

专业限选系列课程

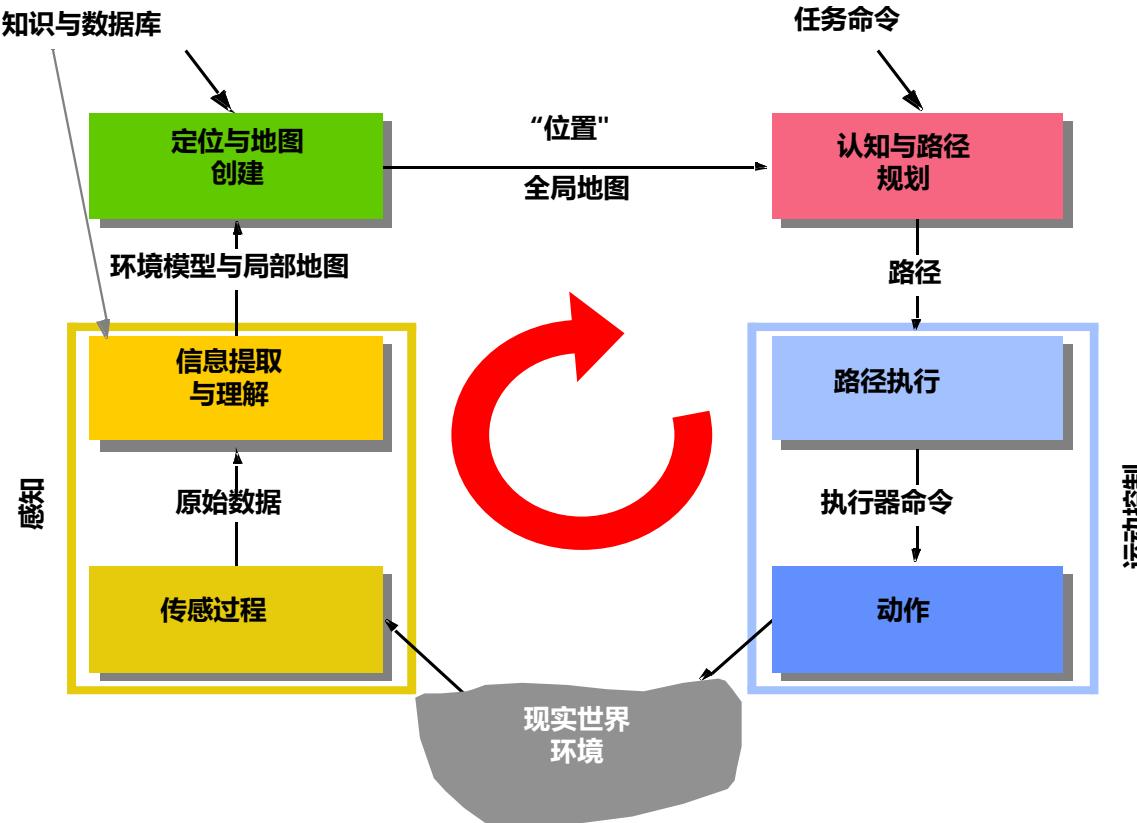
# 智能机器人技术

---Introduction to Intelligent Robotics

赵振刚 gavin@ustc.edu.cn

# 机器人学的知识架构

- 1. 感知**: 视觉传感器、图像传感器、触觉和力传感器、惯导等；
- 2. 认知**: 人工智能、知识表达、规划、任务调度、机器学习等；
- 3. 行为**: 运动学、动力学、控制、manipulation和locomotion等；
- 4. 数学基础**: 最优估计、微分几何、计算几何、运筹学等；



在机器人领域，越来越多地通过人工智能，特别是计算机视觉研究，实现机器人的自主识别行为



# Questions

1. 成像的过程如何量化？
2. 如何描述和定义一帧图像？
3. 如何搜索和分割感兴趣区域？
4. 如何准确、高速地识别出目标？
5. 运动目标的跟踪方法有哪些？

# 第三章 机器人的视觉分析

## 目标:

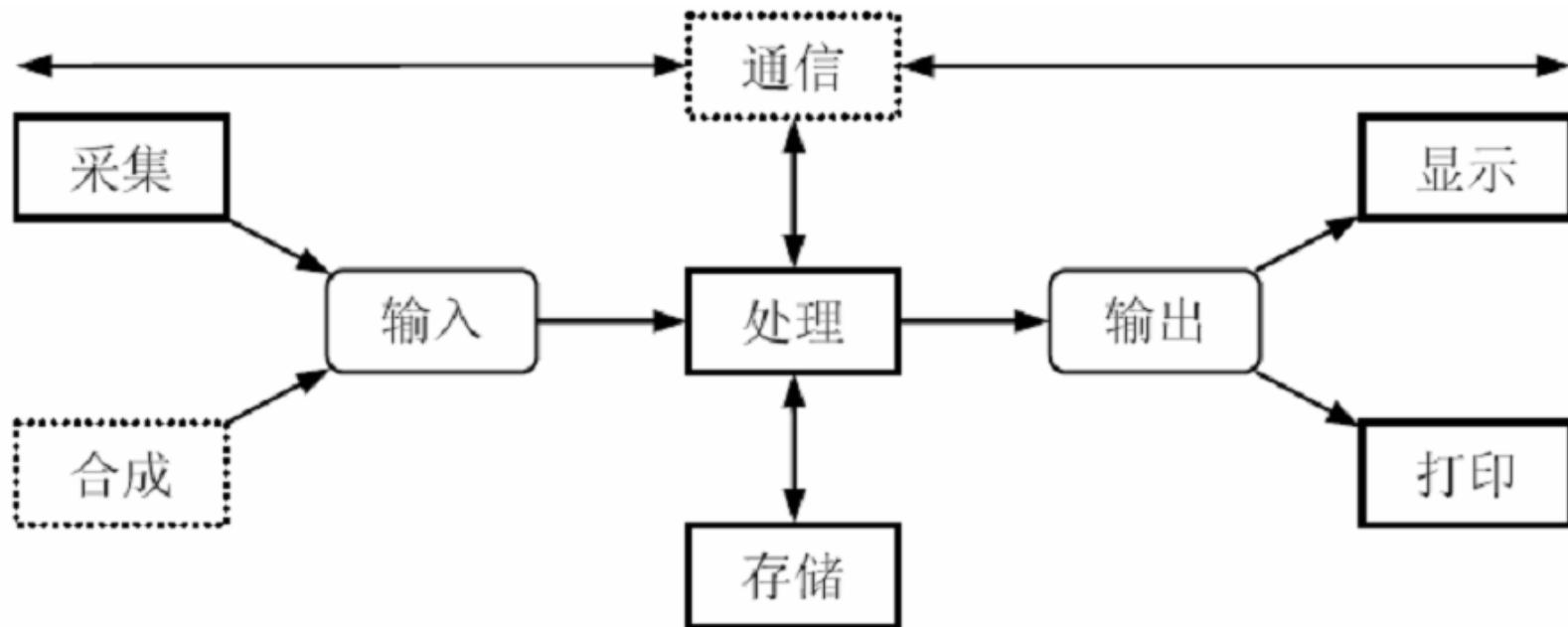
- ❖ 掌握图像处理的一般流程和属性
- ❖ 掌握图像滤波的原理和方法
- ❖ 了解特征分析的工具和方法

