数字图像处理
Digital Image Processing

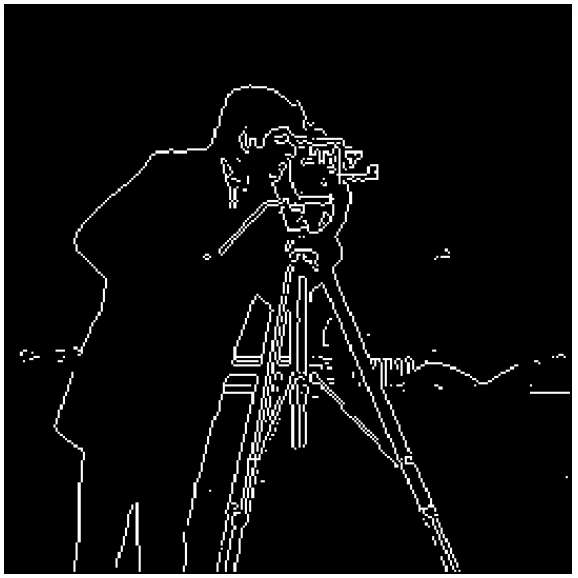
图像边缘提取及处理
Image Edge Detection

图像边缘提取及处理

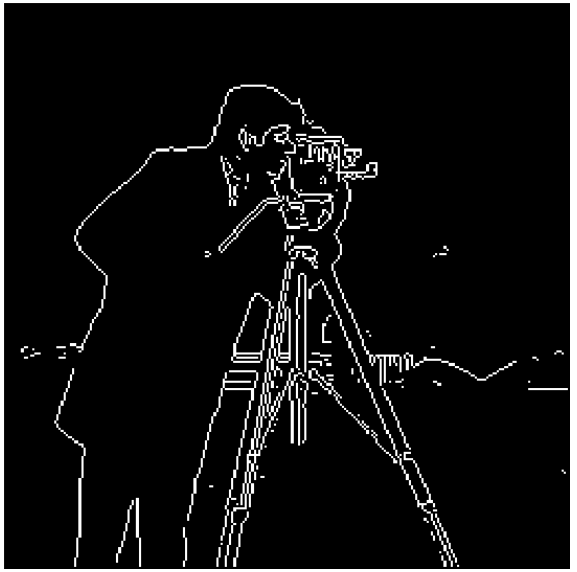
1. 边缘检测
2. Hough变换
3. 边界特征表达及描述

几种常用算子的比较

roberts

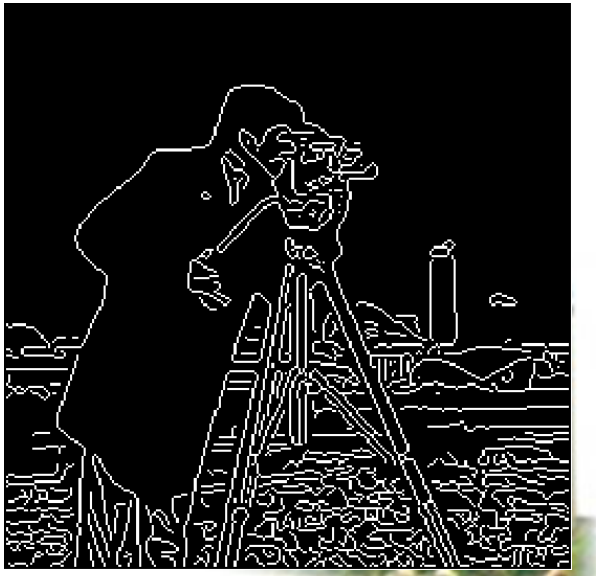


sobel



prewitt

log



canny

图像的边缘

- 图像的边缘对人类的视觉系统具有重要的意义，它是人类判别物体的重要依据，是图像的最基本特征。
- 所谓边缘(又称为边沿)，是指其周围像素灰度有阶跃变化或屋顶变化的那些像素的集合。
- 边缘广泛存在于物体与背景之间、物体与物体之间、基元与基元之间。因此，它是图像分割所依赖的重要特征。

1、不连续性检测

- 点检测
- 线检测
- 边缘检测

不连续性检测 Detection of Discontinuities

基于图像灰度值的不连续性，在图像分割技术中主要有点、线和边缘检测等检测类型。

用于检测图像灰度值不连续的常用方法是空间滤波。

w1	w2	w3
w4	w5	w6
w7	w8	w9

$$R = w_1 z_1 + w_2 z_2 + \cdots + w_9 z_9$$

$$= \sum_{i=1}^{9} w_i z_i$$

模板的响应与其中心相关。

不连续性检测——点检测 Point Detection

对于嵌在常数区域（或图像中亮度基本不变的区域）中的孤立点的检测，可以采用如下模板：

−1	−1	−1
−1	8	−1
−1	−1	−1

若 $|R| \geq T$ 就可在模板的中心位置检测出一个孤立的点。

点检测的MATLAB实现：

```
g=abs(imfilter(double(f),w))>=T
```

不连续性检测——点检测 Point Detection

用空域的高通滤波器来检测孤立点

例：

图像

8	8	8
8	128	8
8	8	8

模板

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

$$R = (-1 * 8 * 8 + 128 * 8) / 9$$

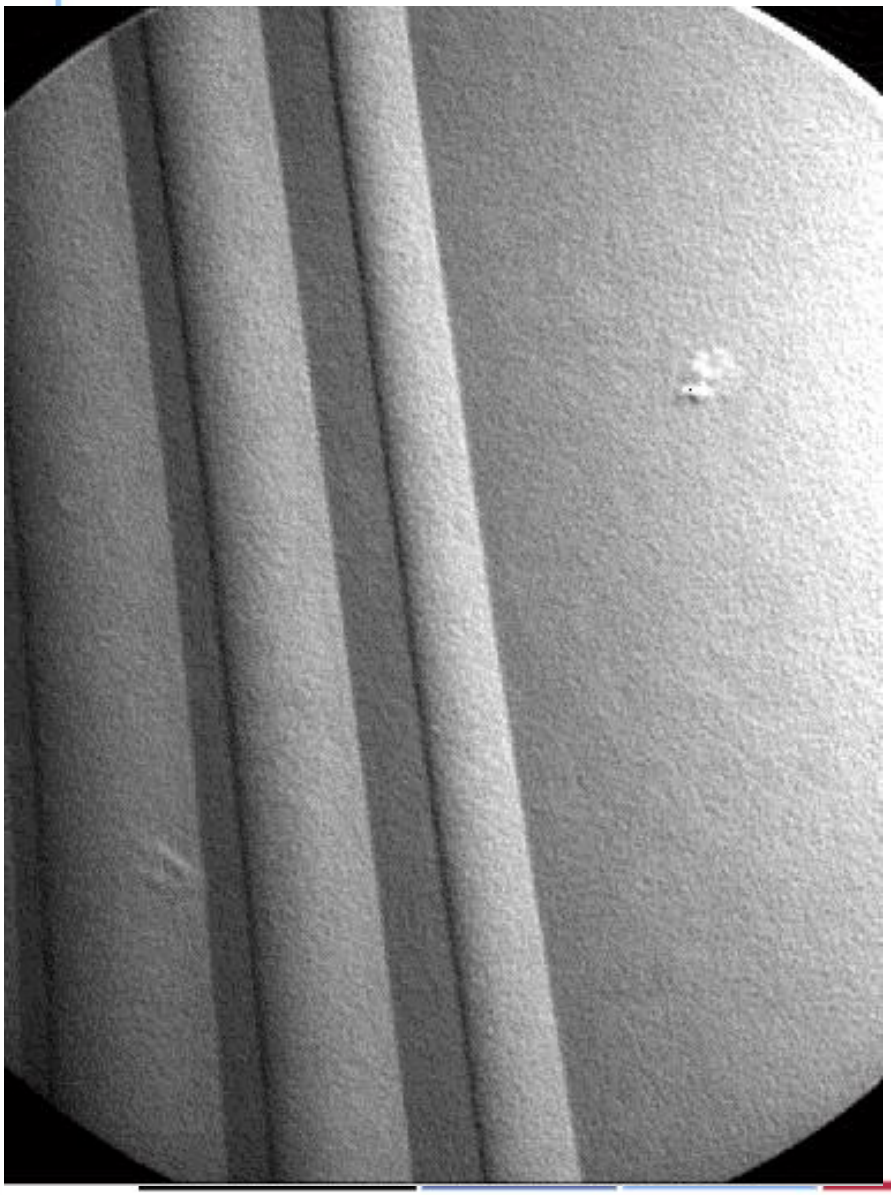
$$= (120 * 8) / 9 = 960 / 9 = 106$$

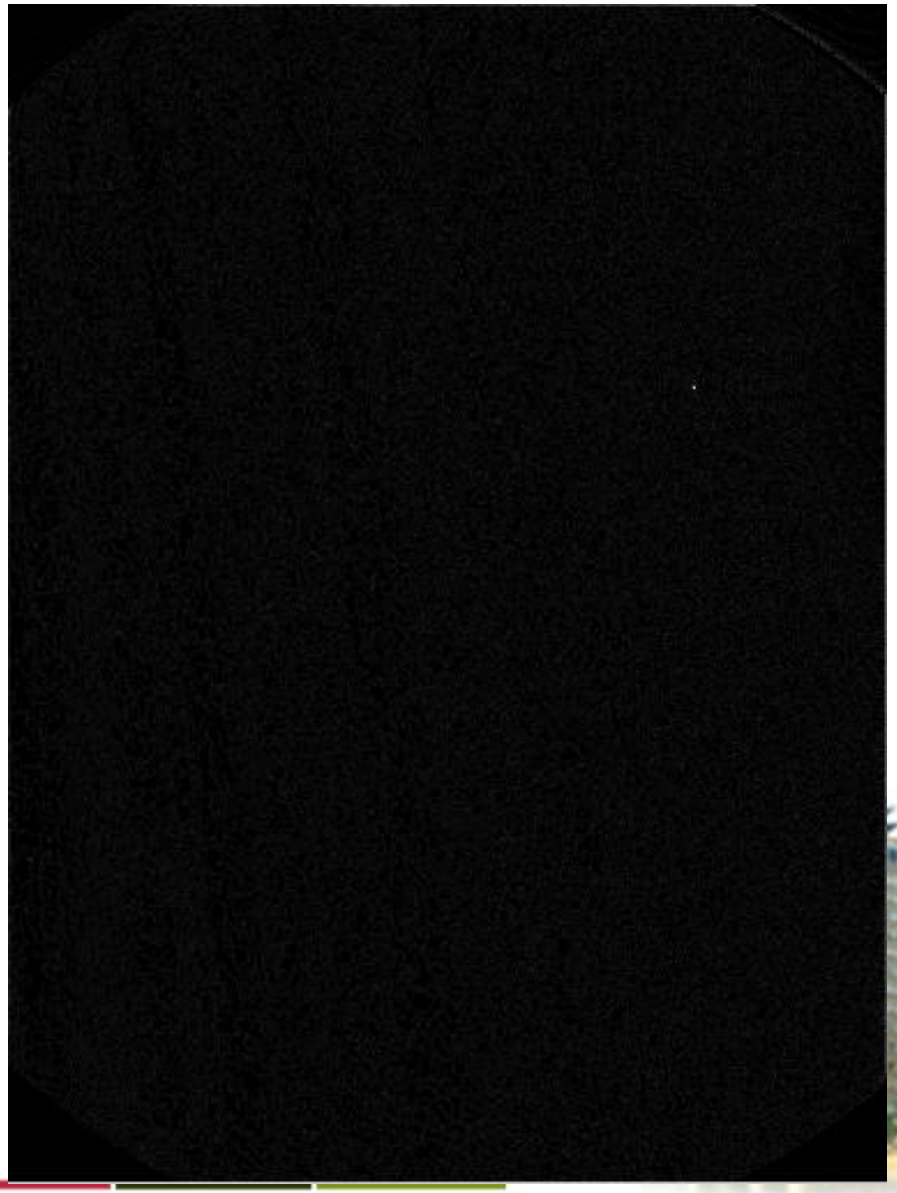
设：阈值：T = 64　　R > T

点检测——算法描述

- 设定阈值 T，如T = 32、64、128等,并计算高通滤波值R；
- 如果R值等于0或接近为0，说明当前检测点的灰度值与周围点的相同或接近；
- 当R的值足够大时，说明该点的值与周围的点非常不同，是孤立点。通过阈值T来判断

