数字图像处理

第八讲

边缘检测与细化

主要内容

- 边缘检测
- 细化

边缘检测

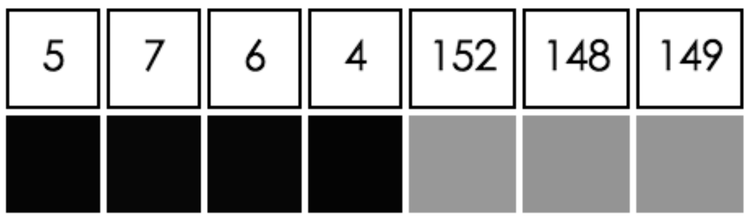
- 边缘检测是图像处理和计算机视觉中的基本问题
  - 目的：标识数字图像中亮度变化明显的点。它存在于目标与背景、目标与目标、区域与区域之间。
  - 图像分割、图像压缩、特征提取等方面都把边缘检测作为基本的工具

边缘检测

- 定义：<u>边缘是指图像中灰度发生急剧变化的区域</u>。
- 边缘与边界的区别
  - 边界是边缘
  - 边缘不一定是边界
  - 二值图像时，边界=边缘

边缘检测

- 如何反映：图像灰度的变化可以用图像的梯度反映。
- 边缘检测并不容易
  - 图1

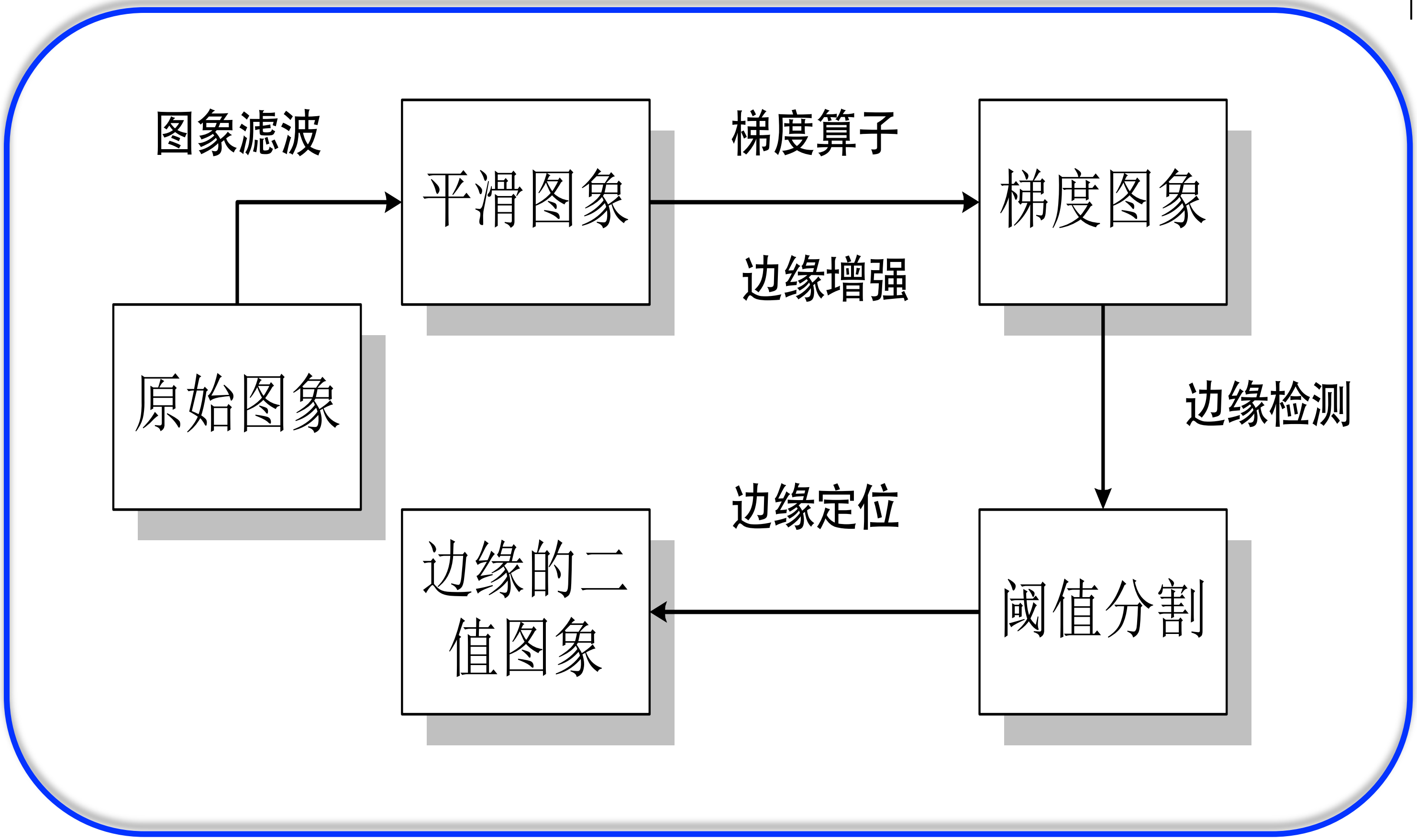


边缘比较明显

  - 图2

边缘变得没有那么明显

边缘检测的基本步骤



边缘检测的基本步骤

- 边缘检测算法的基本步骤
  - （1）滤波。滤波器的目标是降低噪声的影响。但有可能会导致边缘强度的损失。
  - （2）增强。增强算法将邻域中灰度有显著变化的点突出显示。一般通过计算梯度幅值完成。
  - （3）检测。**检测算法的目标是检测出真边缘。最简单的边缘检测是梯度幅值阈值判定。**
  - （4）定位。精确确定边缘的位置。

边缘检测

- 原理：通过梯度的局部最大值来确定边缘
- 做法：求连续图像f(x,y)梯度的局部最大值及其方向。

$$f(x,y)\text{沿}r\text{的梯度}$$

$$\frac{\partial f}{\partial r}=\frac{\partial f}{\partial x}\frac{\partial x}{\partial r}+\frac{\partial f}{\partial y}\frac{\partial y}{\partial r}=f_x\cos\theta+f_y\sin\theta$$

$$\text{使}\frac{\partial f}{\partial r}\text{最大的条件是}\frac{\partial\frac{\partial f}{\partial r}}{\partial\theta}=0$$

