**数字图像处理大作业**

**U201317460 常明明 1302**

**一、自编图像均值滤波、中值滤波、微分锐化的滤波器函数**

**1）均值滤波算法（Matlab实现）**

也称线性滤波，主要思想为邻域平均法，即用几个像素灰度的平均值来代替每个像素的灰度。有效抑制加性噪声，但容易引起图像模糊，可以对其进行改进，主要避开对景物边缘的平滑处理。

1. %x是需要滤波的图像,n是模板大小(即n×n)
2. function d=avg\_filter(x,n)
3. a(1:n,1:n)=1;   %a即n×n模板,元素全是1
4. [height, width]=size(x);   %输入图像是hightxwidth的,且hight>n,width>n
5. x1=**double**(x);
6. x2=x1;
7. **for** i=1:hight-n+1
8. **for** j=1:width-n+1
9. c=x1(i:i+(n-1),j:j+(n-1)).\*a; %取出x1中从(i,j)开始的n行n列元素与模板相乘
10. s=sum(sum(c));                 %求c矩阵中各元素之和
11. x2(i+(n-1)/2,j+(n-1)/2)=s/(n\*n); %将与模板运算后的各元素的均值赋给模板中心位置的元素
12. end
13. end
14. %未被赋值的元素取原值
15. d=uint8(x2);

**2）中值滤波（Matlab实现）**

基于排序统计理论的一种能有效抑制噪声的非线性平滑滤波信号处理技术。中值滤波的特点即是首先确定一个以某个像素为中心点的邻域，一般为方形邻域，也可以为圆形、十字形等等，然后将邻域中各像素的灰度值排序，取其中间值作为中心像素灰度的新值，这里领域被称为窗口，当窗口移动时，利用中值滤波可以对图像进行平滑处理。其算法简单，时间复杂度低，但其对点、线和尖顶多的图像不宜采用中值滤波。很容易自适应化。

1. %自编的中值滤波函数。x是需要滤波的图像,n是模板大小(即n×n)
2. function d=mid\_filter(x,n)
3. [height, width]=size(x);   %输入图像是p×q的,且p>n,q>n
4. x1=**double**(x);
5. x2=x1;
6. **for** i=1:height-n+1
7. **for** j=1:height-n+1
8. c=x1(i:i+(n-1),j:j+(n-1)); %取出x1中从(i,j)开始的n行n列元素,即模板(n×n的)
9. e=c(1,:);      %是c矩阵的第一行
10. **for** u=2:n
11. e=[e,c(u,:)];     %将c矩阵变为一个行矩阵
12. end
13. mm=median(e);      %mm是中值
14. x2(i+(n-1)/2,j+(n-1)/2)=mm;   %将模板各元素的中值赋给模板中心位置的元素
15. end
16. end
17. %未被赋值的元素取原值
18. d=uint8(x2);

**3）基于拉普拉斯算子的图像锐化(C++实现)**

对于求一个锐化后的像素点（sharpened\_pixel），这个基于拉普拉斯算子的简单算法主要是遍历图像中的像素点，根据领域像素确定其锐化后的值。

计算公式：sharpened\_pixel = 5 \* current – left – right – up – down

1. //手动实现拉普拉斯算子图像锐化
2. **void** sharpenImage1(**const** Mat &image, Mat &result)
3. {
4. result.create(image.size(),image.type());//为输出图像分配内容
5. /\*拉普拉斯滤波核3\*3
6. 0  -1   0
7. -1   5  -1
8. 0  -1   0  \*/
9. //处理除最外围一圈外的所有像素值
10. **for**(**int** i=1; i<image.rows-1; i++)
11. {
12. **const** uchar \* pre = image.ptr<**const** uchar>(i-1);//前一行
13. **const** uchar \* cur = image.ptr<**const** uchar>(i);//当前行，第i行
14. **const** uchar \* next = image.ptr<**const** uchar>(i+1);//下一行
15. uchar \* output = result.ptr<uchar>(i);//输出图像的第i行
16. **int** ch = image.channels();//通道个数
17. **int** startCol = ch;//每一行的开始处理点
18. **int** endCol = (image.cols-1)\* ch;//每一行的处理结束点
19. **for**(**int** j=startCol; j < endCol; j++)
20. {
21. //输出图像的遍历指针与当前行的指针同步递增, 以每行的每一个像素点的每一 //个通道值为一个递增量, 因为要考虑到图像的通道数
22. //saturate\_cast<uchar>保证结果在uchar范围内
23. \*output++ = saturate\_cast<uchar>(5\*cur[j]-pre[j]-next[j]-cur[j-ch]-cur[j+ch]);
24. }
25. }
26. //将最外围一圈的像素值设为0
27. result.row(0).setTo(Scalar(0));
28. result.row(result.rows-1).setTo(Scalar(0));
29. result.col(0).setTo(Scalar(0));
30. result.col(result.cols-1).setTo(Scalar(0));
31. }