$|R| > T$　检测到一个孤立点





不连续性检测——线检测 Line Detection

与点检测相比，线检测要复杂一些。图像分割中主要有四个模板用来检测不同方向的线条。

−1	−1	−1
2	2	2
−1	−1	−1

水平

−1	2	−1
−1	2	−1
−1	2	−1

垂直

−1	−1	2
−1	2	−1
2	−1	−1

+45°

2	−1	−1
−1	2	−1
−1	−1	2

-45°

线的检测

- 通过比较典型模板的计算值，确定一个点是否在某个方向的线上

-1	-1	-1
2	2	2
-1	-1	-1

水平模板

-1	-1	2
-1	2	-1
2	-1	-1

45度模板

-1	2	-1
-1	2	-1
-1	2	-1

垂直模板

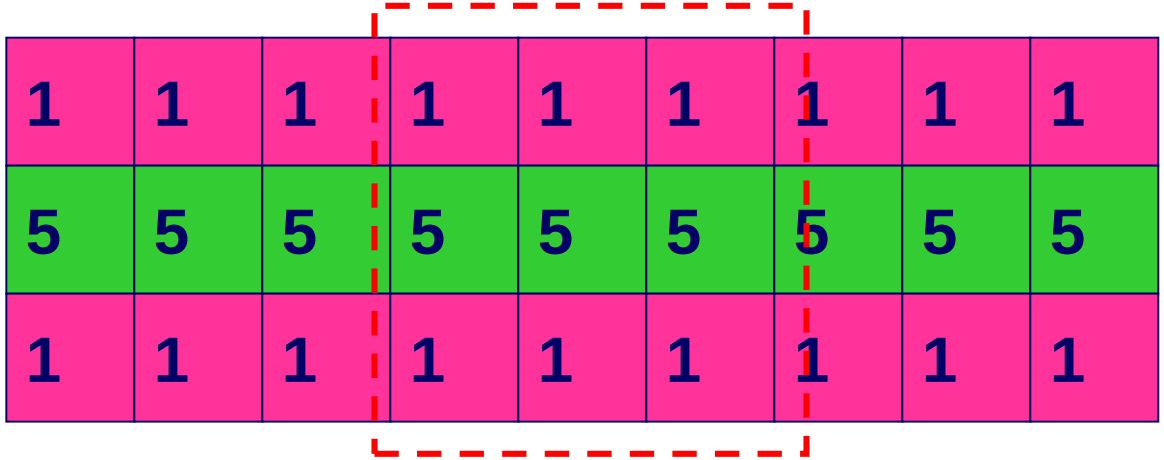
2	-1	-1
-1	2	-1
-1	-1	2

135度模板

线的检测

例：

图像



用4种模板分别计算

$R_{水平} = -6 + 30 = 24$

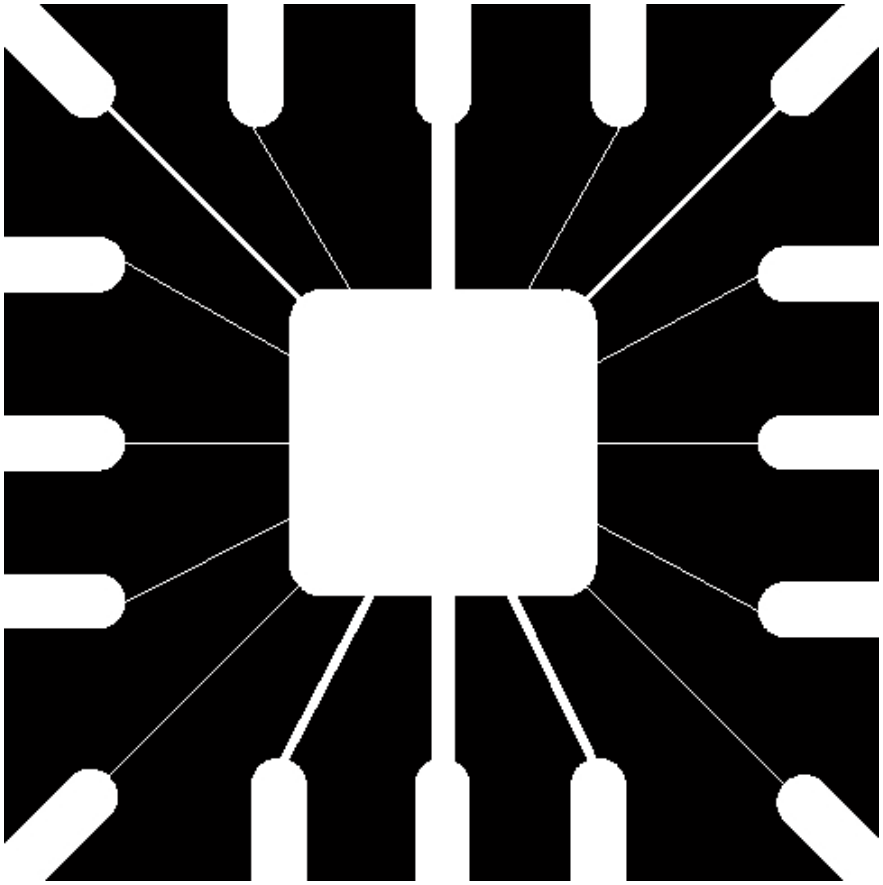
$R_{45度} = -14 + 14 = 0$

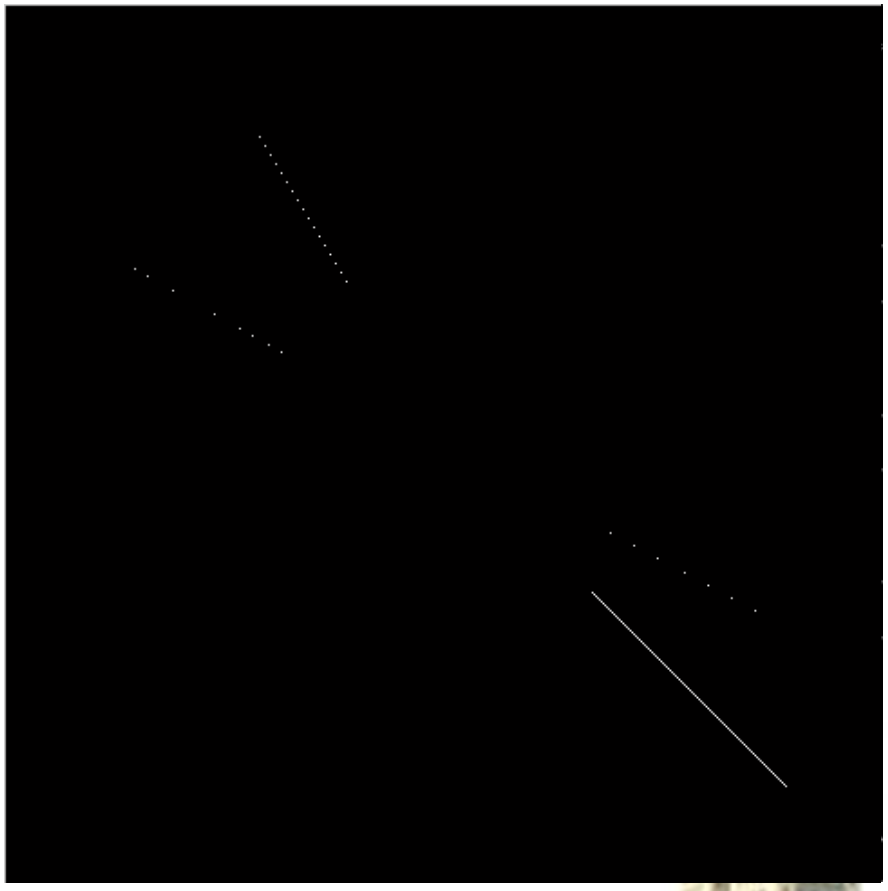
$R_{垂直} = -14 + 14 = 0$

$R_{135度} = -14 + 14 = 0$

- **线的检测——算法描述**
  - 依次计算4个方向的典型检测模板，得到R_i i=1,2,3,4
  - 如 $|R_i| > |R_j|$（对于所有的j不等于i），那么这个点被称为在方向上更接近模板i 所代表的线

- **设计任意方向的检测模板**
  - 可能大于3x3
  - 模板系数和为0
  - 感兴趣的方向的系数大。





不连续性检测——边缘检测 Edge Detection

虽然点检测和线检测在任何关于图像分割的讨论中确实很重要，但到目前为止，边缘检测还是检测图像灰度不连续性最通用的方法。

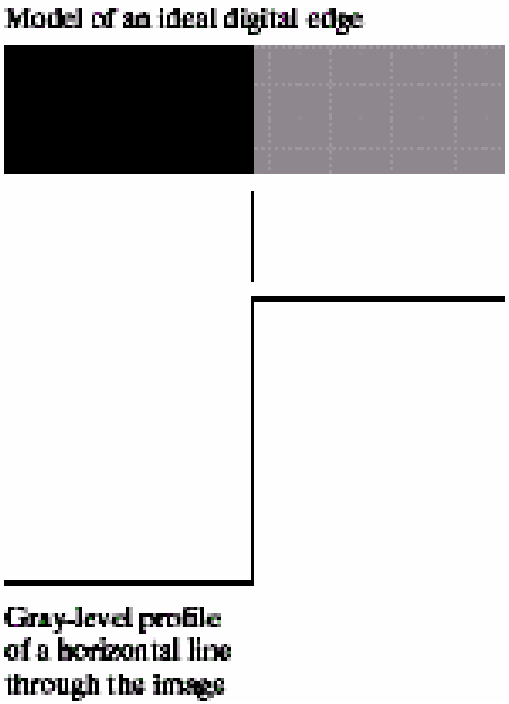
何谓图像边缘？

图像边缘是一组象素的集合，它有方向和幅度两个特性。为了解边缘检测，我们先从直观上对边缘建模开始。

边缘检测

1.理想数字边缘

该模型生成的图像边缘是一组相连的象素集合（垂直方向上），每个象素都处在灰度跃变的一个垂直的台阶上。

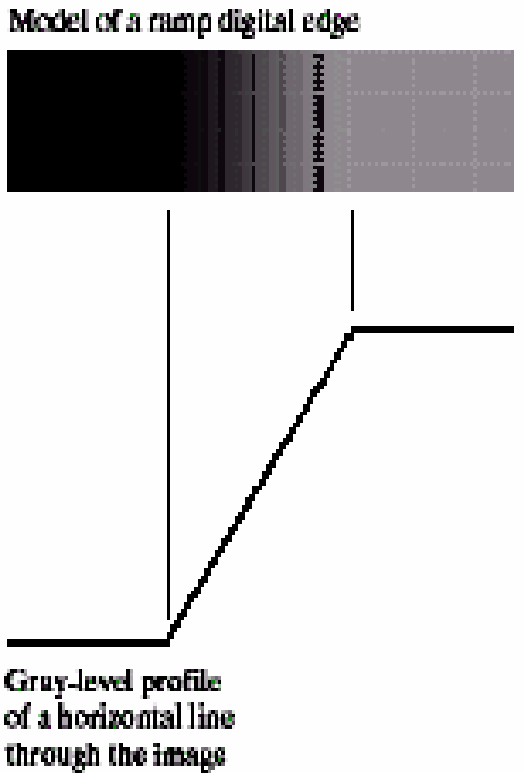


理想数字边缘

边缘检测

2. 斜坡数字边缘

实际上，由于光学系统、取样和其他图像采集的不完善性引起图像的边缘模糊，图像的模糊程度取决于图像采集系统的性能、取样率和获取图像的照明条件等因素。结果，边缘被更精确地模拟成具有“类斜面”的剖面。



