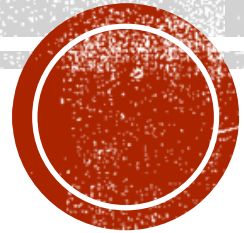
遥感数字图像处理实验课

边缘检测、二值化和轮廓提取与跟踪

李荣昊

2021年12月



实验内容与目的

- 图像边缘检测
- 图像二值化
- 二值化图像轮廓提取与跟踪

边缘检测

- 利用边缘邻近一阶或二阶方向导数变化规律来检测边缘。
- 常见算子
  - ✓ Roberts
  - ✓ Sobel
  - ✓ Prewitt
  - ✓ Laplace
  - ✓ …

边缘检测——Roberts算子

- Roberts算子是利用局部差分算子寻找边缘的算子

$$f_x = f(i, j) - f(i+1, j+1)$$
$$f_y = f(i, j+1) - f(i+1, j)$$

f_x和f_y的卷积算子：

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

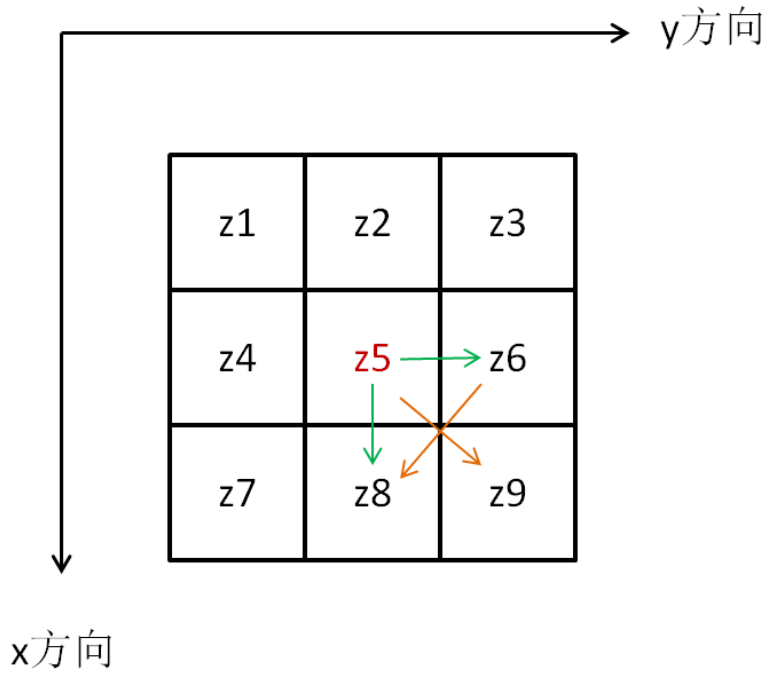
为了简便梯度的计算，可以用以下几种方法简化：

$$g(x,y) = \sqrt{f_x^2 + f_y^2} \quad \text{或者} \quad g(x,y) = |f_x| + |f_y|$$

$$\text{或者} \quad g(x,y) = \max(|f_x|, |f_y|)$$

边缘检测——Roberts算子

- 算法实现



Roberts交叉梯度算子：

f_x= z5-z9

f_y= z6-z8

Roberts交叉梯度:

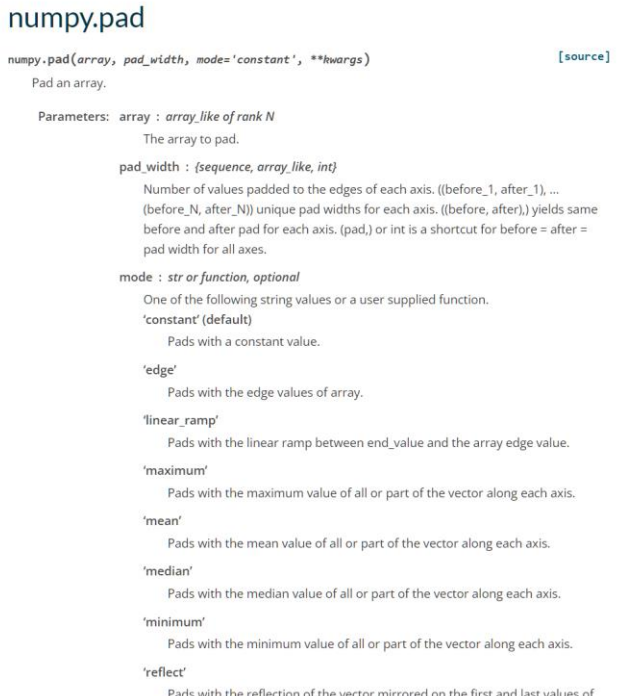
$$g(x,y) = |f_x| + |f_y|$$

边缘检测——Roberts算子

- 实现过程应注意的问题

➢ 将灰度值限制在0-255的范围，计算出来的梯度值很有可能超出这个范围

➢ 图像边缘的处理
扩增图像边缘：扩增边缘灰度值为零，复制原图像的边缘灰度值……



numpy.pad

numpy.pad(*array, pad_width, mode='constant', **kwargs*) [source]

Pad an array.

Parameters: **array** : *array_like of rank N*
The array to pad.

pad_width : *{sequence, array_like, int}*
Number of values padded to the edges of each axis. ((before_1, after_1), ... (before_N, after_N)) unique pad widths for each axis. ((before, after),) yields same before and after pad for each axis. (pad,) or int is a shortcut for before = after = pad width for all axes.

mode : *str or function, optional*
One of the following string values or a user supplied function.
'constant' (default)
Pads with a constant value.
'edge'
Pads with the edge values of array.
'linear_ramp'
Pads with the linear ramp between end_value and the array edge value.
'maximum'
Pads with the maximum value of all or part of the vector along each axis.
'mean'
Pads with the mean value of all or part of the vector along each axis.
'median'
Pads with the median value of all or part of the vector along each axis.
'minimum'
Pads with the minimum value of all or part of the vector along each axis.
'reflect'
Pads with the reflection of the vector mirrored on the first and last values of

