实验一 图像的灰度变换与空域滤波

|  |  |  |  |
| --- | --- | --- | --- |
| **姓名：** | **学号：** | **日期：2021-10-26** | **截止日期：Next Class** |

**一、实验目的**

1. 掌握Opencv空域滤波的基本原理及方法
2. 掌握Opencv频域滤波的基本原理及方法

**二、实验报告要求**

1. 实验报告中，**按题目顺序给出实验内容对应的代码及相应结果**，**体现实验步骤和方法**。要求报告整洁，程序清晰，代码简捷，有必要的注释
2. **报告中的图片**要求清晰可见。 如果是屏幕截图，**请不要截取有效结果之外的无关区域**。
3. **报告中的代码**，一般情况下紧随内容。如果代码太长，建议按题目序号，附在报告末尾。

**三、实验内容及要求**

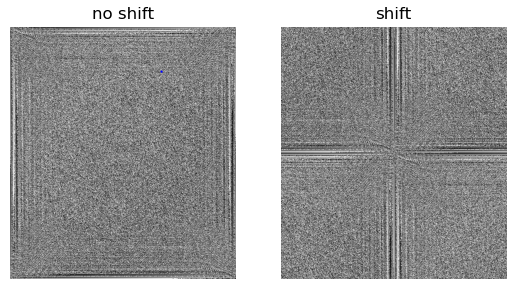
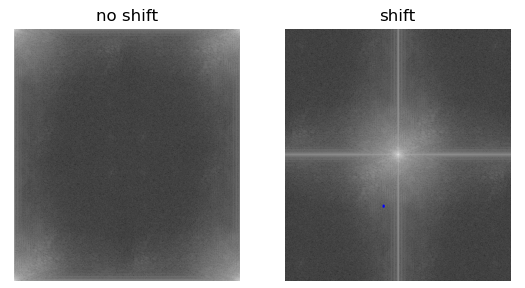
1. 编写代码，画出girl.png 图像的频域的幅度和相位图，通过傅里叶反变换，求得重建之后的图像，并计算重建图和原图之间的PSNR差异。PSNR计算如下：





其中，

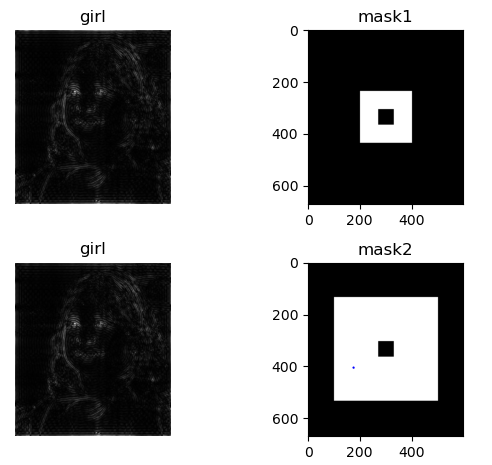
1. **import** cv2 as cv
2. **import** numpy as np
3. **import** matplotlib.pyplot as plt
4. %matplotlib
5. img = cv.imread('girl.png', cv.IMREAD\_GRAYSCALE)
6. F = np.fft.fft2(img)
7. Fshift = np.fft.fftshift(F)
9. #获取幅值图
10. F1 = np.log(abs(F) + 1)
11. Fshift1 = np.log(abs(Fshift) + 1)
12. plt.figure()
13. plt.subplot2grid((1, 2), (0, 0)), plt.imshow(F1, 'gray'), plt.title('no shift'), plt.axis('off')
14. plt.subplot2grid((1, 2), (0, 1)), plt.imshow(Fshift1, 'gray'), plt.title('shift'), plt.axis('off')
15. #获取相位图
16. ph\_F = np.angle(F)
17. ph\_Fshift = np.angle(Fshift)
18. plt.figure()
19. plt.subplot2grid((1, 2), (0, 0)), plt.imshow(ph\_F, 'gray'), plt.title('no shift'), plt.axis('off')
20. plt.subplot2grid((1, 2), (0, 1)), plt.imshow(ph\_Fshift, 'gray'), plt.title('shift'), plt.axis('off')
21. #重建
22. img\_back1 = np.fft.ifft2(F)
23. img\_back1 = np.abs(img\_back1)
24. img\_back2 = np.fft.ifftshift(Fshift)
25. img\_back2 = np.fft.ifft2(img\_back2)
26. img\_back2 = np.abs(img\_back2)
27. plt.figure()
28. plt.subplot2grid((1, 3), (0, 0)), plt.imshow(img, 'gray'), plt.title('img'), plt.axis('off')
29. plt.subplot2grid((1, 3), (0, 1)), plt.imshow(img\_back1, 'gray'), plt.title('no\_shift\_back'), plt.axis('off')
30. plt.subplot2grid((1, 3), (0, 2)), plt.imshow(img\_back2, 'gray'), plt.title('shift\_back'), plt.axis('off')
31. plt.tight\_layout()
32. #计算PSNR差异
33. MSE = np.sum(np.power((img\_back2 - img), 2)) / (img.shape[0] \* img.shape[1])
34. PSNR = 20 \* np.log10(255 / np.sqrt(MSE))
35. **print**(f'PSNR : {PSNR}')





