计算机视觉实验指导书

实验一：图像处理：模糊、锐化、去噪、直方图、色彩空间转换

华为智能基座课程

# 



广东工业大学

计算机学院

2024年2月

# 实验介绍

## 实验介绍

### 关于本实验

实验前理论课应该完成low-level视觉部分的讲解，学生已了解、掌握常见直方图均衡化、空间域和频率域滤波的基本原理。本实验介绍了Window/Linux系统通过Miniconda安装opencv-python、搭建python虚拟环境。

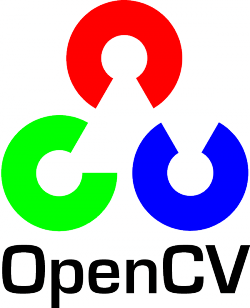
### 软件版本介绍

|  |  |  |  |
| --- | --- | --- | --- |
| 类别 | 版本 | 获取方式 | 说明 |
| Windows | Windows11、10 | / | 需要是64位系统，CPU支持AVX2指令集 |
| Ubuntu | Ubuntu 22.04，18.04.4 | <https://ubuntu.com/download/desktop> | 需要是64位系统，CPU支持AVX2指令集 |
| PyCharm | 2020.1.4 Community Edition | <https://www.jetbrains.com/> | / |
| VScode | 1.77 | https://code.visualstudio.com/download | / |
| Miniconda | Python3.x | 官方下载地址：  <https://docs.conda.io/en/latest/miniconda.html>  清华镜像源地址：  <https://mirrors.tuna.tsinghua.edu.cn/anaconda/miniconda/> | Miniconda可在线安装不同的Python版本，无需刻意下载特定版本，但需要下载64位，Python3.x版本 |

其中Pycharm和VScode任选其一。

## 软件介绍

### OpenCV介绍

OpenCV是著名的开源计算机视觉算法库，其被设计成跨平台的。目前OpenCV也支持深度学习算法。该库底层用C语言编写实现，这使得OpenCV几乎可以移植到任何商业系统中，从PowerPC Macs到机器狗。从2.0版本开始，OpenCV包括其传统的C语言界面以及新的C++界面。在多数情况下，新的OpenCV算法现在都是用C++开发的。同时，为了鼓励更多的人采用，还开发了Python和Java等语言的封装器。OpenCV可以在桌面（Windows、Linux、Android、MacOS、FreeBSD、OpenBSD）和移动（Android、Maemo、iOS）上运行。

### Miniconda介绍

Conda是一款Python环境管理软件，可以方便安装各种Python所需的第三方库，同时也可以创建互相独立的虚拟环境，类似于电脑中的虚拟机，Miniconda只包含了Conda和Python，由于Conda安装包的时候源在国外，国内直连速度较慢，因此本实验只用到了Conda创建虚拟环境的功能。

# 环境搭建

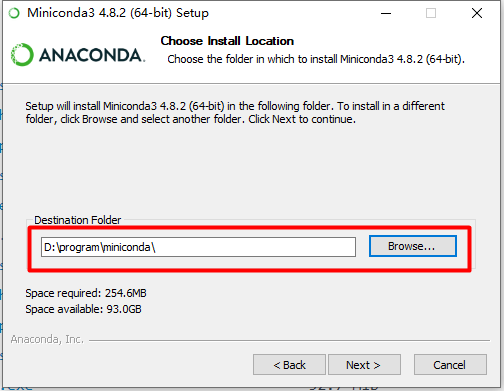
## 本地Windows环境搭建

### Miniconda安装

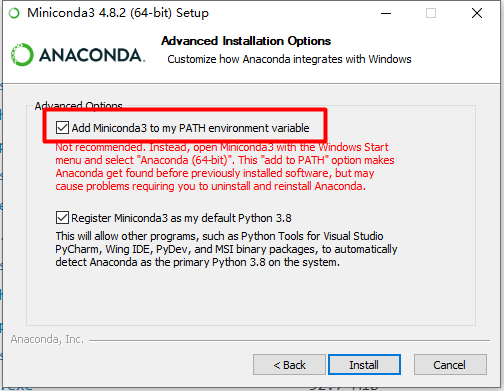
从1.1.2提供的链接下载Miniconda的Windows版本对应的64位安装包，由于官方源下载速度慢，实验所用安装包为清华源下载，带有x86\_64的为64位安装包。