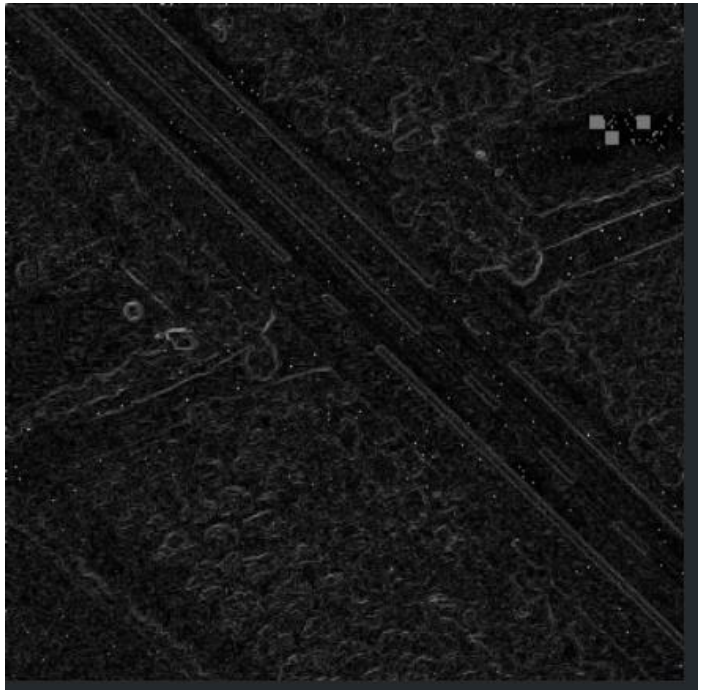
https://numpy.org/doc/stable/reference/generated/numpy.pad.html

边缘检测——Roberts算子

```python
def roberts(image,threshold = 10):
    h,w = image.shape
    image_pad = np.pad(image,((1,1),(1,1)),"edge")
    out = np.zeros((h,w))
    for i in range(h):
        for j in range(w):
            fx = int(image_pad[i][j]) - int(image_pad[i+1][j+1])
            fy = int(image_pad[i][j+1]) - int(image_pad[i+1][j])
            g = abs(fx)+abs(fy)
            if g > threshold:
                out[i][j] = g
            else:
                out[i][j] = image[i][j]
    out = np.clip(out,0,255)
    return np.uint8(out)
```

边缘检测——Roberts算子

灰度影像原图



Roberts算子

阈值为30

边缘检测——Sobel&Prewitt算子

-1	-1	-1
0	0	0
1	1	1

-1	0	1
-1	0	1
-1	0	1

Prewitt边缘检测算子

-1	-2	-1
0	0	0
1	2	1

-1	0	1
-2	0	2
-1	0	1

Sobel边缘检测算子

边缘检测——Sobel算子

- Sobel算子f_x和f_y的卷积算子

$$Gx=\begin{bmatrix}-1 & -2 & -1\\ 0 & 0 & 0\\ 1 & 2 & 1\end{bmatrix}*f(x,y)_{3\times3} \qquad Gy=\begin{bmatrix}-1 & 0 & 1\\ -2 & 0 & 2\\ -1 & 0 & 1\end{bmatrix}*f(x,y)_{3\times3}$$

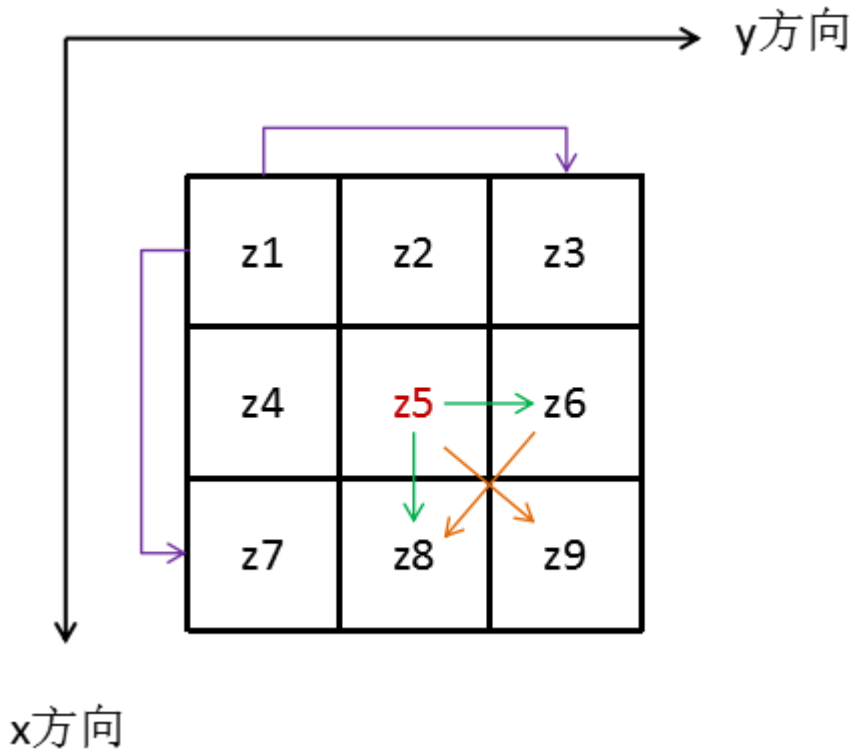
- Sobel算子梯度大小

$$g(x,y)=\sqrt{f_x^2+f_y^2} \quad \text{或者} \quad g(x,y)=|f_x|+|f_y|$$

$$\text{或者} \quad g(x,y)=\max(|f_x|,|f_y|)$$

边缘检测——Sobel算子

- 算法实现



Sobel梯度算子：

gx = (z7+2*z8+z9)-(z1+2*z2+z3)
gy = (z3+2*z6+z9)-(z1+2*z4+z7)

Sobel梯度:

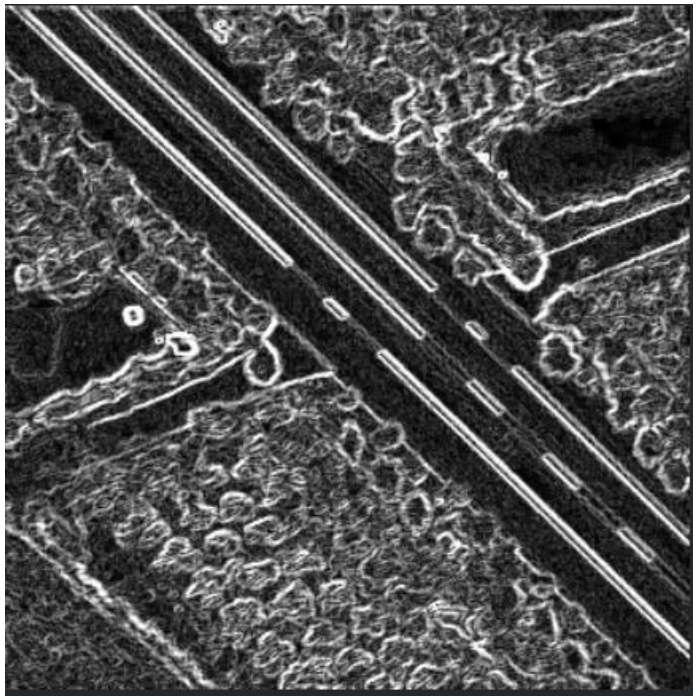
$$g(x,y) = |f_x| + |f_y|$$

边缘检测——Sobel算子

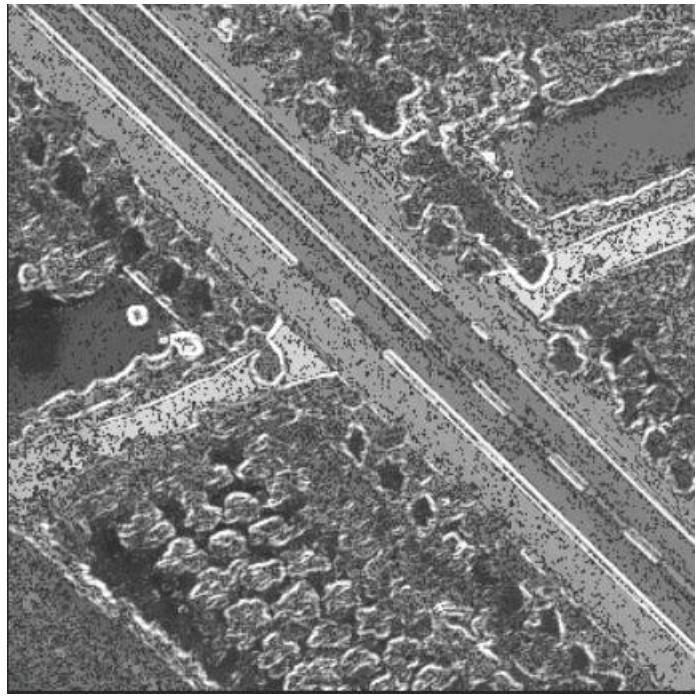
```
def sobel(image,threshold = 10):
    h,w = image.shape
    image_pad = np.pad(image,((1,1),(1,1)),"edge").astype(np.int32)
    out = np.zeros((h,w))
    for i in range(h):
        for j in range(w):
            fx = image_pad[i+2][j]+2*image_pad[i+2][j+1]+image_pad[i+2][j+2]-image_pad[i][j]-2*image_pad[i][j+1]-image_pad[i][j+2]
            fy = image_pad[i][j+2]+2*image_pad[i+1][j+2]+image_pad[i+2][j+2]-image_pad[i][j]-2*image_pad[i+1][j]-image_pad[i+2][j]
            g = abs(fx)+abs(fy)
            if g >= threshold:
                out[i][j] = g
            else:
                out[i][j] = image[i][j]
    out = np.clip(out,0,255)
    return np.uint8(out)
```

边缘检测——Sobel算子

灰度影像原图



Sobel算子



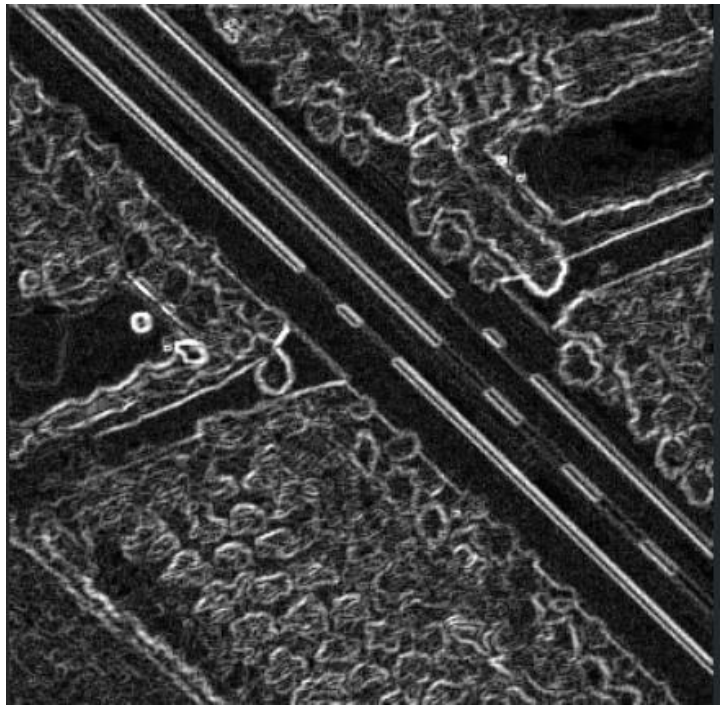
阈值为50

边缘检测——Prewitt算子

```python
def prewitt(image,threshold = 10):
    h,w = image.shape
    image_pad = np.pad(image,((1,1),(1,1)),"edge").astype(np.int32)
    out = np.zeros((h,w))
    for i in range(h):
        for j in range(w):
            fx = image_pad[i+2][j]+image_pad[i+2][j+1]+image_pad[i+2][j+2]-image_pad[i][j]-image_pad[i][j+1]-image_pad[i][j+2]
            fy = image_pad[i][j+2]+image_pad[i+1][j+2]+image_pad[i+2][j+2]-image_pad[i][j]-image_pad[i+1][j]-image_pad[i+2][j]
            g = abs(fx)+abs(fy)
            if g >= threshold:
                out[i][j] = g
            else:
                out[i][j] = image[i][j]
    out = np.clip(out,0,255)
    return np.uint8(out)
```

边缘检测——Prewitt算子

灰度影像原图



Prewitt算子

阈值为50

边缘检测——Laplace算子

- Laplace算子模板

$$\nabla^2 f(x,y)=\frac{\partial^2 f(x,y)}{\partial x^2}+\frac{\partial^2 f(x,y)}{\partial y^2} \quad (1)$$