边缘检测

- 计算梯度最大值及其方向

$$f_x \sin\theta_\varphi - f_y \cos\theta_\varphi = 0$$

$$\theta_\varphi = \tan^{-1}\left(f_y \big/ f_x \right) \text{或} \theta_\varphi + \pi$$

$$\text{梯度最大值} \varphi = \sqrt{f_x^{\,2} + f_y^{\,2}}$$

- 怎么方便计算呢? **算子**

边缘检测中常用的梯度算子

- Roberts 算子
- Prewitt 算子
- Sobel 算子
- Laplician 算子
- Marr(或LoG) 算子
- Canny 算子

算法从简单到复杂

功能从单薄到完善

Roberts算子

- 描述：是一种利用局部差分算子寻找边缘的算子，它在2×2邻域上计算对角导数

$$g(i,j)=\sqrt{\left(f(i,j)-f(i+1,j+1)\right)^2+\left(f(i+1,j)-f(i,j+1)\right)^2}$$

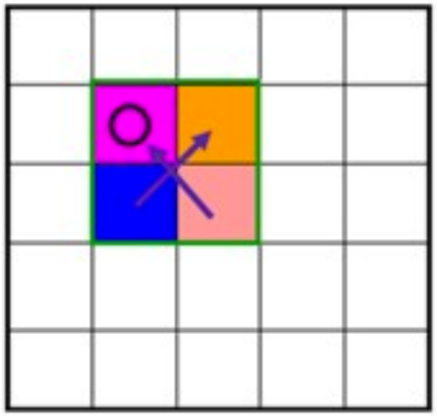
- 简化计算，用梯度的绝对值近似

$$g(i,j)=|f(i,j)-f(i+1,j+1)|+|f(i+1,j)-f(i,j+1)|$$

- 如果 $g(i,j)\geq\theta$ ，则为边缘；否则不为边缘

Roberts算子的计算

$$g(i,j)=|f(i+1,j+1)-f(i,j)|+|f(i+1,j)-f(i,j+1)|$$



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

对应于空间域滤波的模板计算

示例

Roberts算子效果

Lenna的Roberts边缘

Roberts算子点评

- 特点
  - 对于边界陡峭且噪声比较小的图像检测效果比较好
  - 对噪声比较敏感

Prewitt算子

- 思想：采用3x3领域

$$f_i' = f(i-1,j+1) + f(i,j+1) + f(i+1,j+1) \\ -f(i-1,j-1) - f(i,j-1) - f(i+1,j-1)$$

$$f_j' = f(i+1,j-1) + f(i+1,j) + f(i+1,j+1) \\ -f(i-1,j-1) - f(i-1,j) - f(i-1,j+1)$$

- 如果$|f_i'| + |f_j'| \geq \theta$, 则为边缘；否则不为边缘

Prewitt算子的计算

$$g(i,j)=\{d_x^2(i,j)+d_y^2(i,j)\}^{\frac{1}{2}}$$

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

示例

Prewitt算子效果

Lenna的Prewitt边缘

Prewitt算子点评

- 特点：
  - 考虑更大的领域，对噪声有抑制作用
  - 较Roberts算子减少了对噪声的影响

Sobel算子

- 动机：不同近邻对梯度的贡献应该有所不同
- 做法：采用一种**加权**的方式

$$g(i,j)=\{d_x^2(i,j)+d_y^2(i,j)\}^{\frac{1}{2}}$$

$$d_x=\begin{bmatrix}-1 & 0 & 1\\ \boxed{-2} & 0 & \boxed{2}\\ -1 & 0 & 1\end{bmatrix}\quad d_y=\begin{bmatrix}-1 & \boxed{-2} & -1\\ 0 & 0 & 0\\ 1 & \boxed{2} & 1\end{bmatrix}$$

其它变种

- Isotropic Sobel算子

$$H_1 = \begin{bmatrix} -1 & 0 & 1 \\ -\sqrt{2} & 0 & \sqrt{2} \\ -1 & 0 & 1 \end{bmatrix} \quad H_2 = \begin{bmatrix} -1 & -\sqrt{2} & -1 \\ 0 & 0 & 0 \\ 1 & \sqrt{2} & 1 \end{bmatrix}$$

- Sobel算子由于它更为精细，比Roberts, Prewitt算子更为常用

示例

算子效果比较

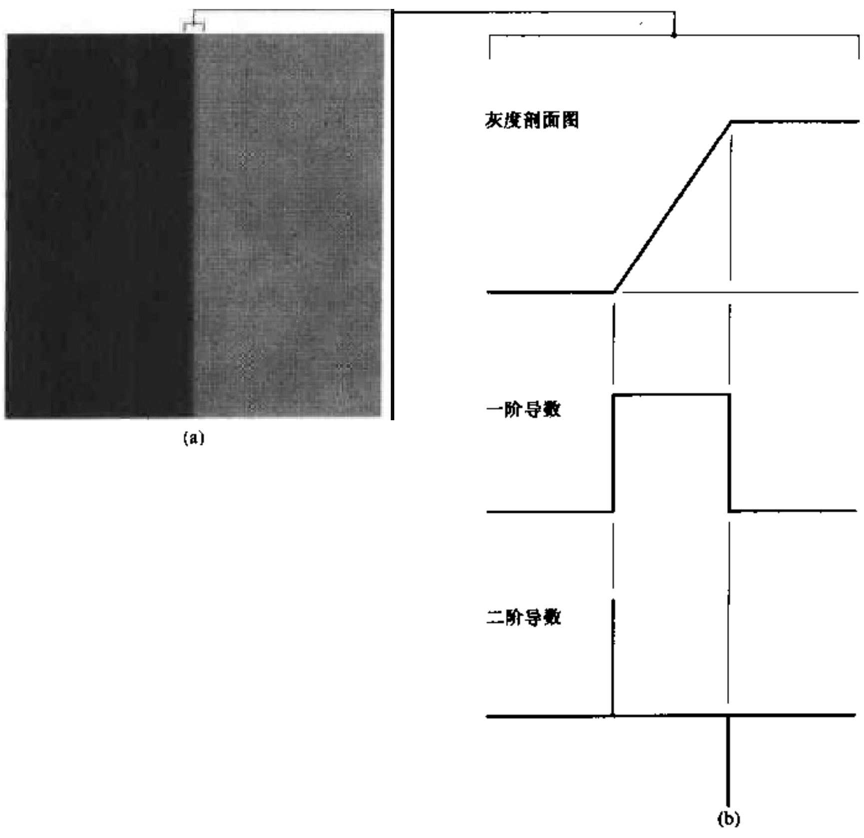
Lenna的 Roberts边缘

Lenna的 Prewitt边缘

Lenna的 Sobel边缘

二阶算子

- Roberts, Prewitt以及Sobel算子考虑一阶梯度信息检测边缘，我们也可以用二阶梯度来检测边缘



邻域灰度直方图

一阶梯度

二阶算子可以判断“亮”与“暗”，从而可以通过过零点，确定边缘的起点和终点。

二阶过零点，二阶梯度大的两头确定边缘的起点和终点

一阶算子与二阶算子

- 二阶算子和一阶算子一样，都是通过阈值来判断边缘，高于阈值为边缘
- 但阈值的选取方式不同。一阶算子的阈值一般跟图片灰度级相关，二阶算子的阈值一般定在0附近
- 二阶算子得到边缘点数少，一阶算子得到边缘点数多