## 目录:

- ❖ 数字图像概念
- ❖ 数字图像处理基本步骤
- ❖ 图像预处理
- ❖ 图像区域表示
- ❖ 二值图像运算
- ❖ 图像特征比对
- ❖ OpenCV平台使用

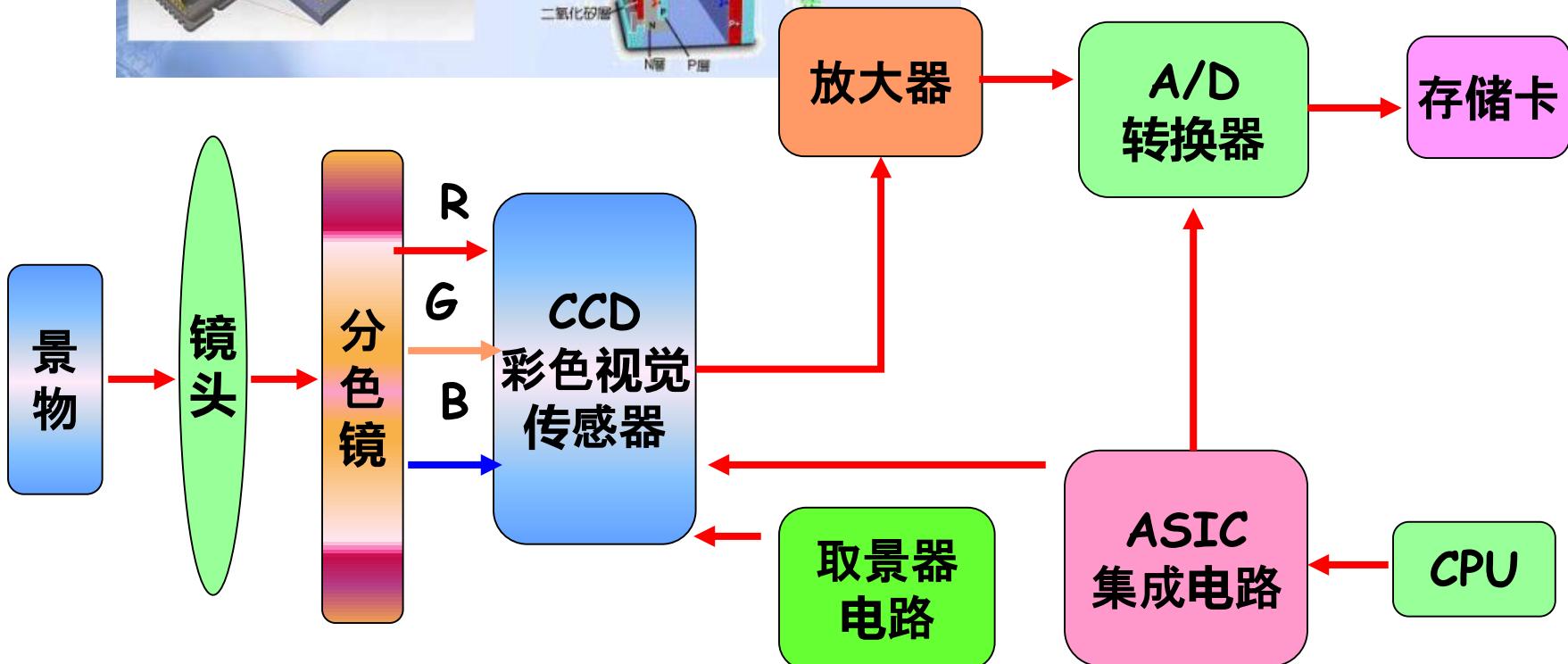
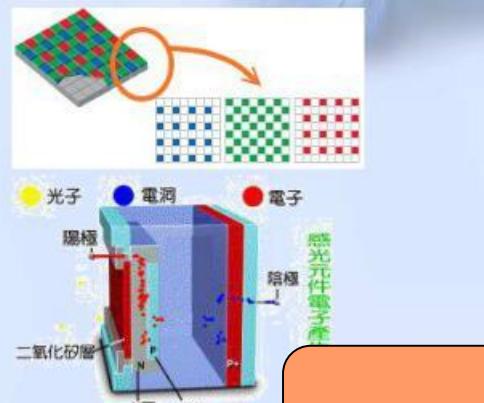
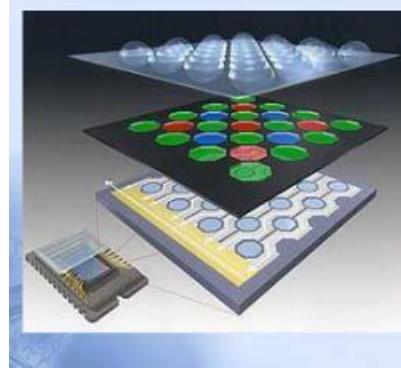
## 3.1 数字图像概念

### ❖ 视觉系统的构成



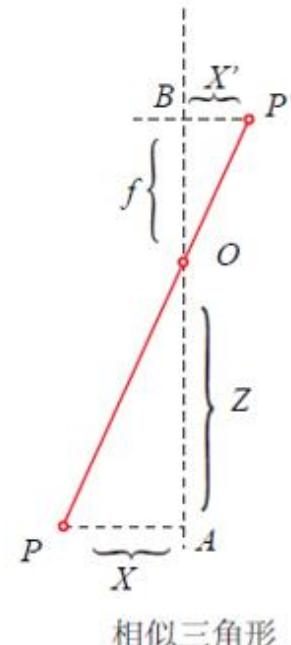
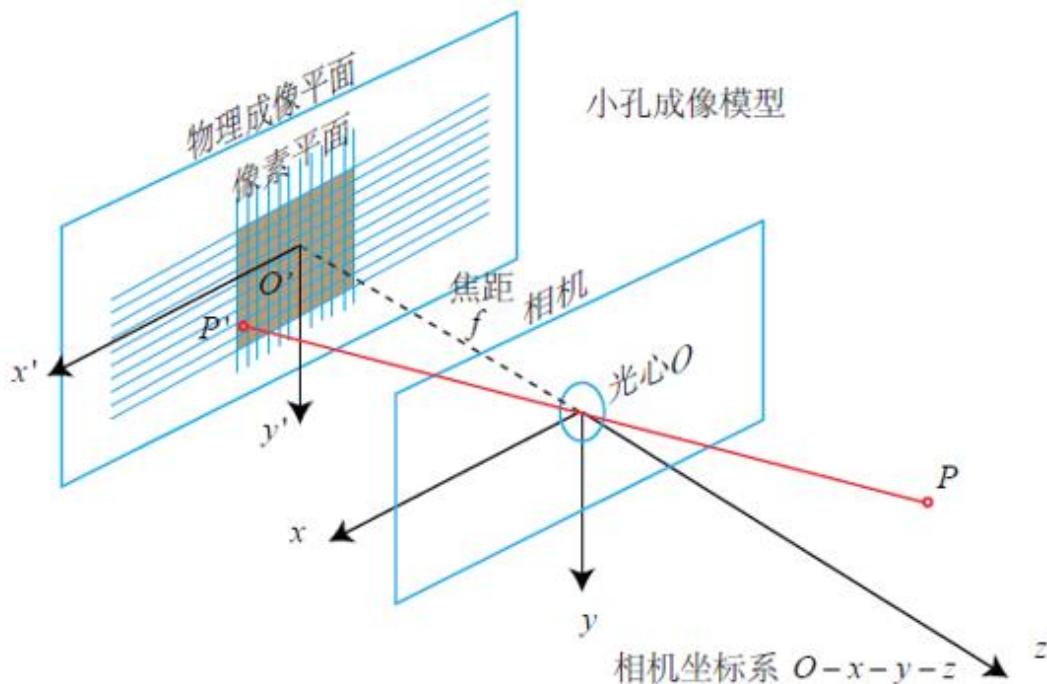
# 视觉传感器