为了适合数字图像处理，通常将其简化为：

标准模板

0	1	0
1	-4	1
0	1	0

$$\nabla^2 f(i,j)=\left|4f(i,j)-f(i+1,j)-f(i-1,j)-f(i,j+1)-f(i,j-1)\right| \quad (2)$$

or

$$\nabla^2 f(i,j)=\left|8f(i,j)-f(i+1,j)-f(i-1,j)-f(i,j+1)-f(i,j-1)-f(i-1,j-1)-f(i-1,j+1)-f(i+1,j-1)-f(i+1,j+1)\right|$$

扩展模板

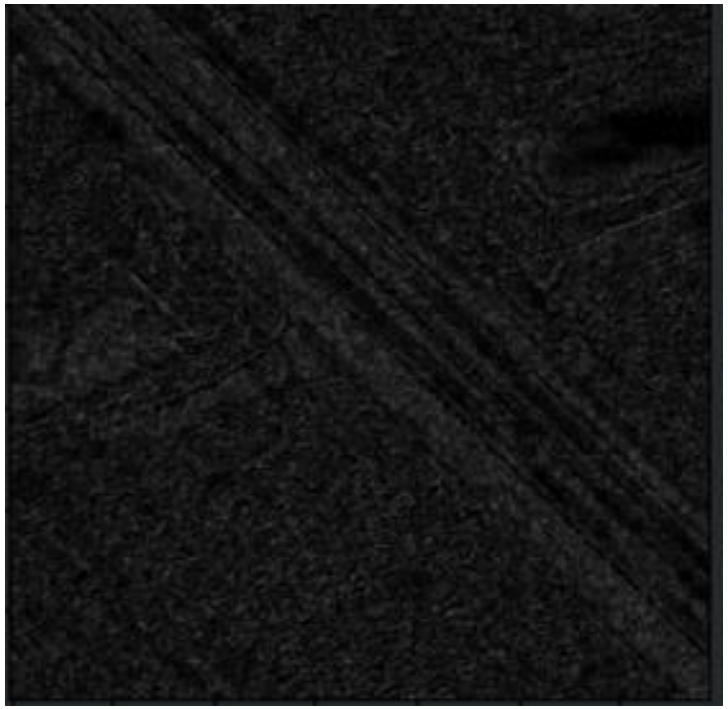
1	1	1
1	-8	1
1	1	1

0	-1	0
-1	4	-1
0	-1	0

-1	-1	-1
-1	8	-1
-1	-1	-1

边缘检测——Laplace算子

灰度影像原图



Laplace算子

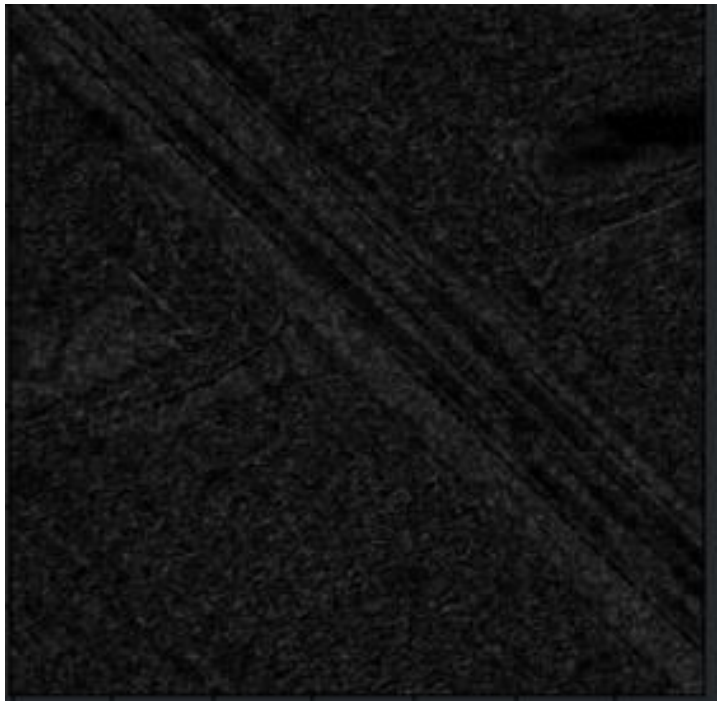
边缘检测——Gauss-Laplace算子

- 将高斯滤波和Laplace算子结合在一起，先用Gauss算子对图像进行平滑，然后采用Laplace算子检测边缘
- 目的：在一定程度上减小噪声的影响
- 高斯滤波器：