斜坡数字边缘

斜坡的斜率与边缘的模糊程度成正比。

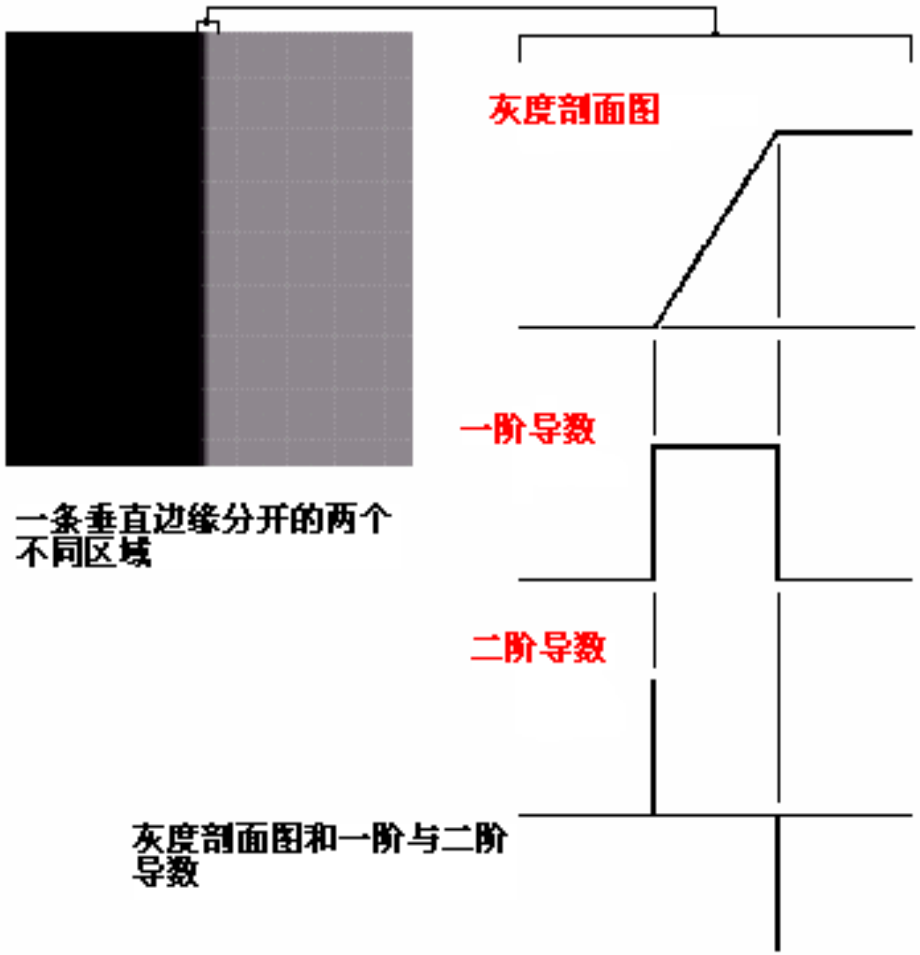
主要讨论基于一阶导数和二阶导数的图像边缘检测方法。

➢ 对灰度值剖面的一阶导数

在图像由暗变明的位置处有一个向上的跳变，而在其他位置为0，这表明可用一阶导数的幅度值来检测边缘的存在。

➢ 对灰度值剖面的二阶导数

在一阶导数的阶跃上升区有一个向上的脉冲，而在一阶导数的阶跃下降区有一个向下的脉冲。这2个阶跃之间的过零点正对应原图像中的边缘位置。

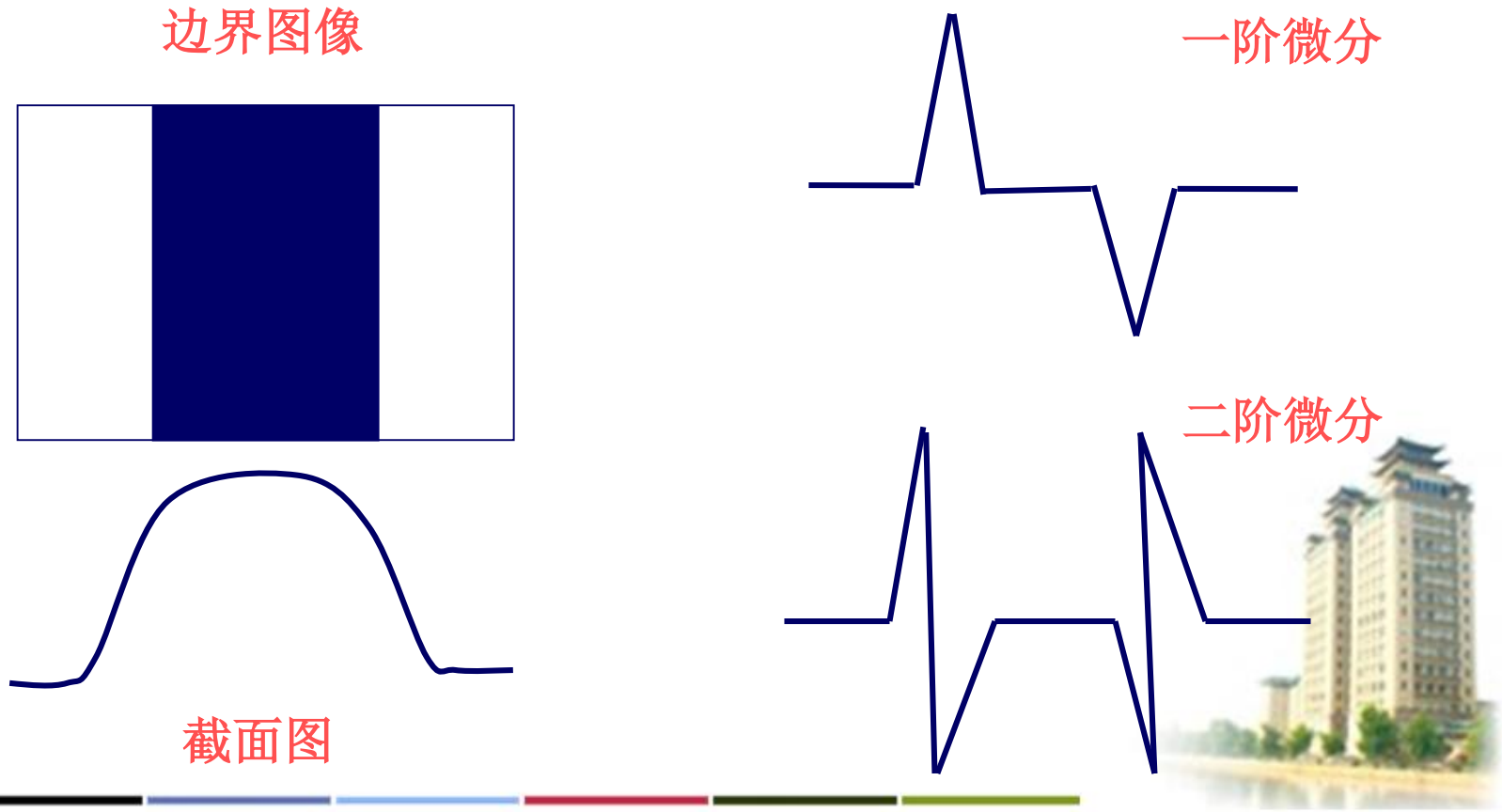


一条垂直边缘分开的两个不同区域

灰度剖面图和一阶与二阶导数

结论：

- 一阶导数的幅度峰值对应边缘位置；
- 二阶导数的过零点对应边缘位置，二阶导数在过零点附近的符号确定边缘象素在图像边缘的暗区或明区。



- 一阶导数可以用于检测图像中的一个点是否是边缘的点，即是否是斜坡边缘上的点；
- 二阶导数的符号可以用于判断一个边缘象素是在边缘亮的一边还是暗的一边。

二阶导数性质：

- 对图像中的每条边缘，二阶导数生成两个值。
- 边缘二阶导数正负值之间存在一个过零点。

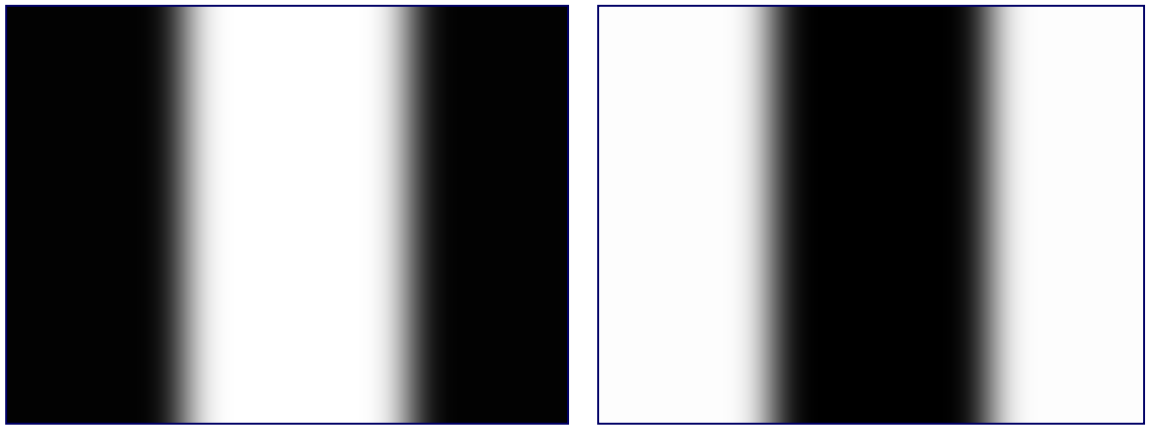
注意：

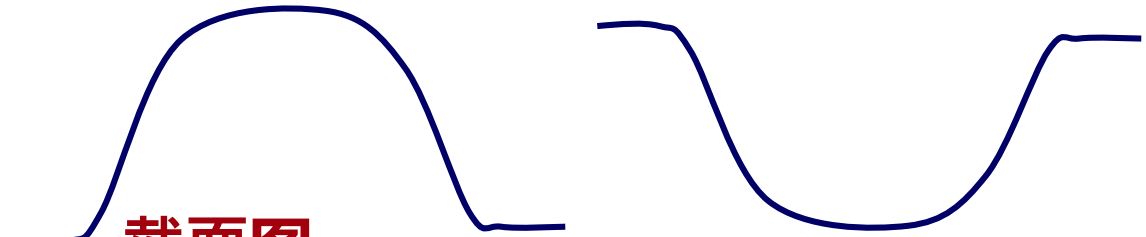
对混有噪声的图像，二阶导数边缘检测的方法要慎用！

图像分割：边界分割法

边缘检测的基本思想：计算局部微分算子

边界图像



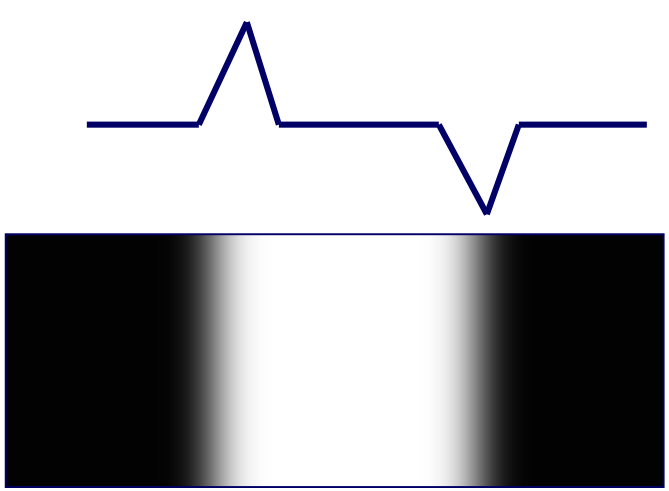


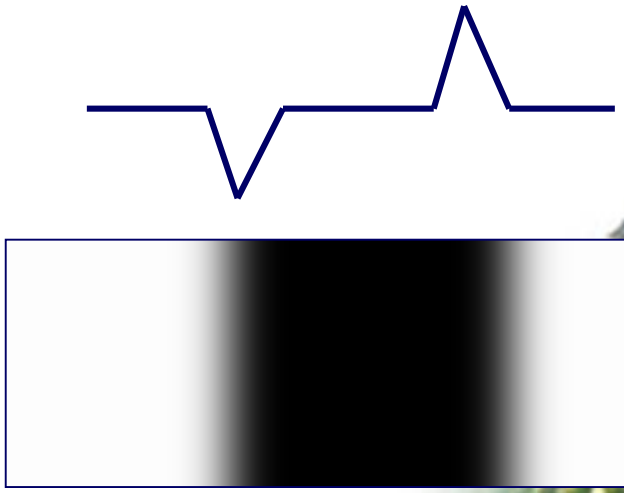
截面图

图像分割：边界分割法

边缘检测

- ◆一阶微分：用梯度算子来计算
  - ●特点：
    - ➢ 对于亮的边，边的变化起点是正的，结束是负的；
    - ➢ 对于暗边，结论相反；
    - ➢ 常数部分为零。
  - ●用途：用于检测图像中边的存在