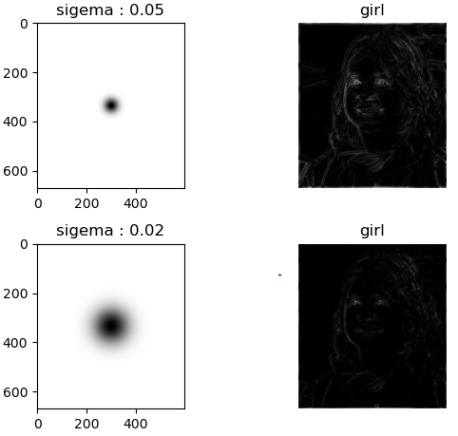
2. 以girl.png为例，设计理想带通滤波器，并给出2种滤波器下的滤波效果。

1. **import** cv2 as cv
2. **import** numpy as np
3. **import** matplotlib.pyplot as plt
4. %matplotlib
5. img = cv.imread('girl.png', cv.IMREAD\_GRAYSCALE)
6. F = np.fft.fft2(img)
7. Fshift = np.fft.fftshift(F)
8. #带通滤波器构造
9. mask = np.zeros(F.shape).astype(np.uint8)
10. row, col = mask.shape[0], mask.shape[1]
11. mask[int(row / 2) - 100 : int(row / 2) + 100, int(col / 2) - 100 : int(col / 2) + 100] = 1
12. mask[int(row / 2) - 30 : int(row / 2) + 30, int(col / 2) - 30 : int(col / 2) + 30] = 0
13. mask1 = mask
15. # fig = plt.figure(figsize=(10, 5))
16. Fshift = mask1 \* Fshift
17. img\_back = np.fft.ifftshift(Fshift)
18. img\_back = np.fft.ifft2(img\_back)
19. img\_back = np.abs(img\_back)
21. plt.subplot2grid((2, 2), (0, 0)), plt.imshow(img\_back, 'gray'), plt.title('girl'), plt.axis('off')
22. plt.subplot2grid((2, 2), (0, 1)), plt.imshow(mask1, 'gray'), plt.title('mask1')
24. mask[int(row / 2) - 200 : int(row / 2) + 200, int(col / 2) - 200 : int(col / 2) + 200] = 1
25. mask[int(row / 2) - 30 : int(row / 2) + 30, int(col / 2) - 30 : int(col / 2) + 30] = 0
26. mask2 = mask
28. Fshift = mask2 \* Fshift
29. img\_back1 = np.fft.ifftshift(Fshift)
30. img\_back1 = np.fft.ifft2(img\_back1)
31. img\_back1 = np.abs(img\_back1)
32. plt.subplot2grid((2, 2), (1, 0)), plt.imshow(img\_back1, 'gray'), plt.title('girl'), plt.axis('off')
33. plt.subplot2grid((2, 2), (1, 1)), plt.imshow(mask2, 'gray'), plt.title('mask2')
34. plt.tight\_layout()



3. 以girl.png为例，设计高斯高通滤波器， 并给出2种滤波器下的滤波效果。高斯高通滤波器定义为, 其中d为像素到图像中心的欧式距离, 为可调参数，影响滤波效果。

