DIP-Lab1

黄川懿 520030910268

2022 年 11 月 3 日

摘要

本次 Lab 的主要目的是复现论文 Lisheng Wang, Jing Bai, Threshold selection by clustering gray levels of boundary, Pattern Recognition Letters, 24 (2003) 1983–1999 的基于边界灰度聚类选择阈值 (其最佳阈值由位于其连续边界上的点的灰度值的平均值确定) 的算法，并且对于给定的图像进行处理分析

目录

1 简介

1.1 实验目的

本次实验的主要目的是通过复现论文 Lisheng Wang, Jing Bai, Threshold selection by clustering gray levels of boundary, Pattern Recognition Letters, 24 (2003) 1983–1999 的阈值选择算法, 进一步了解图像分割中的一项重要技术：阈值选择，阈值假定图像呈现许多组件，每个组件基本都有一个均匀的值，并且可以通过选择适当的强度阈值分离组件，已经学习过的阈值算法有大律法，直方图法，自适应阈值等，这些算法对于阈值的选择都是基于整个图像或者进行了适当分割的图像的特性。而这次复现的论文则引入了优化的思想，通过选取隐含定义的连续曲线, 即边界（两边的灰度值都有急剧的变化），通过对误差函数取最小值求取域值

1.2 环境需求

- OS X environment: Anconda 4.12.0 Python 3.7, Numpy 1.19.5, cv2 4.5.5, scipy 1.5.4 and matplotlib 3.1.1.
- Visual Studio Code V1.71.2

1.3 原理及过程

主体分为选取边界离散点, 以及阈值选择两部分：

选取边界离散点：将计算二维图像内位于边界上的点的灰度值的离散采样，并从这些离散采样中估计平均值，而二维图像被视为从二维规则网格的网格点采样的离散采样数据，如下图所示，其中所有正方形构成二维图像占据的连续区域。由于二维图像中对象的边界是包含在连续区域中的一些连续曲线，因此它们会将所有正方形的集合分为两类：边缘单元，即与边界相交的正方形，以及非边缘单元。边界包含在所有边缘单元的集合中。在每个边缘单元的四个边中，至少有两条边与边界相交，所以通过检查每个正方形中是否存在至少两个交互边缘，可以识别所有边缘单元。具体方法如下：计算图像的像素点的梯度幅值和拉普拉斯值, 对于选定为正方形一条边顶点的点, 通过下面式子判断是否和边界相交

$$\begin{aligned} l(p_1) \cdot l(p_2) &< 0 \\ g(p_1) + g(p_2) &\geq 2T, \end{aligned} \tag{1}$$

第一个式子说明两个顶点 $l(p_1) \cdot l(p_2)$ 是一对过零点，第二个式子说明亮点梯度大于预定义的梯度阈值通过统计正方形中有交点的边的个数，可以判断哪些正方形是边界单元。即可以找出边界离散点。此外，边界离散点的灰度值由线性插值获得。阈值选择：分为双层阈值和多层阈值选择：

对于双层阈值选择：最佳阈值由位于其边界上的点的灰度值的平均值确定。其计算原

理为：解决优化问题：

$$\min_{r} \int_{C_{(x,y)}} (f(x,y)-r)^2 d(x,y), r \in R \tag{2}$$

令 $F(r)=\int_{C_{(x,y)}}(f(x,y)-r)^2 d(x,y)$ 对 r 求导，则有

$$\dot{F}(r)=\int_{C_{(x,y)}} 2f(x,y)d(x,y)-\int_{C_{(x,y)}} 2rd(x,y), r \in R \tag{3}$$

求得为边界上的点的灰度值的平均值：

$$r=\frac{\int_{C_{(x,y)}} f(x,y)d(x,y)}{\int_{C_{(x,y)}} d(x,y)} \tag{4}$$

对于多层阈值选择：由于每个边界的离散采样点的灰度值会围绕它们的均值聚集在一起，所以每个边界的离散采样点的灰度值将在所有边界的离散采样点的直方图中显示为不同的簇。所以可以获取边界离散点的直方图，通过峰值检测方法，即选择每个主簇峰值处的灰度级作为阈值。这里选择了 scipy.signal 中的 find_peaks () 函数来选择峰值。

2 具体实现与源码解释

为了方便处理，将整个过程封装成库：Threshold_Processer, 其初始化为（为了区别边界单元和边界，令 EDGE 代表边界单元，edge 为边界）

```
class Threshold_Processer:
    def __init__(self,filepath,t,rev_flag,multi_flag) ->
        None:
        self.img=load_img(filepath)
        self.gray = cv2.cvtColor(self.img, cv2.
            COLOR_BGR2GRAY)\\灰度图
        self.w,self.h=self.gray.shape
        self.T=t
        self.k=1
        self.flag=rev_flag
        self.mul_flag=multi_flag
```