一阶算子与二阶算子

- 二阶算子计算复杂度更高，一阶算子计算复杂度更低
- 二阶算子更善于发现剧烈变化的边缘
- 二阶算子对噪声非常敏感，一般不独立使用。要配合降噪方法一起使用

Laplician算子

- 优势：是不依赖边缘方向的二阶微分算子，<u>具有旋转不变性即各向同性的性质</u>

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

$$\therefore \nabla^2 f = 4f(i,j) - f(i+1,j) - f(i-1,j) - f(i,j+1) - f(i,j-1)$$

$$H_1 = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Laplician算子模板

- 标准模板及其它的一些变种

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

有时希望邻域中心点具有更大的权值

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix} \text{或}$$

$$\begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix}$$

Laplician算子与一阶算子的比较

- 考虑四种基本结构（孤立点、端点、直线、以及阶跃）

$$
\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1^* & 0 \\ 0 & 0 & 0 \end{bmatrix}
\quad
\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1^* & 0 \\ 0 & 1 & 0 \end{bmatrix}
\quad
\begin{bmatrix} 0 & \vdots & 0 \\ & 1 & \\ 0 & 1^* & 0 \\ 0 & 1 & 0 \\ & \vdots & \end{bmatrix}
\quad
\begin{bmatrix} 0 & \vdots & \vdots & \cdots \\ & 1 & 1 & \cdots \\ 0 & 1^* & 1 & \cdots \\ 0 & 1 & 1 & \cdots \\ & \vdots & \vdots & \cdots \end{bmatrix}
$$

- 分别采用一阶算子和Laplician算子

结果

一阶差分梯度图象 $G(x,y)=\max\left(\left|\Delta_x f(x,y)\right|,\left|\Delta_y f(x,y)\right|\right)$ 向左和向下计算

$$\begin{bmatrix} & 1 & \\ & 1^* & 1 \\ & & \end{bmatrix} \begin{bmatrix} & 1 & \\ & 1^* & 1 \\ & 1 & 1 \end{bmatrix} \begin{bmatrix} & 1 & 1 \\ & 1^* & 1 \\ & 1 & 1 \end{bmatrix} \begin{bmatrix} & 1 & \\ & 1 & \\ & 1 & \end{bmatrix}$$

拉普拉斯图

$$\begin{bmatrix} 0 & +1 & 0 \\ +1 & -4^* & +1 \\ 0 & +1 & 0 \end{bmatrix} \quad \begin{bmatrix} 0 & 1 & 0 \\ 1 & -3^* & 1 \\ 1 & -2 & 1 \end{bmatrix} \quad \begin{bmatrix} 1 & -2 & 1 \\ 1 & -2^* & 1 \\ 1 & -2 & 1 \end{bmatrix} \quad \begin{bmatrix} 1 & -1 & 0 \\ 1 & -1^* & 0 \\ 1 & -1 & 0 \end{bmatrix}$$

a图 b图 c图 d图

Laplician算子点评

- A图中对孤立的点，输出的是一个扩大略带模糊的点和线。B图和C图中对线的端点和线，输出的是加粗了的端点和线。D中对阶跃线，输出的只有一条线。
- 对梯度运算，梯度算子的灰度保持不变。而对拉氏算子，孤立点增加4倍，端点增加3倍，线增加2倍，界线不变。
- 拉氏算子在实际应用中对噪声敏感。因此在实际中通常不直接使用，怎么办?
- 降噪

