《计算机视觉（1）》实验报告

实验五 利用边缘改进全局阈值处理

实验小组成员 （学号+班级+姓名）	分工及主要完成任务	成绩
201800810253+数据科学+王荣	图 2 的处理、实验报告	
201800820149+数据科学+徐潇涵	图 1 的处理	

山东大学

2021 年 4 月

完成《数字图像处理》P485页例10.17和10.18的编程实验，编程语言可以选择Matlab，C，C++，OpenCV，Python等。设计方案可参照教科书中的分析，也可以自行设计新的方案。

（1） 图10.42(a)为一幅包含均值为零、标准差为10个灰度级的加性高斯噪声的图像，其中相对于背景，目标所占像素比例极小。采用以梯度为基础的边缘信息结合全局阈值处理方法，实现小目标的分割。

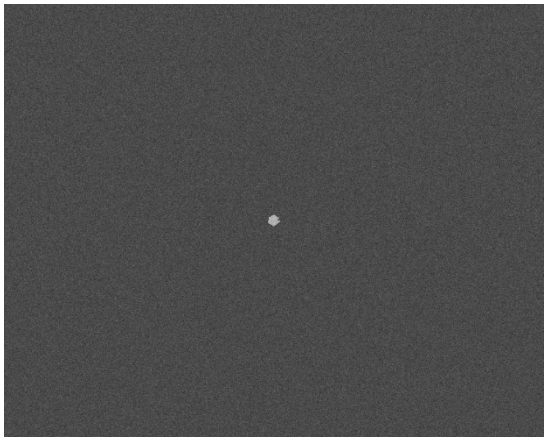


图10.42(a)

（2）图10.43(a)显示了一幅酵母细胞的8比特图像，采用以拉普拉斯为基础的边缘信息结合全局阈值处理方法，实现亮点目标的分割。

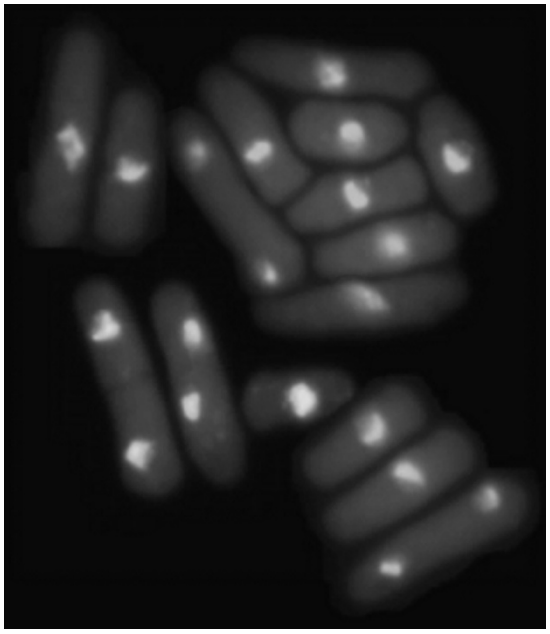


图10.43(a)

原始图像的电子版图像在 Images 文件夹中。实验报告写在如下空白处，页数不限。

实验报告

一、 图像 1 的处理

1. 总体思路

我们对图像 1 采用以梯度为基础的边缘信息结合全局阈值处理方法，实现中心圆点目标的分割。主要的三步骤如下：

（1）计算原图经阈值处理后的梯度幅度图像；

（2）将原图与（1）中图像进行乘积，绘制新图中非零像素的直方图；

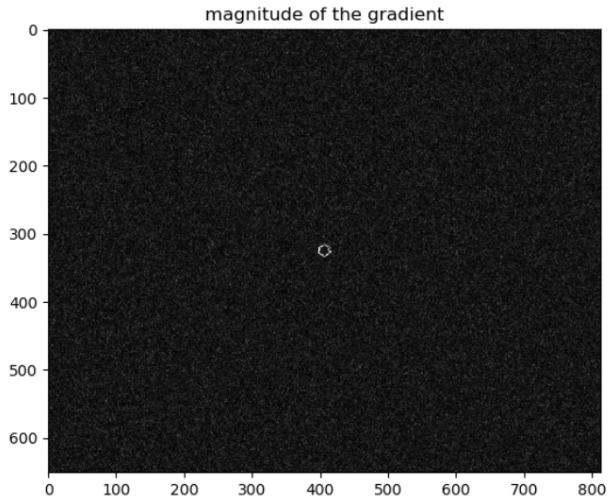
（3）以(2)中的直方图为基础，使用 Otsu 方法对原图像进行阈值处理，即可得到中心圆点分割的图像。