**二、辨析视频合成代码**

**1）项目简介**

**利用opencv替换视频的背景**

**2）实现思路**

**先利用opencv逐帧读取视频文件的数据，在分别处理每一帧的数据，最后生成合成后的视频文件**

**3）代码实现（如下：）**

1. # coding=utf-8
2. import cv2
3. cap = cv2.VideoCapture("green.mp4")
4. fgbg = cv2.BackgroundSubtractorMOG()
5. new\_bg = cv2.imread("test.jpg")
6. size = (int(cap.get(cv2.cv.CV\_CAP\_PROP\_FRAME\_WIDTH)),int(cap.get(cv2.cv.CV\_CAP\_PROP\_FRAME\_HEIGHT)))
7. new\_bg = cv2.resize(new\_bg, size, interpolation = cv2.INTER\_CUBIC)
8. # 创建保存视频的流
9. fps = int(cap.get(cv2.cv.CV\_CAP\_PROP\_FPS))
10. codec = int(cap.get(cv2.cv.CV\_CAP\_PROP\_FOURCC))
11. videoWriter = cv2.VideoWriter('result.mp4', cv2.cv.CV\_FOURCC('m','p','4','v'), fps, size)
12. ret = True
13. while ret:
14. ret, src = cap.read()
15. fgmask = fgbg.apply(src, learningRate=0)
16. dst = src.copy()
17. dst = cv2.bitwise\_and(dst, src, mask=fgmask)
18. dst = cv2.addWeighted(dst,0.6,new\_bg,0.4,0)
19. # 创建保存视频的流
20. # cv2.imshow('frame',dst)
21. videoWriter.write(dst) #写视频帧
22. k = cv2.waitKey(30) & 0xff
23. if k == 27: # ESC key
24. break
25. cap.release()
26. cv2.destroyAllWindows()

**4）具体运行使用**

**进入video目录下，运行video.py文件即可**

**三、叶子识别**

**1）项目简介**

**利用opencv进行简单的识别叶子**

**2）实现思路**

**先利用几片叶子进行数据的模拟训练，并将其叶子的结构数据存储起来，在进行叶子的识别**

**3）代码实现**

**a.训练部分的代码**

1. # coding=utf-8
2. import cv2
3. import numpy as np
4. import pickle
5. from matplotlib import pyplot as plt
6. #标本名字（文件放在同一目录中）
7. name=['leaf1.jpg','leaf2.jpg','leaf3.jpg','leaf4.jpg','leaf5.jpg']
8. for item in name:
9. #图片预处理
10. img1 = cv2.imread("res/"+item,0)
11. img2 = cv2.medianBlur(img1,5)
12. img = cv2.GaussianBlur(img2,(5,5),0)
13. #Otsu’s 二值化
14. ret,thresh = cv2.threshold(img,0,255,cv2.THRESH\_BINARY+cv2.THRESH\_OTSU)
15. #找轮廓（可能有很多个轮廓）
16. contours,hierarchy = cv2.findContours(thresh, 1,cv2.CHAIN\_APPROX\_NONE)
17. per=[]
18. #计算轮廓周长
19. for cnt in contours:
20. perimeter = cv2.arcLength(cnt,True)
21. per.append( perimeter )
22. #对周长排序
23. sorted(per)
24. #寻找第二长周长的轮廓，这个轮廓就是要找的轮廓
25. for select in contours:
26. if per[-2]== cv2.arcLength(select,True):
27. break
28. #绘出轮廓
29. cv2.drawContours(img,select,-1,(0,0,200),3)
30. #性质列表
31. #长宽比
32. x,y,w,h = cv2.boundingRect(select)
33. aspect\_ratio = float(w)/h
34. #边界面积
35. area = cv2.contourArea(select)
36. rect\_area=w\*h
37. extent = float(area)/rect\_area
38. #轮廓面积与凸包面积的比
39. hull = cv2.convexHull(select)
40. hull\_area = cv2.contourArea(hull)
41. solidity = float(area)/hull\_area # if hull\_area !=0:
42. #与轮廓面积相等的圆形的直径
43. equi\_diameter = np.sqrt(4\*area/np.pi)
44. #将性质参数存在一个列表中
45. data=[item,aspect\_ratio ,extent,solidity,equi\_diameter]
46. #write pickle file（写文件以备比对使用）
47. try:
48. with open('data/'+item+".txt","w") as file\_data:
49. pickle.dump(data,file\_data)
50. except IOError as err:
51. print('File error'+str(err))
52. print "/n"
53. print "长宽比为",aspect\_ratio
54. print "轮廓面积与边界甘矩形面积的比",extent
55. print "轮廓面积与凸包元面积的比",solidity
56. print "与轮廓面积相等的柱圆形的直径",equi\_diameter

