****

计算机视觉实验

**实验二**

**专 业：**

**班 级：**

**姓 名：**

**学 号：**

**指导教师：**

**成 绩：**

**计算机学院**

**年 月 日**

# 实验目的

选取一张自己的生活照，实现：

任选两种结构对图像进行腐蚀和膨胀。

边缘检测（分别使用Prewitt，两种大小的Sobel算子，不同参数的Canny算子，LoG算子），并对比各个算子实现差异及不同参数条件下的实验效果。

# 实验过程

## 图像腐蚀和膨胀

实验图片

所用代码

import cv2

import numpy as np

def threshold(image, threshold):

out = image.copy()

out[out < threshold] = 0

out[out > threshold] = 1

out = out.astype(np.uint8)

return out

# 1. 读取图像文件

image = cv2.imread('img/in.jpeg', cv2.IMREAD\_GRAYSCALE)

# 2. 缩放图像

print('Shape: {}'.format(image.shape))

image = cv2.resize(image, (256, 256), cv2.INTER\_LINEAR)

print('Shape: {}'.format(image.shape))

# 3. 均一阈值分割

binary = threshold(image, threshold=200)

# 4. 腐蚀

kernel\_size = 5

erode = cv2.erode(binary, kernel=np.ones((kernel\_size, kernel\_size)))

# 5. 膨胀

dilate = cv2.dilate(binary, kernel=np.ones((kernel\_size, kernel\_size)))

# 6. 显示图像

cv2.imshow('Image', image)

cv2.imshow('Binary', binary \* 255)

cv2.imshow('Erode', erode \* 255)

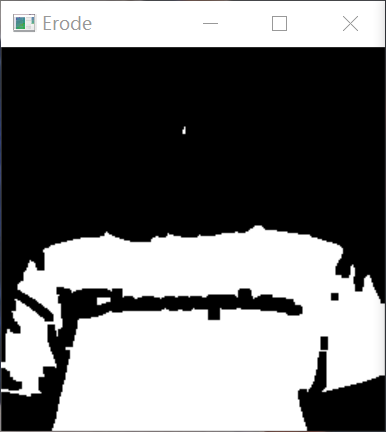
cv2.imshow('Dilate', dilate \* 255)

cv2.waitKey(0)

实验效果



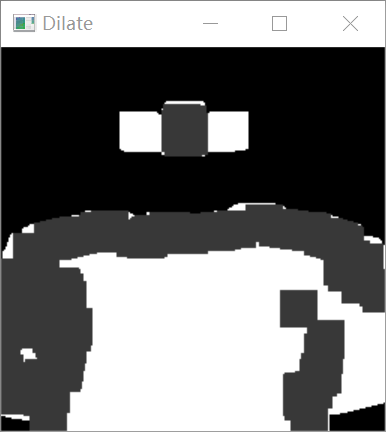
Kernel=5





Kernel=25





代码分析及现象

腐蚀，消除连通的边界，使边界向内收缩。腐蚀处理可以将粘连在一起的不同目标物分离，并可以将小的颗粒噪声去除。随着kernel越大，腐蚀效果越明显

膨胀，将目标点融合到背景中，向外部扩展。膨胀处理可以将断裂开的目标物进行合并，便于对其整体的提取。随着kernel越大，膨胀效果越明显

## 图像边缘检测

实验图片

### Prewitt算子

所用代码

import cv2

import numpy as np

def edge\_prewitt(image):

"""

:param image:

:return:

Prewitt:

[ 1, 0, -1] [ 1, 1, 1]

G\_x = [ 1, 0, -1] G\_y = [ 1, 0, -1]

[ 1, 0, -1] [-1, 0, -1]

Prewitt = threshold(max(G\_x \* I, G\_y \* I))

"""

g\_x = np.array([[1, 0, -1], [1, 0, -1], [1, 0, -1]], dtype=int)

g\_y = np.array([[1, 1, 1], [0, 0, 0], [-1, -1, -1]], dtype=int)

x = cv2.filter2D(image, cv2.CV\_16S, g\_x)

y = cv2.filter2D(image, cv2.CV\_16S, g\_y)

absX = cv2.convertScaleAbs(x)

absY = cv2.convertScaleAbs(y)

result = cv2.max(absX, absY)

\_, result = cv2.threshold(result, 0, 255, cv2.THRESH\_OTSU)

return result

# 1. 读取图像文件

image = cv2.imread('img/in.jpeg', cv2.IMREAD\_GRAYSCALE)

# 2. Prewitt

prewitt = edge\_prewitt(image)