由过程所示：

2.1 选取边界离散点

首先，选取二维规则网格的网格点 (正方形), 这里设置正方形的边长为 2 像素点：

```
    def GET_EDGE(self,loc):
        edge1=(loc,[loc[0]+self.k,loc[1]])
        edge2=(loc,[loc[0],loc[1]+self.k])
        edge3=([loc[0]+self.k,loc[1]],[loc[0]+1,loc[1]+
            self.k])
        edge4=([loc[0],loc[1]+self.k],[loc[0]+1,loc[1]+
            self.k])
        return [edge1,edge2,edge3,edge4]
```

接下来计算二维图中每个点的梯度幅值:

```
    def Gradient_Mag(self):
        # Prewitt 算子计算梯度
        kernelx = np.array([[-1,0,1],[-1,0,1],[-1,0,1]],
            dtype=int)
        kernely = np.array([[1,1,1],[0,0,0],[-1,-1,-1]],
            dtype=int)
        x = cv2.filter2D(self.gray, cv2.CV_64F, kernelx)
        y = cv2.filter2D(self.gray, cv2.CV_64F, kernely)
        Grad_Mag = sqrt(x**2+y**2)
        return Grad_Mag
```

接下来计算二维图中每个点的拉普拉斯值

```
    def Laplacian(self,use_CV2_mol=false):
        if (use_CV2_mol):
            cv_show('lap',cv2.Laplacian(self.gray,cv2.
                CV_64F))
            return cv2.Laplacian(self.gray,cv2.CV_64F)
        else:
            lap_kernel=np.array
                ([[0,1,0],[1,-4,1],[0,1,0]],dtype=int)
            lap=cv2.filter2D(self.gray,cv2.CV_64F,
                lap_kernel)
            return lap
```

接下来通过（1）中二式检查是否是边界单元

```
    def check_EDGE(self,lap,grad,edges):
        count=0
```

```
        flag=false
        cross_edges=[]# 记录相交的edge
        for edge in edges:
            if (lap[edge[0][0],edge[0][1]]*lap[edge[1][0],
                edge[1][1]]<0 and grad[edge[0][0],edge
                [0][1]]+grad[edge[0][0],edge[0][1]]>=2*self
                .T):
                count+=1
                cross_edges.append(edge)
            if (count>=2):
            flag=True
        return flag,cross_edges
```

遍历图像，找到所有的边界单元，选取边界离散点

```
    def edge_dectect(self):
        FIN_edge=[]
        FIN_EDGE_IMG=np.zeros((self.w,self.h))
        lap=self.Laplacian()
        grad=self.Gradient_Mag()
        for i in range(self.w-self.k):
            for j in range(self.h-self.k):
                edges=self.GET_EDGE([i,j])
                is_FIN_edge,cross_edges=self.check_EDGE(
                    lap,grad,edges)
                if (is_FIN_edge):
                    for cross_edge in cross_edges:
                        FIN_edge.append(self.
                            _interpolation(cross_edge,lap))
                    FIN_EDGE_IMG[i][j]=self.gray[i][j]
        return FIN_edge,FIN_EDGE_IMG
```

在遍历途中，若确定一个边界单元，则通过线性插值获得其灰度值：

```
\\类外
def _interp(x,xp,fp):
    output = fp[0] + (x - int(xp[0])) * ((int(fp[1]) - int
        (fp[0]))/(int(xp[1]) -int(xp[0])))
    return output
\\类中
```

```
    def _interpolation(self,edge,lap):
        start=lap[edge[0][0]][edge[0][1]]
        end=lap[edge[1][0]][edge[1][1]]
        inter_gray=_interp(0,[start,end],[self.gray[edge
            [0][0]][edge[0][1]],self.gray[edge[1][0]][edge
            [1][1]]])
        return inter_gray
```

找出边界离散点后，通过

2.2 阈值选择

首先观察直方图，选择以那种方式进行处理（双层阈值和多层阈值）直方图的获取如下所示:

```
    def histogram(self,filename):
        fin_edge,_=self.edge_dectect()
        fd=np.array(fin_edge).astype(np.float32)
        plt.figure(figsize=(5.5, 2), dpi=40)
        plt.hist(fd, 256*2, [0, 256])
        plt.savefig(filename)
        plt.close()
```

