可视化作业2

18300290007 加兴华

Note: Please provide a report file (in PPT/word/pdf) of your homework consisting of everything including description of data, code and result. Also please submit a zip file to elearning including the report, data/image if you use, and source code (with comments).

文件说明:

附件包含 otsu+bgt阈值法.py、local_otsu.m、linear_interpolation.m分别对应作业题1, 2, 3;

另,所使用照片位于"图片库"文件夹中,将其移至相应环境根目录或者修改代码中路径即可正常运行程序;

以及第三题保存的处理图像也位于"图片库"文件夹中。

1

Restate the Basic Global Thresholding (BGT) algorithm so that it uses the histogram of an image instead of the image itself. (Please refer to the statement of OSTU algorithm)

代码: (python)

```
import matplotlib.pyplot as plt
 2 from skimage import io
 3
 4 # 定义函数实现寻找otsu阈值
 5 def otsu_threshold(img):
 6
       rows,cols = img.shape
 7
      N = rows*cols
8
       bin=[0 for x in range(256)]
9
       # 得到图片的灰度分布列
10
       for x in range(rows):
11
           for y in range(cols):
               bin[img[x,y]] = bin[img[x,y]]+1
12
13
       s_{max} = (0, -1)
       # 遍历寻找最优划分阈值
14
15
       for threshold in range(0,255):
           n_0 = sum(bin[:threshold]) # 阈值以下像素数
16
           n_1 = sum(bin[threshold:]) # 阈值以上像素数
17
18
           # 两侧频率
19
           w_0 = n_0/N
20
           w_1 = n_1/N
           # 阈值下平均灰度
21
           u_0 = sum([i*bin[i] for i in range(0, threshold)])/n_0 if n_0>0 else
22
    0 #考虑极端划分
           # 阈值上平均灰度
23
24
           u_1 = sum([i*bin[i] for i in range(threshold, 256)])/n_1 if n_1>0
    else 0
```

```
# 总平均灰度
25
26
           u = w_0*u_0 + w_1*u_1
27
           # 类间方差
28
           Dbet2 = w_0*((u_0-u)**2) + w_1*((u_1-u)**2)
29
           # 跟先前最优比较(取绝对大者意味着右偏,二分时阈值需分配给左半边)
30
           if Dbet2>s_max[1]:
31
               s_{max}=(threshold,Dbet2)
32
       return s_max[0]
33
34
    # 定义函数实现寻找bgt阈值
35
    def bgt_threshold(img):
36
       rows,cols = img.shape
37
       bin=[0 for x in range(256)]
       # 设定初始阈值
38
39
       threshold=123
       t=0
40
41
       # 得到图片的灰度分布列
42
       for x in range(rows):
43
           for y in range(cols):
44
               bin[img[x,y]] = bin[img[x,y]]+1
       # 迭代至阈值收敛
45
46
       while True:
47
           t = threshold
           n_0 = sum(bin[:int(threshold)]) # 阈值以下像素数
48
49
           n_1 = sum(bin[int(threshold):]) # 阈值以上像素数
           # 阈值下平均灰度
           u_0 = sum([i*bin[i] for i in range(0, int(threshold))])/n_0 if n_0>0
51
    else 0 #考虑极端划分
52
           # 阈值上平均灰度
53
           u_1 = sum([i*bin[i] for i in range(int(threshold), 256)])/n_1 if
    n_1>0 else 0
54
           # 得到新阈值
55
           threshold = (u_0+u_1)/2
56
           if abs(t-threshold)<0.1:
57
               break
58
       return threshold
59
60 #主函数部分
   path='图片库//finger.tif'
61
62
   img = io.imread(path)
63 # 原始图像
64
    plt.subplot(1,3,1)
65
   io.imshow(img)
66 | plt.title('origin image')
67
    plt.axis("off")
68
69
    # otsu法
70
   threshold=otsu_threshold(img)
71 print("otsu_threshold is",threshold)
72
   # 根据阈值二分
73
   rows,cols = img.shape
74
    for i in range(rows):
75
        for j in range(cols):
76
           img[i,j]=0 if img[i,j]<=threshold else 255</pre>
77
    plt.subplot(1,3,2)
78
   io.imshow(img)
79
    plt.title('after otsu_threshold')
    plt.axis("off")
```

```
81
82 # bgt法
83 | img = io.imread(path)
84 threshold=bgt_threshold(img)
85 print("bgt_threshold is",threshold)
86 rows, cols = img.shape
87 # 根据阈值二分
88 for i in range(rows):
89
       for j in range(cols):
90
            img[i,j]=0 if img[i,j]<=threshold else 255</pre>
91 plt.subplot(1,3,3)
92 io.imshow(img)
93 plt.title('after bgt_threshold')
94 plt.axis("off")
95 # 展示
96 plt.show()
```