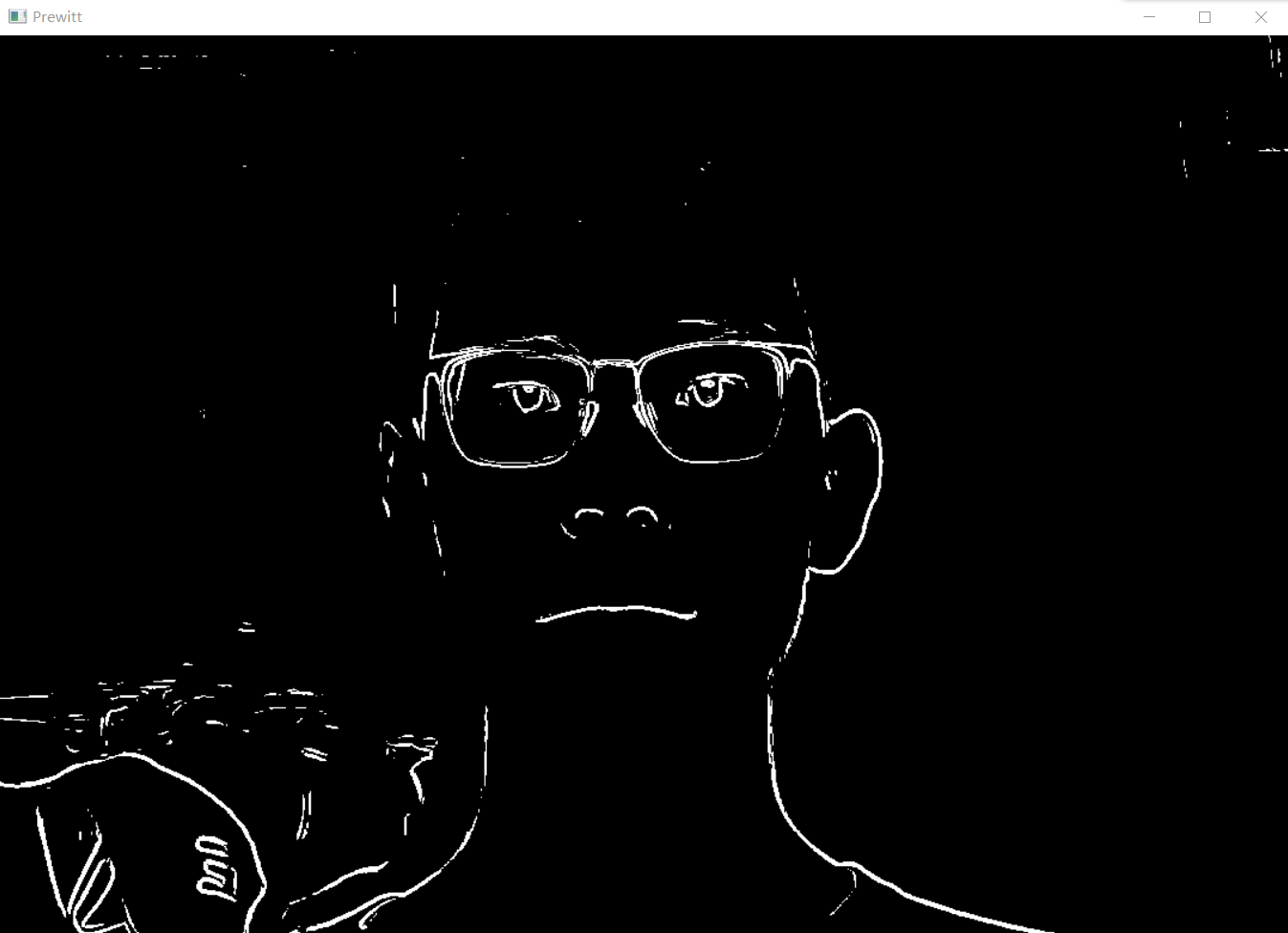
# 5. 显示

cv2.imshow('Image', image)

cv2.imshow('Prewitt', prewitt)

cv2.waitKey(0)

实验效果



代码分析及现象

对边缘检测效果一般

### Sobel算子

所用代码

import cv2

import numpy as np

def edge\_sobel(image, kernel\_size=3):

x = cv2.Sobel(image, cv2.CV\_16S, 1, 0, ksize=kernel\_size)

y = cv2.Sobel(image, cv2.CV\_16S, 0, 1, ksize=kernel\_size)

absX = cv2.convertScaleAbs(x)

absY = cv2.convertScaleAbs(y)

result = cv2.addWeighted(absX, 0.5, absY, 0.5, 0)