CCD 的三层结构：上：增光镜片、中：色块网格  
下：感应线路



# 1. 相机模型

- 小孔成像模型



原始形式  $\frac{Z}{f} = -\frac{X}{X'} = -\frac{Y}{Y'}.$

翻转到O点前面

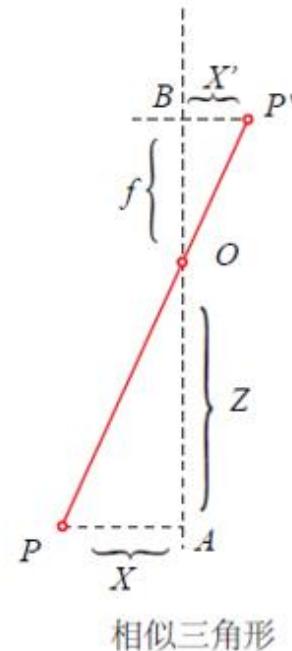
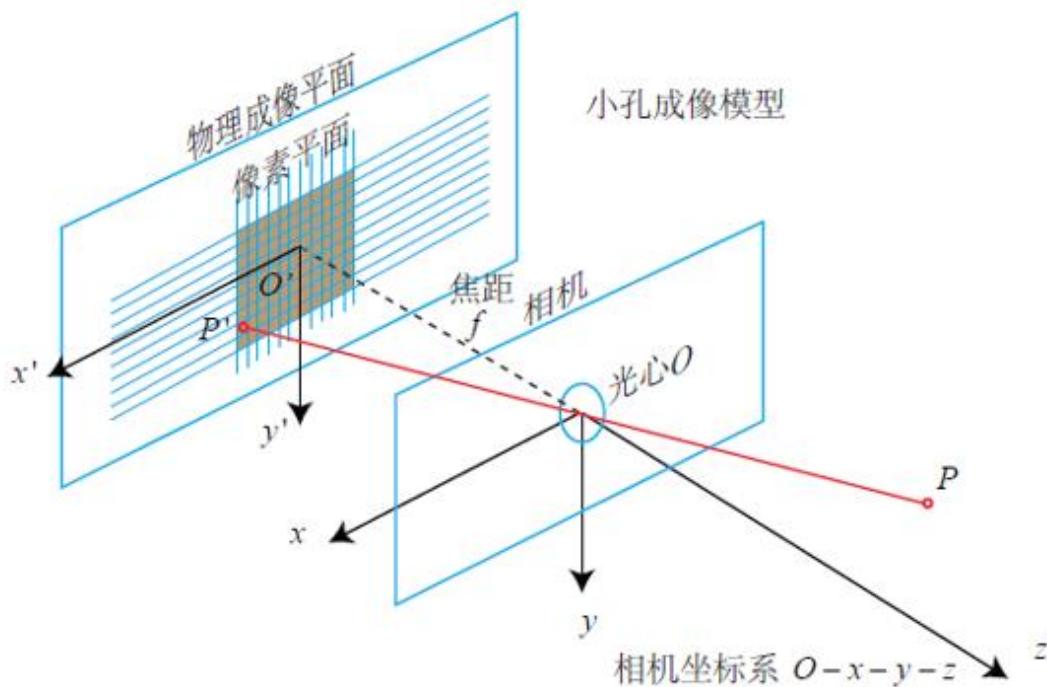
$$\frac{Z}{f} = \frac{X}{X'} = \frac{Y}{Y'}.$$

整理之：

$$X' = f \frac{X}{Z}$$
$$Y' = f \frac{Y}{Z}$$

# 1. 相机模型

- 小孔成像模型



- 成像平面到像素坐标 归一化变换

$$\begin{cases} u = \alpha X' + c_x \\ v = \beta Y' + c_y \end{cases} \quad \text{代入} \quad \begin{cases} X' = f \frac{X}{Z} \\ Y' = f \frac{Y}{Z} \end{cases} \quad \text{得} \quad \begin{cases} u = f_x \frac{X}{Z} + c_x \\ v = f_y \frac{Y}{Z} + c_y \end{cases}$$

# 1. 相机模型

- 小孔成像模型

展开形式 
$$\begin{cases} u = f_x \frac{X}{Z} + c_x \\ v = f_y \frac{Y}{Z} + c_y \end{cases}.$$

齐次坐标写法

矩阵形式 
$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \frac{1}{Z} \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \triangleq \frac{1}{Z} KP.$$
       $Z \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \triangleq KP.$

中间矩阵称为内参数K

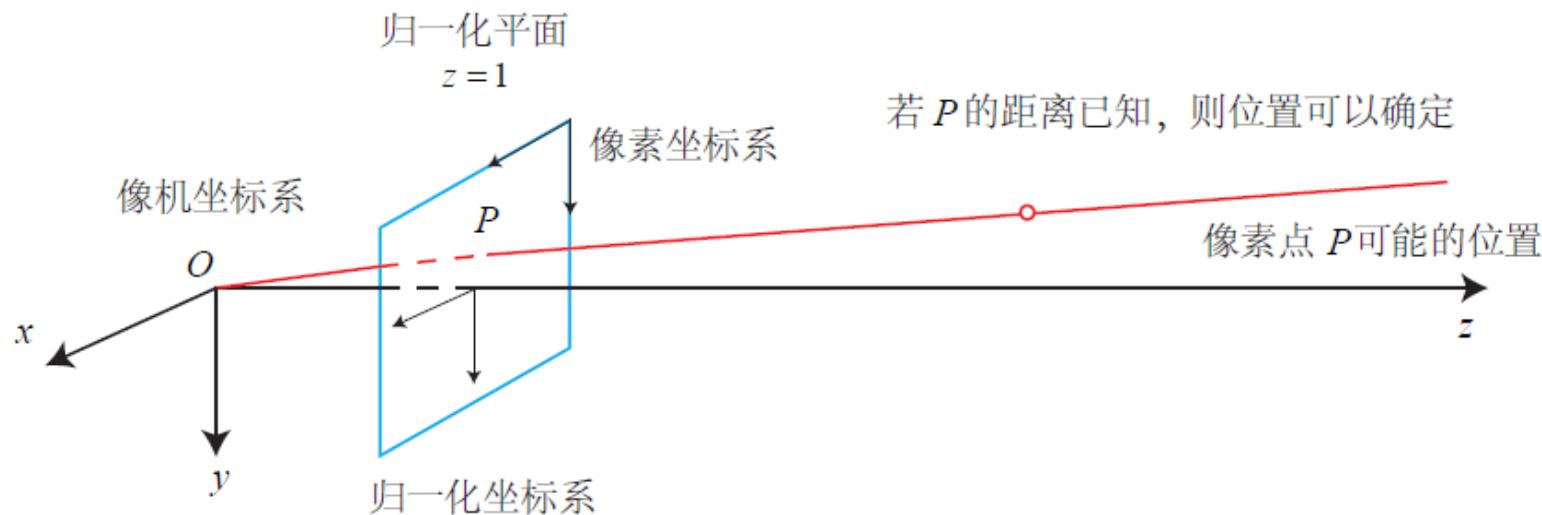
内参通常在相机生产之后就已固定

# 1. 相机模型

- 小孔成像模型

$$Z \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \triangleq KP.$$

同一直线上的投影点仍是同一个



# 回顾 视觉坐标系统

- ❖ 摄像机坐标, 场景坐标:



摄像机坐标



场景坐标

# 1. 相机模型

- 小孔成像模型
- ❖ 除内参外，相机坐标系与世界坐标系还相差一个变换：

$$ZP_{uv} = Z \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K(RP_w + t) = KTP_w.$$

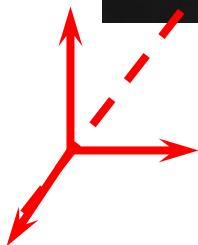
先把P从世界坐标变到  
相机坐标系下

这里 $P_w$  是点P的世界坐标

- ❖ 旋转矩阵R, 平移向量t 组成的 T 是相机当前姿态，  
称为外参

# 回顾 视觉坐标系统

❖ 视觉坐标系统：  
摄像机坐标, 场景坐标:

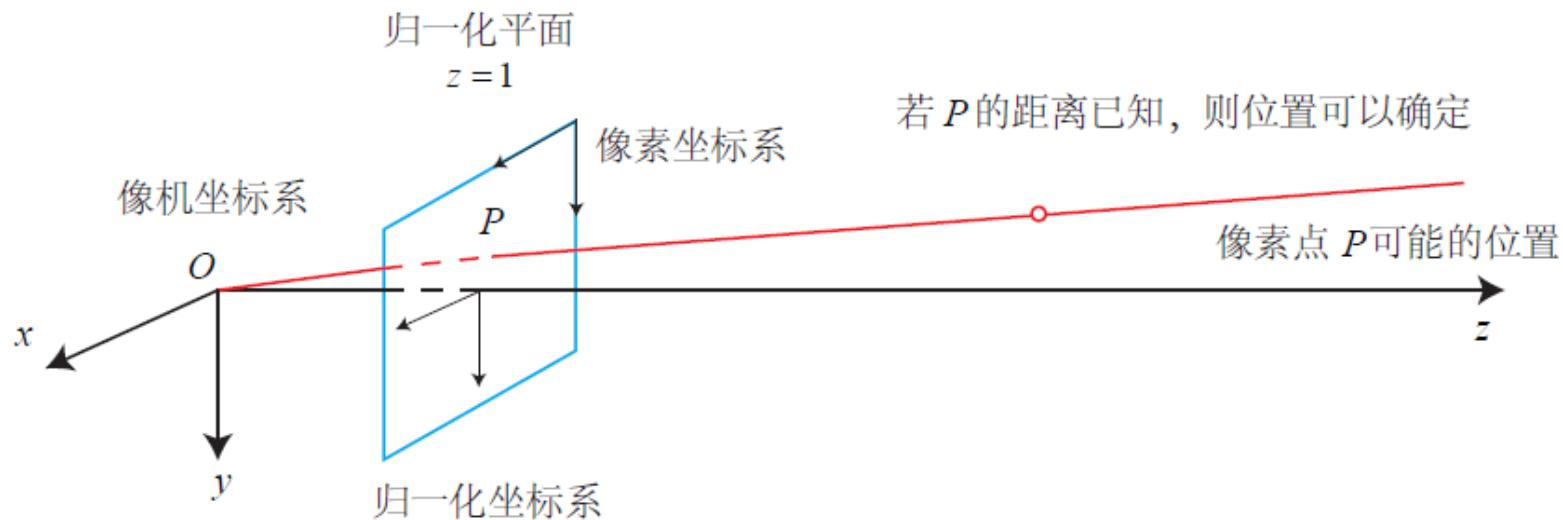


|    |     |     |     |     |     |     |    |    |    |    |    |
|----|-----|-----|-----|-----|-----|-----|----|----|----|----|----|
| 32 | 113 | 251 | 252 | 252 | 252 | 252 | 31 | 30 | 30 | 29 | 29 |
|----|-----|-----|-----|-----|-----|-----|----|----|----|----|----|