运行结果:

otsu_threshold is 160 bgt_threshold is 159.40149844112722



分析: bgt法与otsu法得到的阈值接近且处理图片结果都很好,两种方法可以相互印证

2

Design an algorithm of locally adaptive thresholding based on local OTSU or maximum of local entropy; implement the algorithm and test it on exemplar image(s).

代码: (matlab) (基于局部otsu)

```
1 clear all;clc;close all;
2 img=imread("图片库//writting.tif");
```

```
3 subplot(2,2,1);
   imshow(img);
   title("origin image");
6 bin=makebin(img);
   t=otsu_find(bin,0);
8
   img2=partition(img,t);
9
   subplot(2,2,2);
   imshow(img2);
10
11 | title("global otsu")
12
   img3=partition(img,0);
13
   subplot(2,2,3);
14
   imshow(img3);
15
   title("threshold = 0")
16 img4=local_otsu_do(img,5);
17
   subplot(2,2,4);
   imshow(img4);
18
19
   title("local otsu with k=5")
20
21 % otsu寻找阈值的函数
22
   function t=otsu_find(bin,k)
23 s_{max} = [0,0];
24 N=sum(bin(:));
25
   i=0:255;
              %递增行向量,用于算期望
  for threshold=1:256
26
27
       u=0;
       n_0 = sum(bin(1:threshold)); % 阈值以下像素数
28
29
       n_1 = sum(bin(threshold:256)); % 阈值以上像素数
30
       % 两侧频率
31
       w_0 = n_0/N;
32
       w_1 = n_1/N;
33
       % 阈值下平均灰度
34
       if(n_0>0)
35
           u_0 = i(1:threshold)*bin(1:threshold)/n_0;
36
       else
37
           u_0=0; %考虑极端划分
38
       end
39
       % 阈值上平均灰度
40
       if(n_1>0)
41
           u_1 = i(threshold:256)*bin(threshold:256)/n_1;
42
       else
43
           u_1=0; %考虑极端划分
44
       end
45
       % 总平均灰度
46
       u = w_0*u_0 + w_1*u_1;
47
       % 类间方差
       Dbet2 = w_0*((u_0-u)^2) + w_1*((u_1-u)^2);
48
49
       % 跟先前最优比较(取绝对大者意味着右偏,二分时阈值需分配给左半边)
50
       if (Dbet2>s_max(2))
51
           s_max=[threshold-1,Dbet2];
       end
52
53
   end
   t=s_max(1);
   %局部法需开启的修正模式: 当选区中存在奇异点,则让更新点跟随大流实现减弱噪音干扰
55
56
   if(k==1)
       if(w_0>0.95|w_1>0.95)
57
58
           if(w_0>0.5)
59
               t=0;
60
           else
```

```
61
                t=255;
 62
            end
 63
         end
 64
     end
 65
     end
 66
 67
     % 生成bin的函数
 68
     function bin=makebin(img)
    [m,n]=size(img);
 69
 70
    bin=zeros(256,1);
     for i=1:m
 71
 72
         for j=1:n
 73
            bin(img(i,j)+1)=bin(img(i,j)+1)+1;
 74
         end
 75
     end
 76
     end
 77
 78
     % 根据阈值进行二分的函数
 79
     function img_new=partition(img,t) % t为阈值
     [m,n]=size(imq);
 81
     imq_new=zeros(m,n);
 82
     for i=1:m
 83
         for j=1:n
 84
            if(img(i,j) \le t)
 85
                img_new(i,j)=0;
 86
            else
 87
