计算机视觉实验指导书

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

华为智能基座课程

广东工业大学

计算机学院

2024 年 2 月

1. 实验介绍

1.1 实验介绍

1.1.1 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。

1.1.2 实验目标

1. 熟悉适用 OpenCV 库进行直方图统计、均衡化、空间域模糊、去噪、锐化等 Low-level vision 的实现方法；
2. 熟悉使用 OpenCV 库进行色彩空间转换方法。

1.1.3 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（VScode Windows 下友好一些）。另：Anaconda 和 Miniconda 任选其一（**注意，已经安装过 Anaconda 的同学不需要再安装 miniconda！！！**）。

1.2 软件介绍

1.2.1 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。

1.2.2 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。

2 环境搭建

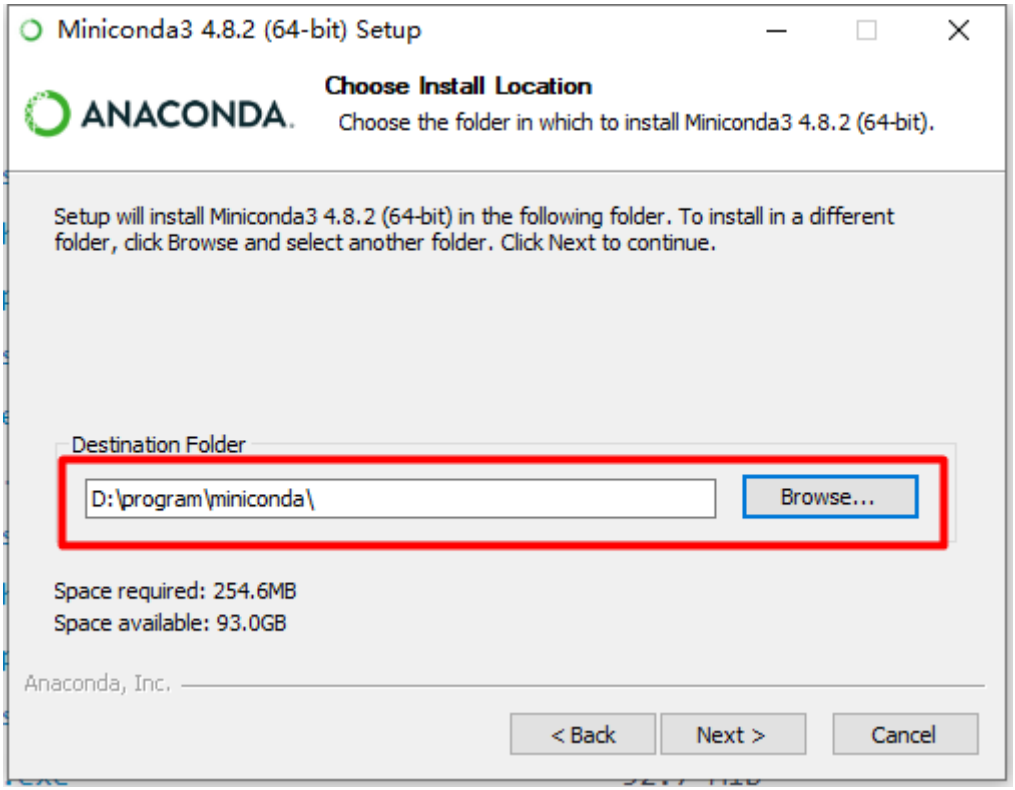
2.1 本地 Windows 环境搭建（有 Linux 使用经验的同学，推荐使用 Ubuntu 系统安装）

2.1.1 Miniconda 安装（机器上已经装了 Anaconda 的同学可以不用再安装 Miniconda！！！）

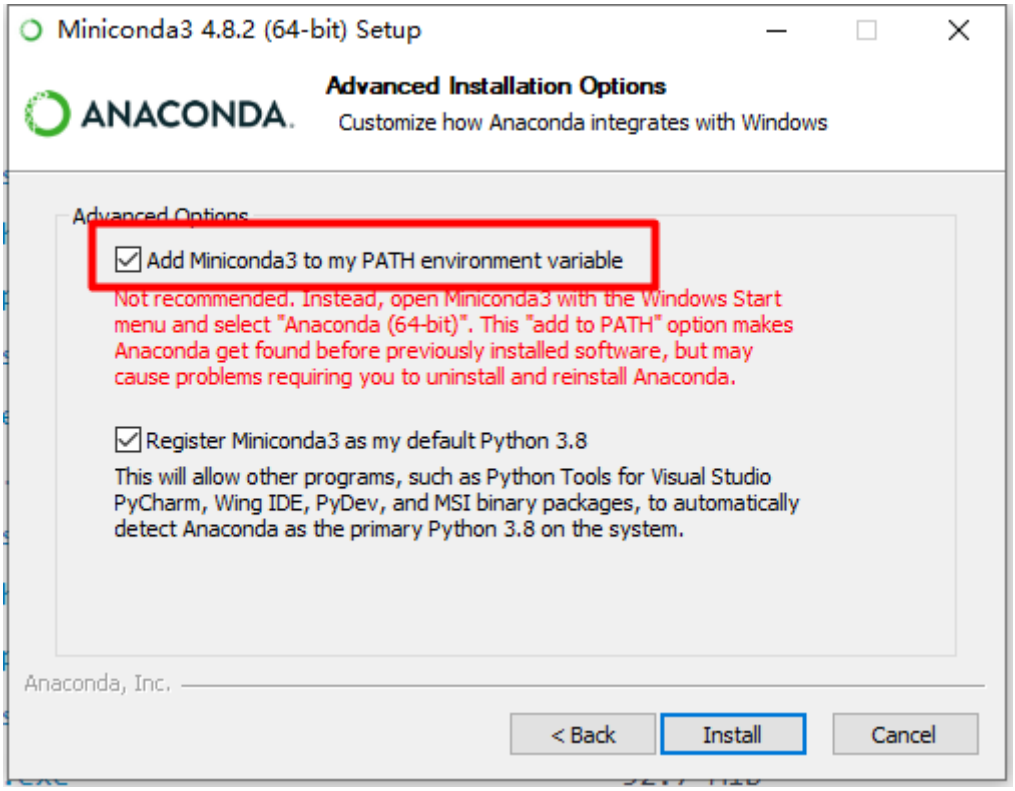
从 1.1.2 提供的链接下载 Miniconda 的 Windows 版本对应的 64 位安装包，由于官方源下载速度慢，实验所用安装包为清华源下载，带有 x86_64 的为 64 位安装包。

