round(gray\_intensity).astype(np.uint8)  
 return gray  
  
  
def paddingWithZero(img):  
 padding\_img = np.zeros((img.shape[0] + 2, img.shape[1] + 2), np.uint8)  
 padding\_img[1: img.shape[0] + 1, 1: img.shape[1] + 1] = img  
 return padding\_img  
  
  
def paddingWithNeighbor(img):  
 padding\_img = np.zeros((img.shape[0] + 2, img.shape[1] + 2), np.uint8)  
 padding\_img[1: img.shape[0] + 1, 1: img.shape[1] + 1] = img  
 for i in range(1, img.shape[0] + 1):  
 padding\_img[i][0] = img[i - 1][0] # 第一列  
 padding\_img[i][img.shape[1] + 1] = img[i - 1][img.shape[1] - 1] # 最后一列  
  
 for i in range(1, img.shape[1] + 1):  
 padding\_img[0][i] = img[0][i - 1] # 第一行  
 padding\_img[img.shape[0] + 1][i] = img[img.shape[0] - 1][i - 1] # 最后一行  
 return padding\_img  
  
  
def Filtering2D(img, filter):  
 # 申请变量, 存储输出图像大小  
 filtered\_img = np.zeros((img.shape[0] - 2, img.shape[1] - 2), np.uint8)  
 # img 转变为float 类型  
 img = img.astype(np.float32)  
 for i in range(0, filtered\_img.shape[0]):  
 for j in range(0, filtered\_img.shape[1]):  
 # 这里编程实现统计滤波公式  
 pixel = np.sum(img[i:i + 3, j:j + 3] \* filter)  
 filtered\_img[i][j] = np.clip(pixel, 0.0, 255.0).astype(np.uint8)  
 return filtered\_img  
  
  
def denoisewithOrderStatisticsFilter(img, filter\_type='median'):  
 filtered\_img = np.zeros((img.shape[0] - 2, img.shape[1] - 2), np.uint8)  
 for i in range(0, filtered\_img.shape[0]):  
 for j in range(0, filtered\_img.shape[1]):  
 window = img[i:i + 3, j:j + 3].flatten()  
 if filter\_type == 'median':  
 pixel = np.median(window)  
 elif filter\_type == 'mean':  
 pixel = np.mean(window)  
 elif filter\_type == 'max':  
 pixel = np.max(window)  
 elif filter\_type == 'min':  
 pixel = np.min(window)  
 else:  
 raise ValueError("Invalid filter type specified")  
 filtered\_img[i][j] = pixel  
 return filtered\_img  
  
  
def getPSNR(ori\_img, en\_img):  
 MAX = 255  
 total = 0  
 ori\_img = ori\_img.astype(np.float32)  
 en\_img = en\_img.astype(np.float32)  
 for i in range(ori\_img.shape[0]):  
 for j in range(ori\_img.shape[1]):  
 total = total + (ori\_img[i][j] - en\_img[i][j]) \*\* 2  
 MSE = total / (ori\_img.shape[0] \* ori\_img.shape[1])  
 PSNR = 10 \* math.log(MAX \* MAX / MSE, 10)  
 return PSNR  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 # 1. 从test文件夹中选一张图进行平滑低通滤波  
 img = cv2.imread("1\_smooth.jpg")  
 img = getGrayImg(img)  
 cv2.imshow('original image', img)  
 img\_padding = paddingWithNeighbor(img)  
 filtered\_img = Filtering2D(img\_padding, weightedAverageKernel)  
 cv2.imshow('filtered image', filtered\_img)  
 cv2.imwrite("1\_enhanced.jpg", filtered\_img)  
  
 # 2. 将平滑后的图像进行锐化高通滤波查看结果  
 sharpened\_img = Filtering2D(paddingWithNeighbor(filtered\_img), laplacianKernel)  
 cv2.imshow('sharpened image', sharpened\_img)  
 cv2.imwrite("1\_sharpened.jpg", sharpened\_img)  
  