Laplician算子模板

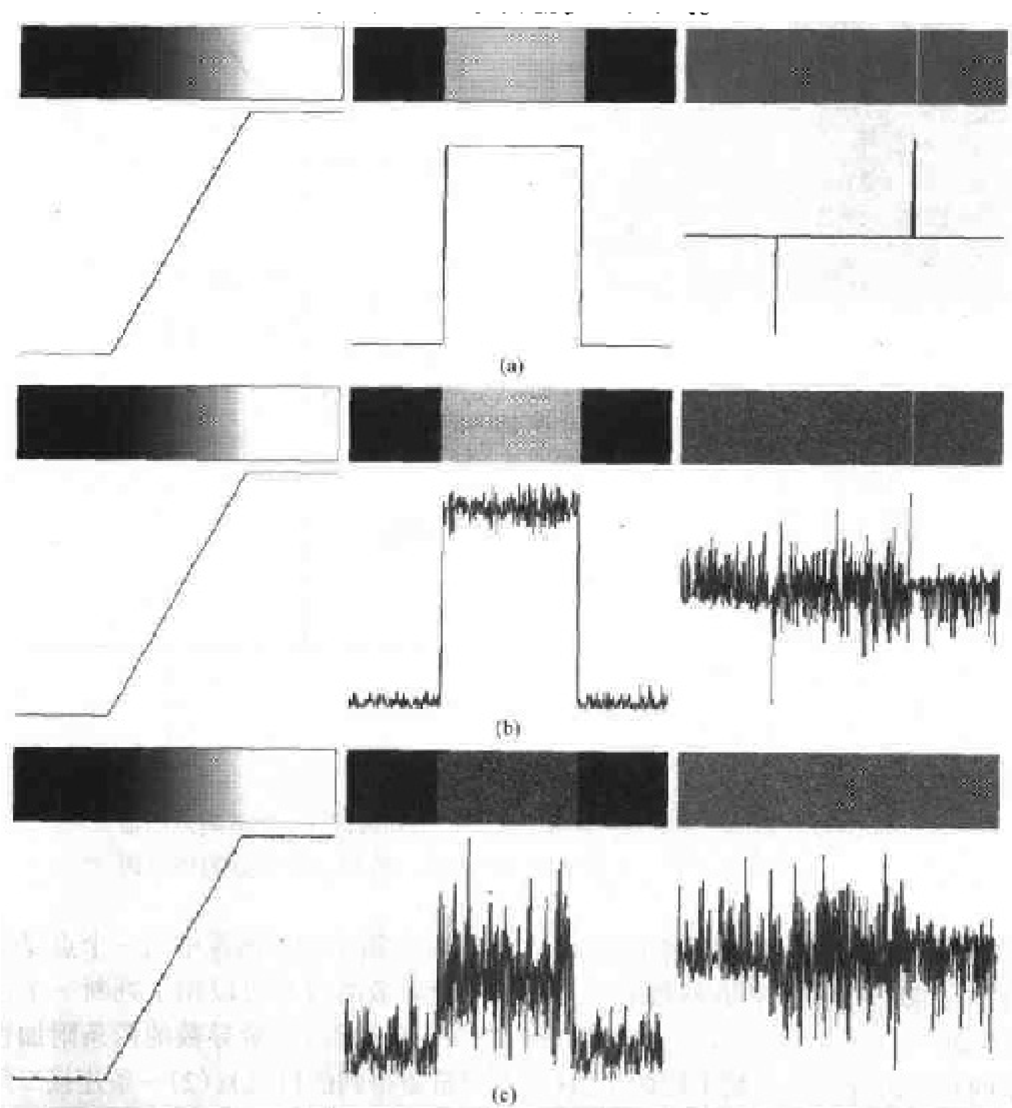
代码，但是直接基于这个代码并不奏效，详细看下图

```
lenna = imread('lenna.pgm');
%-------------------------------------------------------------------
lenna_3=mat2gray(lenna);   %图像矩阵的归一化
[m,n]=size(lenna_3);
lenna_4=lenna_3;       %保留图像的边缘一个像素
L=0;
t=0.2;          %设定阈值
%Laplace算子
for j=2:m-1
    for k=2:n-1
        L=abs(4*lenna_3(j,k)-lenna_3(j-1,k)-lenna_3(j+1,k)-lenna_3(j,k
+1)-lenna_3(j,k-1));
        if(L > t)
            lenna_4(j,k)=255;  %白
        else
            lenna_4(j,k)=0;    %黑
        end
    end
end
figure;
imshow(lenna_4,[]);title('Laplacian 0.2')
```

Laplician算子模板

拉普拉斯算子对噪声非常敏感，一般不单独使用

二阶算子对噪声敏感



Marr算子

- Marr算子，也称为Laplacian of Guassian或LOG算子
- 题外话
  - 计算机视觉会议论文最高奖Marr奖
- 思想：考虑将高斯滤波和Laplacian算子结合在一起进行边缘检测

Marr算子的步骤

- 第一步：对图像进行平滑高斯滤波

$$G(x,y)=\frac{1}{2\pi\sigma^2}\exp\left(-\frac{1}{2\pi\sigma^2}(x^2+y^2)\right)$$

$$g(x,y)=f(x,y)*G(x,y)$$

- 第二步：对平滑后的图像采用Laplacian算子
- 第三步：通过零交叉点判断边缘
- 第四步：采用线性插值的方法估计边缘的位置

选择高斯滤波的原因

- 一个原因是高斯滤波可以平滑噪声，减少Laplician算子对噪声的影响

- 另外一个原因是高斯滤波函数很平滑，任意阶可导，可以配合Laplician算子使用。

$$h(x)=\frac{1}{\sqrt{2\pi\sigma}}e^{-\frac{x^2}{2\sigma^2}},h'(x)=\frac{-x}{\sqrt{2\pi\sigma^3}}e^{-\frac{x^2}{2\sigma^2}},h''(x)=\frac{1}{\sqrt{2\pi\sigma}}e^{-\frac{x^2}{2\sigma^2}\left(\frac{x^2}{\sigma^2}-1\right)}$$

Marr算子

- 联合形式

$$h(x,y)=\frac{1}{2\pi\sigma^2}e^{-\frac{x^2+y^2}{2\sigma^2}}$$

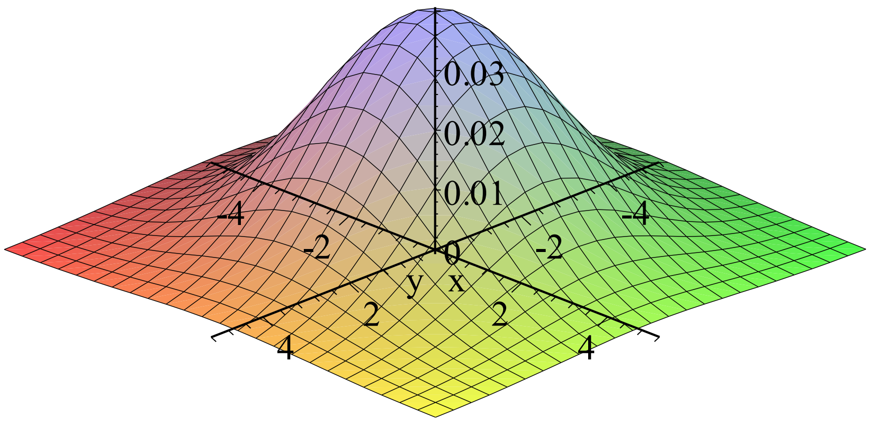
$$LoG=\nabla^2 h(x,y)=\frac{\partial^2 h}{\partial x^2}+\frac{\partial^2 h}{\partial y^2}$$

$$=\frac{1}{\pi\sigma^4}\left(\frac{x^2+y^2}{2\sigma^2}-1\right)e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Marr算子的计算

- 离散拉普拉斯高斯模板（5*5，delta=2）

$$\begin{vmatrix} 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & -2 & -1 & 0 \\ -1 & -2 & 16 & -2 & -1 \\ 0 & -1 & -2 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 \end{vmatrix}$$



Marr算子的其它变种

- 高斯滤波可以进一步推广为DoG滤波

$$DOG = \frac{1}{2\pi\sigma_1^{\ 2}} e^{-\frac{x^2+y^2}{2\sigma_1^{\ 2}}} - \frac{1}{2\pi\sigma_2^{\ 2}} e^{-\frac{x^2+y^2}{2\sigma_2^{\ 2}}}$$