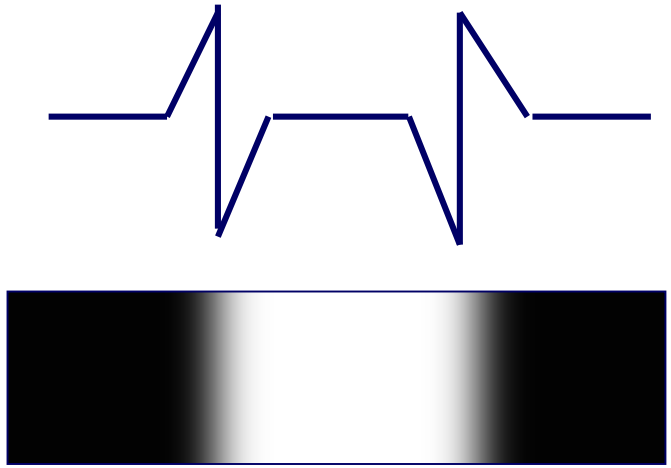




图像分割：边界分割法

边的检测

◆二阶微分：通过拉普拉斯来计算



- 特点
  - 二阶微分在亮的一边是负的；
  - 在暗的一边是正的；
  - 常数部分为零。
- 用途
  - 二次导数的符号，用于确定边上的像素是在亮的一边，还是暗的一边。
  - 0跨越，确定边的准确位置

边缘检测Edge Detection

一、梯度算子

定义图像f(x,y)在点(x,y)的梯度为G[f(x,y)]，即：

$$\nabla f(x,y) = G[f(x,y)] = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

其幅度：

$$G = |\nabla f(x,y)| = \left[\left(\frac{\partial f}{\partial x} \right)^2 + \left(\frac{\partial f}{\partial y} \right)^2 \right]^{\frac{1}{2}}$$

为了便于计算机处理，用差分的绝对值代替乘方和开方，即

$$G = |f(x,y) - f(x+1,y)| + |f(x,y) - f(x,y+1)|$$

1. Robert 算子

上述求梯度时，*f(x,y)*和*f(x+1,y+1)*的灰度差的象元位置关系并未考虑。Roberts把上述梯度做了修改，定义为Roberts梯度G_R。

表示成模板

1	
	−1

	1
−1	

Roberts

Robert算子边缘检测结果

2. Sobel 算子

Sobel梯度算子先加权平均，然后微分。

$$\Delta_x f(x,y) = [f(x-1,y+1) + 2f(x,y+1) + f(x+1,y+1)]$$
$$-[f(x-1,y-1) + 2f(x,y-1) + f(x+1,y-1)]$$

$$\Delta_y f(x,y) = [f(x-1,y-1) + 2f(x-1,y) + f(x-1,y+1)]$$
$$-[f(x+1,y-1) + 2f(x+1,y) + f(x+1,y+1)]$$

表示成模板

−1	−2	−1
1	2	1

−1		1
−2		2
−1		1

Sobel梯度算子的使用与分析

1.直接计算∂y、∂x可以检测到边的存在，以及从暗到亮，从亮到暗的变化

2.仅计算|∂x|，产生最强的响应是正交于x轴的边； |∂y|则是正交于y轴的边。

3.Soleb算子具有平滑效果，由于微分增强了噪音，这一点是特别引人注意的特性

3. Prewitt 算子

- 普瑞维特（Prewitt）边缘检测算子是一种利用局部差分平均方法寻找边缘的算子，它体现了3对像素点像素值之差的平均概念，算子形式如下：

−1	−1	−1
1	1	1

−1		1
−1		1
−1		1

Prewitt

上述三个算子分别对G_x和G_y各用一个模板两个梯度分量组合起来构成一个梯度算子

$$\nabla f = |G_x| + |G_y|$$

示例：

示例：

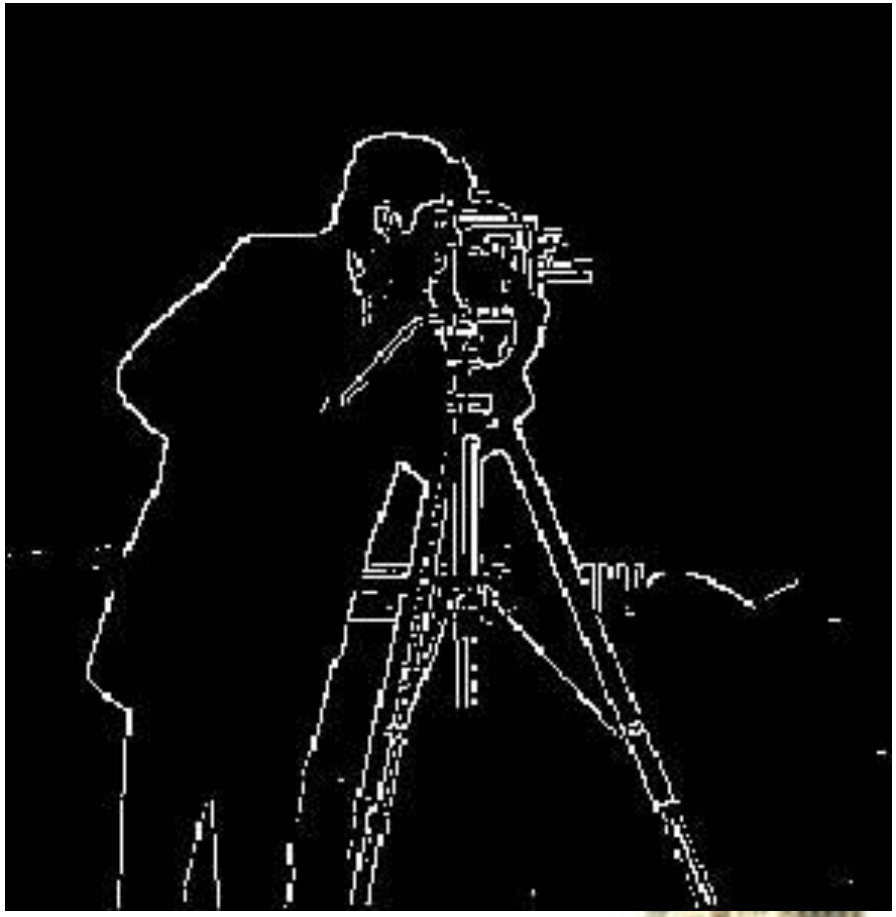
a b
c d

FIGURE 10.10
(a) Original image. (b) $|G_x|$, component of the gradient in the x-direction. (c) $|G_y|$, component in the y-direction. (d) Gradient image, $|G_x| + |G_y|$.

Sobel 算子

Prewitt 算子

Roberts 算子



二、拉普拉斯算子

图像f(x,y)的拉普拉斯算子是不依赖于边缘方向的二阶导数：

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

同样对拉普拉斯算子采用差分近似微分的方法：

$$\frac{\partial^2 f}{\partial x^2} = [f_x(i,j) - f_x(i+1,j)]$$
$$= [f(i,j) - f(i-1,j)] - [f(i+1,j) - f(i,j)]$$
$$\frac{\partial^2 f}{\partial y^2} = [f_y(i,j) - f_y(i,j+1)]$$
$$= [f(i,j) - f(i,j-1)] - [f(i,j+1) - f(i,j)]$$

这样，拉普拉斯算子可表示为：

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

◆ 定义数字形式的拉普拉斯的基本要求

- 作用于中心像素的系数是一个正数；
- 而且其周围像素的系数为负数；
- 系数之和必为0。

◆ 表示成模板

	−1	
−1	4	−1
	−1	

或

	1	
1	−4	1
	1	

◆ 拉普拉斯算子一般不用于单纯的边缘检测？

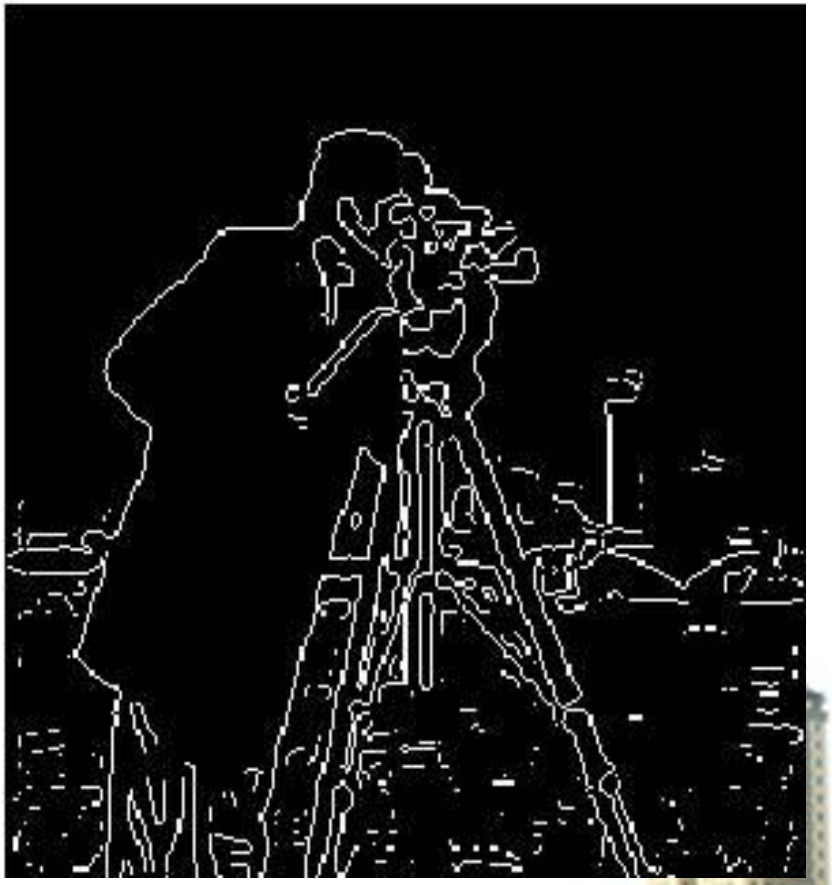
- 作为一个二阶导数，拉普拉斯算子对噪声具有无法接受的敏感性；
- 拉普拉斯算子的幅度产生双边缘，这是复杂的分割不希望产生的结果；
- 拉普拉斯算子不能检测边缘的方向。

◆ 拉普拉斯算子在图像分割中的作用？

- 利用零交叉的性质进行边缘定位；
- 确定一个象素是在边缘暗的一边还是亮的一边。

Laplacian 算子

Laplacian 算子



边缘检测Edge Detection

三、LOG算子(Laplacian of Gaussian)