 # 3. 利用均值、中值、最大值、最小值对椒盐、椒、盐噪声图像进行去噪并查看结果  
 noisy\_img = cv2.imread("2.jpg")  
 noisy\_img = getGrayImg(noisy\_img)  
 cv2.imshow('noisy image', noisy\_img)  
  
 denoised\_mean = denoisewithOrderStatisticsFilter(paddingWithNeighbor(noisy\_img), 'mean')  
 denoised\_median = denoisewithOrderStatisticsFilter(paddingWithNeighbor(noisy\_img), 'median')  
 denoised\_max = denoisewithOrderStatisticsFilter(paddingWithNeighbor(noisy\_img), 'max')  
 denoised\_min = denoisewithOrderStatisticsFilter(paddingWithNeighbor(noisy\_img), 'min')  
  
 cv2.imshow('denoised mean image', denoised\_mean)  
 cv2.imshow('denoised median image', denoised\_median)  
 cv2.imshow('denoised max image', denoised\_max)  
 cv2.imshow('denoised min image', denoised\_min)  
  
 cv2.imwrite("2\_denoised\_mean.jpg", denoised\_mean)  
 cv2.imwrite("2\_denoised\_median.jpg", denoised\_median)  
 cv2.imwrite("2\_denoised\_max.jpg", denoised\_max)  
 cv2.imwrite("2\_denoised\_min.jpg", denoised\_min)  
  
 cv2.waitKey(0)  
 cv2.destroyAllWindows()  
  
 print("PSNR (original vs. filtered):", getPSNR(img, filtered\_img))  
 print("PSNR (original vs. sharpened):", getPSNR(img, sharpened\_img))  
 print("PSNR (noisy vs. denoised mean):", getPSNR(noisy\_img, denoised\_mean))  
 print("PSNR (noisy vs. denoised median):", getPSNR(noisy\_img, denoised\_median))  
 print("PSNR (noisy vs. denoised max):", getPSNR(noisy\_img, denoised\_max))  
 print("PSNR (noisy vs. denoised min):", getPSNR(noisy\_img, denoised\_min))

**2、直方图均衡化实验完整代码：**

import cv2  
import numpy as np  
import math  
import matplotlib.pyplot as plt  
  
  
def getGrayImg(img):  
 gray = np.zeros((img.shape[0], img.shape[1]), np.uint8)  
 timg = img.astype(np.float32)  
 for i in range(timg.shape[0]):  
 for j in range(timg.shape[1]):  
 # R\*0.299 + G\*0.587 + B\*0.114  
 gray\_intensity = timg[i][j][2] \* 0.299 + timg[i][j][1] \* 0.587 + timg[i][j][0] \* 0.114  
 gray[i][j] = np.round(gray\_intensity).astype(np.uint8)  
 return gray  
  
  
def get\_histogram(gray\_img):  
 # 利用hash table实现统计直方图  
 Pr = np.zeros(256, np.int32) # Hash Table  
 for i in range(gray\_img.shape[0]):  
 for j in range(gray\_img.shape[1]):  
 Pr[gray\_img[i][j]] += 1  
  
 # 归一化直方图，获得概率分布  
 Pr = Pr / (gray\_img.shape[0] \* gray\_img.shape[1])  
  
 S = np.zeros(256, np.float32)  
 pre\_sum = 0  
 for i in range(256):  
 pre\_sum += Pr[i]  
 S[i] = pre\_sum  
  
 S = S \* 255  
 S = np.round(S).astype(np.uint8)  
  
 Ps = np.zeros(256, np.float32)  
 for i in range(256):  
 Ps[S[i]] += Pr[i]  
  
 return S, Ps  
  
  
def image\_equalization(img, S):  
 img\_eq = np.zeros((img.shape[0], img.shape[1]), np.uint8)  
 for i in range(img.shape[0]):  
 for j in range(img.shape[1]):  
 img\_eq[i][j] = S[img[i][j]]  
 return img\_eq  
  
  
def getPSNR(ori\_img, en\_img):  
 MAX = 255.0  
 total = 0.0  
 for i in range(ori\_img.shape[0]):  
 for j in range(ori\_img.shape[1]):  
 total += (ori\_img[i][j] - en\_img[i][j]) \*\* 2  
 MSE = total / (ori\_img.shape[0] \* ori\_img.shape[1])  
 PSNR = 10 \* math.log10(MAX \* MAX / MSE)  
 return PSNR  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 # 改变不同文件，查看效果  
 img = cv2.imread("Lena.jpg")  
 gray = getGrayImg(img)  
 S, Ps = get\_histogram(gray)  
 img\_eq = image\_equalization(gray, S)  
 psnr = getPSNR(gray, img\_eq)  
 print(f"PSNR: {psnr}")  
  
 cv2.imwrite("LenaRGBLow1\_enhanced.jpg", img\_eq)  
  