双击安装包进行安装，点击next，然后选择安装位置。



环境变量打勾，这样可以直接在命令行中启动Miniconda。

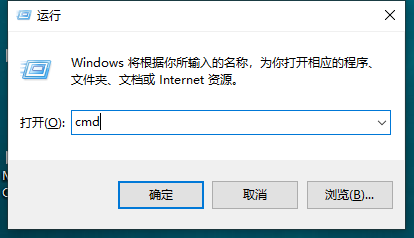


等待安装成功，然后点击Finish。

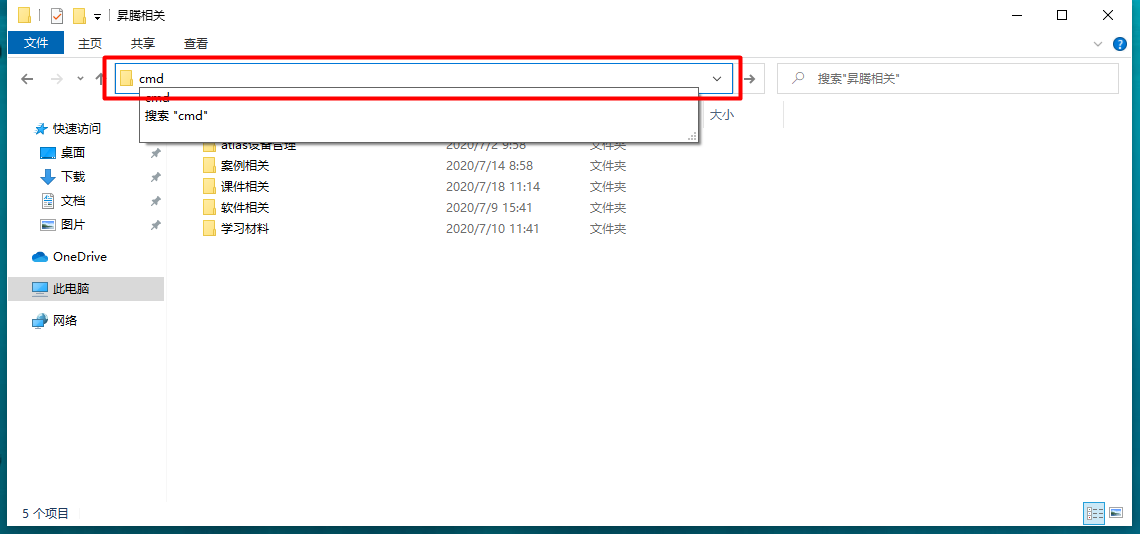
### 创建虚拟环境

在Window中有多种方式开启命令行窗口，这里介绍两种，按下win+R键，然后输