确定方式，进行处理：双层阈值求取边界离散点的灰度平均值，多层阈值选取峰值对应灰度值

```
    def select_Threshold(self):
        fin_edge,_=self.edge_dectect()
        return np.average(fin_edge)
    def select_Threshold_mul(self):
        fin_edge,_=self.edge_dectect()
        hist,gray_scales=np.histogram(fin_edge
            ,256*2,(0,255))
        peaks,_ = scipy.signal.find_peaks(hist,distance
            =150)
        threshes=[]
        for peak in peaks:
            threshes.append(gray_scales[peak])
        return threshes
```

最后，获取阈值分离后的图像并保存

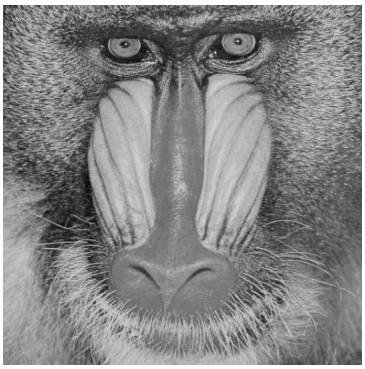
```
    def process(self,filename):
        if self.mul_flag:
            threshes=self.select_Threshold_mul()
            res=[]
            back_groud_index=0
            for ts in threshes:
                _,thresh = cv2.threshold(self.gray,ts,255,
                    cv2.THRESH_BINARY)
                if back_groud_index==1:
                    res.append(res[0]-thresh)
                back_groud_index+=1
                res.append(thresh)
            res=np.hstack(res)
            cv2.imwrite(filename,res)
        else:
            ts=self.select_Threshold()
            if (self.flag):
                ret,thresh = cv2.threshold(self.gray,ts
                    ,255,cv2.THRESH_BINARY_INV)
            else:
                ret,thresh = cv2.threshold(self.gray,ts
                    ,255,cv2.THRESH_BINARY)
            cv2.imwrite(filename,thresh)
```

3 实验结果及数据

3.1 双层阈值实验结果

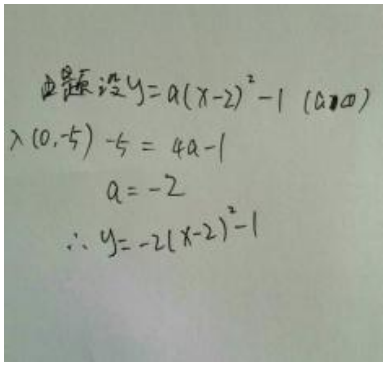
通过观察离散边界点的直方图，可以发现以下图像 fig1-5 (图1)的离散边界点的直方图 fig6-10(图2) 都表现出一个主峰，或者唯一明显的聚类. 其中 fig6,7,8,9,10 分别对应为 fig1,2,3,4,5 的直方图，

