数字图像处理-作业 1

程铭

2020 年 4 月 14 日

1 不同阈值大小的阈值分割结果及分析

此次作业中，采用全局阈值分割算法，且共设定了 5 种阈值大小：25，75，125，175，225. 阈值分割结果（简单图像、复杂图像）如下：

2 简单图像

简单图像的阈值分割结果如图 1 所示。

由上述结果可知，阈值大小的选取对阈值分割结果有较大的影响。在图 1中，当阈值为 125 时，分割结果较好，可以将鸟、树枝与白色的背景分离开来；阈值过小（25）则会将物体都当做背景来看待，导致图片白色部分较多，物体分割效果不好；阈值过大（225）则会将背景都当做物体来看待，导致图片黑色部分较多，物体分割效果不好。因此选择合适大小的阈值十分重要。

3 复杂图像

复杂图像的阈值分割结果如图 2 所示。

由图 2可知，当阈值过小时，大部分物体内容都被分为背景，导致图片较白：例如，阈值取 75 时（图 2(c)），大象的耳朵、鼻子等都被当做是背

(a) 简单图像，原图

(b) 简单图像，阈值 25

(c) 简单图像，阈值 75

(d) 简单图像，阈值 125

(e) 简单图像，阈值 175

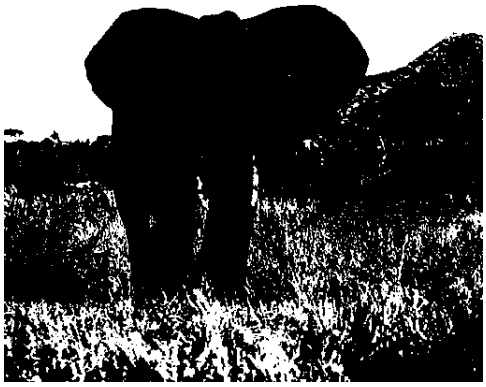
(f) 简单图像，阈值 225

图 1: 简单图像不同阈值大小的阈值分割结果展示

(a) 复杂图像，原图

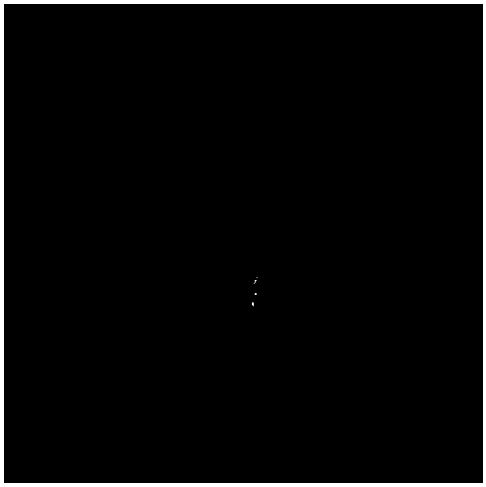
(b) 复杂图像，阈值 25

(c) 复杂图像，阈值 75



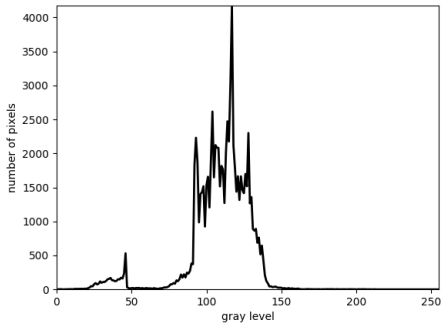
(d) 复杂图像，阈值 125

(e) 复杂图像，阈值 175

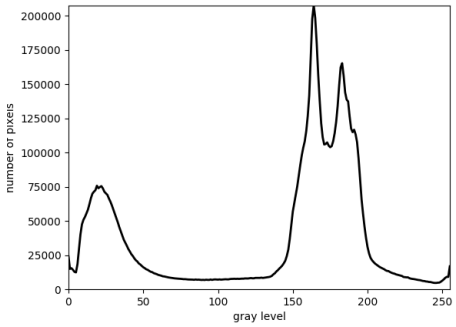


(f) 复杂图像，阈值 225

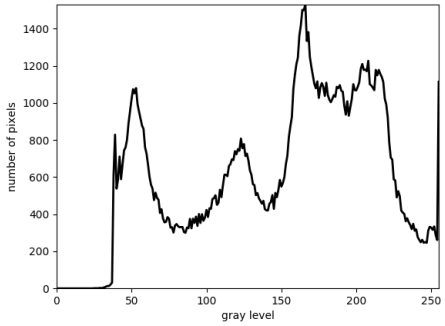
图 2: 复杂图像不同阈值大小的阈值分割结果展示



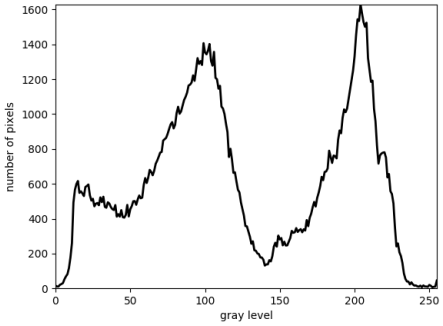
(a) 简单图像事例 1



(b) 简单图像事例 2



(c) 复杂图像事例 1



(d) 复杂图像事例 2

图 3: 简单图像和复杂图像的灰度分布图展示

景；当阈值过大时，大部分背景内容都被当做物体，导致图片较黑：例如，阈值取 125 时（图 2(d)），山脉等背景内容都被当做是物体，呈黑色，无法将其与大象分离开来。

另外，对于复杂图像，可以看到无论阈值取多少，都不能得到很好的分割结果。原因分析如下：

图 3展示了简单图像和复杂图像的灰度分布结果图。由图可知，对于简单图像，灰度分布的波峰较少，且各个波峰之间间隔较远、存在一定的波谷；而对于复杂图像，灰度分布的波峰较多，且分布在较多的像素范围，没有明显的波谷区域（即在不同的灰度范围内都有分布）。对于阈值分割算法，通常将像素阈值选取在波谷处，可以较好地分隔波峰。例如，对于图 3(a)和图 3(b), 分别选取阈值为 75，125，可以得到较好的分隔结果。而对于图 3(c)和图 3(d), 没有明显的波谷存在，因此无论选取的阈值大小如何，都无法分离各个波峰，因此不能得到较好的分割结果。

4 阈值分割算法适用场景

- 由于阈值分割算法实现了“将图像像素分类“的任务，因此较适用于目标和背景占据不同灰度范围的图像，即图像背景和目标灰度值区别明显：对应的灰度分布图即为几个较为独立的波峰。将阈值选在波谷处，即可以得到较好的分割结果。对于较为复杂的图像，且目标和图像背景的灰度范围较为接近，则不适用。

- 由于阈值分割算法只考虑像素点灰度值本身的特征，一般不考虑空间特征，因此对噪声较为敏感，适用于没有被噪声污染的图像。

代码

```
import numpy as np
import os
import cv2 as cv2
import matplotlib.pyplot as plt

def getFigName(path):
    fig_lis = []
    for _, _, fname in os.walk(path):
        for fig in fname:
            str_ = path + fig
            fig_lis.append(str_)