入cmd点击确定，或者任意打开一个文件夹，在上方地址栏输入cmd，然后按回车键。

运行打开命令行界面如下图所示：



地址栏打开命令行界面如下图所示：



打开命令行窗口之后，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

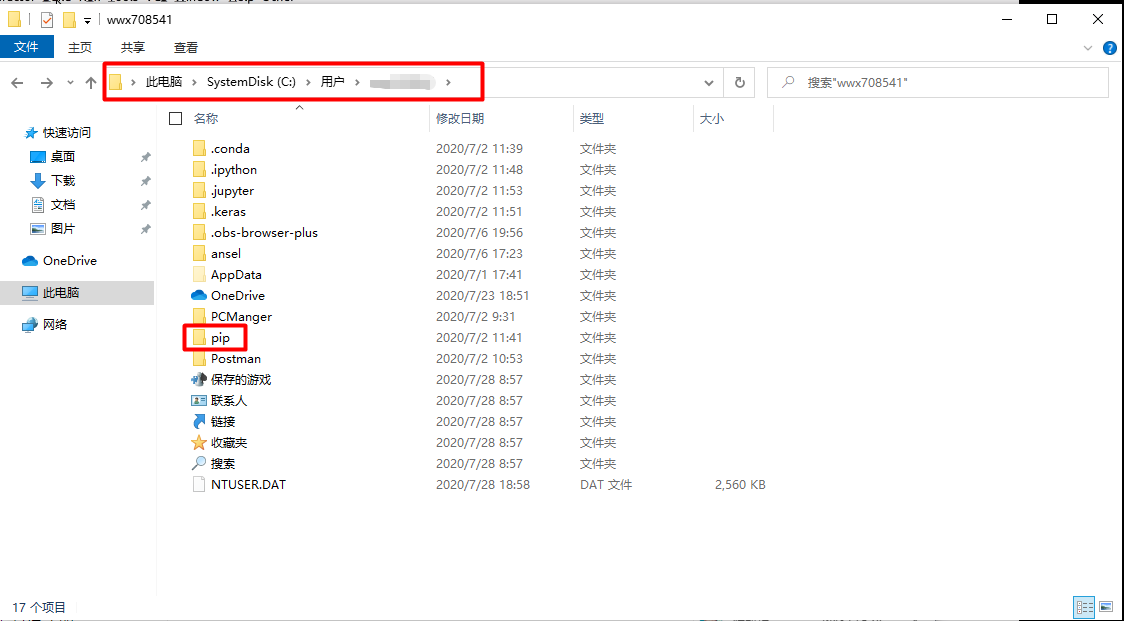
虚拟环境创建成功后输入对应名称即可进入对应虚拟环境