# 1. 相机模型

- 小孔成像模型
- ❖ 投影顺序: 世界——相机——归一化平面——像素



# 1. 相机模型

- 小孔成像模型
- ❖ 畸变
  - 针孔前的镜头会引入畸变



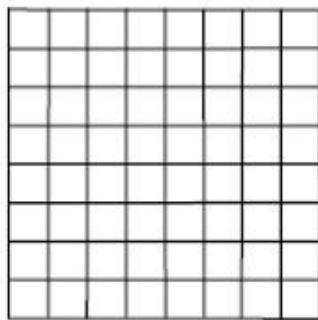
广角镜头畸变



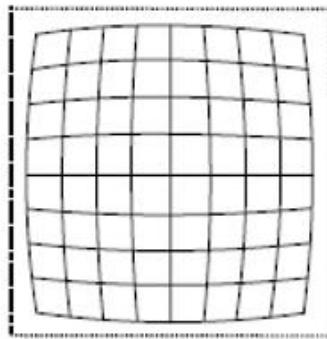
鱼眼镜头畸变

# 1. 相机模型

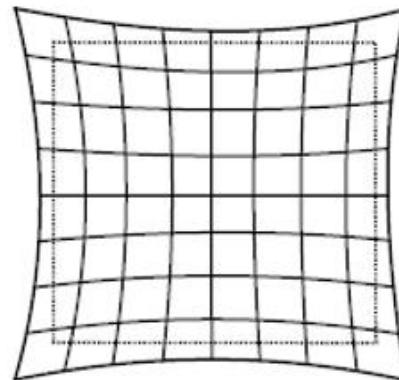
- 小孔成像模型
- ❖ 主要的畸变类型: 径向畸变



正常图像



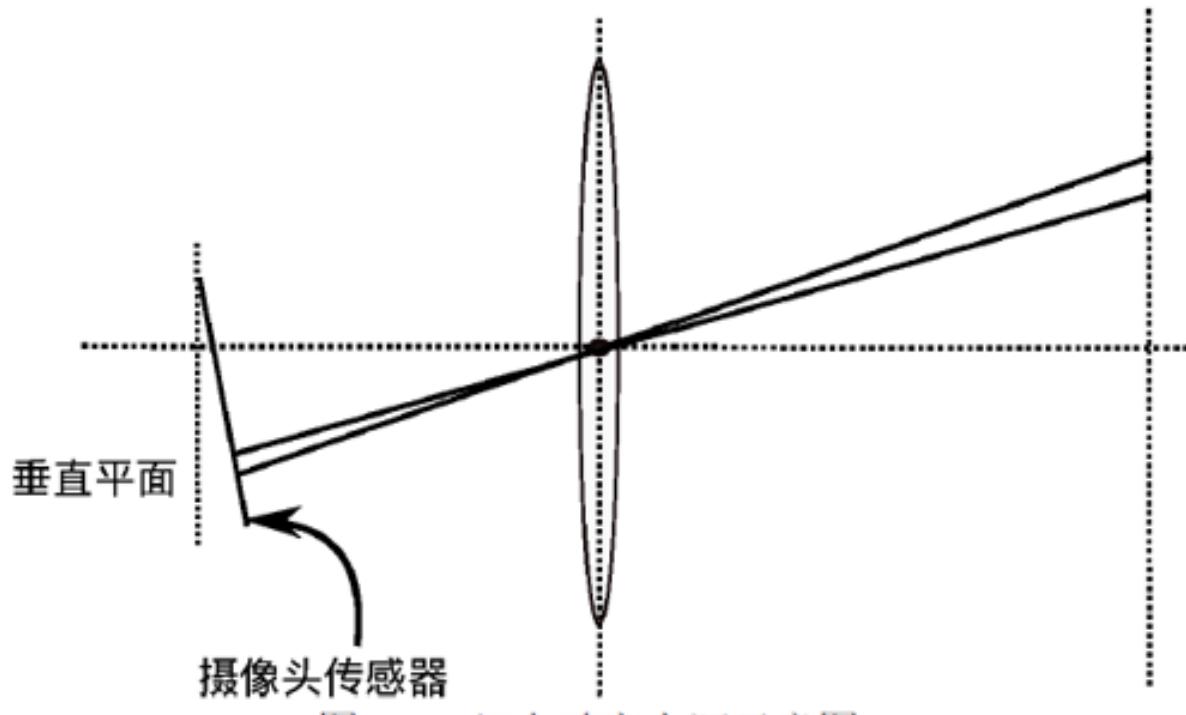
桶形失真



枕形失真

# 1. 相机模型

- 小孔成像模型
- ❖ 主要的畸变类型: 切向畸变



# 1. 相机模型

- 小孔成像模型
- ❖ 数学模型
  - 畸变可以用归一化坐标的变换来描述

$$\begin{aligned}x_{distorted} &= x \left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right) & x_{distorted} &= x + 2p_1 xy + p_2 \left(r^2 + 2x^2\right) \\y_{distorted} &= y \left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right) & y_{distorted} &= y + p_1 \left(r^2 + 2y^2\right) + 2p_2 xy\end{aligned}$$

径向畸变: 多项式描述

切向畸变: 多项式描述

放在一起:

$$\begin{cases} x_{distorted} = x \left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right) + 2p_1 xy + p_2 \left(r^2 + 2x^2\right) \\ y_{distorted} = y \left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right) + p_1 \left(r^2 + 2y^2\right) + 2p_2 xy \end{cases}$$

实际当中可灵活保留各项系数

# 1. 相机模型

---

- 小孔成像模型小结

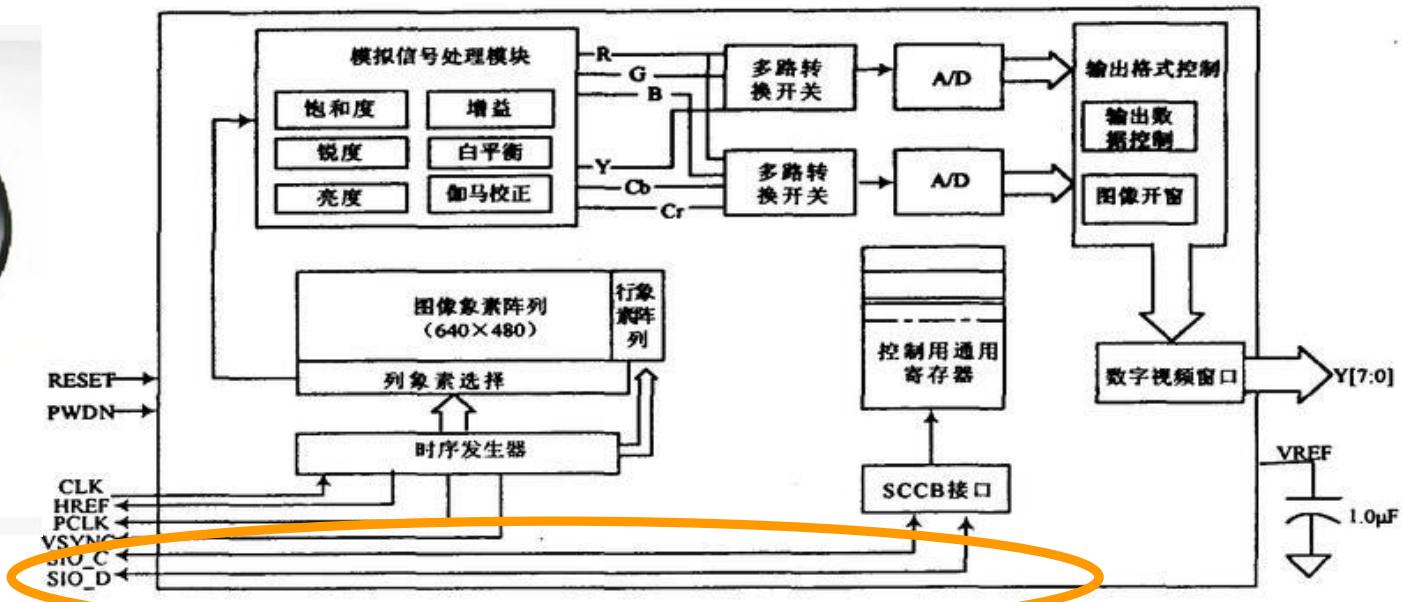
1. 首先，世界坐标系下有一个固定的点  $P$ ，世界坐标为  $\mathbf{P}_w$ ；
2. 由于相机在运动，它的运动由  $\mathbf{R}, \mathbf{t}$  或变换矩阵  $\mathbf{T} \in SE(3)$  描述。 $P$  的相机坐标为：  
$$\tilde{\mathbf{P}}_c = \mathbf{R}\mathbf{P}_w + \mathbf{t}.$$
3. 这时的  $\tilde{\mathbf{P}}_c$  仍有  $X, Y, Z$  三个量，把它们投影到归一化平面  $Z = 1$  上，得到  $P$  的归一化相机坐标：  
$$\mathbf{P}_c = [X/Z, Y/Z, 1]^T \text{①}.$$
4. 最后， $P$  的归一化坐标经过内参后，对应到它的像素坐标：  
$$\mathbf{P}_{uv} = \mathbf{K}\mathbf{P}_c.$$