                img_new(i,j)=255;
 88
            end
 89
         end
 90
     end
 91
     end
 92
 93
     % 局部otsu的函数
 94
    function img_new=local_otsu_do(img,k) % k为滑窗大小,应为奇数
 95
     [m,n]=size(img);
 96
    img_new=zeros(m,n);
 97
     q=(k-1)/2; % 滑窗半径
 98
     b=zeros(m+2*q,n+2*q); %扩张工作空间减少边界讨论
99
    b(1+q:m+q,1+q:n+q)=img; %搬运原数组
100
    % 柔化扩充部分边界
    b(1:q,1+q:n+q)=flipud(img(2:q+1,1:n));
101
102
     b(m+q+1:m+2*q,1+q:n+q)=flipud(img(m-q:m-1,1:n));
103
     b(1+q:m+q,1:q)=fliplr(img(1:m,2:q+1));
104
     b(1+q:m+q,1+n+q:n+2*q)=fliplr(img(1:m,n-q:n-1));
105
     bin=makebin(b(1:1+2*q,1:1+2*q)); %初始化
106
    flag=1; % 采用缝针式滑窗, flag用于判断当前行是左滑还是右滑以及更新bin
107
     j=1+q;
108
    % 直接对上次作业滑窗代码进行改写
109
    for i=1+q:m+q
110
         t=otsu_find(bin,1);
111
         img_new(i-q,j-q)=partition(b(i,j),t);
112
         sym=(1-flag)/2; %配合flag实现判断当前行是左滑还是右滑
113
         qqq=1; %判断是不是一行的开始的标志,如果是则跳过一次更新
114
         for j=(sym*(2*q+1+n)+flag*(1+q)):flag:(sym*(2*q+1+n)+flag*(n+q))
115
            if(qqq==1)
116
                qqq=0;
117
                continue;
118
            end
```

```
119
              for x=i-q:i+q
120
                  bin(b(x,j-flag*(q+1))+1) = bin(b(x,j-flag*(q+1))+1)-1;
                  bin(b(x,j+flag*q)+1) = bin(b(x,j+flag*q)+1)+1;
121
122
              end
123
              t=otsu_find(bin,1);
124
              img_new(i-q,j-q)=partition(b(i,j),t);
125
         end
126
         flag=-1*flag;
127
         if(i\sim=m+q)
128
              for x=j-q:j+q
129
                  bin(b(i-q,x)+1) = bin(b(i-q,x)+1)-1;
130
                  bin(b(i+1+q,x)+1) = bin(b(i+q+1,x)+1)+1;
131
              end
132
          end
133
     end
134
     end
```

运行结果:



infrinte six between stakling of fringe different father of turnessy histories and father of turnessy histories and totally sometimes for all the same for the same for all same father through the same hand had he they breaked athe buf by they presents of all alien exports and a father who has heard and a band a father of parall of a same arrey on, throwsand after same arrey on, throwsand arre



local obsu with k=5
Ind rinkty by betness Itakkey
of May Carefitat of Europes
Luther Jaker of the other park
Lat afor Baif of the other park
Lat afor Baif of the other park
Lat afor Baif of two thousand
the Jam of two thousand
hand haid the tweet presents
but baif of tweet presents
but land as their presents
but aline who of and confer
Custom that of parall of La
Dandaren on parall of La

分析:

- 1.我使用的图片素材为上课使用的,但经过我研究认为这一素材不适合用于局部otsu,原因在于:正如上面左下图 (t=0) 处理结果与局部otsu基本一样,原因在于字迹点灰度0,其余均严格 > 0,因此使用固定阈值就可以处理这样的图像,比局部otsu高效太多了。
- 2.局部otsu需要配合修正使用,如果不修正,那么背景内部点常被误分为前景点,素材中体现为有背景为黑色(如下图,为运行更快只对图片的一小块应用代码)



2. (续) 但是!如果修正的条件不合适,可能会不能完全修正,比如我第一次使用类间方差很小作为条件,结果修正图像正中间亮度最大处有坏点(如下图)。经研究发现使用集中性条件最好,即选区内极多数点都被划分到一侧时进行修正。

indrinty by letnew & f Kny and Stay of the hudrew Jackson of the that aforesaid of the other aid stocker Donelson of the Sum of two the hand paid the true praise of alien enfort and Jackson his heirs Ertain traits or hards

3

编程实现线性插值算法(不能调用某个算法库里面的插值函数),并应用:读出一幅图像,利用线性插值把图片空间分辨率放大N倍,然后保存图片。

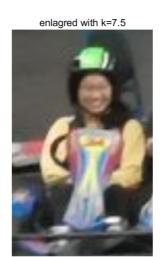
代码: (matlab)

```
1 | clc;clear all;close all;
   2
            img=imread('图片库//yxy.jpg');%读取图像信息
            % 可灰度化,下文代码兼容
   3
   4
           % img=rgb2gray(img);
   5
            subplot(121);
   6
           imshow(img);%显示原图
  7
            title('origin image');
   8
             subplot(122);
  9
            b=enlarge(img, 7.5);
10
            imshow(b);%显示处理后图
11
            title('enlagred with k=7.5')
             imwrite(b,'./图片库/双线性插值放大结果.png'); % 将图片保存到图片库
12
13
            %进行双线性放大的函数
14
15
             function b=enlarge(img,k) % k为放大倍率,可以为非整数,但要大于一
16
             [m,n,c]=size(img); % 记录原图数组三维
             B=zeros(m+1,n+1,c); % 扩张原图数组省去(1)处讨论边界
17
18
            B(1:m,1:n,:)=img; % 搬运
19
           % 生成目标图数组的工作空间
20
            M=ceil(k*m); %向上取整
21
            N=ceil(k*n);
22
             b=zeros(M,N,c);
            % 对目标图数组每个元素依此双线性插值
23
24
           for i=1:M
25
                       for j=1:N
26
                                   % 将新图坐标仿射变换到原图, (x,y)为插值所用点的左上点坐标, u、v为组合系数
27
                                    x=floor((i-1)*(m-1)/(M-1)+1);
28
                                    y=floor((j-1)*(n-1)/(N-1)+1);
29
                                    u=(i-1)*(m-1)/(M-1)+1-x;
30
                                    v=(j-1)*(n-1)/(N-1)+1-y;
31
                                    b(i,j,:)=(1-v)*(1-u)*B(x,y,:)+v*(1-u)*B(x,y+1,:)+(1-u)*B(x,y+1,:)+(1-u)*B(x,y+1,:)+(1-u)*B(x,y+1,:)+(1-u)*B(x,y+1,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B(x,y,:)+(1-u)*B
             v)*u*B(x+1,y,:)+u*v*B(x+1,y+1,:); % (1)
32
                        end
33
             end
```

运行结果:







分析: 经双线性插值放大后明显看到像素块不那么明显了,说明处理确实增加了分辨率。

在写代码时我最初将m*n阵中(i, j)元素先映到(ki, kj)再插值,但完善后左边和上边会存在宽度为k-1的黑边,若直接用邻域像素填充会让图形扭曲,实际只把原图放大到右下角(km-k+1,kn-k+1)一块,放大倍率小于k,且全图不是均匀放大,对此我不能接受,最后采用了放射变换整数化进行插值,实现真正意义上等比放大k倍。