(a) fig1

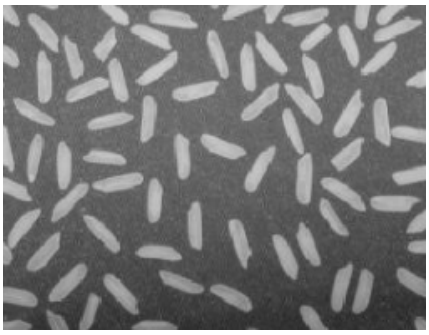


(b) fig2

(c) fig3

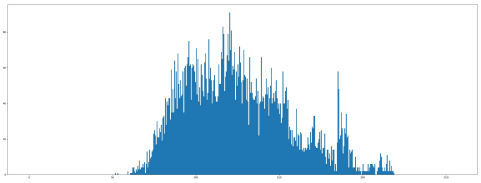


(d) fig4

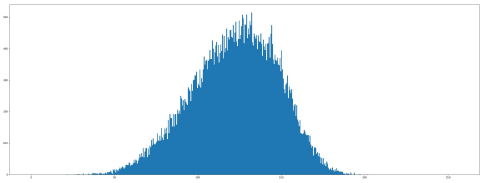


(e) fig5

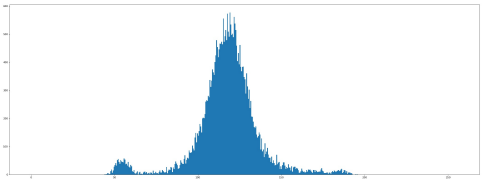
图 1: OriginFigures



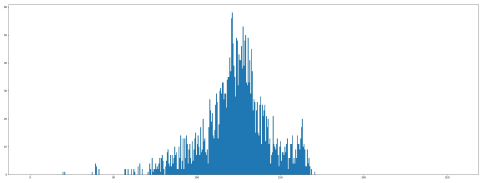
(a) fig6



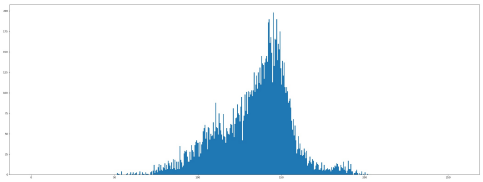
(b) fig7



(c) fig8



(d) fig9



(e) fig10

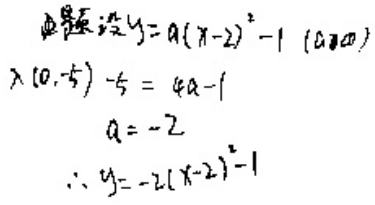
图 2: HistFigures

这样的图像可以求取边界离散点的灰度平均值作为阈值进行分离，实验结果为 fig11-15 （图3）. 其中 fig11-fig15 对应于 fig1-fig5 的结果

(a) fig11 t=165

(b) fig12 t=122

(c) fig13 t=90



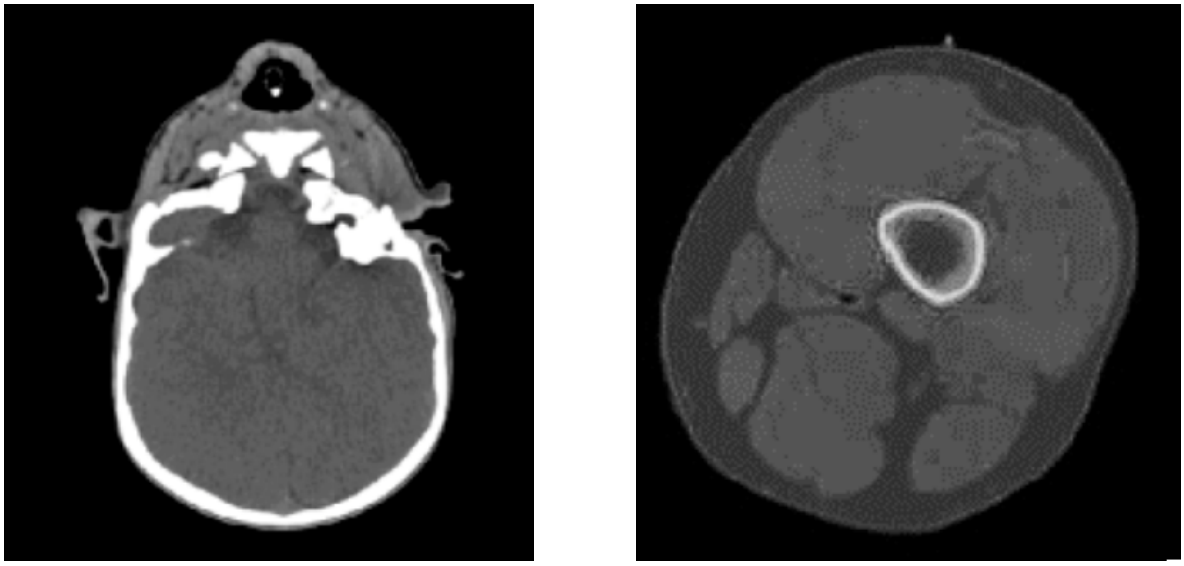
(d) fig14 t=148

(e) fig15 t=75

图 3: ResultFigures

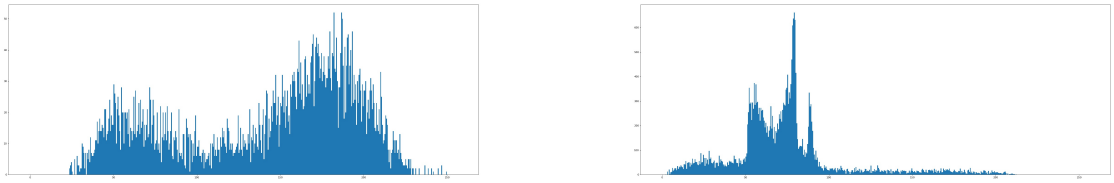
3.2 多级阈值选择

通过观察离散边界点的直方图，可以发现以下图像 fig16-17(图4) 的离散边界点的直方图 fig18-19(图5) 都表现出多个主峰，围绕其均值聚集成不同的簇。. 其中 fig18,19 分别对应为 fig16,17 的直方图，



(a) fig16 (b) fig17

图 4: Multi-threshold OriginFigures



(a) fig18 (b) fig19

图 5: Multi-threshold HistFigures

可以发现:

在 fig16(图4 (a)) 中，考虑从包含两个不同对象（背景、骨骼和软组织）的 2D CT 头部图像中进行多级阈值选择。图像的直方图如 fig18(图5左一) 所示。它有一个宽阔的山谷, 存在两个明显的簇。它们分别对应于软问题和骨骼的边界。通过计算两个聚类的峰值对应的灰度值，得到骨骼和软组织的阈值。相应的分割结果如 fig208(图6(a)) 所示

在 fig17(图4 (b)) 中，从包含三个不同对象（骨骼、肌肉和结缔组织）的 2D CT（腿部切片）图像中进行多级阈值选择。图像的直方图如 fig19(图5左二) 所示。它有一个宽阔的山谷和一个不太明显的山谷，以及三个不规则的山峰。意味着存在三个明显的簇，分别对应于骨骼、肌肉和结缔组织的边界。通过计算三个聚类的峰值对应的灰度值，得到骨骼、肌肉和结缔组织的阈值。相应的分割结果如 fig21(图6(b)) 所示

(a) fig20

(b) fig21

图 6: Multi-threshold ResultFigures

4 敏感度分析

4.1 噪声敏感度分析

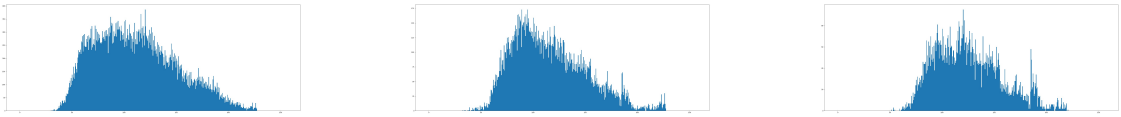
由于没有在提供的图像中找到原论文中分析所用的图像, 仅给出对理论分析的理解: 由于边界离散采样点的计算是由二维图像的拉普拉斯函数值决定, 所以其对于噪声的鲁棒性取决于拉普拉斯算子对噪声的敏感性。一般来说拉普拉斯算子受噪声影响大, 但是该方法由于只选取高梯度点而不是零交叉点。故其对于噪声的鲁棒性有一定程度的提升。

4.2 对梯度阈值 T 的敏感性分析

原论文对比梯度算子产生的边界，边界离散采样点的直方图以及阈值方法计算的双层阈值图像受梯度阈值 T 的变化程度来分析敏感度，实验结果 (从左到右依次选择梯度阈值 40 100 160) 如下：

(a) Prewitt Edge

图 7: Sensitivity of Prewitt Edge



(a) Histogram t=40 (b) Histogram t=100 (c) Histogram t=160

图 8: Sensitivity of Histogram

(a) Threshold Figure

图 9: Sensitivity of Threshold Figure

可以发现当梯度阈值 T 选取三个不同的值，边缘图 ((图7)) 变化很大，边界离散采样点的直方图 ((图8)) 发生变化。然而，通过所提出的阈值方法计算的双层阈值图 ((图9)) 只有很小的变化。特别是它们对应的二值图像差别不大。表明提出的阈值方法可以在一定程度上保持稳定

5 总结

通过复现论文的阈值选择算法，进一步了解图像分割中的一项重要技术：阈值选择。了解到了阈值选择的一些优点，并且从更多的角度了解如何对于图像进行处理，比如这里引入了优化的思想。拓宽了我对于图像处理的思路，十分感谢能给予我这次复现论文的机会。

6 附录: 对梯度阈值 T 的敏感度分析源码

```
from ThresholdProcessor import Threshold_Processer
import matplotlib.pyplot as plt
import numpy as np
from genericpath import isdir
import cv2
import os
T=[40,100,160]
edge_imgs=[]
thresh_imgs=[]
if not isdir('Sensitivity_Analysis'):
    os.mkdir('Sensitivity_Analysis')
for t in T:
    tp=Threshold_Processer('PRLetter-images/1_gray.bmp',t,
        False,False)
    _,edge=cv2.threshold(tp.Gradient_Mag(),t,255,cv2.
        THRESH_BINARY_INV)
    edge_imgs.append(edge)
    ts=tp.select_Threshold()
    _,thresh=cv2.threshold(tp.gray,ts,255,cv2.
        THRESH_BINARY)
    thresh_imgs.append(thresh)
    tp.histogram('Sensitivity_Analysis/hist'+str(t)+'.jpg'
        )
edge_fin=np.hstack(edge_imgs)
thresh_fin=np.hstack(thresh_imgs)
cv2.imwrite('Sensitivity_Analysis/edge.jpg',edge_fin)
cv2.imwrite('Sensitivity_Analysis/thresh.jpg',thresh_fin)
```