在实际应用中，取$\sigma_1 / \sigma_2 = 1.6$，此时

$$DOG \rightarrow LOG$$

DoG:
Difference
of Gaussian

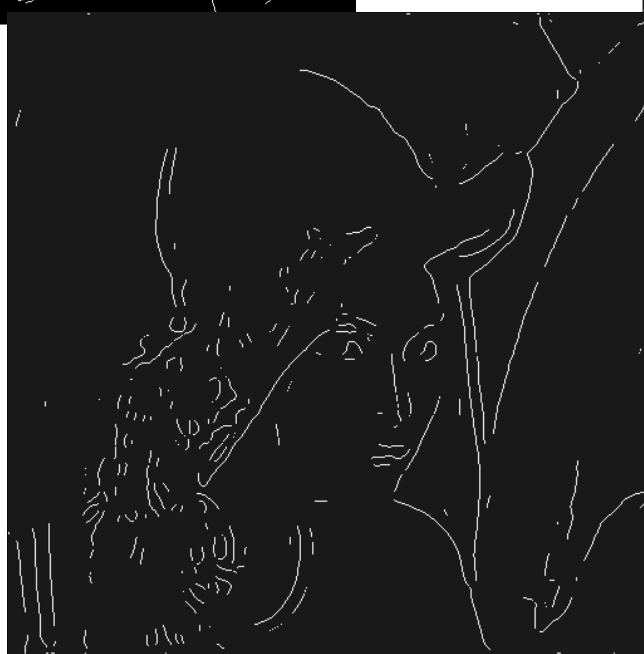
- 更加精细

Marr算子点评

- 过零点（Zero-crossing）的检测与参数delta有关，但边缘位置与delta的选择无关。Delta的选择是个问题
  - 若只关心全局性的边缘可以选取比较大的邻域(如delta= 4 时，邻域接近40个像素宽)来获取明显的边缘。
- 可能会因为过度平滑形状，丢失一些边缘；