2. 实验细节

1) 计算梯度幅度图像

我们利用 Sobel 算子分别计算出图像 x 方向和 y 方向的一阶导数，再利用公式计算得到梯度值。下面依次展示了相关代码和梯度幅度图像。

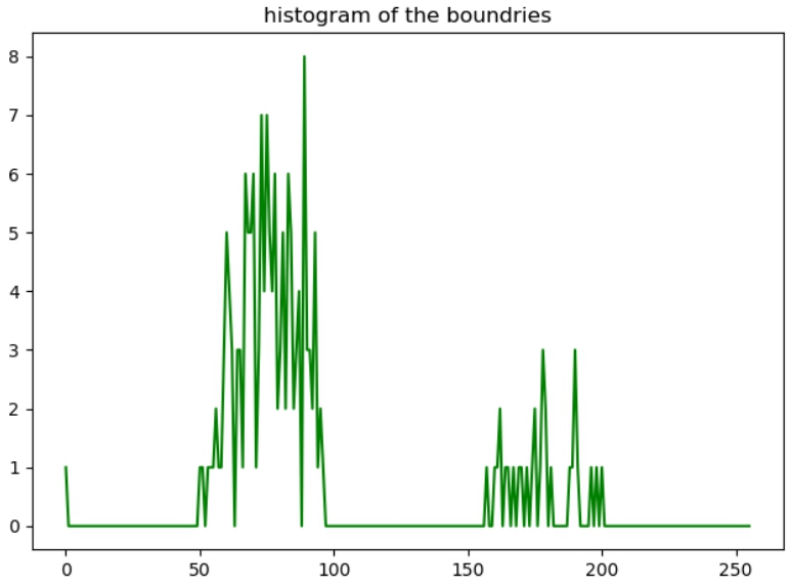
```
img = cv2.imread('pic1.tif', 0)
gray1 = np.double(img)
x = cv2.Sobel(gray1, -1, 1, 0)  # x方向的一阶导数
y = cv2.Sobel(gray1, -1, 0, 1)  # y方向的一阶导数
grad = (np.sqrt(x**2 + y**2))
plt.figure(), plt.imshow(grad, cmap='gray')
plt.title('magnitude of the gradient')
```



2) 绘制边缘像素的直方图

我们将梯度幅度图像当作遮罩图像，利用 cv2.calcHist 函数选择性地绘制非零像素的直方图。

```
hist = cv2.calcHist([img], [0], mask_bw, [256], [0, 256])
hist[0] = 1
plt.figure(), plt.plot(hist, color = 'g')
plt.title('histogram of the boundries')
```



3) 根据 Otsu 算法对原图进行阈值处理

如下是自定义的 Otsu 算法。根据算法的原理，我们遍历 0-255 的所有取值，在每个取值处计算两个类的类间方差。最后函数返回使得类间方差最大的阈值取值。

```
def OTSU(img_gray):
    max_g = 0
    suitable_th = 0
    th_begin = 0
    th_end = 256
    for threshold in range(th_begin, th_end):
        bin_img = img_gray > threshold
        bin_img_inv = img_gray <= threshold
        fore_pix = np.sum(bin_img)
        back_pix = np.sum(bin_img_inv)
        if 0 == fore_pix:
            break
        if 0 == back_pix:
            continue

        w0 = float(fore_pix) / img_gray.size
        u0 = float(np.sum(img_gray * bin_img)) / fore_pix
        w1 = float(back_pix) / img_gray.size
        u1 = float(np.sum(img_gray * bin_img_inv)) / back_pix
```