>> activate cv

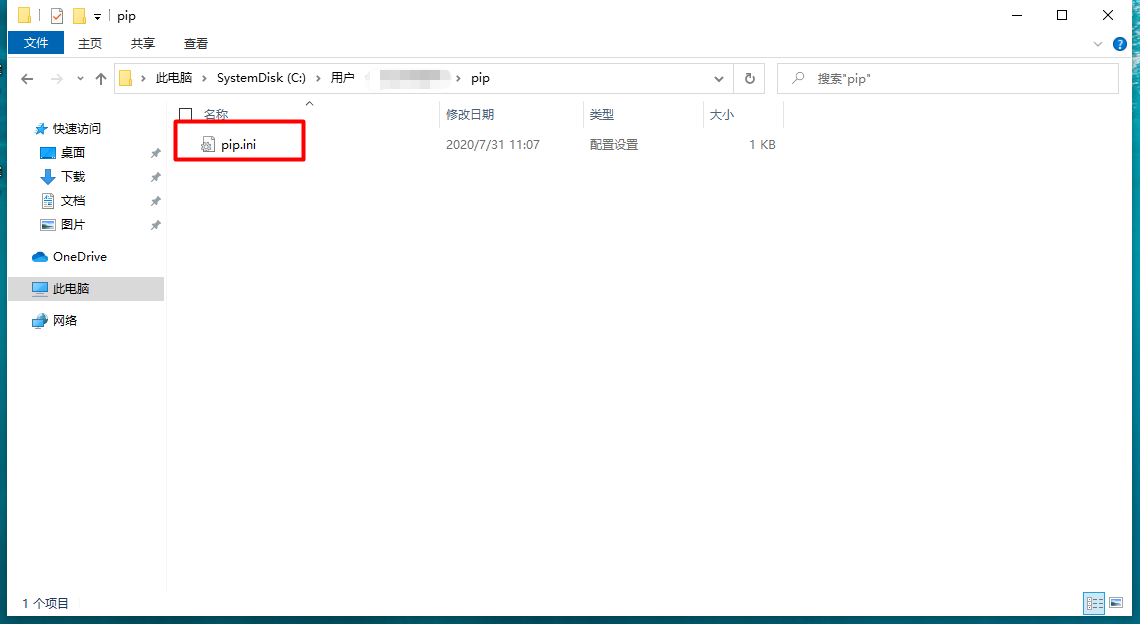
### Pip换源

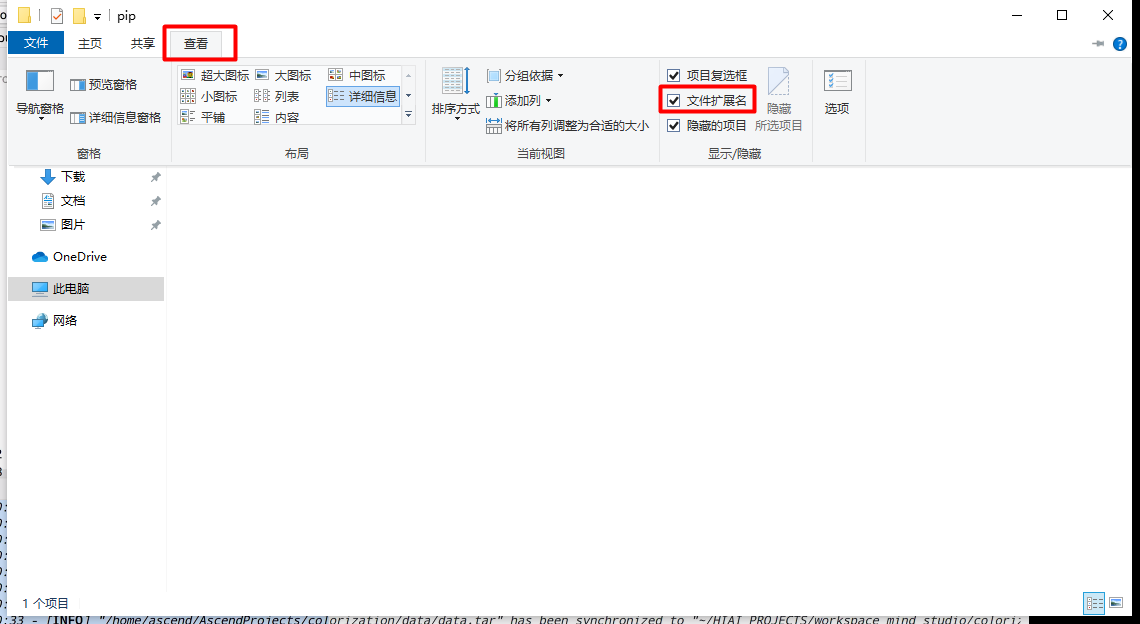
Python可以通过pip和conda两种方式来安装包，但是两者所安装的包并不完全兼容，在实际使用过程中建议只选择一种方式来安装包，本实验使用的是pip，但是由于pip的官方源在国外，直连速度较慢，因此需要换为国内的镜像源。