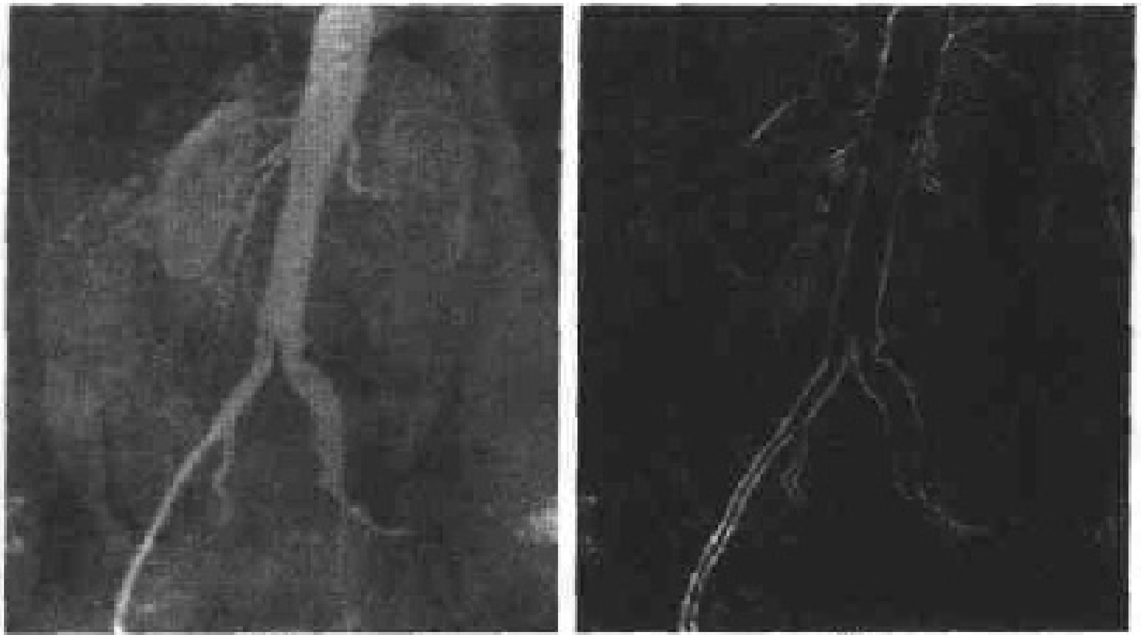
Marr算子效果

Marr边缘
Delta=2



Marr边缘
Delta=4

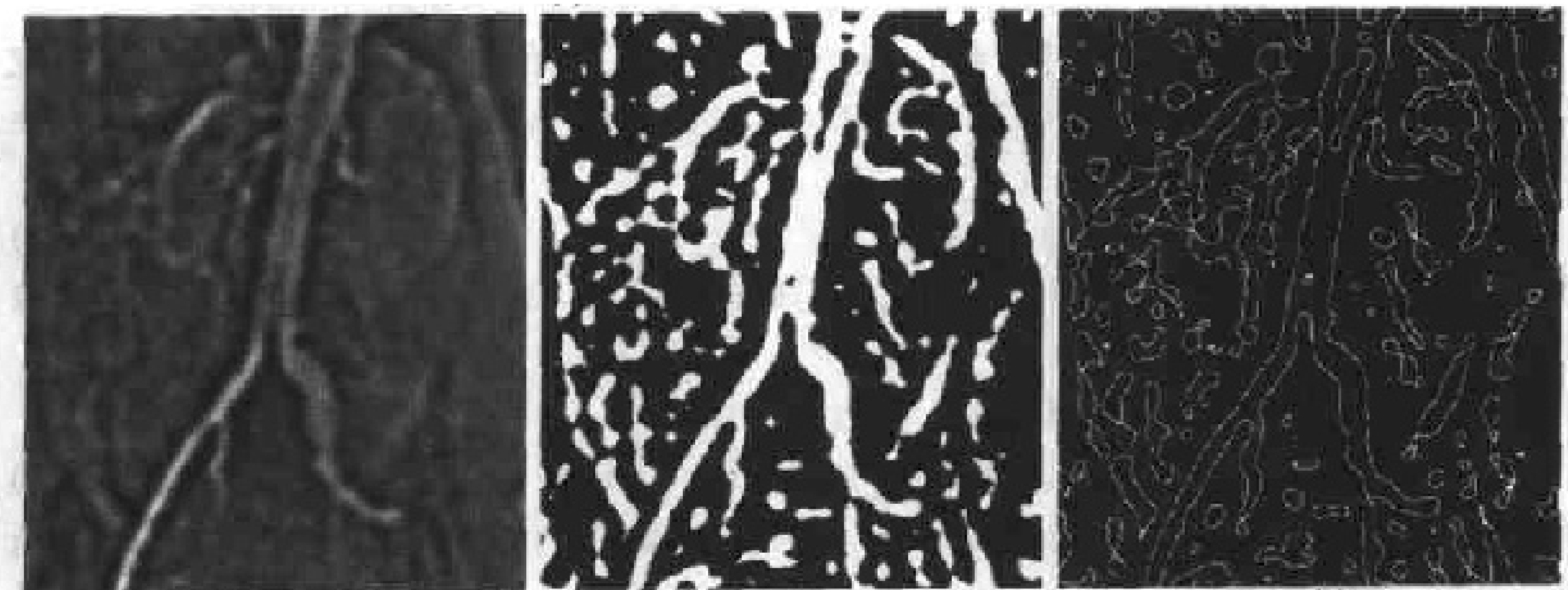
一阶算子与二阶算子



(a) (b)

一阶Sobel算子

一阶算子与二阶算子


(e) (f) (g)

二阶Laplcian算子

集大成者：Canny算子

- 最优的阶梯型边缘检测算法
- 原理
  - 图像边缘检测必须满足两个条件：一能有效地抑制噪声；二必须尽量精确确定边缘的位置。
  - 根据对信噪比与定位乘积进行测度，得到最优化逼近算子。这就是Canny边缘检测算子。
  - 类似与Marr（LoG）边缘检测方法，也属于先平滑后求导数的方法。

Canny算子的基本步骤

- step1:用高斯滤波器平滑图像；
- step2:用一阶偏导的有限差分来计算梯度的幅值和方向；
- step3:对梯度幅值进行非极大值抑制；
- step4:用双阈值算法检测和连接边缘。

Canny算子

- Step1：用高斯滤波器平滑图像

$$H(x,y)=e^{-\frac{x^2+y^2}{2\sigma^2}}$$

$$G(x,y)=f(x,y)*H(x,y)$$

Canny算子

- step2:一阶差分卷积模板:

$$H_1 = \begin{vmatrix} -1 & -1 \\ 1 & 1 \end{vmatrix} \qquad H_2 = \begin{vmatrix} 1 & -1 \\ 1 & -1 \end{vmatrix}$$

$$\varphi_1(m,n) = f(m,n) * H_1(m,n)$$

$$\varphi_2(m,n) = f(m,n) * H_2(m,n)$$

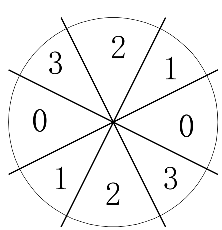
$$\varphi(m,n) = \sqrt{\varphi_1^{\,2}(m,n) + \varphi_2^{\,2}(m,n)}$$

$$\theta_\varphi(m,n) = \tan^{-1}\frac{\varphi_2(m,n)}{\varphi_1(m,n)}$$

Canny算子

- Step3:非极大值抑制
  - 仅仅得到全局的梯度并不足以确定边缘，因此为确定边缘，必须保留局部梯度最大的点，而抑制非极大值。（non-maxima suppression, NMS）
  - 解决方法：利用梯度的方向。

$$\xi[i,j]=\mathbf{Sector}(\theta[i,j])$$



1	2	3
8		4
7	6	5

Canny算子

- Step3（续）
  - 四个扇区的标号为0到3，对应3*3邻域的四种可能组合。
  - 在每一点上，邻域的中心像素M与沿着梯度线的两个像素相比。如果M的梯度值不比沿梯度线的两个相邻像素梯度值大，则令M=0。
  - 即：

$$N[i,j] = NMS(M[i,j], \xi[i,j])$$

Canny算子

- Step4:阈值化
  - 减少假边缘段数量的典型方法是对N[i，j]使用一个阈值。将低于阈值的所有值赋零值。但问题是如何选取阈值?
  - 解决方法：**双阈值算法**
    - 设置两个阈值a1,a2，通常2.5 a1 = a2
    - a2阈值下假边缘少，但是轮廓有间断；对于轮廓的端点，利用a1阈值的8邻点位置寻找可以连接到轮廓上的边缘。算法不断地收集边缘，知道将轮廓连接起来为止

Canny算子效果

Canny边缘

a=2

Canny算子效果

Canny边缘

a=4

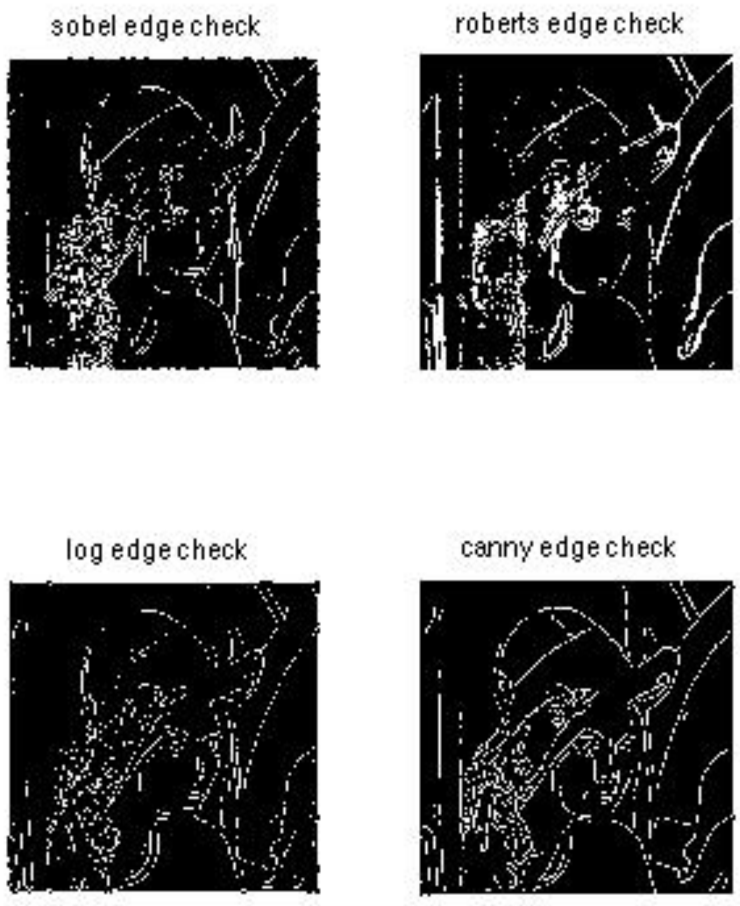
边缘检测算法例子

- function my_edge_detection()
-
- I=imread('lenna.png');% 提取图像
- I=I(:,:,1);
-
- BW1=edge(I,'sobel',0.04); %用SOBEL算子进行边缘检测
- BW2=edge(I,'roberts',0.04);%用Roberts算子进行边缘检测
- BW3=edge(I,'prewitt',0.04); %用prewitt算子进行边缘检测
- BW4=edge(I,'log'); %用log算子进行边缘检测
- BW5=edge(I,‘canny’); %用canny算子进行边缘检测