 # visualization  
 plt.figure()  
 plt.suptitle('Histo Eq. Result')  
 plt.subplot(221)  
 plt.imshow(gray, cmap='gray')  
 plt.title('Original Gray Image')  
 plt.subplot(222)  
 plt.imshow(img\_eq, cmap='gray')  
 plt.title('Equalized Image')  
 plt.subplot(223)  
 p1 = gray.reshape(gray.shape[0] \* gray.shape[1], )  
 plt.hist(p1, bins=256, range=[0, 256])  
 plt.title('Histogram of Original Image')  
 plt.subplot(224)  
 p2 = img\_eq.reshape(img\_eq.shape[0] \* img\_eq.shape[1], )  
 plt.hist(p2, bins=256, range=[0, 256])  
 plt.title('Histogram of Equalized Image')  
 plt.show()

**3、YUV色彩空间变换实验完整代码：**

import numpy as np  
import cv2  
import math  
import os  
  
def RGB2YUV\_enhance(img, lightness\_en=3.5):  
 temp\_YUV = np.zeros((img.shape[0], img.shape[1], 3), np.float32)  
 res\_RGB = np.zeros((img.shape[0], img.shape[1], 3), np.float32)  
 timg = img.astype(np.float32)  
 for i in range(timg.shape[0]):  
 for j in range(timg.shape[1]):  
 ##############################################################  
 # Note that, should be careful about the RGB or BGR order  
 # Hint: check the transformation matrix to convert RGB to YUV  
 ##############################################################  
 ## write your code here  
 # OpenCV uses BGR by default, so we need to adjust accordingly  
 B = timg[i, j, 0]  
 G = timg[i, j, 1]  
 R = timg[i, j, 2]  
  
 # Convert RGB to YUV  
 Y = 0.299 \* R + 0.587 \* G + 0.114 \* B  
 U = -0.14713 \* R - 0.28886 \* G + 0.436 \* B  
 V = 0.615 \* R - 0.51498 \* G - 0.10001 \* B  
  
 ## 1. save temp\_YUV for visualization  
 temp\_YUV[i, j, 0] = Y  
 temp\_YUV[i, j, 1] = U  
 temp\_YUV[i, j, 2] = V  
  
 ## 2. enhance Y and convert YUV back to the RGB  
 Y\_enhanced = Y \* lightness\_en  
  
 R\_enhanced = Y\_enhanced + 1.13983 \* V  
 G\_enhanced = Y\_enhanced - 0.39465 \* U - 0.58060 \* V  
 B\_enhanced = Y\_enhanced + 2.03211 \* U  
  
 ## 3. store the enhanced RGB  
 res\_RGB[i, j, 0] = np.clip(B\_enhanced, 0, 255)  
 res\_RGB[i, j, 1] = np.clip(G\_enhanced, 0, 255)  
 res\_RGB[i, j, 2] = np.clip(R\_enhanced, 0, 255)  
  
 temp\_YUV = temp\_YUV.astype(np.uint8)  
 res\_RGB = res\_RGB.astype(np.uint8)  
 return temp\_YUV, res\_RGB  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 img = cv2.imread("Lena.jpg")  
 imgyuv, res\_rgb = RGB2YUV\_enhance(img)  
 cv2.imshow('Original Image', img)  
 cv2.imshow('Y', imgyuv[:,:,0])  
 cv2.imshow('U', imgyuv[:,:,1])  
 cv2.imshow('V', imgyuv[:,:,2])  
 cv2.imshow('Enhanced Light', res\_rgb)  
 cv2.imwrite("enhanced\_image.jpg", res\_rgb) # Save the enhanced image  
 cv2.waitKey(0)  
 cv2.destroyAllWindows()