实验报告

课程名称 数字图形处理

实验名称 图像分割

专业班级 计算机技术

姓　　名 孙熙春

学　　号 182306057026

一、实验目的

掌握常用的边缘提取算法，从图像中提取感兴趣的区域，实现图像分割。在图像中，寻找灰度相同或相似的区域，区分图像中的背景区域和目标区域，利用 Python 实现图像的边缘检测，进行图像分割。

二、实验环境

软件：Anaconda3， Python3.6

三、实验内容

图像边缘是图像中特性（如像素灰度、纹理等）分布的不连续处，图像周围特性有阶跃变化或屋脊状变化的那些像素的集合。图像边缘存在于目标与背景、目标与目标、基元与基元的边界，标示出目标物体或基元的实际含量，是图像识别信息最集中的地方。

图像分割处理主要用于检测出图像中的轮廓边缘、细节以及灰度跳变部分，形成完整的物体边界，达到将物体从图像中分离出来或将表示同一物体表面的区域检测出来的目的。常用的分割方法是边缘检测。边缘检测是采用多种边缘算子实现突出图像边缘，抑制图像中非边缘信息，使图像轮廓更加清晰。

1. 梯度算子法

对于图像 f(x,y)，它在点 f(x,y)处的梯度是一个矢量，定义为：

$$G[f(x,y)]=\left(\frac{\partial f}{\partial x}\quad\frac{\partial f}{\partial y}\right)^{T}$$

梯度的方向在函数 f(x,y)最大变化率的方向上，梯度的幅值为：

$$|\nabla f|=\sqrt{\left(\frac{\partial f}{\partial x}\right)^{2}+\left(\frac{\partial f}{\partial y}\right)^{2}}$$

梯度的数值就是 f(x,y)在其最大变化率方向上的单位距离所增加的量。对于图像而言，微分运算可以用差分运算来近似。

$$\|\nabla\|=|\Delta x|+|\Delta y|=|f(x,y)-f(x-1,y)|+|f(x,y)-f(x,y-1)|$$

简化成模板可以表示成如下形式：

Robert 梯度算子：$\begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix}$ $\begin{bmatrix}0 & 1\\ -1 & 0\end{bmatrix}$

- 梯度直接输出

当梯度计算完后，可采用以下几种形式突出图像的轮廓。使各点的灰度 g(x,y)等于该点的梯度，即

$$g(x,y)=G[f(x,y)]$$

这种方法简单、直接。但增强的图像仅显示灰度变化比较陡的边缘轮廓，而灰度变换比较平缓的区域则呈暗色。

- 加阈值的梯度输出

加阈值的梯度输出表达式为

$$g(x,y)=\begin{cases}G[f(x,y)] & G[f(x,y)]\geq T\\ f(x,y) & \text{其他}\end{cases}$$

式中，T 是一个非负的阈值，适当选取 T，既可以使明显的边缘得到突出，又不会破坏原来灰度变化比较平缓的背景。

给边缘指定一个特定的灰度级

$$g(x,y)=\begin{cases}L_G & G[f(x,y)]\geq T\\ f(x,y) & \text{其他}\end{cases}$$

式中 LG 是根据需要指定的一个灰度级，它将明显的边缘用一个固定的灰度级表现，而其他的非边缘区域的灰度级仍保持不变。

给背景指定一个特定的灰度级

$$g(x,y)=\begin{cases}G[f(x,y)] & G[f(x,y)]\geq T\\ L_G & \text{其他}\end{cases}$$

该方法将背景用一个固定灰度级 LG 表现，便于研究边缘灰度的变化。

- 二值图像输出

在某些场合（如字符识别等），既不关心非边缘像素的灰度级差别，又不关心边缘像素的灰度级差别，只关心每个像素是边缘像素还是非边缘像素，这时可采用二值化图像输出方式，其表达式为

$$g(x,y)=\begin{cases}L_G & G[f(x,y)]\geq T\\ L_B & \text{其他}\end{cases}$$

此法将背景和边缘用二值图像表示，便于研究边缘所在位置。

2. Sobel 算子法

Sobel 相对于先对图像进行加权平均再做差分。

对于图像的 3×3 窗口 $\begin{bmatrix}a & b & c\\ d & e & f\\ g & h & i\end{bmatrix}$，设 $\begin{cases}X=(c+2f+i)-(a+2d+g)\\ Y=(a+2b+c)-(g+2h+i)\end{cases}$