边缘检测算法例子

- subplot(2,3,1), imshow(BW1);
- title('sobel edge check');
- subplot(2,3,2), imshow(BW2);
- title('roberts edge check');
- subplot(2,3,3), imshow(BW3);
- title('prewitt edge check');
- subplot(2,3,4), imshow(BW4);
- title('log edge check');
- subplot(2,3,5), imshow(BW5);
- title('canny edge check');

边缘检测算法效果图


sobel edge check
roberts edge check
prewitt edge check
log edge check
canny edge check

Canny算法点评

- 实际效果通常要优于其它算子，特别对于有噪声，或者存在假边缘的图像

主要内容

- 边缘检测
- 细化

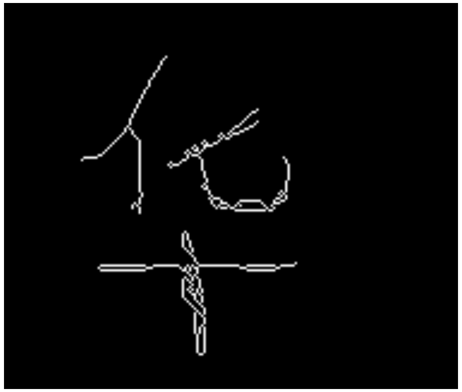
细化

- 1）什么是细化?
- 2）回顾一些基本概念
- 3）细化的要求
- 4）细化算法

细化（thinning）

- 定义
  - 细化是一种二值图像处理运算。可以把二值图像区域缩成线条，以逼近区域的中心线。
  - 细化的目的是减少图像成分，只留下区域最基本的信息，以便进一步分析和处理。
  - 细化一般用于文本分析预处理阶段。

细化示例



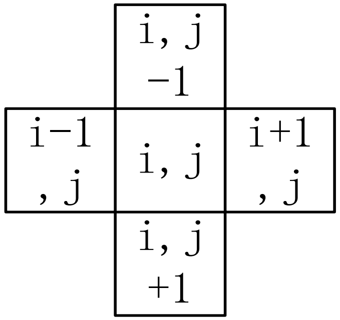
回顾一些基本概念

- （1）近邻
  - 4邻点（4-neighbors）：如果两个像素有公共边界，则称它们互为4邻点。
  - 8邻点（8-neighbors）：如果两个像素至少共享一个顶角，则称它们互为8邻点。
- （2）连通
  - 一个像素与它的4邻点是4连通（4-connected）关系；
  - 一个像素与它的8邻点是8连通（8-connected）关系；

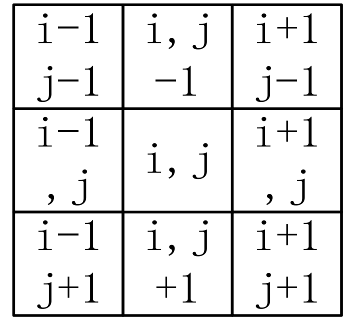
路径、前景背景

- （3）<u>路径</u>
  - 从像素0到像素n的路径是指一个像素序列，0,1,…,k,…,n，其中k与k+1像素互为邻点。
  - 如果邻点关系是4连通的，则是4路径；
  - 如果邻点关系是8连通的，则是8路径；
- （4）<u>前景背景</u>
  - 图像中值为1的全部像素的集合称为前景(foreground)，用S来表示。

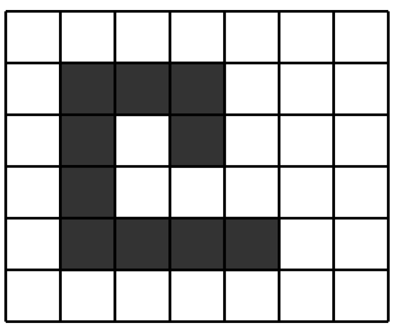
4路径，8路径



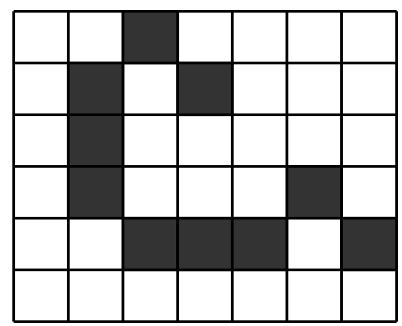
4邻点



8邻点



4路径



8路径

连通性、连通分支、边界

- （5）<u>连通性</u>
  - 已知像素$p,q \in S$，如果存在一条p到q的路径，且路径上全部像素都包含在S中，则称p与q是连通的。
  - 连通性具有：自反性、互换性和传递性。
- （6）<u>连通分支</u>
  - 一个像素集合，如果集合中每一个像素与其他像素连通，则称该集合是连通分支（connected component）。
- （7）<u>简单边界点</u>
  - S中的一个边界点P，如果其邻域中（不包括P点）只有一个连通成分，则P是简单边界点。

边界点

- 判断下图中哪些是简单边界点?