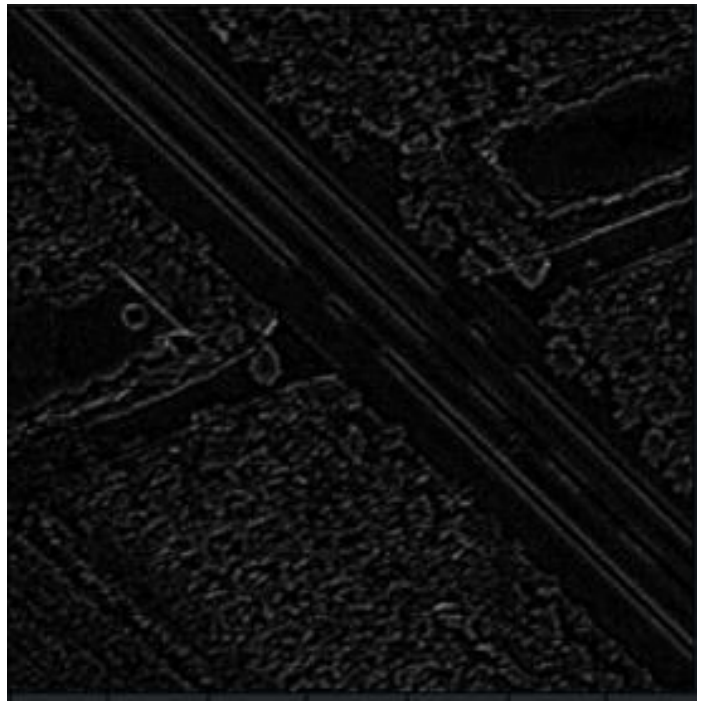
$$H(u,v) = e^{-D^2(u,v)/2D_0^2}$$

边缘检测——Gauss-Laplace算子

灰度影像原图



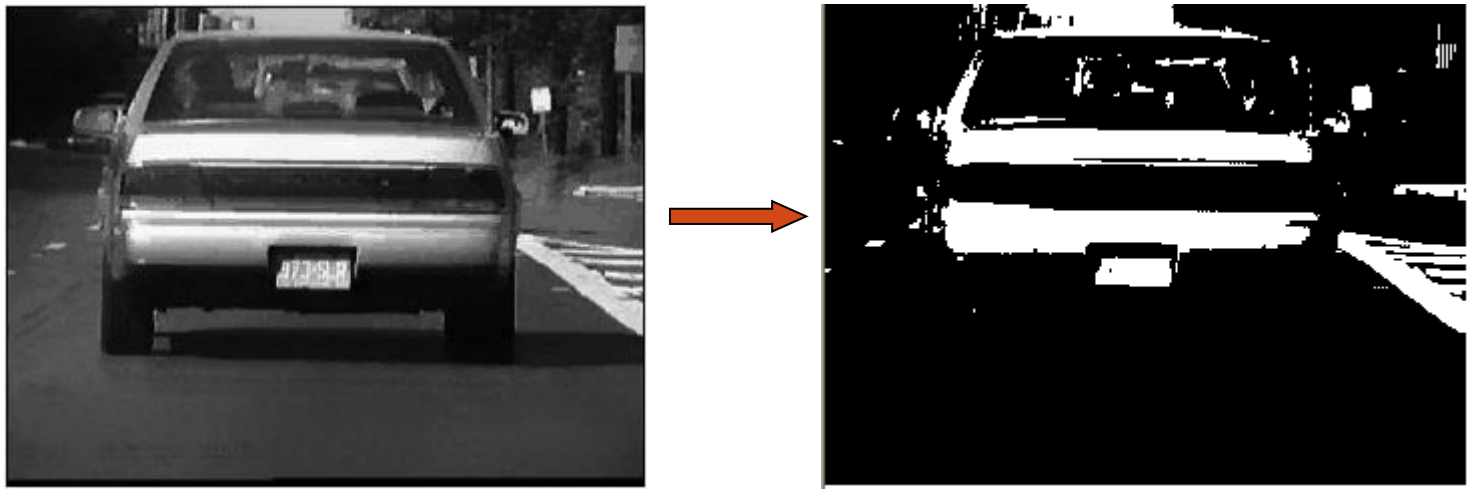
Laplace算子



Gauss-Laplace算子

图像二值化

- 二值化就是将原来的灰度图像转换成只有黑和白两种颜色的图像。

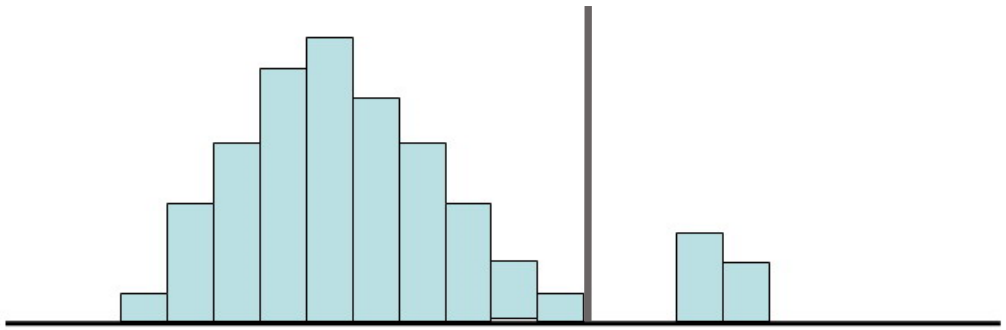


图像二值化

- 利用灰度图像直方图阈值二值化
- 灰度级切片法二值化
- 等灰度片法二值化

图像二值化

- 利用灰度图像直方图阈值二值化
  - 对于大多数灰度图像来说，图像中的物体和背景是有明显的区别的，我们通过选择阈值，区分图像和背景，以便对物体进行处理。
  - 设定一个阈值，若像素的颜色值大于阈值则取255，否则就取0。
  - 最简单的情况——双峰直方图

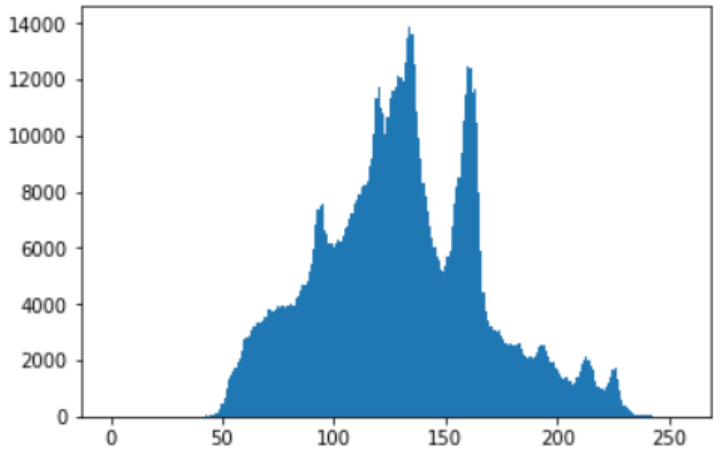


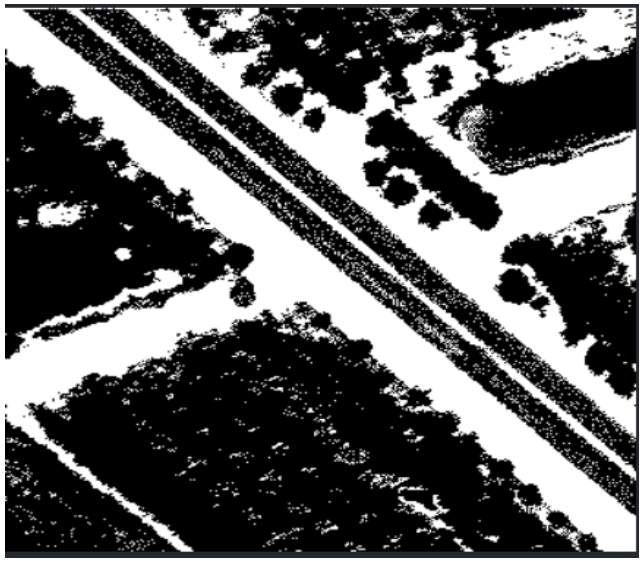
图像二值化

- 利用灰度图像直方图阈值二值化

```
def binary_image(image,threshold = 100):
    h,w = image.shape
    out = np.zeros((h,w))
    for i in range(h):
        for j in range(w):
            if image[i][j] >= threshold:
                out[i][j] = 255
    return np.uint8(out)
```

```
out = binary_image(image_np,threshold=140)
Image.fromarray(out)
```