打开此电脑，进入C盘、用户、用户名，然后新建一个pip文件夹。



新建一个文本文件，然后改名pip.ini，该文件就是pip的配置文件，如果改完之后图标没变化，说明没有显示文件扩展名，点击查看，随后勾选显示文件扩展名。





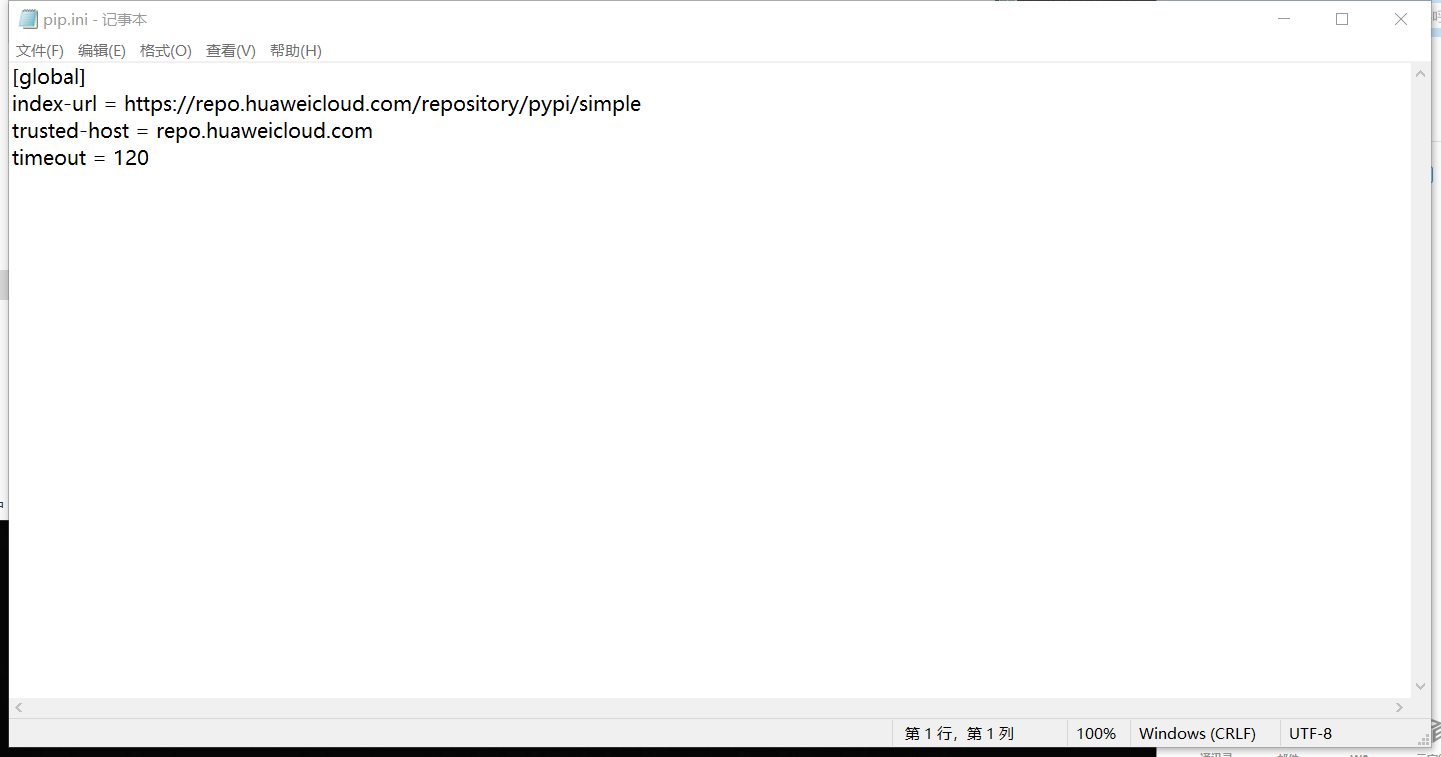
打开pip.ini文件，将以下内容粘贴进去并保存。

[global]