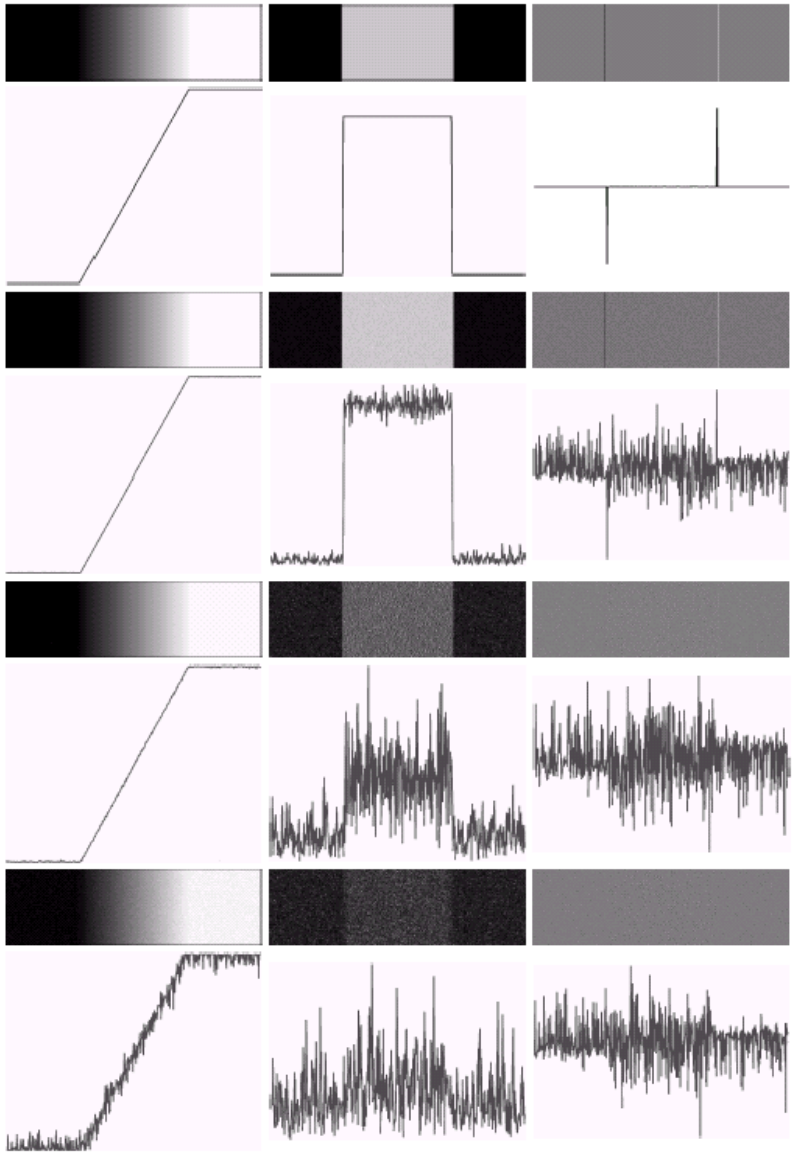
- 由于微分运算对噪声有放大作用，因此梯度算子和拉普拉斯算子对噪声比较敏感。

◆ 改进方法：

- 先对图像进行适当平滑，抑制噪声，然后再进行微分。

◆ LOG算子基本思想：

- Marr和Hildreth将高斯滤波和拉普拉斯边缘检测结合在一起，使用了Gaussian来进行噪声去除并使用Laplacian来进行边缘检测。



◆ LOG算子是一种用的较多的先平滑后微分的方法，算法如下：

该算子先用高斯函数平滑：

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

平滑后的图像：

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

对平滑后的图像求二阶导数：

$$\nabla^2 g(x,y)=\nabla^2(G(x,y,\sigma)*f(x,y))$$

根据卷积求导法有

$$=(\nabla^2 G(x,y,\sigma)*f(x,y))$$

而

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

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

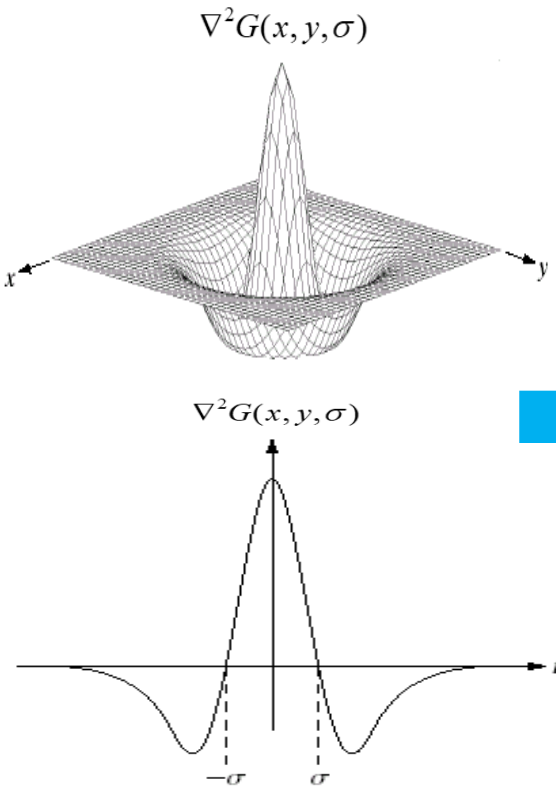
Laplacian of Gaussian（LOG）滤波器
Marr边缘检测算子

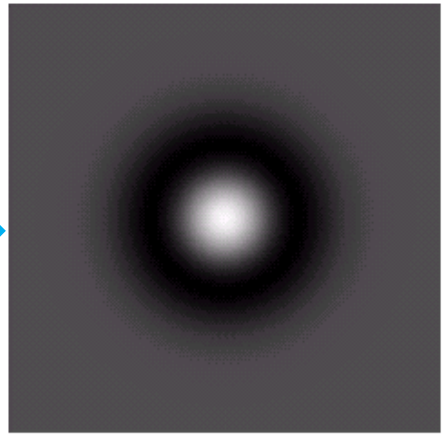
$$\nabla^2 G(x,y,\sigma)=\frac{\partial^2 G}{\partial x^2}+\frac{\partial^2 G}{\partial y^2}$$

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

墨西哥草帽算子

- 2D高斯拉普拉斯算子可以通过任何一个方形核进行逼近；
- 只要保证该核的所有元素的和或均值为0；
- 下边为一个5×5的核进行逼近。





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

LOG边缘检测算子

边缘检测Edge Detection

四、Canny算子

- 动机
  - 好的检测结果：对边缘的错误检测率要尽可能低，在检测出图像真实的边缘的同时要避免检测出现虚假的边缘。
  - 好的边缘定位精度：标记出的边缘位置要和图像上真正边缘的位置尽量接近。
  - 对同一边缘要有低的响应次数：有的算子会对一个边缘产生多个响应。也就是说图像上本来只有一个边缘点的，可是检测出来就会出现多个边缘点。
  - 克服噪声的影响。

◆Canny边缘检测基本原理：

- 图象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 LoG 边缘检测方法，也属于先平滑后求导数的方法。

◆Canny边缘检测算法：

1. 用高斯滤波器平滑图像；
2. 用一阶偏导有限差分计算梯度幅值和方向；
3. 对梯度幅值应用非极大值抑制；
4. 用双阈值算法检测和连接边缘。

（1）高斯平滑滤波器

高斯滤波器是一种线性滤波模板，模板各位置的权重是根据高斯分布函数确定的。由于高斯滤波器的二维可分性（X 轴与 Y 轴方向进行高斯滤波互不干扰），我们可以用两个一维高斯滤波器的连续卷积来实现一个二维高斯滤波器：

（1）对图像使用一维高斯卷积模板，在一个方向上进行滤波（例如水平方向）。

（2）转置图像；

（3）对转置以后的图像使用同一个高斯卷积模板，在同样方向进行滤波。

（4）将图像转置回原来的位置，我们就得到了经过二维滤波的图像。

一维高斯卷积模板可以由二项式展开的系数来模拟

a）3×3 模板

$$\frac{1}{4} \times \begin{array}{|c|c|c|} \hline 1 & 2 & 1 \\ \hline \end{array}$$

b）5×5 模板

$$\frac{1}{16} \times \begin{array}{|c|c|c|c|c|} \hline 1 & 4 & 6 & 4 & 1 \\ \hline \end{array}$$

c）7×7 模板

$$\frac{1}{64} \times \begin{array}{|c|c|c|c|c|c|c|} \hline 1 & 6 & 15 & 20 & 15 & 6 & 1 \\ \hline \end{array}$$

（2）使用一阶有限差分计算偏导数的两个阵列P与Q：

$$P[y,x] \approx (S[y,x+1] - S[y,x] + S[y+1,x+1] - S[y+1,x])/2$$

$$Q[y,x] \approx (S[y+1,x] - S[y,x] + S[y+1,x+1] - S[y,x+1])/2$$

$$\begin{matrix} 0 & 1 \\ 0 & 1 \end{matrix} \qquad \begin{matrix} 0 & 0 \\ 1 & 1 \end{matrix}$$

（3）幅值和方位角：

$$M[y,x] = \sqrt{P[y,x]^2 + Q[y,x]^2}$$

$$\theta[y,x] = \arctan(Q[y,x]/P[y,x])$$

（4）非极大值抑制（NMS non-maxima suppression ）：

- 仅仅得到全局的梯度并不足以确定边缘；
- 保留局部梯度最大的点，而抑制非极大值；
- 解决方法：利用梯度的方向。
  - 将梯度角的变化范围减小到圆周的四个扇区之一；
  - 四个扇区的标号为0到3，对应3*3邻域的四种可能组合方向；
  - 每一点上邻域的中心象素M与沿着梯度线的两个象素比较；
  - 如果M梯度值不比沿梯度线的两个相邻象素梯度值大，则令M=0。

1	2	3
8	M	4
7	6	5

（4）双阈值：

- 问题：
  - 将低于阈值的所有值赋零，得到图像的边缘阵列；
  - 阈值 τ 取得太低→假边缘；
  - 阈值 τ 取得太高→部分轮廓丢失。

- 解决方案：选用两个阈值
- 基本思想：
  - 取高低两个阈值作用在幅值图 N[i,j]，t1 = 2*t2；
  - 得到两个边缘图，高阈值边缘图和低阈值边缘图；
  - 连接高阈值边缘图；
  - 在低阈值边缘图中的8邻点域搜寻边缘点以补充连接高阈值边缘图。

Canny算子的实现步骤

- 用高斯滤波器平滑图像；
- 计算滤波后图像梯度的幅值和方向；
- 对梯度幅值应用非极大值抑制，其过程为找出图像梯度中的局部极大值点，把其它非局部极大值点置零以得到细化的边缘；
- 用双阈值算法检测和连接边缘，使用两个阈值T1和T2(T1>T2)，T1用来找到每条线段，T2用来在这些线段的两个方向上延伸寻找边缘的断裂处，并连接这些边缘。

Canny算子检测结果

A=edge(I,'canny')

边缘检测的MATLAB调用语法

- **Sobel检测** [g,t]=edge(f,'sobel',T,dir)
- **Prewitt检测** [g,t]=edge(f,'prewitt',T,dir)
- **Roberts检测** [g,t]=edge(f,'roberts',T,dir)
  - f 为输入图像，T 为指定阈值，
  - dir 指定检测边缘的方向：'horizontal'、'vertical'、'both'(默认值)
  - 输出参数t是可选的，它是函数edge所用的阈值。
- **Laplacian of a Gaussian(LoG)检测** [g,t]=edge(f,'log',T,sigma)
  - sigma是标准偏差，默认值为2。
- **Canny检测** [g,t]=edge(f,'canny',T,sigma)
  - sigma是平滑滤波器标准偏差。

几种常用算子的比较

- Roberts算子：Roberts算子利用局部差分算子寻找边缘，边缘定位精度高，但容易丢失一部分边缘，同时由于图像没经过平滑处理，因此不具备能抑制噪声能力。该算子对陡峭边缘且含噪声少的图像效果较好。
- Sobel算子和Prewitt算子：都是对图像先做加权平滑处理，然后再做微分运算，所不同的是平滑部分的权值有些差异，因此对噪声具有一定的抑制能力，但不能完全排除检测结果中出现的虚假边缘。虽然这两个算子边缘定位效果不错，但检测出的边缘容易出现多像素宽度。
- Laplacian算子：是不依赖于边缘方向的二阶微分算子算子，对图像中的阶跃型边缘点定位准确，该算子对噪声非常敏感，它使噪声成分得到加强，这两个特性使得该算子容易丢失一部分边缘的方向信息，造成一些不连续的检测边缘，同时抗噪声能力比较差。

几种常用算子的比较

- LOG算子：该算子首先用高斯函数对图像作平滑滤波处理，然后才使用Laplacian算子检测边缘，因此克服了Laplacian算子抗噪声能力比较差的缺点，但是在抑制噪声的同时也可能将原有的比较尖锐的边缘也平滑掉了，造成这些尖锐边缘无法检被测到。应用LOG算子，高斯函数中方差参数的选择很关键，对图像边缘检测效果有很大的影响。高斯滤波器为低通滤波器，越大，通频带越窄，对较高频率的噪声的抑制作用越大，避免了虚假边缘的检出，同时信号的边缘也被平滑了，造成某些边缘点的丢失。反之，越小，通频带越宽，可以检测到的图像更高频率的细节，但对噪声的抑制能力相对下降，容易出现虚假边缘。因此应用LOG算子，为取得更佳的效果，对于不同图像应选择不同参数。

- Canny算子：Canny算子虽然是基于最优化思想推导出的边缘检测算子，实际效果并不一定最优，原因在于理论和实际有许多不一致的地方。该算子同样采用高斯函数对图像作平滑处理，因此具有较强的抑制噪声能力，同样该算子也会将一些高频边缘平滑掉，造成边缘丢失。Canny算子其后所采用双阈值算法检测和连接边缘，采用的多尺度检测和方向性搜索较LOG算子要好。