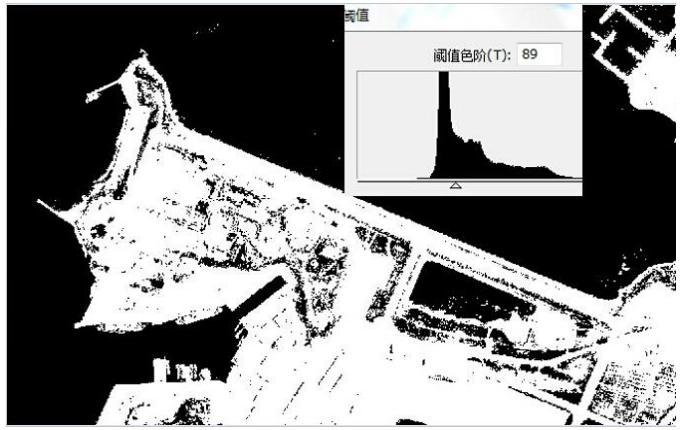




图像二值化

原图

直方图阈值

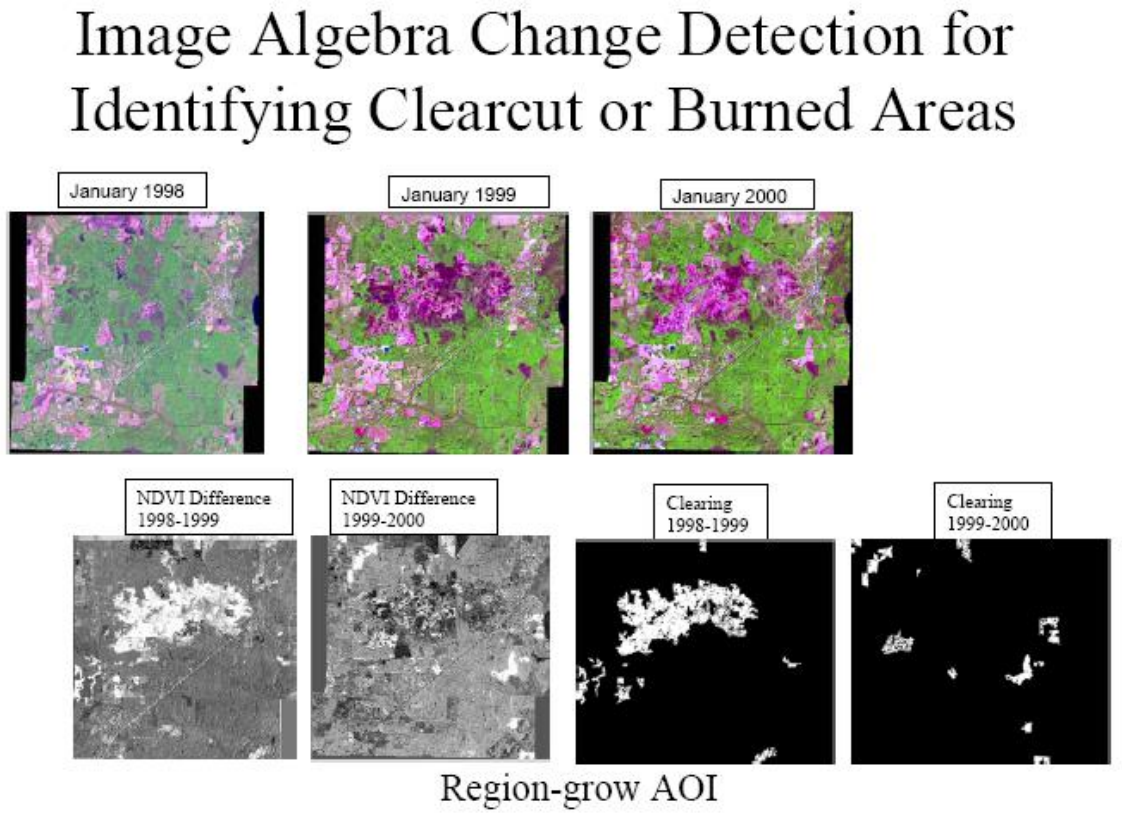


均值阈值

中位数阈值

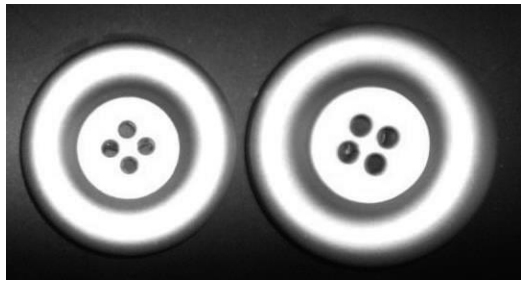
图像二值化

- 图像阈值二值化遥感方面应用
  - 变化检测
    - —不同时相图像作差
    - —对差图像取阈值进行二值化处理
      - 变化与无变化两类



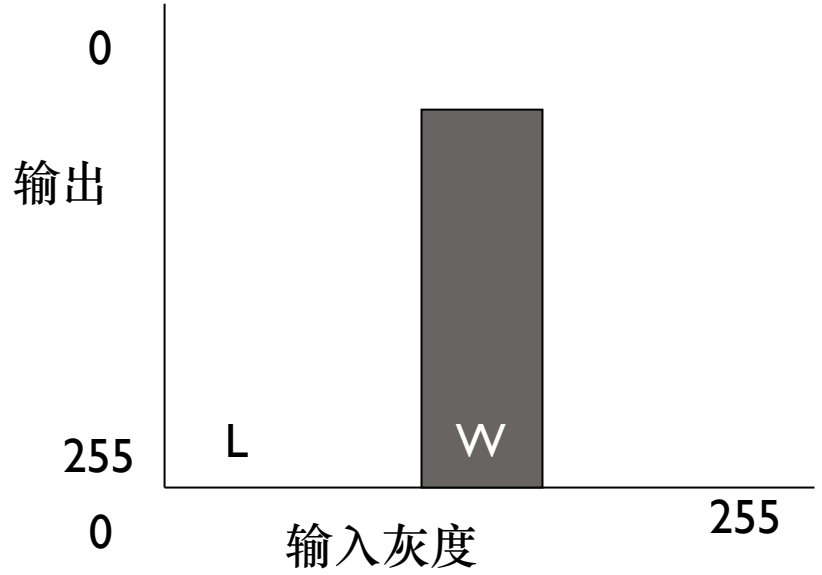
图像二值化