则定义 Sobel 算子为 $g(x,y)=\left(X^2+Y^2\right)^{\frac{1}{2}}$

简化成模板可以表示成如下形式：

Sobel 模板：

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

3. 拉普拉斯运算法

拉普拉斯算子定义图像 f(x,y)的梯度为

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

锐化后的图像 g 为

$$g = f - k\left[\nabla^2 f\right]$$

其中，k 为扩散效应系数。对系数 k 的选择要合理，太大会使图像中的轮廓边缘产生过冲；太小则锐化不明显。

常用 laplacian 算子模板为：

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \qquad \begin{bmatrix} -1 & 0 & -1 \\ 0 & 4 & 0 \\ -1 & 0 & -1 \end{bmatrix}$$

另外还有一些模板也常用于图像增强，如 Prewitt 模板：

$$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

四、实验结果

1. 自动阈值分割

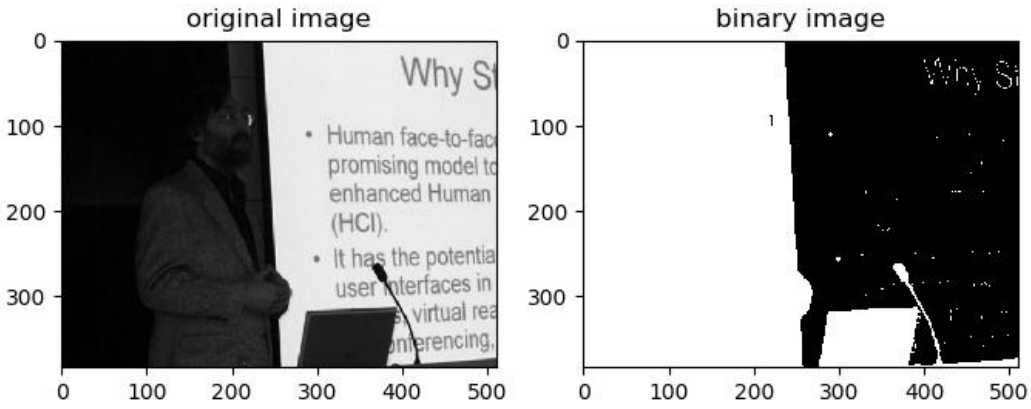
```
from skimage import data,filters
import matplotlib.pyplot as plt
import cv2
image = cv2.imread("../testImage/img3.jpg", 0)

thresh = filters.threshold_otsu(image)     #返回一个阈值

dst =(image <= thresh)*1.0     #根据阈值进行分割

plt.figure('thresh',figsize=(8,8))
plt.subplot(121)
plt.title('original image')
```

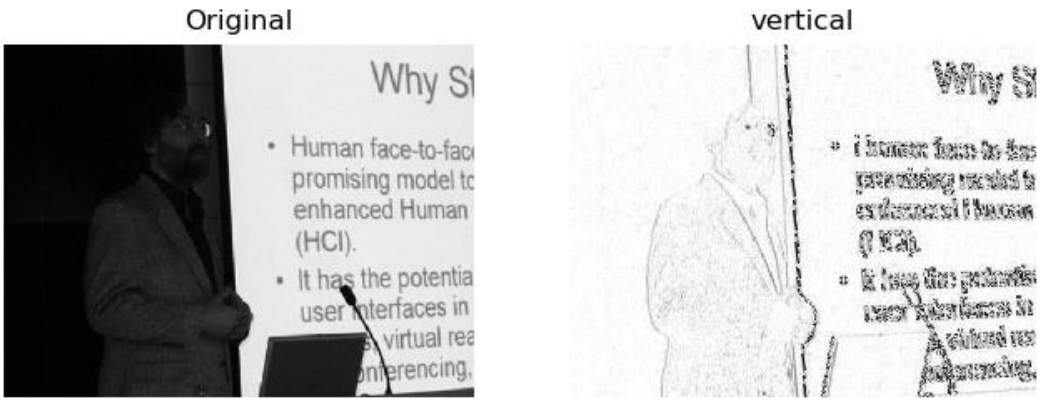
```
plt.imshow(image,plt.cm.gray)
plt.subplot(122)
plt.title('binary image')
plt.imshow(dst,plt.cm.gray)
plt.show()
```



2. Sobel 算子

```
import numpy as np
from skimage import data
import matplotlib.pyplot as plt
import cv2
image = cv2.imread("../testImage/img3.jpg", 0)
def SobelOperator(roi,operator_type):
    if operator_type == "horizontal":
        sobel_operator = np.array([[-1,-2,-1],[0,0,0],[1,2,1]])
    elif operator_type == "vertical":
        sobel_operator = np.array([[-1,0,1],[-2,0,2],[-1,0,1]])
    else:
        raise("type Error")
    result = np.abs(np.sum(roi*sobel_operator))
    return result
def SobelAlogrithm(image,operator_type):
    new_image = np.zeros(image.shape)
    image = cv2.copyMakeBorder(image,1,1,1,1,cv2.BORDER_DEFAULT)
    for i in range(1,image.shape[0]-1):
        for j in range(1,image.shape[1]-1):
            new_image[i-1,j-1]=SobelOperator(image[i-1:i+2,j-
1:j+2],operator_type)
    new_image = new_image*(255/np.max(image))
    return new_image.astype(np.uint8)
```