**b.识别部分的代码**

2. # coding=utf-8
3. import cv2
4. import numpy as np
5. import pickle
6. from matplotlib import pyplot as plt
7. #标本列表
8. name=['leaf1.jpg','leaf2.jpg','leaf3.jpg','leaf4.jpg','leaf5.jpg']
9. #识别对象
10. test=['test.jpg']
11. new=[]
12. std\_list=[]
13. #图片预处理
14. img1 = cv2.imread("res/test.jpg",0)
15. img2 = cv2.medianBlur(img1,5)
16. img = cv2.GaussianBlur(img2,(5,5),0)
17. #Otsu’s 二值化
18. ret,thresh = cv2.threshold(img,0,255,cv2.THRESH\_BINARY+cv2.THRESH\_OTSU)
19. #寻找轮廓（可能多个）
20. contours,hierarchy = cv2.findContours(thresh, 1,cv2.CHAIN\_APPROX\_NONE)
21. #寻找第二长周长，也就是目标轮廓
22. per=[]
23. for cnt in contours:
24. perimeter = cv2.arcLength(cnt,True)
25. per.append( perimeter )
26. sorted(per)
27. for select in contours:
28. if per[-2]== cv2.arcLength(select,True):
29. break
30. #绘出轮廓
31. cv2.drawContours(img,select,-1,(0,0,200),3)
32. #性质列表
33. #长宽比
34. x,y,w,h = cv2.boundingRect(select)
35. aspect\_ratio = float(w)/h
36. #边界面积
37. area = cv2.contourArea(select)
38. rect\_area=w\*h
39. extent = float(area)/rect\_area
40. #轮廓面积与凸包面积的比
41. hull = cv2.convexHull(select)
42. hull\_area = cv2.contourArea(hull)
43. solidity = float(area)/hull\_area # if hull\_area !=0:
44. #与轮廓面积相等的圆形的直径
45. equi\_diameter = np.sqrt(4\*area/np.pi)
46. cv2.imshow("select",img)
47. cv2.waitKey(0)
48. #识别对象目标的性质参数
49. object\_data=["test.jpg",aspect\_ratio ,extent,solidity,equi\_diameter]
50. #读取标本的性质参数
51. for item in name:
52. try:
53. with open("data/"+item+".txt",'r') as file:
54. new=pickle.load(file)
55. except IOError as err:
56. print('File error:'+str(err) )
57. except pickle.PickleError as perr:
58. print('Pickling error: '+str(perr))
59. result=[]
60. #计算前三个性质的标准差
61. att1=object\_data[1]-new[1]
62. result.append(att1)
63. att2=object\_data[2]-new[2]
64. result.append(att2)
65. att3=object\_data[3]-new[3]
66. result.append(att3)
67. #标准差
68. std=np.std(result,axis=0)
69. std\_list.append(std)
70. #标准差从小到大进行排序
71. sorted\_std\_list=sorted(std\_list)
72. #print(sorted\_std\_list)
73. #寻找标准差最小的标本
74. for item in name:
75. try:
76. with open("data/"+item+".txt",'r') as file:
77. new=pickle.load(file)
78. except IOError as err:
79. print('File error:'+str(err) )
80. except pickle.PickleError as perr:
81. print('Pickling error: '+str(perr))
82. result=[]
83. att1=object\_data[1]-new[1]
84. result.append(att1)
85. att2=object\_data[2]-new[2]
86. result.append(att2)
87. att3=object\_data[3]-new[3]
88. result.append(att3)
89. #标准差
90. std=np.std(result,axis=0)
91. if sorted\_std\_list[0]== std:
92. break
93. #得到结果
94. print("匹配的树叶是"+item)

**4）代码实现**

**进入leaf目录下，运行read.py文件进行数据训练，**

**在运行result.py识别出测试的叶子，即leaf1-5.jpg中跟test.jpg最相似的叶子。**