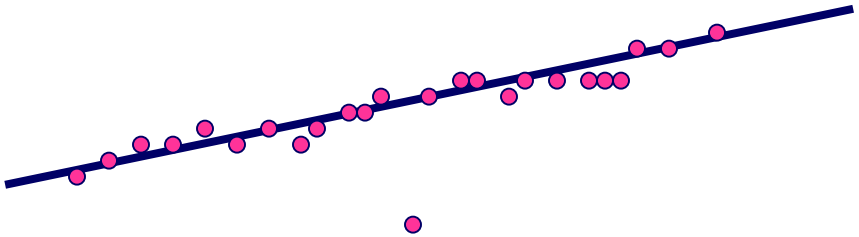
图像边缘提取及处理

1. 边缘检测
2. Hough变换
3. 边界特征表达及描述

边缘的Hough变换

◆ 问题的提出

在找出边界点集之后，需要连接，形成完整的边界图形描述。



◆ 基本原理

利用点与线的对偶性，将图像空间的线条变为参数空间的点，从而检测图像中是否存在给定性质的线条。

直线的检测

设原始图像空间为(x,y)，则直线的方程可表为：

$$y = u\,x + v$$

其中 u 为斜率，v 为截距

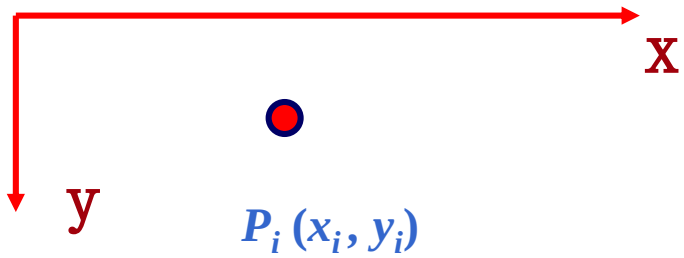
考虑变换的参数空间(u, v)，直线上任意点$P_i\,(x_i, y_i)$，

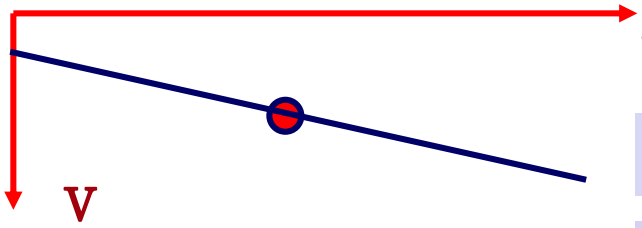
在参数空间内满足 $y_i = u\,x_i + v$

即： $v = -\,x_i\,u + y_i$

因此点 P_i 对应了参数空间内的一条直线。

图像空间内的一点$P_i\,(x_i, y_i)$，对应参数空间内的一条直线。

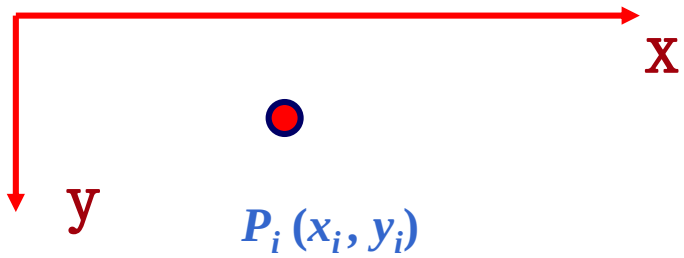


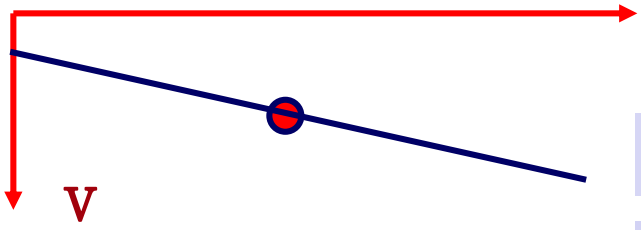


$y_i = u\,x_i + v$

$v = -\,x_i\,u + y_i$

图像空间内的一点$P_i(x_i, y_i)$，对应参数空间内的一条直线。

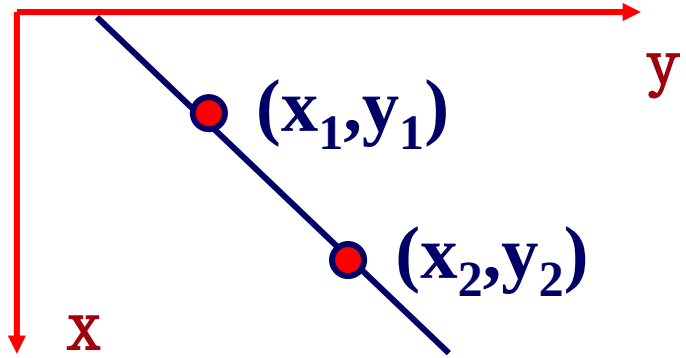




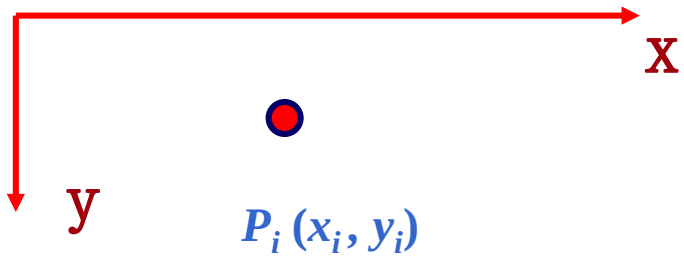
$$y_i = u\,x_i + v$$

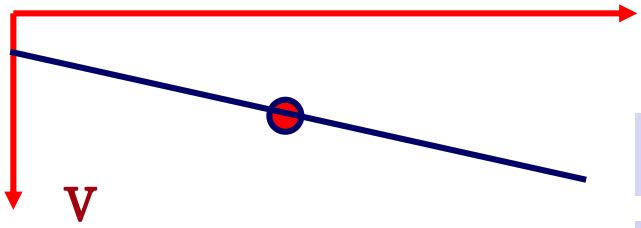
$$v = -\,x_i\,u + y_i$$

如果点(x1, y1)与点(x2, y2)共线，那么这两点在参数uv平面上的直线将……



图像空间内的一点$P_i(x_i, y_i)$，对应参数空间内的一条直线。

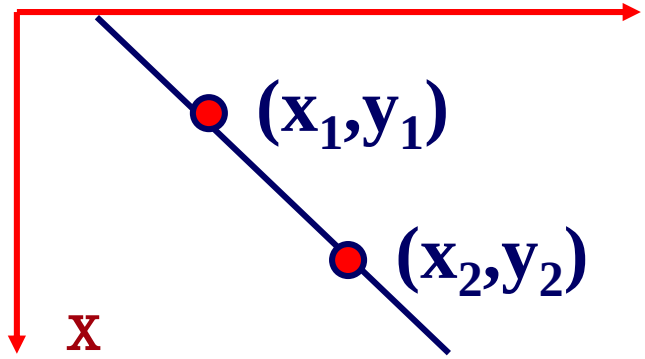


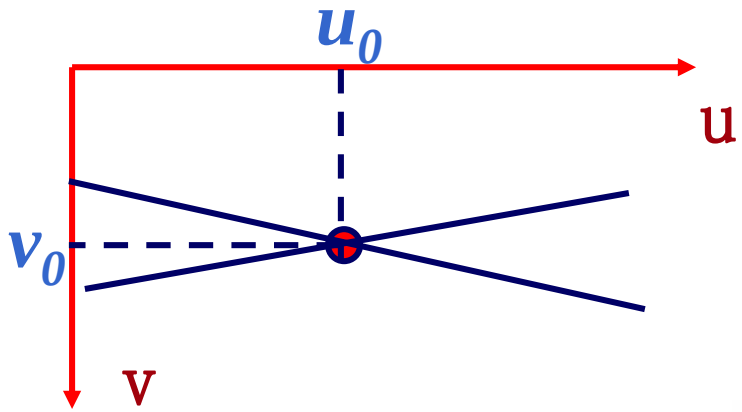


$$y_i = u\,x_i + v$$

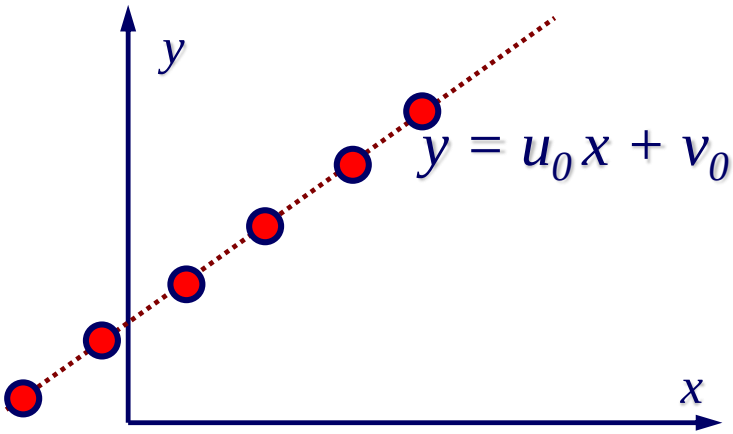
$$v = -\,x_i\,u + y_i$$

如果点(x1, y1)与点(x2, y2)共线，那么这两点在参数uv平面上的直线将有一个交点(u_0, v_0)