1. **import** cv2 as cv
2. **import** numpy as np
3. **import** matplotlib.pyplot as plt
4. %matplotlib
6. img = cv.imread('girl.png', cv.IMREAD\_GRAYSCALE)
7. F = np.fft.fft2(img)
8. Fshift = np.fft.fftshift(F)
10. #高斯高通滤波器构造
11. sigema1 = 0.05
12. #注意不能取uint8型数据
13. mask1 = np.zeros(img.shape)
15. **for** i **in** range(img.shape[0]) :
16. **for** j **in** range(img.shape[1]) :
17. mask1[i][j] = 1 - np.exp((-np.square(i - int(mask1.shape[0] / 2)) - np.square(j - int(mask1.shape[1]  / 2))) / 2 \* sigema1 \*\* 2)
19. Fshift = mask1 \* Fshift
20. img\_back = np.fft.ifftshift(Fshift)
21. img\_back = np.fft.ifft2(img\_back)
22. img\_back = np.abs(img\_back)
23. plt.subplot2grid((2, 2), (0, 0)), plt.imshow(mask1, 'gray'), plt.title('sigema : 0.05')
24. plt.subplot2grid((2, 2), (0, 1)), plt.imshow(img\_back, 'gray'), plt.title('girl'), plt.axis('off')
26. sigema2 = 0.02
27. #注意不能取uint8型数据
28. mask2 = np.zeros(img.shape)
30. **for** i **in** range(img.shape[0]) :
31. **for** j **in** range(img.shape[1]) :
32. mask2[i][j] = 1 - np.exp((-np.square(i - int(mask2.shape[0] / 2)) - np.square(j - int(mask2.shape[1]  / 2))) / 2 \* sigema2 \*\* 2)
34. Fshift = mask2 \* Fshift
35. img\_back = np.fft.ifftshift(Fshift)
36. img\_back = np.fft.ifft2(img\_back)
37. img\_back = np.abs(img\_back)
38. plt.subplot2grid((2, 2), (1, 0)), plt.imshow(mask2, 'gray'), plt.title('sigema : 0.1')
39. plt.subplot2grid((2, 2), (1, 1)), plt.imshow(img\_back, 'gray'), plt.title('girl'), plt.axis('off')
40. plt.tight\_layout()



4. 数字图像加密技术

改变图像像素的最后一位比特(LSB)并不会明显影响图像质量。因此可以用像素的LSB来隐藏信息，达到加密的效果。在本题中，我们尝试把学号信息嵌入到图像中去。步骤如下：

4.1 把你学号的每一个数字用4位bit表示，例如2对应0010。因此二进制串的总长度为数字个数x 4

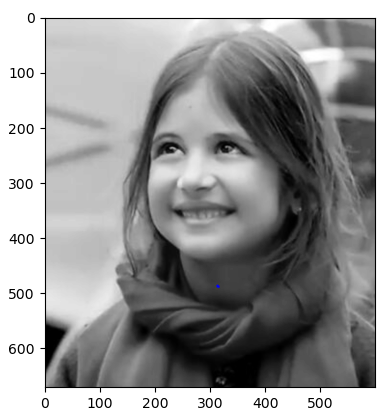
4.2 把图像的每一个像素的最后一个bit变为0，提示：用bitwise\_and

4.3 把4.1中的二进制信息依次写入像素的最后一个bit，即一个像素放一个bit。

**要求：**

* 写出函数img2 = P2\_4\_encode(img, idNumber), 其中输入图像为灰度图，idNumber 为数字字符串, 例如’ 2020281001’, 返回加密后的图像。
* 写出 idNumber = P2\_4\_decode(img) 得到加密信息，即返回你的学号字符串。