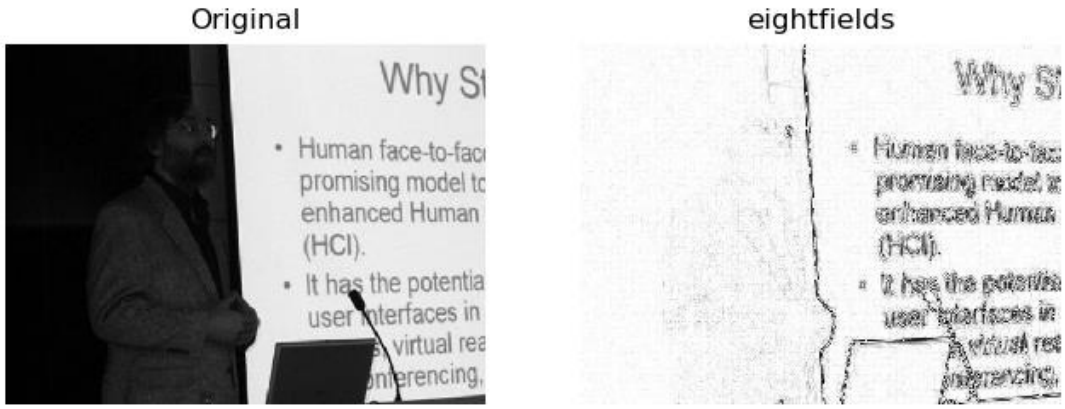
```
plt.figure('thresh',figsize=(8,8))
plt.subplot(121)
plt.title("Original")
plt.imshow(image,plt.cm.gray)
plt.axis("off")
plt.subplot(122)
plt.title("vertical")
plt.imshow(SobelAlogrithm(image,"vertical"),cmap="binary")
plt.axis("off")
plt.show()
```



3. Laplace 算子

```
import numpy as np
from skimage import data
import matplotlib.pyplot as plt
import cv2
image = cv2.imread("../testImage/img3.jpg", 0)
def LaplaceOperator(roi,operator_type):
    if operator_type == "fourfields":
        laplace_operator = np.array([[0,1,0],[1,-4,1],[0,1,0]])
    elif operator_type == "eightfields":
        laplace_operator = np.array([[1,1,1],[1,-8,1],[1,1,1]])
    else:
        raise("type Error")
    result = np.abs(np.sum(roi*laplace_operator))
    return result
def LaplaceAlogrithm(image,operator_type):
    new_image = np.zeros(image.shape)
    image = cv2.copyMakeBorder(image,1,1,1,1,cv2.BORDER_DEFAULT)
    for i in range(1,image.shape[0]-1):
        for j in range(1,image.shape[1]-1):
```

```
                new_image[i-1,j-1]=LaplaceOperator(image[i-1:i+2,j-
1:j+2],operator_type)
        new_image = new_image*(255/np.max(image))
        return new_image.astype(np.uint8)
plt.figure('thresh',figsize=(8,8))
plt.subplot(121)
plt.title("Original")
plt.imshow(image,plt.cm.gray)
plt.axis("off")
plt.subplot(122)
plt.title("eightfields")
plt.imshow(LaplaceAlogrithm(image,"eightfields"),cmap="binary")
plt.axis("off")
plt.show()
```



五、实验总结

Sobel 相对于先对图像进行加权平均再做差分。 在边缘检测中，常用的一种模板是 Sobel 算子。Sobel 算子有三个，一个是检测双向边缘的 ，一个是检测水平边缘的;另一个是检测垂直边缘的。由于 Sobel 算子是一节微分滤波算子的，用于提取边缘，有方向性，从结果可以看出双向 both 的分割效果最好。缺点：Sobel 算子并没有将图像的主体与背景严格地区分开来，换言之就是 Sobel 算子没有基于图像灰度进行处理，由于 Sobel 算子没有严格地模拟人的视觉生理特征，所以提取的图像轮廓有时并不能令人满意。

laplace 算子对边缘的处理明显，它是二阶微分算子，能加强边缘效果，对噪声很敏感。它没有方向性，但是可以改变模板的中间系数，会有不同的效果。