\_, result = cv2.threshold(result, 0, 255, cv2.THRESH\_OTSU)

return result

# 1. 读取图像文件

image = cv2.imread('img/in.jpeg', cv2.IMREAD\_GRAYSCALE)

# 3. Sobel

sobel = edge\_sobel(image, kernel\_size=3)

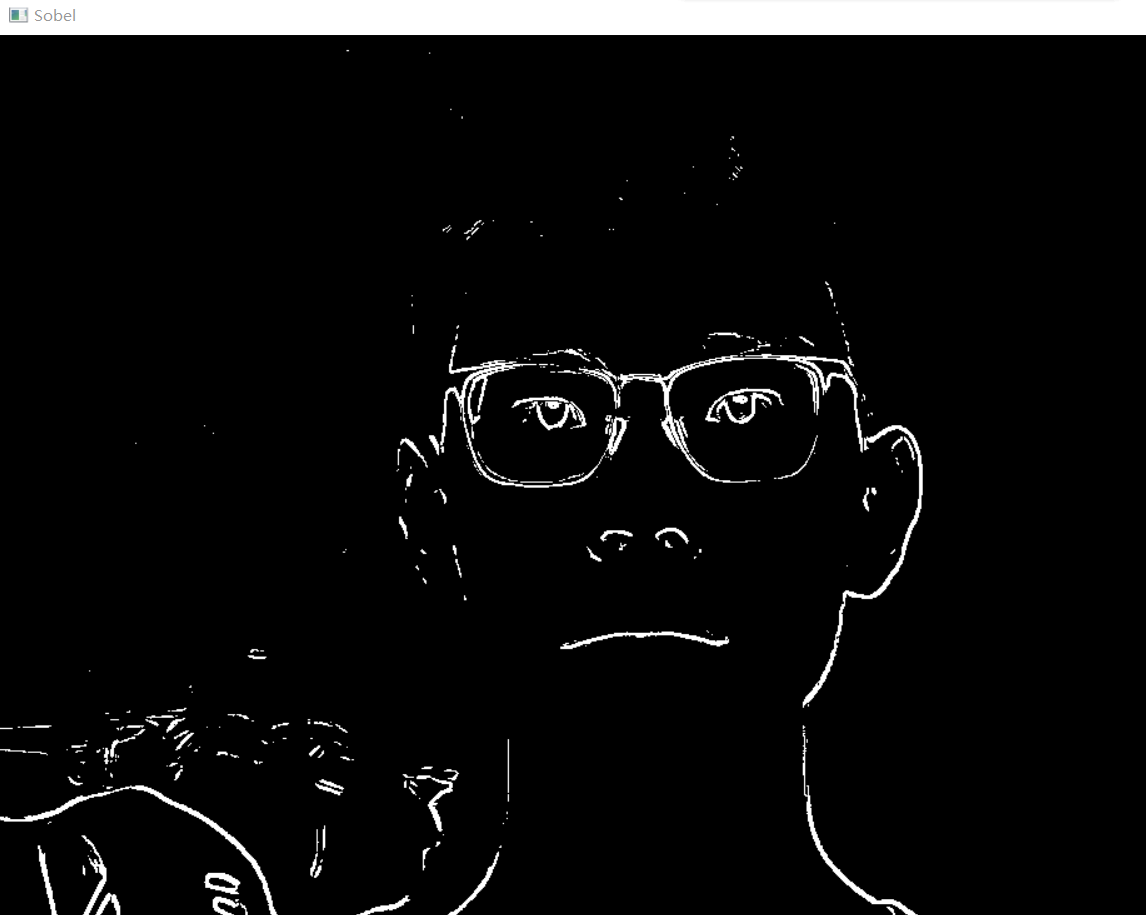
# 5. 显示

cv2.imshow('Image', image)