1. **import** cv2 as cv
2. **def** P2\_4\_encode(img,  idNumber) :
3. mask = np.zeros(img.shape).astype(np.uint8)
4. mask[ : ,  : ] = 254
5. img = cv.bitwise\_and(img, mask)
6. row, col = img.shape[0], img.shape[1]
7. img = img.ravel()
8. New\_idNumnber = []
9. idNumber = list(idNumber)
10. **for** i **in** range(len(idNumber)) :
11. idNumber[i] = str(bin(int(idNumber[i])).replace('0b', ''))
12. idNumber[i] = list(idNumber[i])
13. **if** len(idNumber[i]) == 1 :
14. **for** k **in** range(3) :
15. idNumber[i].insert(0, '0')
16. **elif** len(idNumber[i]) == 2 :
17. **for** k **in** range(2) :
18. idNumber[i].insert(0, '0')
19. **elif** len(idNumber[i]) == 3 :
20. idNumber[i].insert(0, '0')
21. New\_idNumnber.extend(idNumber[i])
22. **for** i **in** range(len(New\_idNumnber)) :
23. img[i] = img[i] | int(New\_idNumnber[i])
24. img = img.reshape(row,col)
25. **return** img
27. **def** P2\_4\_decode(img) :
28. img = img.ravel()
29. idNumber = []
30. tmp\_idNumber = []
31. count = 0
32. **for** i **in** range(length) :
33. tmp\_idNumber.append(img[i] & 1)
34. count += 1
35. **if** count == 4 :
36. count = 0
37. idNumber.append("".join(map(str, tmp\_idNumber)))
38. tmp\_idNumber = []
39. **for** i **in** range(len(idNumber)) :
40. idNumber[i] = int(idNumber[i], 2)
41. **return** "".join(map(str, idNumber))
43. img = cv.imread('girl.png', cv.IMREAD\_GRAYSCALE)
44. idNumber = '2020281024'
45. img2 = P2\_4\_encode(img, idNumber)
46. plt.imshow(img2, 'gray')
48. length = len(idNumber) \* 4
49. idNumber = P2\_4\_decode(img2)
50. **print**(f'The extracted student number is as follows : {idNumber}')



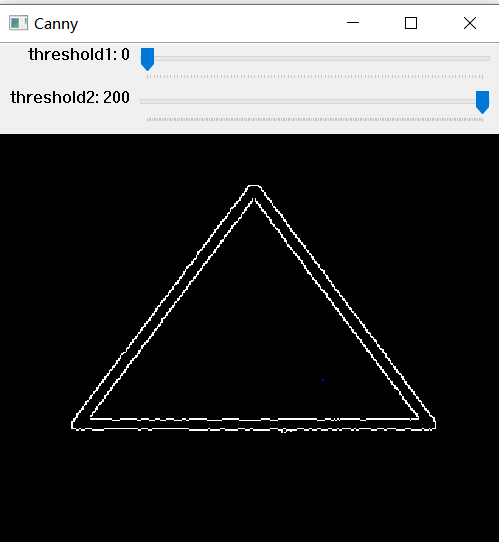
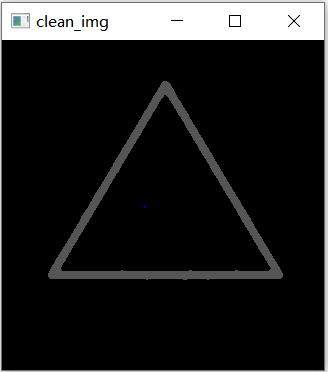
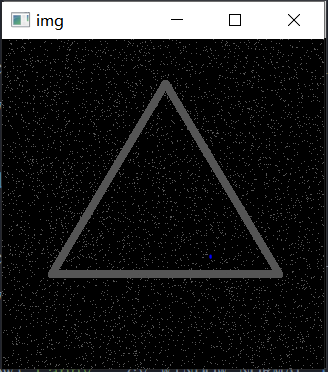


5. 几何形状测量

现在要检查如图triangle\_noise.png 中的三角形跑道，检查底边上跑道的上下边线是否水平以及测量底边跑道的平均宽度（以像素为单位）。用相机对跑道进行拍照，假设相机和跑道底边平行。