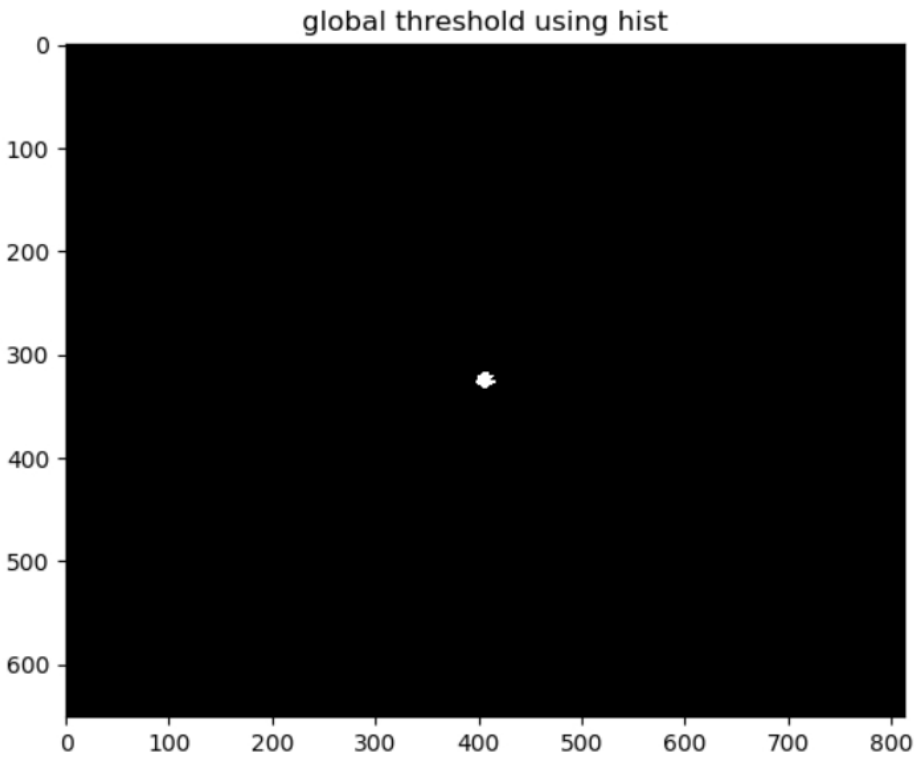
```
    # 类间方差
    g = w0 * w1 * (u0 - u1) * (u0 - u1)
    if g > max_g:
        max_g = g
        suitable_th = threshold

return suitable_th
```

在打印台得到 OTSU 算法得到的最佳阈值为 134。最终效果图如下：

```
ret = OTSU(new)
print(ret)
_, bw = cv2.threshold(img, ret, 255, cv2.THRESH_BINARY)
plt.figure(), plt.imshow(bw, cmap='gray')
plt.title('global threshold using hist')

plt.show()
```



二、 酵母细胞图像的处理

1. 总体思路

我们对酵母细胞图像采用以拉普拉斯为基础的边缘信息结合全局阈值处理方法，实现亮点目标的分割。主要的三步骤如下：

（1）计算原图经阈值处理后的绝对拉普拉斯图像；

（2）将原图与（1）中图像进行乘积，绘制新图中非零像素的直方图；

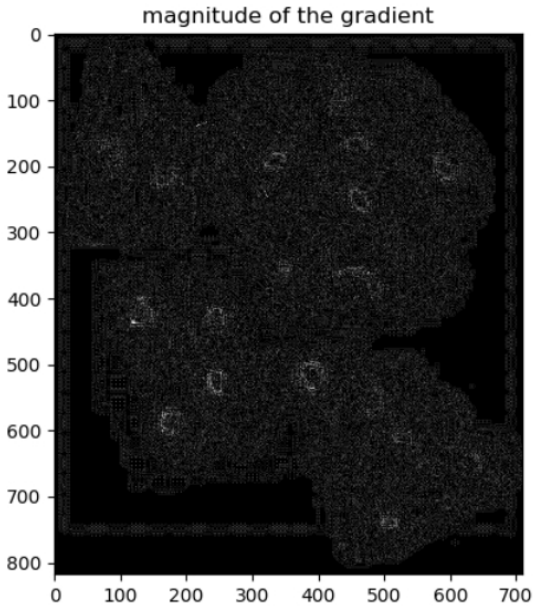
（3）以(2)中的直方图为基础，使用 Otsu 方法对原图像进行阈值处理，即可得到亮点分割的图像。