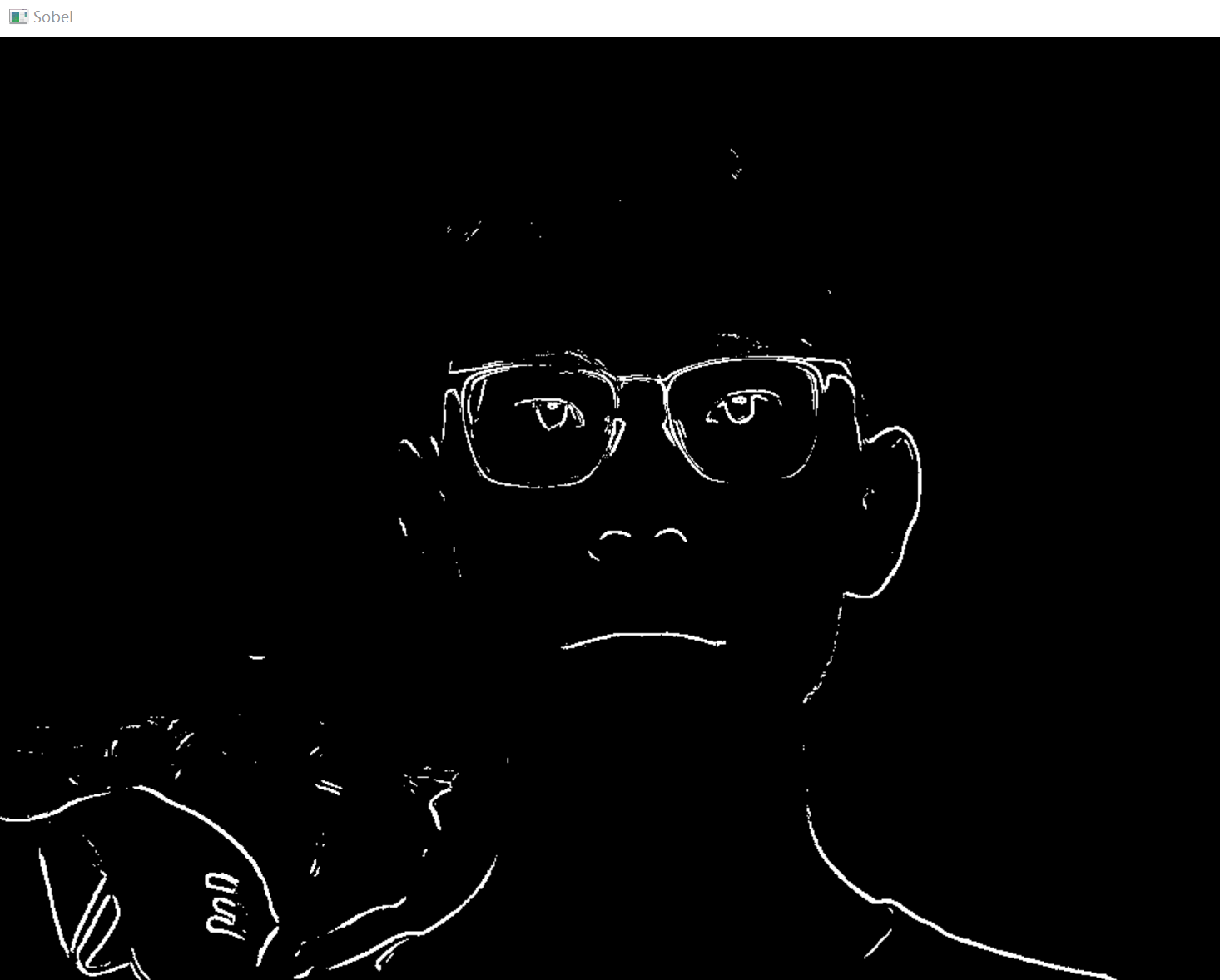
cv2.imshow('Sobel', sobel)

cv2.waitKey(0)

实验效果



Kernerl\_size=1



代码分析及现象

随着kernelsize的增大，边缘检测效果先变好再变差

### Canny算子

所用代码

import cv2

import numpy as np

# 1. 读取图像文件

image = cv2.imread('img/in.jpeg', cv2.IMREAD\_GRAYSCALE)

# 4. Canny

canny = cv2.Canny(image, 50, 150)