# 视觉传感器

## ❖ CMOS图像传感器 OV7648

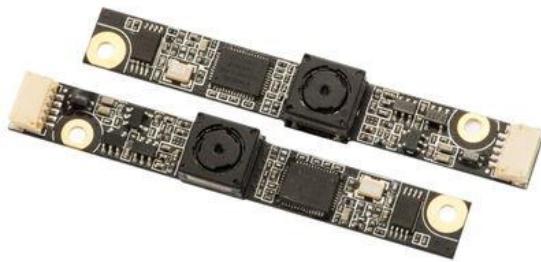
I<sup>2</sup>C控制，标准SCCB接口

芯片内共有128个寄存器，其中41个由用户根据需要通过SCCB接口设置，实现对象素、帧率、曝光控制、白平衡、增益控制、亮度控制等

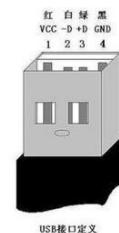


# 视觉传感器

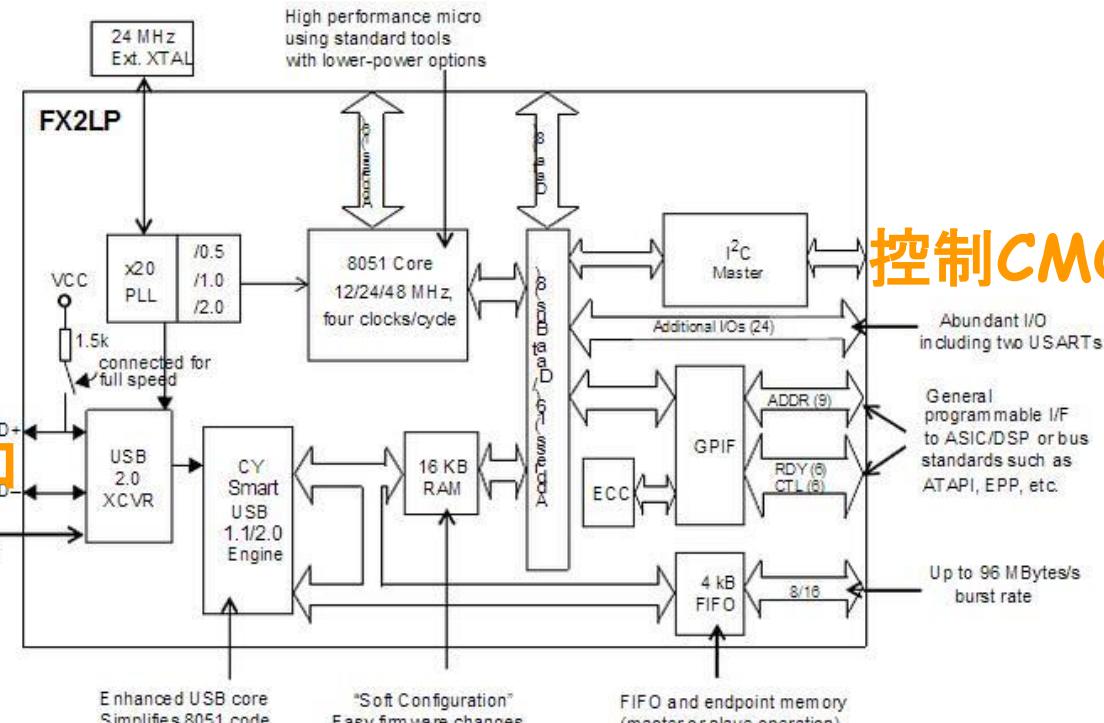
## ❖ CMOS图像传感器 扩展为USB摄像头



USB接口



Integrated full speed and high speed XCVR



USB控制器CY7C68013A

控制CMOS Se

## 3.1 数字图像概念

### ❖ 视觉系统的构成



Kinect

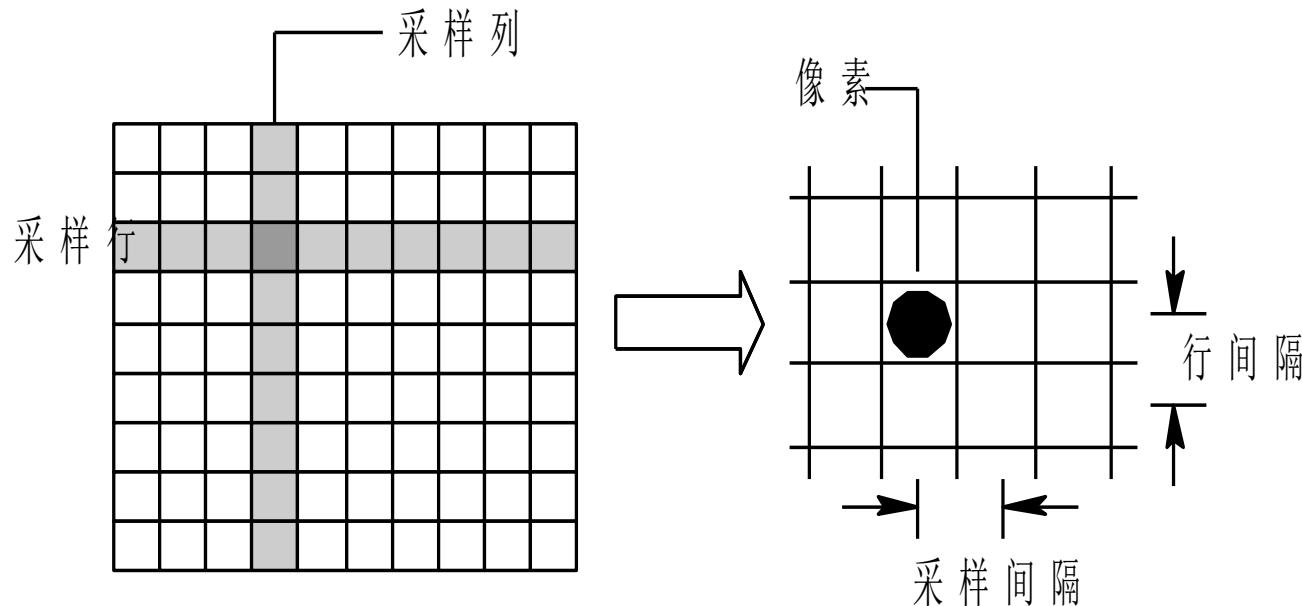


DJI Guidance



## 3.1 数字图像概念

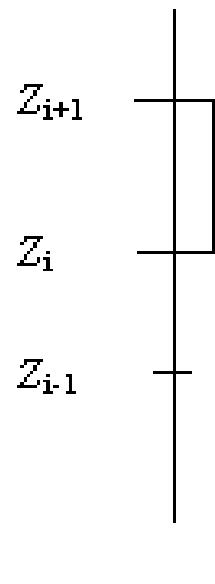
- ❖ 数字图像：用二维阵列表示的影像，阵列元素表示亮度值或灰度值



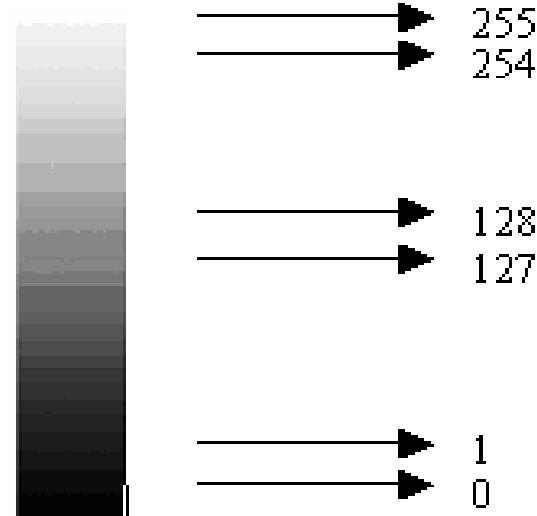
$$f(x, y) = \begin{bmatrix} f(0,0) & f(0,1) & \dots & f(0, N-1) \\ f(1,0) & f(1,1) & \dots & f(1, N-1) \\ \vdots & & & \\ f(M-1,0) & & & f(M-1, N-1) \end{bmatrix}$$

## 3.1 数字图像概念

- 把采样后所得的各像素灰度值从模拟量到离散量的转换称为图像灰度的量化。



(a)



(b)