Miniconda3-py38_4.8.2-Windows-x86_64.exe 52.7 MiB 2020-03-12 00:09

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。

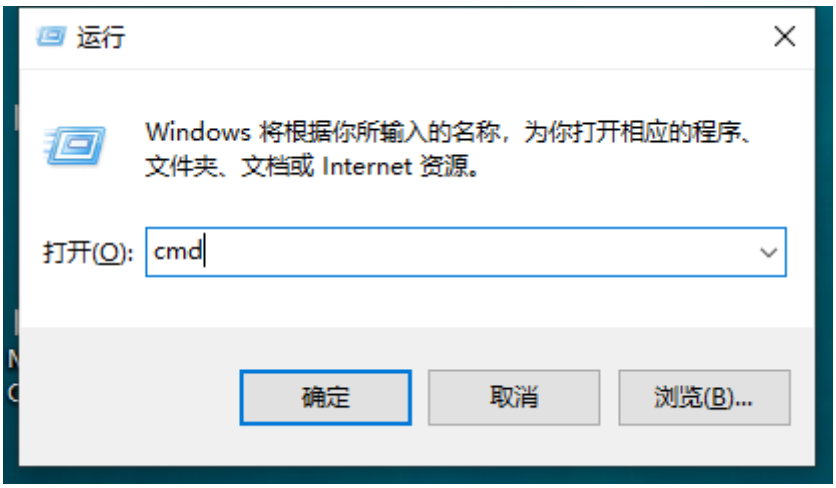


等待安装成功，然后点击 Finish。

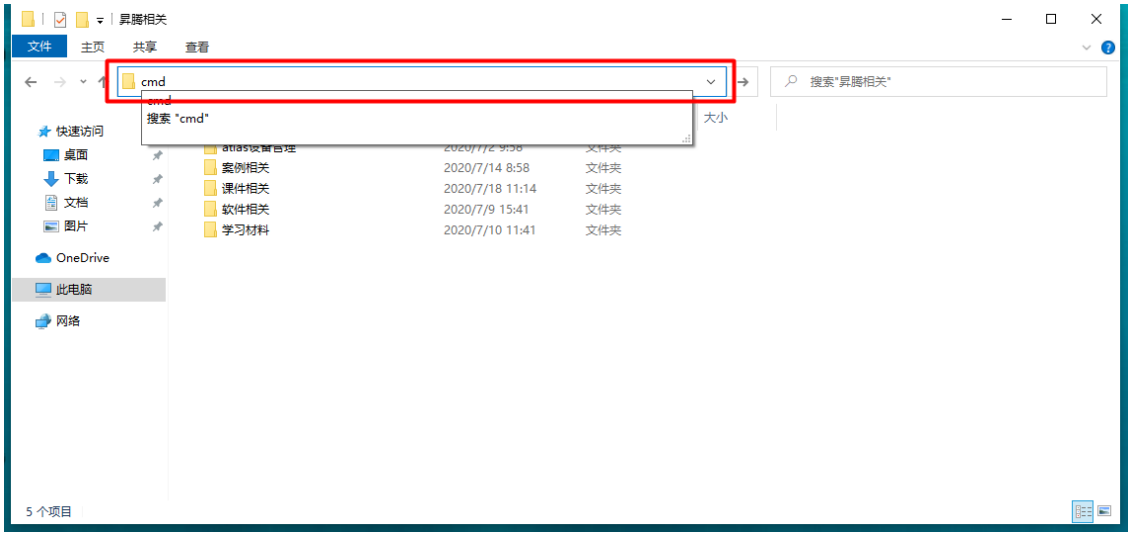
2.1.2 创建虚拟环境

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

注：为加速安装，一般会更改 conda 源为国内镜像源。关于 conda 换国内清华源的方式参见下面的链接：

https://mirrors.tuna.tsinghua.edu.cn/help/anaconda/

2.1.3 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。

更多关于 pip 换源的信息可以参考：

https://mirrors.tuna.tsinghua.edu.cn/help/pypi/

2.1.4 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。**（Window10，11 环境系统的 cmd shell 设置变化较大，同学们注意查看安装的环境是否能被顺利激活，比如激活后，查看 python 版本是否符合预期！！！，使用 PyCharm IDE 的同学也要注意下。很可能应为 shell 的不同，导致环境激活失败！）**

(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'
>>>
```

3 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
    #
    #
    #
    #
    # 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
    #
    #
    #
    #
    #
    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。

4 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("data/Histogram Equalization/LenaRGBLow1.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()
```

5 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
            #
            #
            #
            #
            #

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

            ## 3. store the enhanced 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()
```

（选做部分，可不列入报告）思考及拓展：

1、 测试其他 3×3 滤波器？比如检测边缘、一阶、二阶梯度算子？如果这些算子可学习？手工设计的滤波器与 CNN 的卷积核的关系？

2、 测试其他色彩空间转换效果？

3、 调研彩色图像如何进行直方图均衡化？