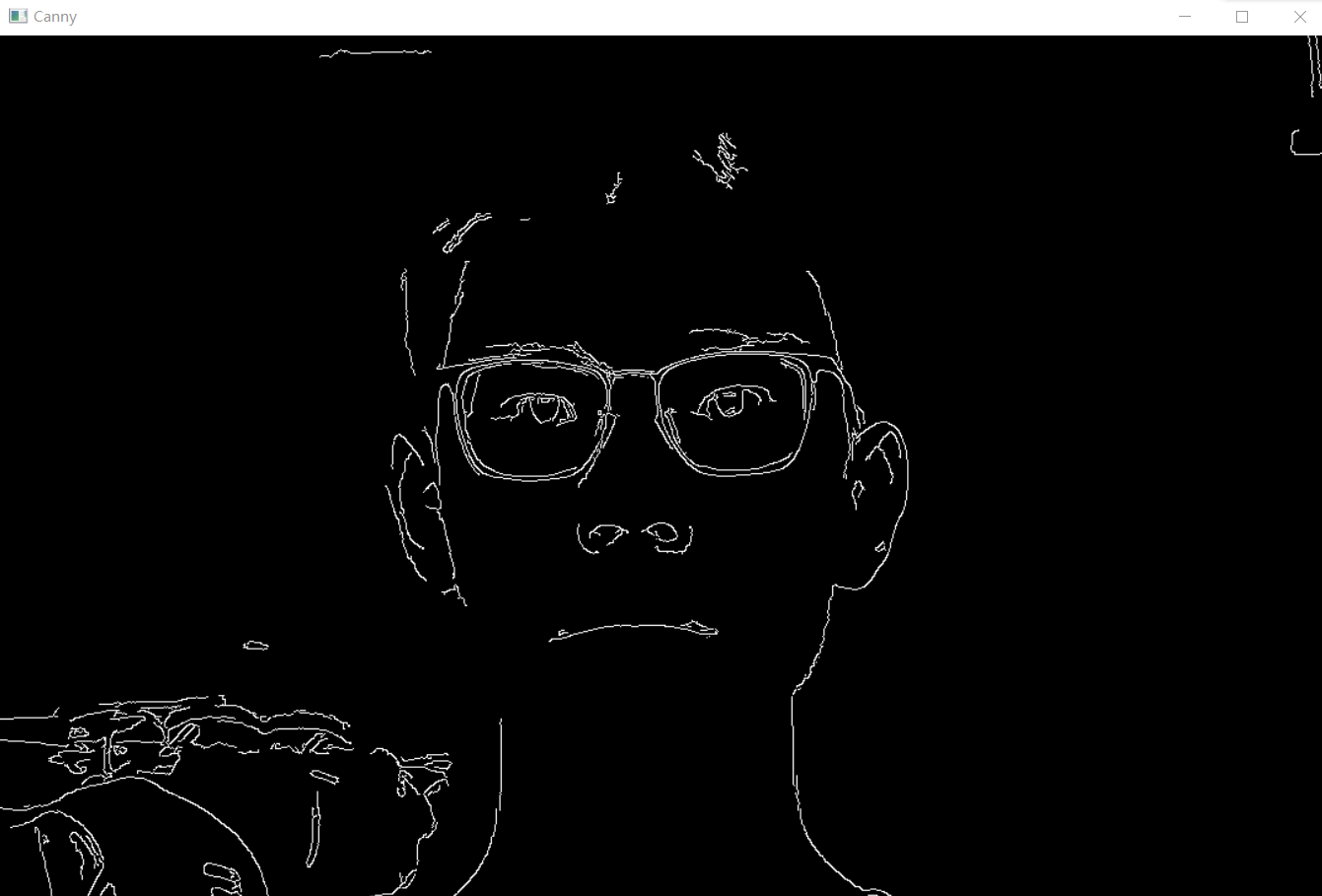
# 5. 显示

cv2.imshow('Image', image)

cv2.imshow('Canny', canny)

cv2.waitKey(0)

实验效果



Canny(image,30,30)



代码分析及现象

当第二个阙值远大于第一个阙值时，检测到了边缘但连接边缘的效果不明显。图像边缘显示也不清晰，连接效果也不好，即当第一个阈值增大时，接近第二个阈值时，图像连接效果更好。

### LoG算子

所用代码

import cv2 as cv

import matplotlib.pyplot as plt

# 读取图像

img = cv.imread("img/in.jpeg")

rgb\_img = cv.cvtColor(img, cv.COLOR\_BGR2RGB)

gray\_img = cv.cvtColor(img, cv.COLOR\_BGR2GRAY)

# 先通过高斯滤波降噪

gaussian = cv.GaussianBlur(gray\_img, (3, 3), 0)

# 再通过拉普拉斯算子做边缘检测

dst = cv.Laplacian(gaussian, cv.CV\_16S, ksize=3)

LOG = cv.convertScaleAbs(dst)

# 用来正常显示中文标签

plt.rcParams['font.sans-serif'] = ['SimHei']

# 显示图形

titles = ['原始图像', 'LOG 算子']

images = [rgb\_img, LOG]

for i in range(2):

plt.subplot(1, 2, i + 1), plt.imshow(images[i], 'gray')

plt.title(titles[i])

plt.xticks([]), plt.yticks([])

plt.show()

实验效果



代码分析及现象

sigma越大，图像越模糊滤除噪声效果越好

# 实验心得

Prewitt 算子和 Sobel 算子检测出的结果类似，相比之下 Sobel 检测的边缘更加突出；LOG 和 Canny 算子检测出的边缘比较连贯，且具有较好的平滑性，边缘细节也更多。