- 图像阈值二值化工业应用
  - 零件质量检测
  - 指纹识别
  - 车牌号自动提取



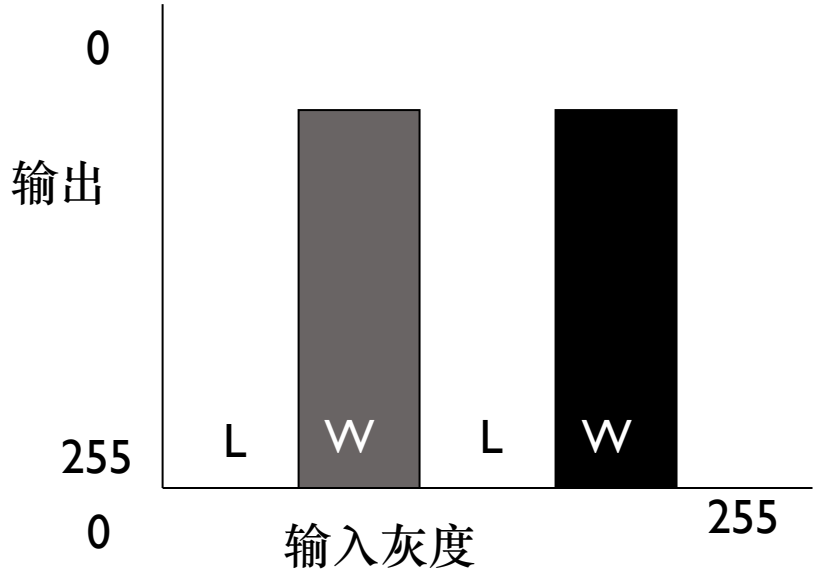
图像二值化

- 灰度级切片法二值化
  - 如下图所示，将输入图像的某一灰度级范围内的所有像素全部置为0（黑），其余灰度级的所有像素全部置为255（白），则生成黑白二值图像。



图像二值化

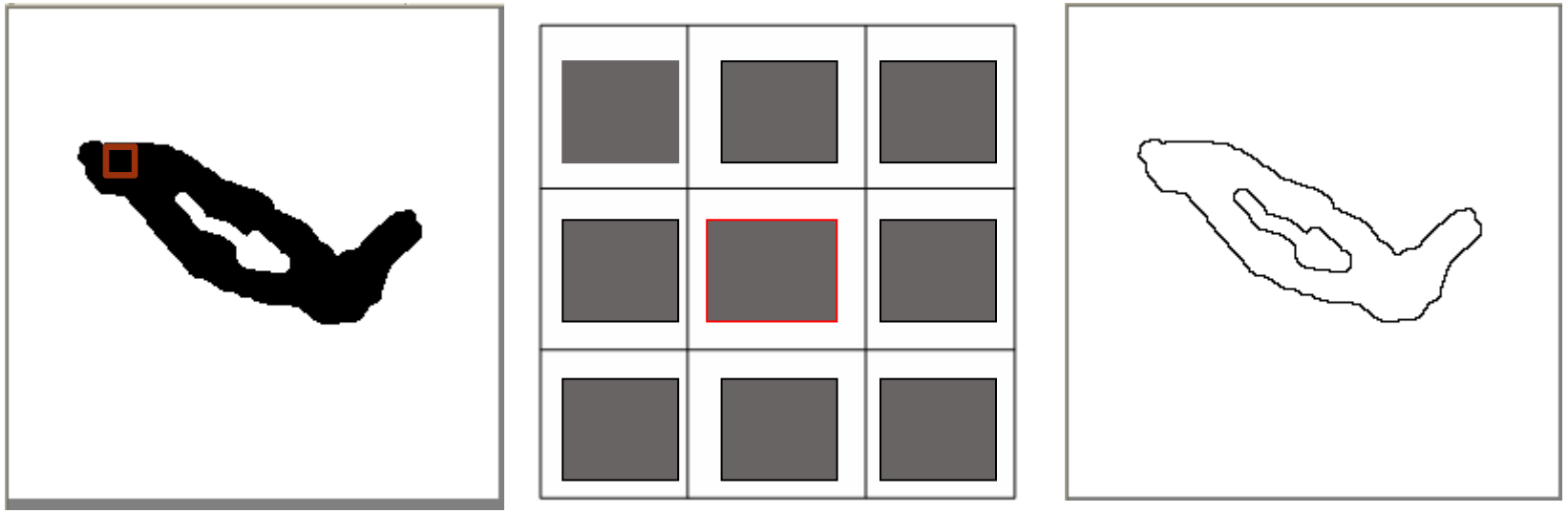
- 等灰度片法二值化
  - 将输入图像在某两个等宽的灰度级范围内的所有像素全部置为0（黑），其余灰度级的所有像素全部置为255（白），则生成黑白二值图像。