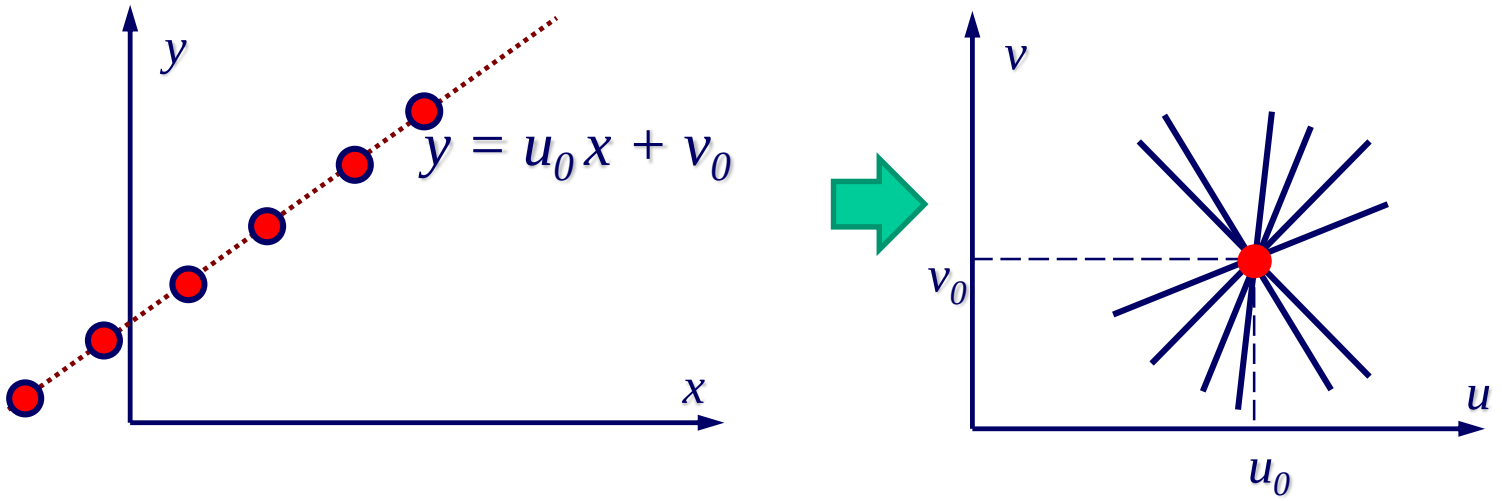




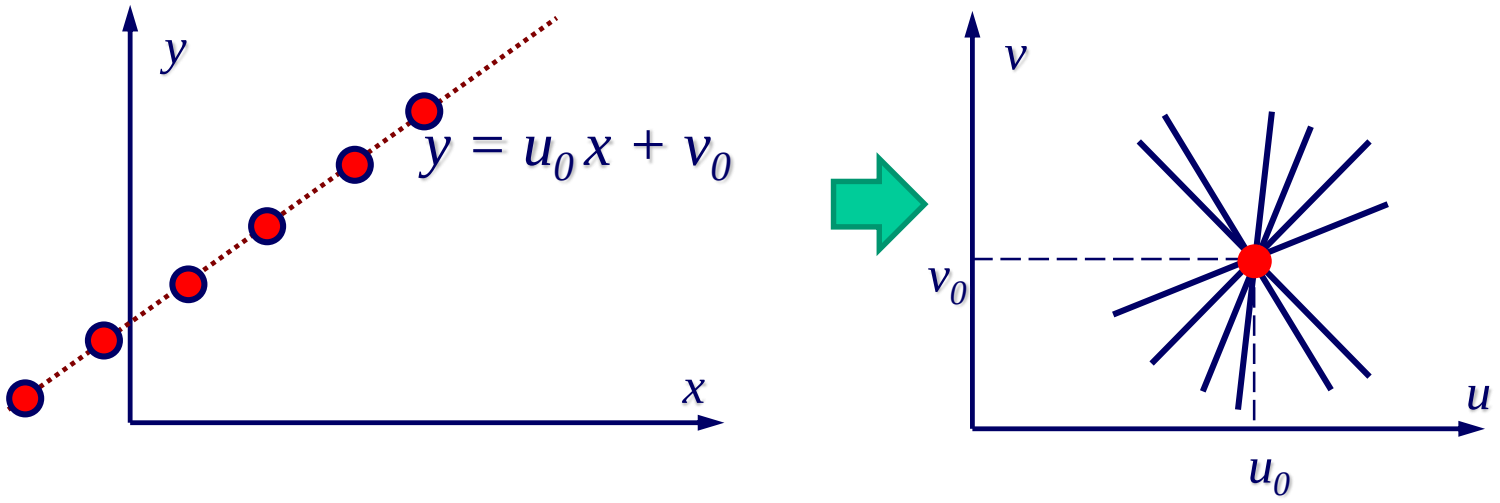
于是，图像空间内共直线的一系列点 ⇨ 对应参数空间内一族直线



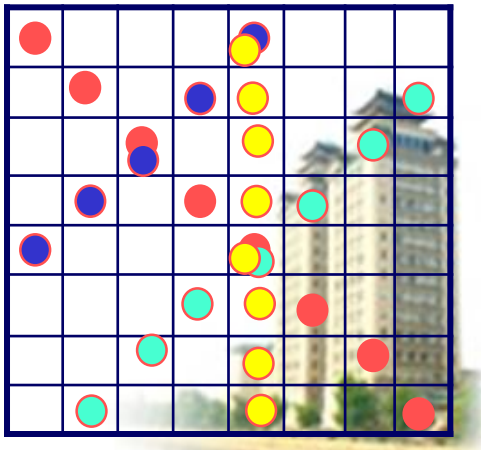
于是，图像空间内共直线的一系列点 ⇒ 对应参数空间内一族直线



于是，图像空间内共直线的一系列点 ⇨ 对应参数空间内一族直线



因此，在参数空间内所有过同一点的直线，对应图像空间内共线的点。若图像空间内有共线的点(实线或虚线)，必然在参数空间内形成过同一点的直线族，**通过累加，检测峰点**，即可以知道是否有所检测的线条。



◆ 算法描述：

1. 在参数空间(u,v)内建立两维数组 $A(u,v)$;
2. 在开始时将数组置零
3. 对图像空间的每一个待检测点(x_i, y_i)，令 u 取遍所有可能的取值，并计算对应的 v。

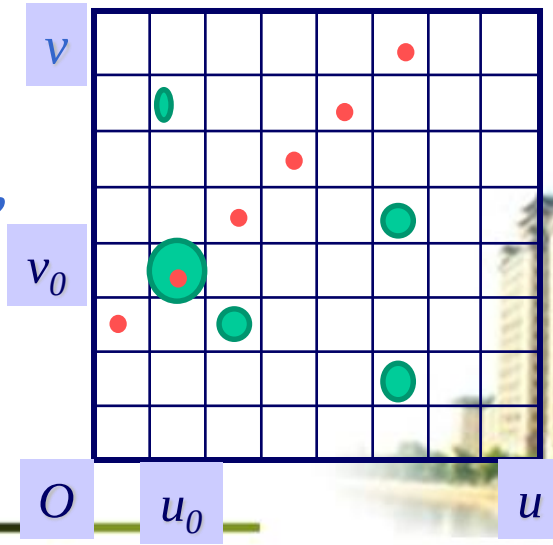
$$v = - x_i u + y_i$$

4. 对计算得到的(u,v)，对$A(u,v)$中相应单元进行累加:

$$A(u, v) = A(u, v) + 1$$

5. 根据 $A(u,v)$的值，确定有多少点是共线的，同时可以知道线条的参数 (u,v)。

$$y = u x + v$$

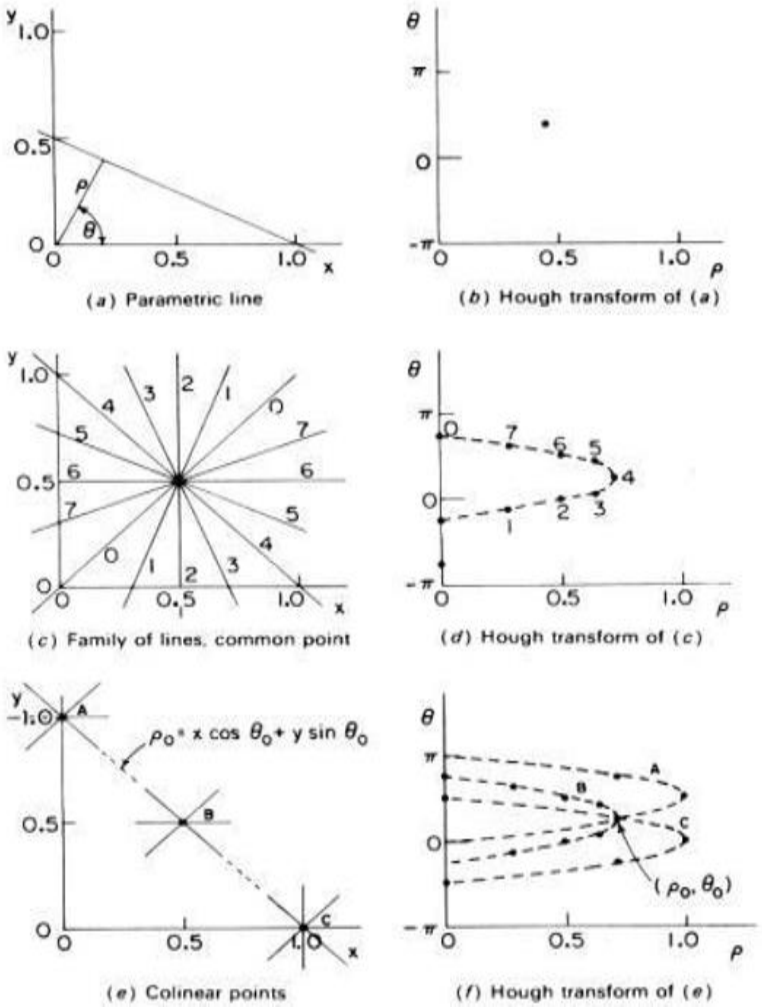


◆ 算法实现（采用极坐标）：

- 由于垂直直线a为无穷大，改用极坐标形式：

  $x\cos\theta + y\sin\theta = \rho$

- 参数平面为θ,ρ ，对应不是直线而是正弦曲线
- 使用交点累加器，或交点统计直方图，找出相交线段最多的参数空间的点
- 然后找出该点对应的xy平面的直线线段



◆Hough变换的扩展讨论：

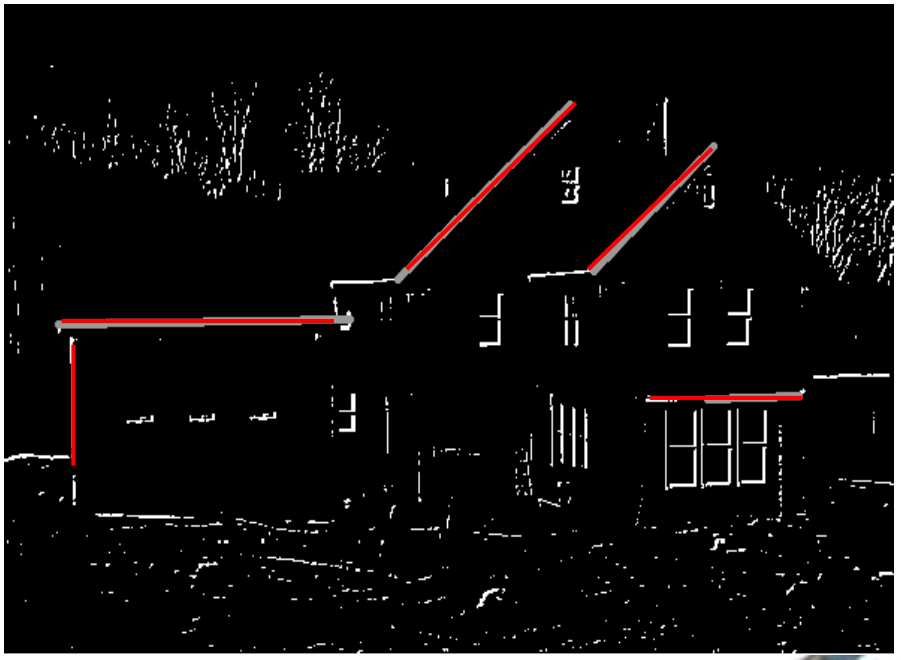
1. Hough变化之后的处理，如何确定直线（不连续）

2. 圆的检测：

   $(x-a)^2+(y-b)^2 = r^2$ 参数为 a, b, r

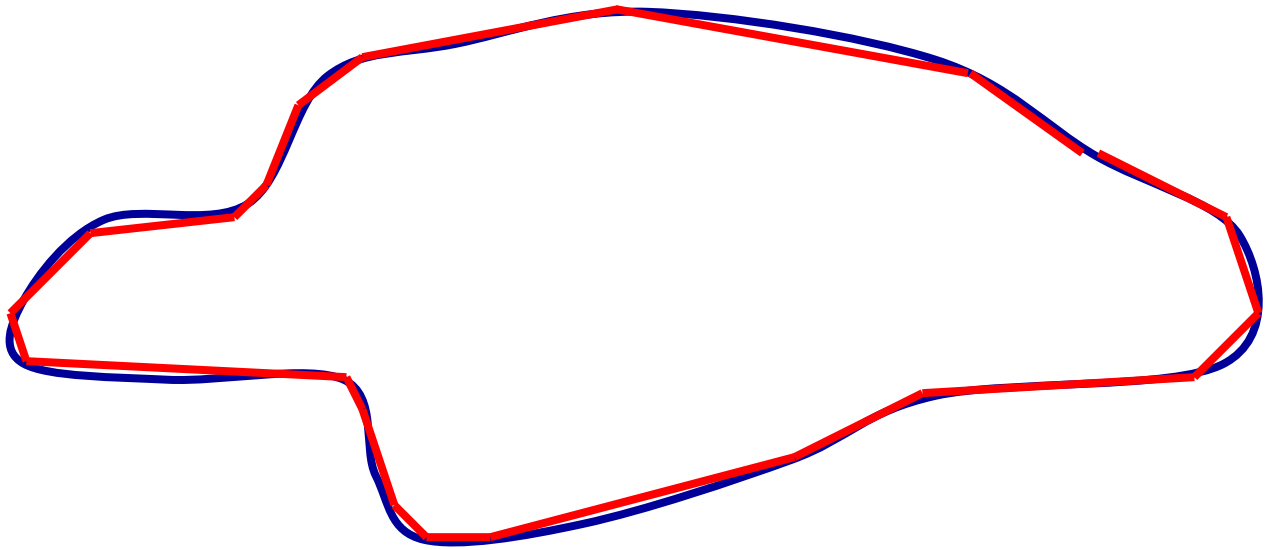
   需要三个参数的参数空间，建立数组 A(a, b, r)来检测

3. 广义 Hough 变换： 检测特定的可描述形状（如椭圆等）



图像边缘提取及处理

1. 边缘检测
2. Hough变换
3. 边界特征表达及描述

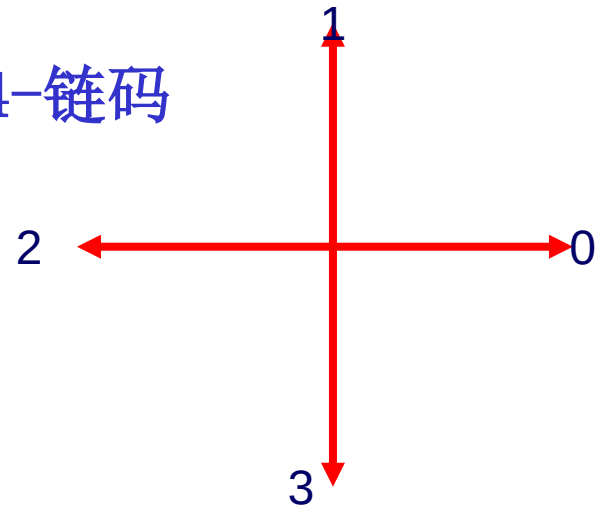


边缘特征表示与描述——链码

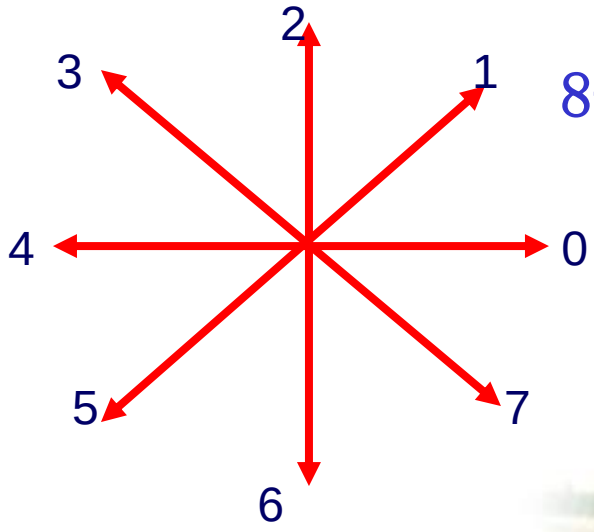
- 链码
  - 定义：1）链码是一种边界的编码表示法。

    2）用边界的方向作为编码依据。为简化边界的描述。一般描述的是边界点集。

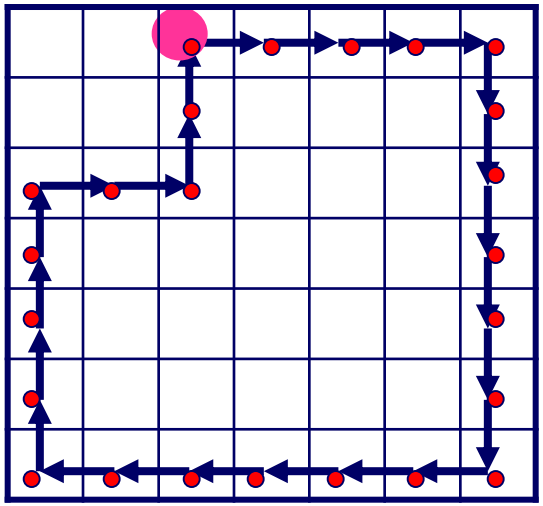
4-链码



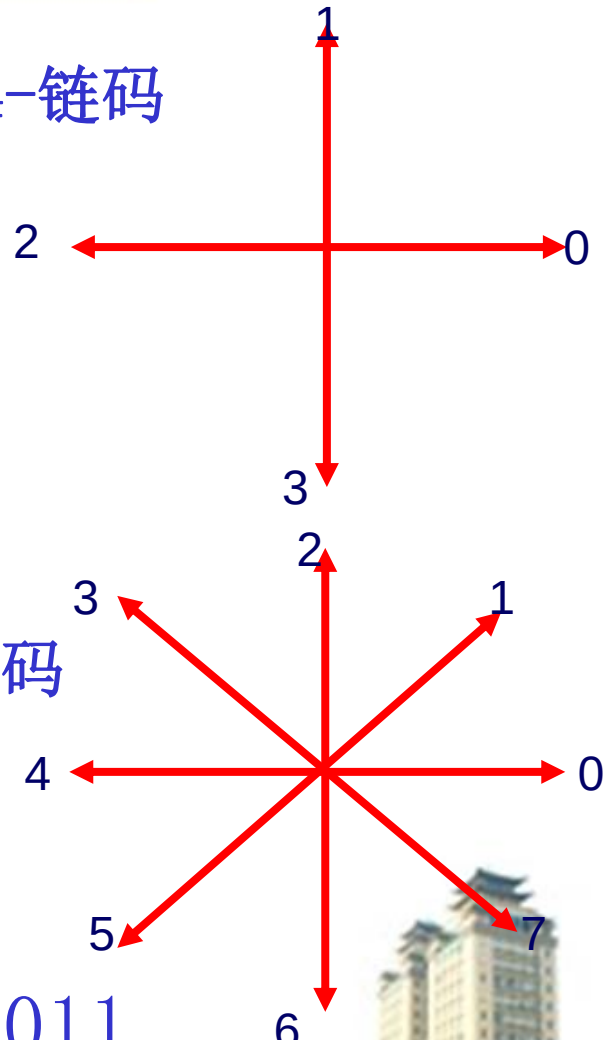
8-链码



- 链码举例：



4-链码



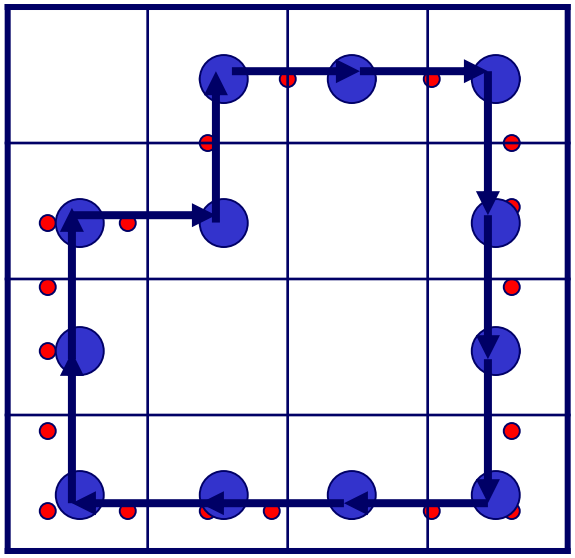
8-链码

4-链码：00003333332222221111001 1

- 链码
  - –算法：
    - 给每一个线段边界一个方向编码。
    - 有4-链码和8-链码两种编码方法。
    - 从起点开始，沿边界编码，至起点被重新碰到，结束一个对象的编码。

- **链码**
  - **问题1：**
    1）链码相当长。
    2）噪音会产生不必要的链码。
  - **改进1：**
    1）加大网格空间。
    2）依据原始边界与结果的接近程度，来确定新点的位置。

- 链码举例：



4-链码：00333221101

- **链码**
  - **问题2：**
    1）由于起点的不同，造成编码的不同
    2）由于角度的不同，造成编码的不同
  - **改进2：**
    1）从固定位置作为起点(最左最上)开始编码
    2）通过使用链码的首差代替码子本身的方式

- **链码**

  – 循环首差链码：用相邻链码的差代替链码

例如：4-链码 10103322 循环首差为：33133030

循环首差：1 - 2 = -1(3)　　3 - 0 = 3

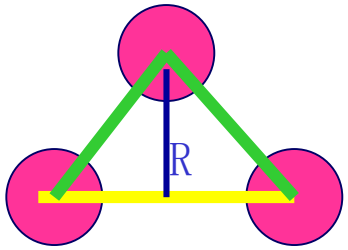
0 - 1 = -1(3)　　3 - 3 = 0

1 - 0 = 1　　2 - 3 = -1(3)

0 - 1 = -1(3)　　2 - 2 = 0

边缘特征表示与描述——多边形逼近

- **多边形逼近**
  - 基本思想：用最少的多边形线段，获取边界形状的本质。
  - 寻找最小基本多边形的方法一般有两种：
    1）点合成法
    2）边分裂法



R < T

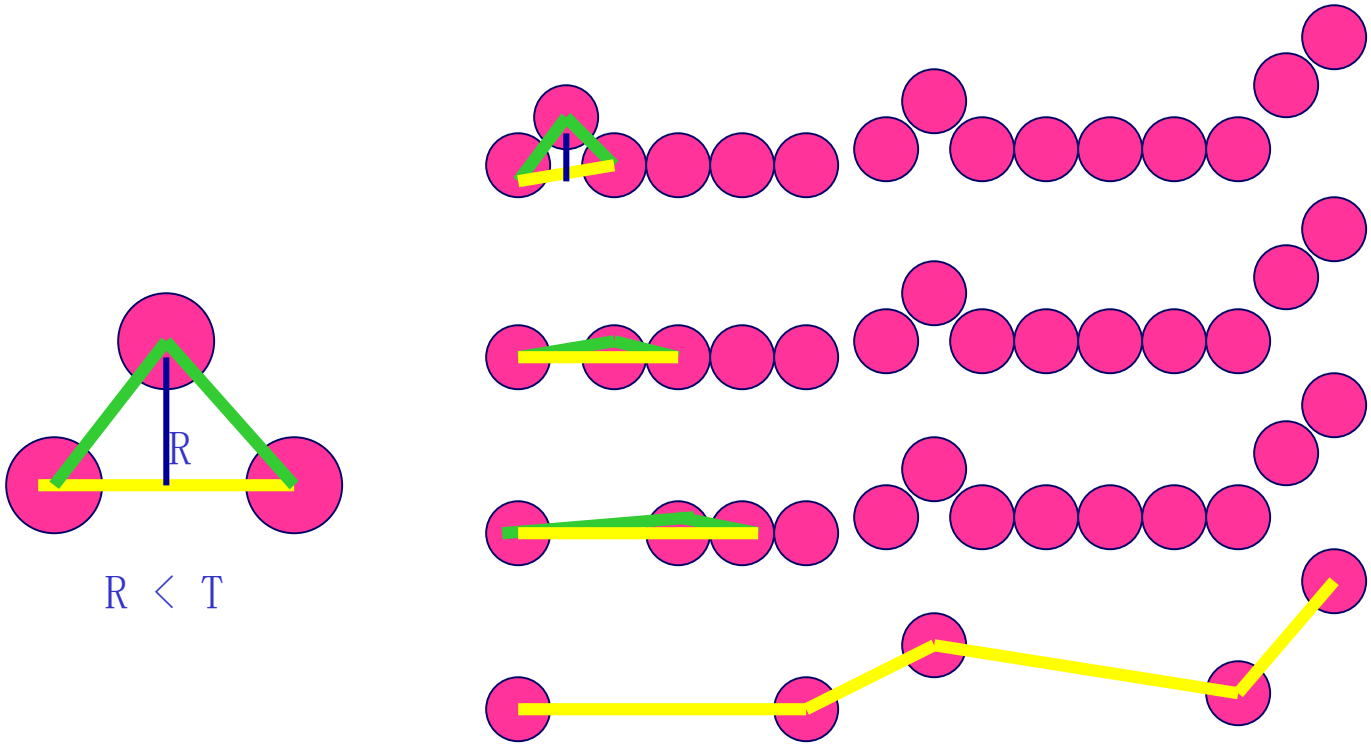
- **多边形逼近**
  - 点合成算法：

    1）沿着边界选两个相邻的点对，计算首尾连接直线段与原始折线段的误差R。

    2）如果误差R小于预先设置的阈值T。去掉中间点，选新点对与下一相邻点对，重复1）；否则，存储线段的参数，置误差为0，选被存储线段的终点为起点，重复1）2）。

    3）当程序的第一个起点被遇到，程序结束。

- **多边形逼近**
  - 点合成算法思想举例：



- **多边形逼近**
  - 点合成算法的问题：
    - 顶点一般不对应于边界的拐点（如拐角）。
    - 因为新的线段直到超过误差的阈值才开始。
    - 下面讲到的分裂法可用于缓解这个问题

- **多边形逼近**
  - 分裂边算法：

（1）连接边界线段的两个端点（如果是封闭边界，连接最远点）；

（2）如果最大正交距离大于阈值，将边界分为两段，最大值点定位一个顶点。重复（1）；

（3）如果没有超过阈值的正交距离，结束。

- 多边形逼近
  - 边分裂算法思想举例：