2. 实验细节

4) 计算绝对拉普拉斯图像

利用 cv2.Laplacian 函数对图像边缘进行检测，并利用 cv2.convertScaleAbs 函数得到绝对拉普拉斯图像。下面依次展示了相关代码和图像。

```
img = cv2.imread('pic2.tif', 0)
gray1 = np.double(img)
laplace = cv2.Laplacian(gray1, cv2.CV_64F)
laplace = cv2.convertScaleAbs(laplace)
plt.figure(), plt.imshow(laplace, cmap='gray')
plt.title('magnitude of the gradient')
```

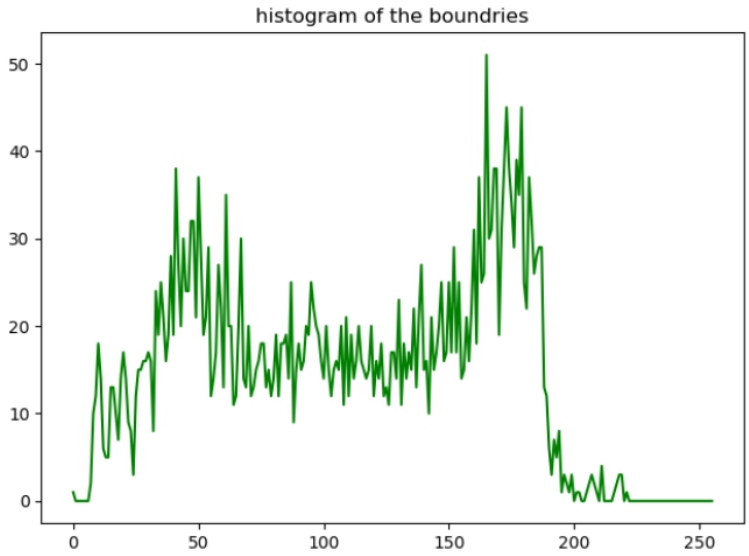


5) 绘制边缘像素的直方图

指定阈值并对原图进行阈值处理，然后将其用作遮罩图像。将原图与遮罩图像进行乘积，得到新图。绘制新图中非零像素的直方图。

```
'''step 2:
   1.指定阈值并对原图进行阈值处理，然后将其用作遮罩图像。 '''
_, mask_bw = cv2.threshold(np.uint8(laplace),
                           np.int(np.max(laplace))*0.28,
                           255, cv2.THRESH_BINARY)
plt.figure(), plt.imshow(mask_bw, cmap='gray')
plt.title('threshold of the magnitude')

'''2.将原图与遮罩图像进行乘积，得到新图。'''
new = np.multiply(img, mask_bw)
'''3.绘制新图中非零像素的直方图。 '''
hist = cv2.calcHist([img], [0], mask_bw, [256], [0, 256])
hist[0] = 1
plt.figure(), plt.plot(hist, color = 'g')
plt.title('histogram of the boundries')
```



6) 根据 Otsu 算法对原图进行阈值处理

Otsu 算法的代码与图像 1 的处理相同，在此处直接调用自定义的 Otsu 函数返回值，对原图进行阈值分割。

```
ret = OTSU(new)
print(ret)
_, bw = cv2.threshold(img, ret, 255, cv2.THRESH_BINARY)
plt.figure(), plt.imshow(bw, cmap='gray')
plt.title('global threshold using hist')

plt.show()
```

在打印台得到 OTSU 算法得到的最佳阈值为 **122**。

最终效果图如下：