A不是		
0	1	1
0	P	1
1	0	0

B是		
0	1	1
0	P	1
0	1	0

C是		
0	0	1
0	P	1
1	1	0

D是		
0	0	0
0	P	0
0	0	1

E不是		
0	1	1
0	P	0
1	1	0

细化要求

- （1）连通区域必须细化成连通线结构；
- （2）细化结果至少是8连通的；
- （3）保留终止线的位置；
- （4）细化结果应该近似于中轴线；
- （5）由细化引起的附加突刺应该是最小的。

细化算法

- 思想：在至少3x3邻域内检查图像前景中的每一个像素，迭代削去简单边界点，直至区域被细化成一条线。

细化算法

- 算法描述：
  - 对于每一个像素，如果
  - A) 没有上邻点（下邻点、左邻点、右邻点）；
  - B) 不是孤立点或孤立线；
  - C) 去除该像素点不会断开连通区域
  - D) 删除该像素点；
  - E) 重复A-D步骤直到没有像素点可以去除。

具体步骤

- 每次细化分4步（<u>不去除只有一个邻点</u>），具体过程如下：
- (1）八连通下北向边界点（n=0, p=1）可删除条件

$$w\bar{s}e+\bar{w}(nw)+(ne)\bar{e}+\bar{e}(se)\bar{s}+\bar{s}(sw)\bar{w}=0$$

- 上式排除下面5种情况：

nw	n	ne
w	p	e
sw	s	se

	0			1	0				0	1			0				0	
1	P	1		0	P				P	0			P	0		0	P	
	0												0	1		1	0	

- （2）八连通下的南向边界点（s=0，p=1）可删除条件：

$$w\bar{n}e+\bar{w}(sw)+(se)\bar{e}+\bar{e}(ne)\bar{n}+\bar{n}(nw)\bar{w}=0$$

	0												0	1		1	0	
1	P	1		0	P				P	0			P	0		0	P	
	0			1	0				0	1			0				0	

- （3）八连通下的西向边界点（w=0，p=1）可删除条件：

$$n\bar{e}s+\bar{s}(sw)+(nw)\bar{n}+\bar{e}(ne)\bar{n}+\bar{s}(se)\bar{e}=0$$

	1							1	0				0	1				
0	P	0		0	P			0	P			0	P	0		0	P	0
	1			1	0												0	1

- （4）八连通下的东向边界点（e=0, p=1）可删除条件：

$$n\bar{w}s+\bar{s}\left(se\right)+\left(ne\right)\bar{n}+\bar{e}\left(nw\right)\bar{n}+\bar{s}\left(sw\right)\bar{w}=0$$

- 排除了下面5种情况：

	1	
0	P	0
	1	

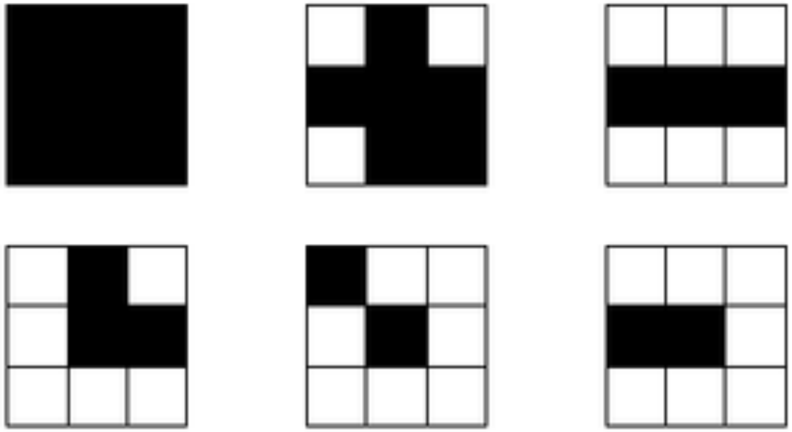
	P	0
	0	1

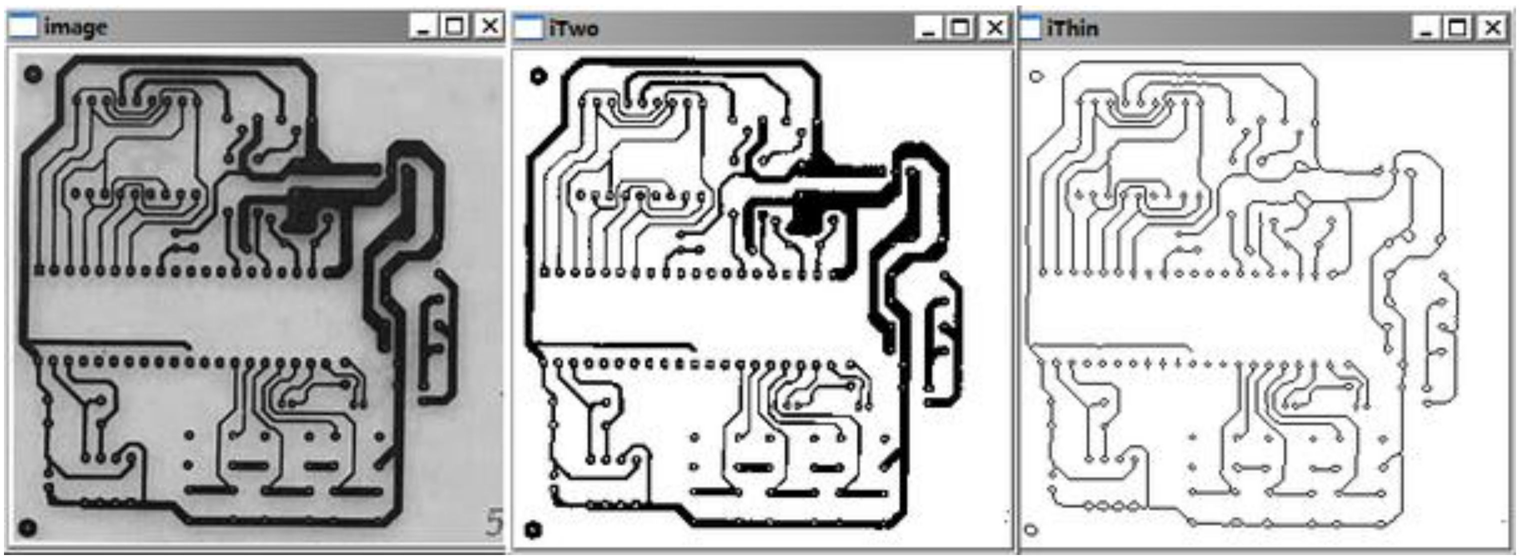
	0	1
	P	0

1	0	
0	P	0

0	P	0
1	0	

细化效果





要点小结

- 边缘检测
  - 边缘检测的基本步骤及其意义
  - 边缘与边界的区别联系
  - 边缘检测的核心在于梯度计算
  - 边缘检测常用的算子及其原理
- 细化
  - 细化的定义，原理及其意义

下一讲