5.1 对该图像进行滤波和提取边缘

1. **import** cv2 as cv
2. **import** copy
4. img = cv.imread('triangle\_noise.png', cv.IMREAD\_GRAYSCALE)
5. cv.imshow('img', img)
6. **print**(img.shape)
8. #对图片进行中值滤波
9. cimg = copy.deepcopy(img)
10. img\_clean = cv.medianBlur(cimg, 5)
11. cv.imshow('clean\_img', img\_clean)
13. #边缘提取
14. threshold1\_min = 0
15. threshold1\_max = 100
16. threshold2\_min = 100
17. threshold2\_max = 200
19. **def** Threshold1(x) :
20. gaussian = cv.GaussianBlur(img, (5, 5), 0)
21. canny = cv.Canny(gaussian, x, threshold2\_min)
22. cv.imshow('Canny', canny)
24. **def** Threshold2(x) :
25. gaussian = cv.GaussianBlur(img, (5, 5), 0)
26. canny = cv.Canny(gaussian, threshold2\_min, x)
27. cv.imshow('Canny', canny)
29. cv.namedWindow('Canny', cv.WINDOW\_NORMAL | cv.WINDOW\_KEEPRATIO)
30. cv.resizeWindow('Canny', (500, 500))
31. cv.createTrackbar('threshold1', 'Canny', threshold1\_min, threshold1\_max, Threshold1)
32. cv.createTrackbar('threshold2', 'Canny', threshold2\_min, threshold2\_max, Threshold2)
34. cv.waitKey(0)



5.2 设计方案，确定底边的两条边缘是否平行以及其平均宽度。可以假设一定的误差范围，例如5个像素的偏差。

1. **import** cv2 as cv
2. **import** copy
3. **import** numpy as np
4. **import** matplotlib.pyplot as plt
6. img = cv.imread('triangle\_noise.png', cv.IMREAD\_COLOR)
8. img\_draw1 = copy.deepcopy(img)
9. img\_draw2 = copy.deepcopy(img)
10. cimg = copy.deepcopy(img)
11. cimg = cv2.cvtColor(cimg, cv2.COLOR\_BGR2GRAY)
12. #canny二值化
13. gaussian = cv.GaussianBlur(cimg, (5, 5), 0)
14. canny = cv.Canny(gaussian, 70, 200)
15. #霍夫直线检测得到每一条直线的r和theta
16. lines = cv.HoughLines(canny, 1, np.pi / 180, 100)
17. #print(lines.shape) output : (6, 1, 2)
19. #做出所有检测到的直线
20. **for** line **in** lines :
21. **for** rho, theta **in** line:
22. a = np.cos(theta)
23. b = np.sin(theta)
24. #中点
25. x0 = a \* rho
26. y0 = b \* rho
27. #y轴的正方向的极远处
28. x1 = int(x0 + 1000 \* (-b))
29. y1 = int(y0 + 1000 \* (a))
30. #y轴的负方向的极远处
31. x2 = int(x0 - 1000 \* (-b))
32. y2 = int(y0 - 1000 \* (a))
33. cv2.line(img\_draw1, (x1, y1), (x2, y2), (0, 0, 255), 2)
34. #底边在第四、五条线
35. **for** i **in** range(4, 6) :
36. line = lines[i]
37. **for** rho, theta **in** line:
38. a = np.cos(theta)
39. b = np.sin(theta)
40. x0 = a \* rho
41. y0 = b \* rho
42. x1 = int(x0 + 1000 \* (-b))
43. y1 = int(y0 + 1000 \* (a))
44. x2 = int(x0 - 1000 \* (-b))
45. y2 = int(y0 - 1000 \* (a))
46. cv2.line(img\_draw2, (x1, y1), (x2, y2), (0, 0, 255), 2)
47. #计算底边的斜率
48. base\_line1 = -np.cos(lines[4][0][1]) / np.sin(lines[4][0][1])
49. base\_line2 = -np.cos(lines[5][0][1]) / np.sin(lines[5][0][1])
50. #print(base\_line1, base\_line2) output : 4.371139e-08 4.371139e-08
51. **assert**(abs(base\_line1 - base\_line2) < 0.00001)
52. #计算底边之间的宽度
53. d = abs(lines[4][0][0] - lines[5][0][0])
54. **print**(f'The slopes of the two bottom edges are : {base\_line1}, {base\_line2}')
55. **print**(f'The two bottom edges are parallel and the average width is : {d}')
56. titles = ['Original Drawing', 'All Edges', 'The Base Edges']
57. imgs = [img, img\_draw1, img\_draw2]
58. fig = plt.figure()
59. plt.ion()
60. **for** i **in** range(3):
61. ax = plt.subplot(1, 3, i + 1)
62. plt.imshow(cv2.cvtColor(imgs[i], cv.COLOR\_BGR2RGB))
63. ax.set\_title(titles[i])
64. plt.ioff()
65. plt.show()
66. cv.waitKey(0)