(a) 量化

(b) 量化为8 bit

## 3.1 数字图像概念

- ❖ 数字图像:用二维阵列表示的影像,阵列元素表示亮度值或灰度值

## 3.1 数字图像概念

量化级数与图像质量之间的关系

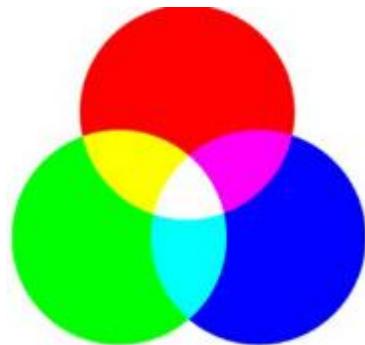


- (a) 量化为2级的Lena图像
- (b) 量化为16级的Lena图像
- (c) 量化为256级的Lena图像

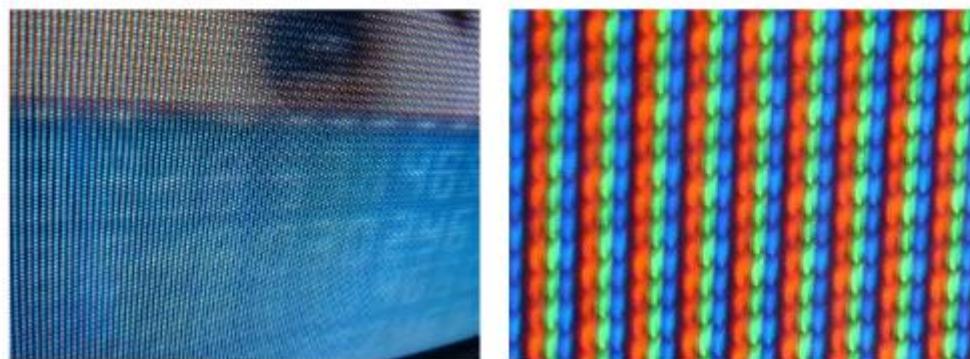
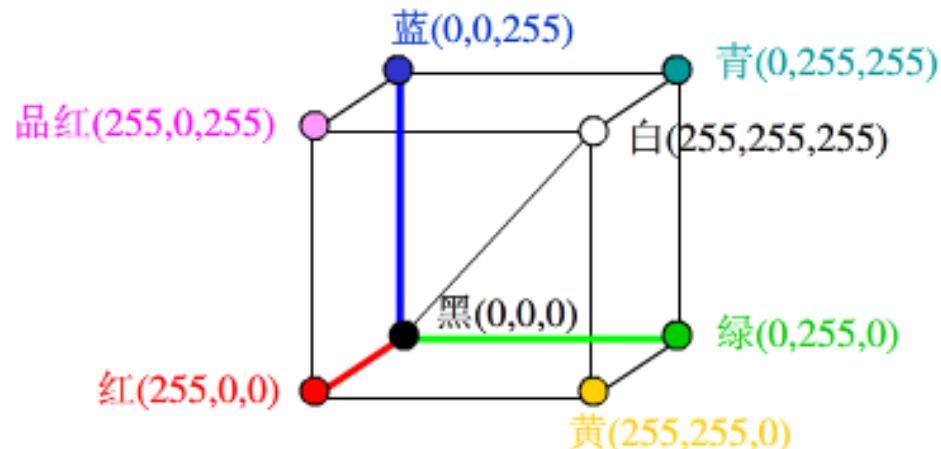
## 3.1 数字图像概念

### ❖ 图像的颜色

RGB：光混合配色体系



RGB



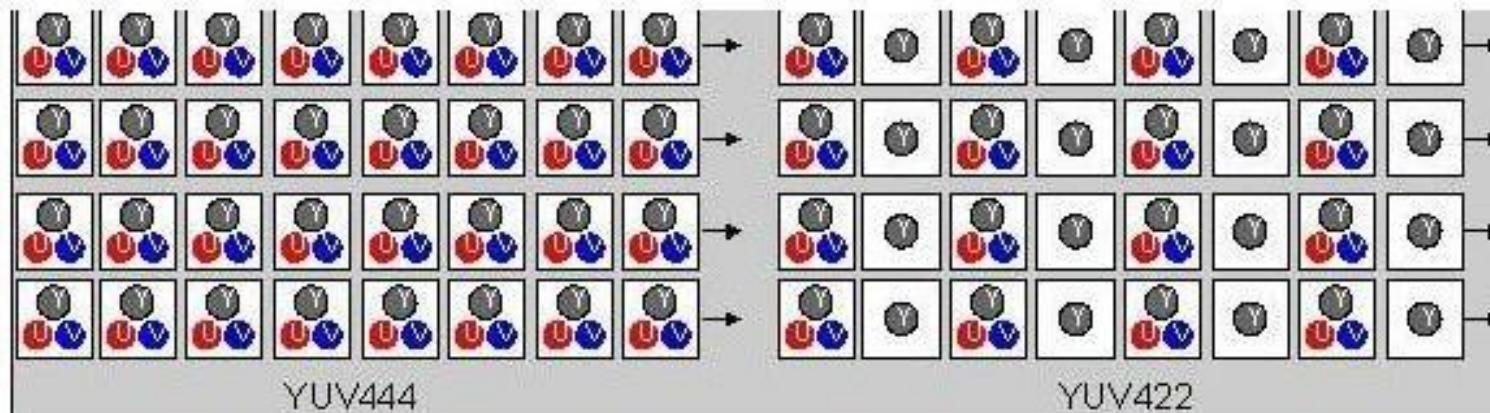
显示器及放大视图

## 3.1 数字图像概念

### ❖ 图像的颜色

**YUV**: 复合颜色视频标准，其实是CCD感光后，RGB再经过矩阵变换电路得到亮度信号Y，再取亮度Y和色差信号(R-Y), (B-Y)，最后发送端将亮度和色差三个信号分别进行编码，用同一信道发送出去。

采用YUV色彩空间的重要性是它的亮度信号Y和色度信号U、V是分离的。如果只有 Y信号分量而没有U、V信号分量，那么这样表示的图像就是黑白灰度图像。



# 3.1 数字图像概念

## ❖ 图像的颜色

YUV：采用YUV色彩空间的重要性是它的亮度信号Y和色度信号U、V是分离的。如果只有Y信号分量而没有U、V信号分量，那么这样表示的图像就是黑白灰度图像。



色差分量(Component)接口采用YPbPr和YCbCr两种标识，前者表示逐行扫描色差输出，后者表示隔行扫描色差输出；  
色差分量接口一般利用3根信号线分别传送亮色和两路色差信号。