二值图像轮廓提取

- 原理

二值图像边界提取算法就是掏空内部点：如果原图中有一点为黑，且它的8个相邻点都是黑色时，则将该点删除。



二值图像轮廓提取

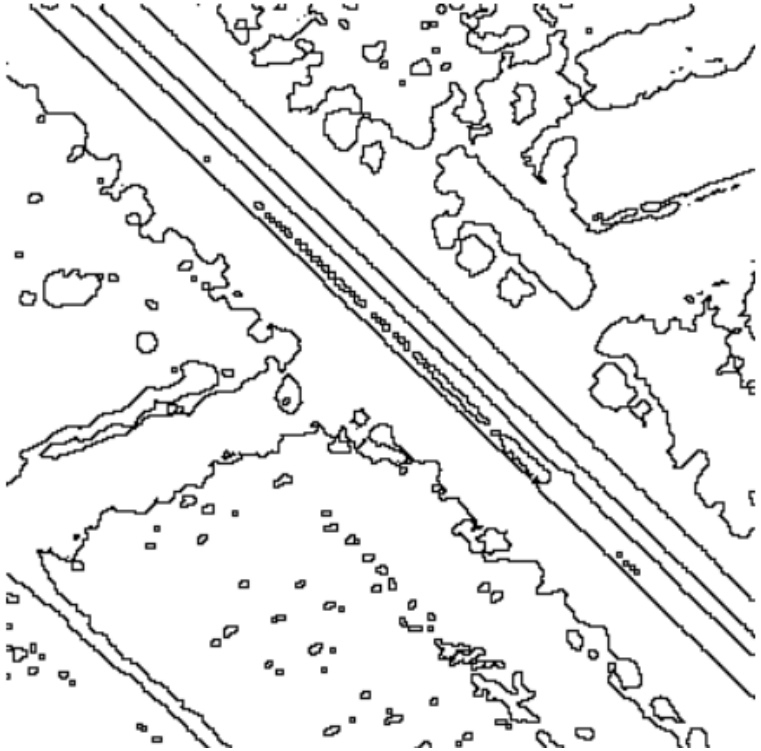
- 算法实现

```
def contour_extraction(image):
    h, w = image.shape
    image_pad = np.pad(image, ((1, 1), (1, 1)), "edge")
    image_pad = np.int32(image_pad)
    out = image.copy()
    for i in range(h):
        for j in range(w):
            if image_pad[i+1][j+1] == 0 and (image_pad[i][j]+image_pad[i][j+1]+image_pad[i][j+2]+image_pad[i+1][j]+image_pad[i+1][j+2]+image_pad[i+2][j]+image_pad[i+2][j+1]+image_pad[i+2][j+2] == 0):
                out[i][j] = 255
    return np.uint8(out)
```

二值图像轮廓提取

- 提取效果

二值化图像
（高斯模糊）



轮廓提取结果

二值图像轮廓跟踪

- 外边界跟踪

➢先根据某种严格的“探测准则”找出目标物体轮廓上的像素，给程序一个初始位置；

探测准则：

首先，找到最左下方的边界点，以这个边界点为起始，按照从左到右，从下到上的顺序搜索，找到的第一个不同灰度的点就是最左下方的边界点，以此点为轮廓跟踪初始点。

二值图像轮廓跟踪

- 外边界跟踪

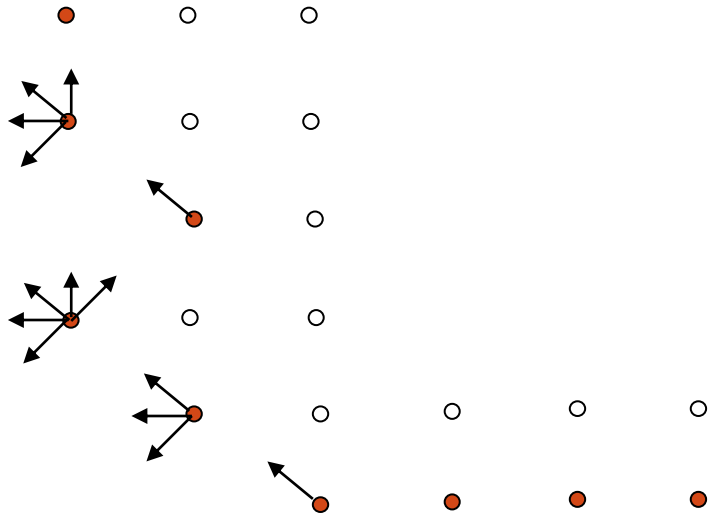
➢再根据这些像素的某些特征，用一定的“跟踪准则”找出轮廓上的其他像素。

跟踪准则：

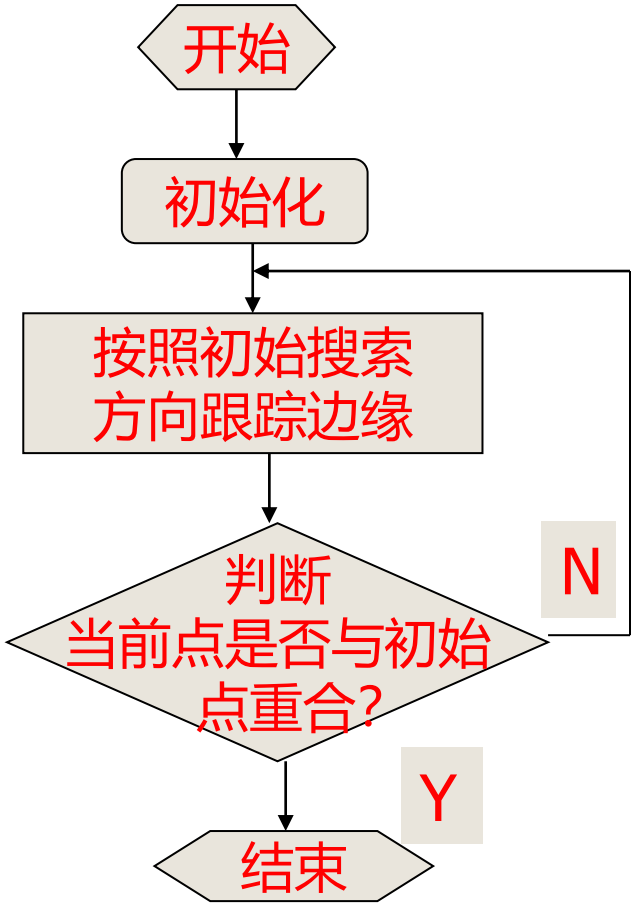
从第一个边界点开始，定义初始的搜索方向为沿左上方；如果上方的点是黑点，则为边界点，否则搜索方向为顺时针旋转45度。这样一直到寻找到第一个黑点为止。然后把这个黑点作为新的边界点，在当前的搜索方向的基础上逆时针旋转90度，继续用同样的方法搜索下一个黑点，直到返回最初的边界点为止。

二值图像轮廓跟踪

- 外边界跟踪



8邻域外边界跟踪过程



作业

- 随机选取一张灰度图，先进行灰度直方图均衡，再随机选取一个算子进行边缘检测；
- 实现二值图像轮廓跟踪算法(只需要考虑一幅图像中只有一个封闭对象的简单情况)；
- 完成实验报告，需要保留中间处理结果图。