    return fig_lis

def getGrayHist(img_lis):
    for img in img_lis:
        image = cv2.imread(img, 0)
        img_name_ = img.split('/')[-1]
        img_name = img_name_.split('.')[0]
        print(img_name)
        rows, cols = image.shape
        grayHist = np.zeros([256], np.uint64)
        x = range(256)
        for r in range(rows):
            for c in range(cols):
                grayHist[image[r][c]] += 1

        plt.plot(x, grayHist, 'r', linewidth=2, c='black')
        y_maxValue = np.max(grayHist)
        plt.axis([0, 255, 0, y_maxValue])
        plt.xlabel("gray level")
        plt.ylabel("number of pixels")

        if 'easy' in img:
            plt.savefig('./hist/easy_' + img_name)
        elif 'complex' in img:
            plt.savefig('./hist/complex_' + img_name)
        # plt.show()

def getThresh(path, thresh):
    for img in path:
        for th_ in thresh:
            print(img)
            img_name = img.split('/')[-1].split('.')[0]
            img_cv2 = cv2.imread(img)
```

```
            ret, th = cv2.threshold(img_cv2, th_, 255, cv2.THRESH_BINARY)
            cv2.waitKey(0)
            cv2.destroyAllWindows()
            if 'easy' in img:
                cv2.imwrite('./thresh/easy_' + str(th_) + '_' + img_name +
'.png', th)
                # plt.savefig('./thresh/easy_' + str(th_) + '_' + th)
            elif 'complex' in img:
                cv2.imwrite('./thresh/complex_' + str(th_) + '_' + img_name +
'.png', th)



if __name__ == "__main__":
    path_easy = '../easy/'
    path_complex = '../complex/'
    ## 遍历图片
    fname_lis_easy = getFigName(path_easy)
    fname_lis_complex = getFigName(path_complex)
    ## 灰度分布图
    getGrayHist(fname_lis_easy)
    getGrayHist(fname_lis_complex)
    ## 阈值分割
    threshold_ = [25, 75, 125, 175, 225]
    getThresh(fname_lis_easy, threshold_)
    getThresh(fname_lis_complex, threshold_)
```