index-url = https://repo.huaweicloud.com/repository/pypi/simple

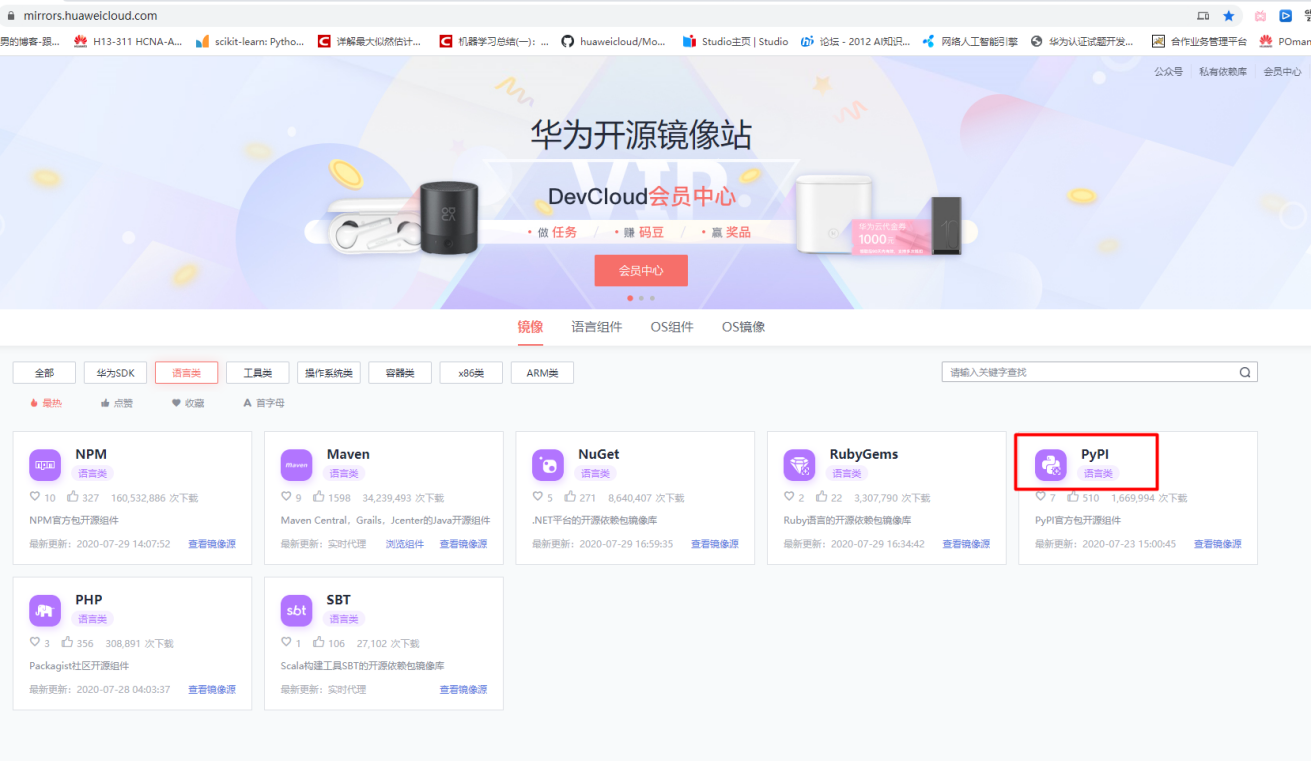
trusted-host = repo.huaweicloud.com

timeout = 120



更多关于pip换源的信息可以参考以下链接：

<https://mirrors.huaweicloud.com/>



### 安装opencv-python

新建一个命令行窗口，输入以下命令激活cv安装虚拟环境。

(Ubuntu/Linux)

>> conda activate cv

(Windows)

>> activate cv

成功激活环境后，利用pip命令安装最新版本的opencv库

>> pip install opencv-python

安装完毕后，可在命令行验证安装是否成功

>> python

>>> import cv2

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

安装正常一般会显示

>>> import cv2

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

'4.6.0'

>>>

备注：

1. 如果已经安装好python环境，则无需重复执行上面的安装操作。

2. 下面的代码仅供参考，具体以压缩包里的 \*.py 文件为准。学生需要对代码进行改动，从而完成基础实验要求。

3. 思考题是拓展实验。需要在报告中列出解题思路、关键代码、执行结果等。

# 空间域滤波

编辑image\_filtering.py 文件，实现滤波函数Flitering2D以及统计滤波器函数denoisewithOrderStatisticFilter。通过选择不同的图片和不同的滤波器，实现不同的滤波效果。

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)

# sharppen kernel

lapalicanKernel = 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] # 第一行

    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 = ?

            # ############## 结束编程 #############

            filtered\_img[i][j] = np.clip(pixel, 0.0, 255.0).astype(np.uint8)

    return filtered\_img

def denoisewithOrderStatisticsFilter(*img*):

    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]):

            # ###### 这里编程实现统计数据公式 ##########

            # pixel = ?

            #

            #

            #

            #

            # ############## 结束编程 #############

            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("test/1\_smooth.jpg")

    img = getGrayImg(img)

    cv2.imshow('orginal 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. 将平滑后的图像行锐化高通滤波 查看结果

    #

    # 更换Filtering2D(img, filter) 的filter，实现锐化

    # 需要锐化的图片是 test/3.jpg

    #

    # 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

    #

    # 需要调整denoisewithOrderStatisticsFilter(img)，来实现不同的功能

    # 每种功能执行一次，输出一个结果，保存

    # 均值、中值滤波处理的图像：test/2.jpg

    # 最大值滤波：421.jpeg, 最小值滤波：419.jpeg

    cv2.waitKey(0)

    cv2.destroyAllWindows()

    print(getPSNR(img, filtered\_img))

完成代码，并调试后，测试滤波结果。注意测试线性平滑滤波（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。

# 直方图均衡化

注：为可视化直方图，需要安装matplotlib。激活环境后，pip安装。

>> pip install matplotlib

编辑histo\_eq.py文件，实现直方图统计函数和直方图均衡化函数，并得到测试效果。

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][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 get\_histogram(*gray\_img*):

    # 利用hash table实现统计直方图

    # 注意统计范围较大，使用int32类型numpy array

    Pr = np.zeros(256, np.int32) # Hash Table

    for i in range(*gray\_img*.shape[0]):

        for j in range(*gray\_img*.shape[1]):

            # ###### 1. 这里编程实现直方图统计 ##########

            #

            #

            #

            # ############## 结束编程 #############

            pass

    # ###### 2. 归一化直方图，获得概率分布计 ##########

    #

    #

    #

    # ############## 结束编程 #############

    pass

    S =  np.zeros(256, np.float32)

    pre\_sum = 0 # 提示1

    for i in range(256):

        # ###### 3. 获得累积概率分布 ##########

        #

        #

        #

        # ############## 结束编程 #############

        pass

    S = S \* 255

    S = np.round(S).astype(np.uint8)

    Ps = np.zeros(256, np.float32)

    for i in range(256):

        # 提示线索

        Ps[S[i]] = Ps[S[i]] + Pr[S[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]):

            # ###### 实现像素值的重映射 ##########

            #

            #

            #

            # ############## 结束编程 #############

            pass

    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 = 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\_\_':

    # 改变不同文件，查看效果

    # img = cv2.imread("test/Lena.jpg")

    img = cv2.imread("data/Histogram Equalization/LenaRGBLow2.jpg")

gray = getGrayImg(img)

    S, Ps = get\_histogram(gray)

img\_eq = image\_equalization(gray, S)

    psnr = getPSNR(gray, img\_eq)

    print(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.subplot(222)

    plt.imshow(img\_eq, *cmap*='gray')

    plt.subplot(223)

    p1 = gray.reshape(gray.shape[0]\*gray.shape[1], )

    plt.hist(p1, 256)

    plt.subplot(224)

    p2 = img\_eq.reshape(img\_eq.shape[0]\*img\_eq.shape[1], )

    plt.hist(p2, 256)

    plt.show()

# 色彩空间变换

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。分析颜色空间转换后进行图像增强的原理。

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.uint8)

    res\_RGB  = np.zeros((*img*.shape[0], *img*.shape[1], 3), np.uint8)

    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

            # Y =

            # U =

            # V =

            ## 1. save temp\_YUV for visualization

            # 把上一步得到的YUV 赋给 temp\_YUV

            # 注意：YUV可能是非整数。可以取四舍五入后赋给temp\_YUV

            #

            #

            #

            ## 2. enhance Y and convert YUV back to the RGB

            # 这一步将Y通道增强。可直接写下面的代码

            # rY = Y \* lightness\_en

            #

# 继续，将 rY U V 还原成RGB

            ## 3. store the enhanced RGB

            # 把RGB 赋给 新图像 res\_RGB

            #

            #

            #

            #

            #############################################################

            # end of your code

            #############################################################

            pass

            #############################################################

            # (Optional) consider more efficent way to implement such a conversion

            #############################################################

    return temp\_YUV, res\_RGB

if \_\_name\_\_ == '\_\_main\_\_':

    img = cv2.imread("test/Lena.jpg")

    imgyuv, res\_rgb = RGB2YUV\_enhance(img)

    cv2.imshow('orginal image', img)

    cv2.imshow('Y', imgyuv[:,:,0])

    cv2.imshow('U', imgyuv[:,:,1])

    cv2.imshow('V', imgyuv[:,:,2])

    cv2.imshow('Enhance Light', res\_rgb)

    # cv2.imwrite("rgb2yuv.jpg", imgyuv)

    cv2.waitKey(0)

    cv2.destroyAllWindows()

思考：

1. 测试其他3×3滤波器,比如检测边缘、一阶、二阶梯度算子。
2. 简述手工设计的滤波器与CNN的卷积核的关系？
3. 测试其他色彩空间转换效果，如RGB -> YIQ or CMY？
4. 彩色图像如何进行直方图均衡化？