# 3.1 数字图像概念

图像RAW格式： YUYV和RGB

$$Y = 0.299R + 0.587G + 0.114B$$

$$U = -0.147R - 0.289G + 0.436B$$

$$V = 0.615R - 0.515G - 0.100B$$

$$R = Y + 1.14V$$

$$G = Y - 0.39U - 0.58V$$

$$B = Y + 2.03U$$



(a) 原RGB图像



(b) Y分量



(c) U分量

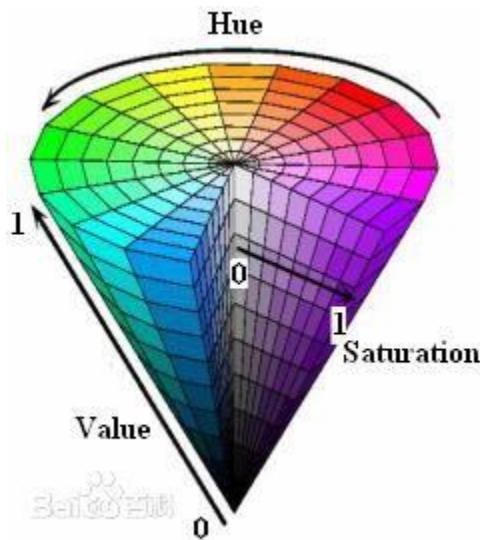


(d) V分量

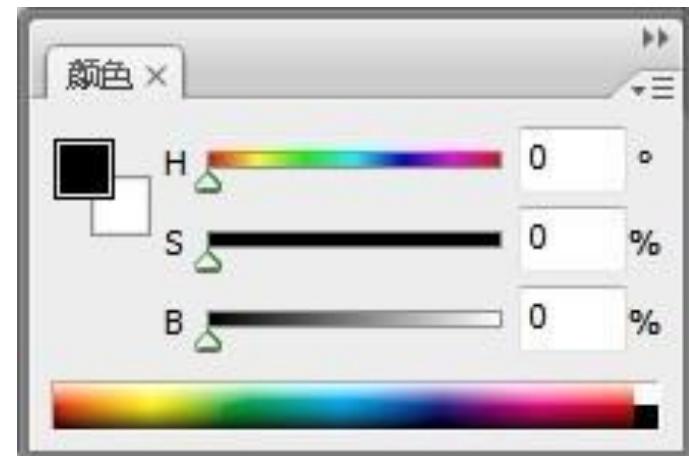
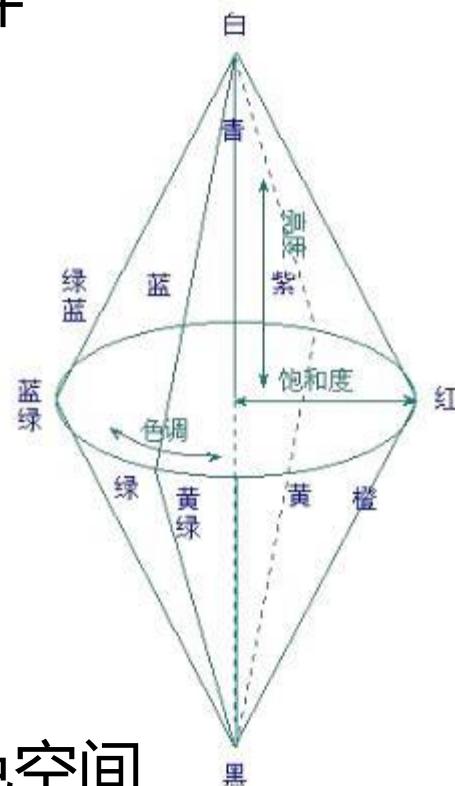
# 3.1 数字图像概念

扩展知识： 颜色显示空间 HSI HSV(HSB)

为色度H 饱和度S 亮度空间V 圆坐标为彩虹颜色排列，垂直线为亮度数值，同心圆层为饱和度，饱和度与白光含量成反比，更符合人眼的颜色感觉和解释，用于印刷品等，常见于 Photoshop等软件



HSV颜色空间



# 3.1 数字图像概念

扩展知识： 颜色显示空间 HSI HSV(HSB)



(a) 原**RGB**图像



(b) *H*分量



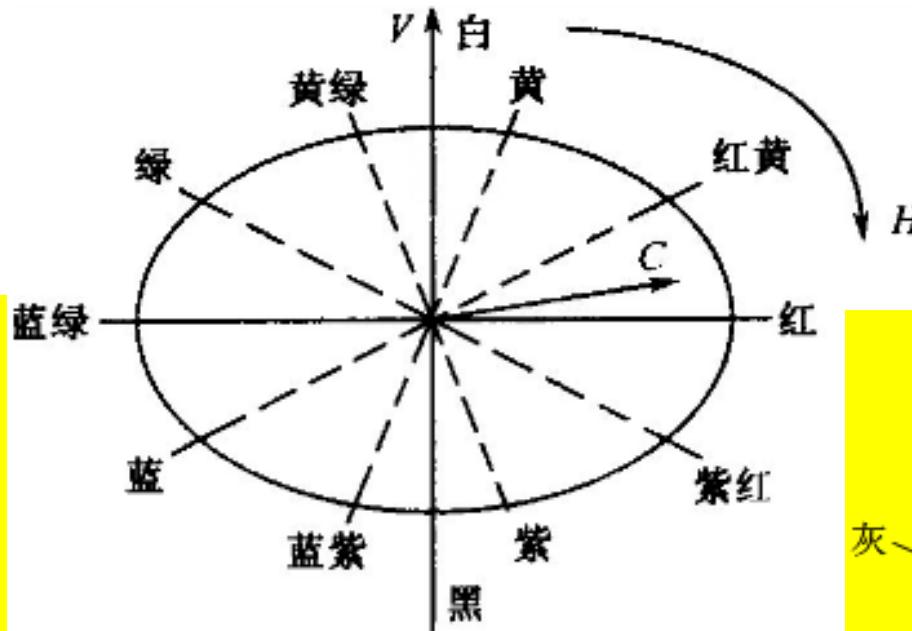
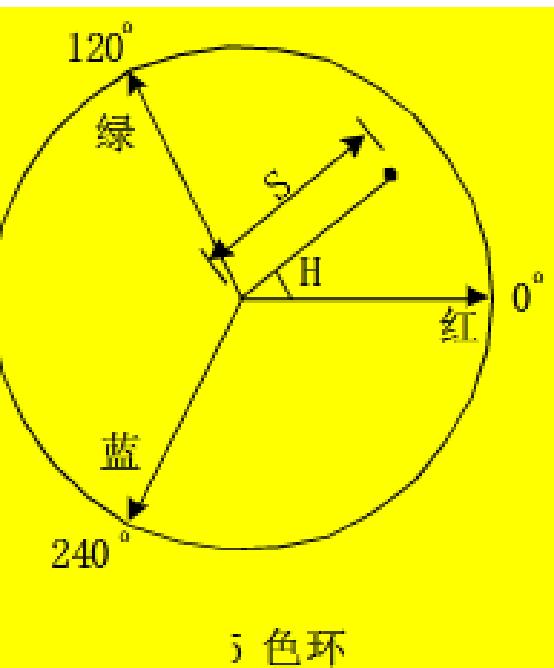
(c) *S*分量



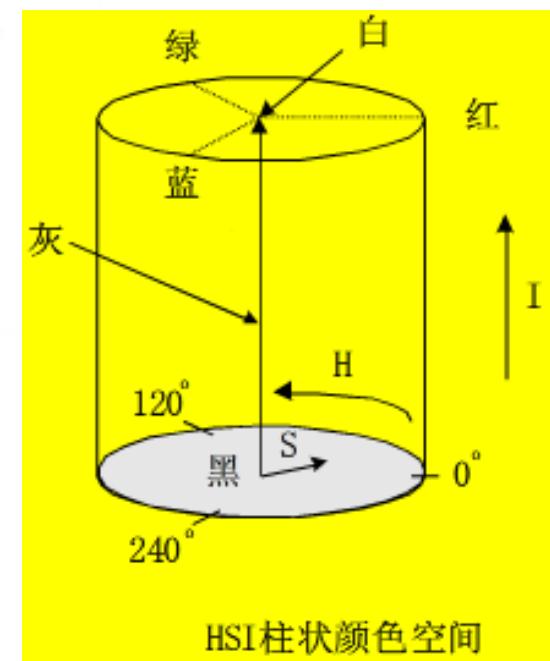
(d) *V*分量

# 3.1 数字图像概念

## ❖ 图像的颜色



HSI 颜色空间



HSI柱状颜色空间

## 3.1 数字图像概念

### ❖ 图像的存储

- 格式(表示格式和文件格式)

- (1) 矢量格式

- 用线段或线段的组合体来表示图像(WMF)

- (2) 光栅格式

- 用许多像素点的集合来表示图像

- BMP格式, GIF格式, TIFF格式, JPEG 格式

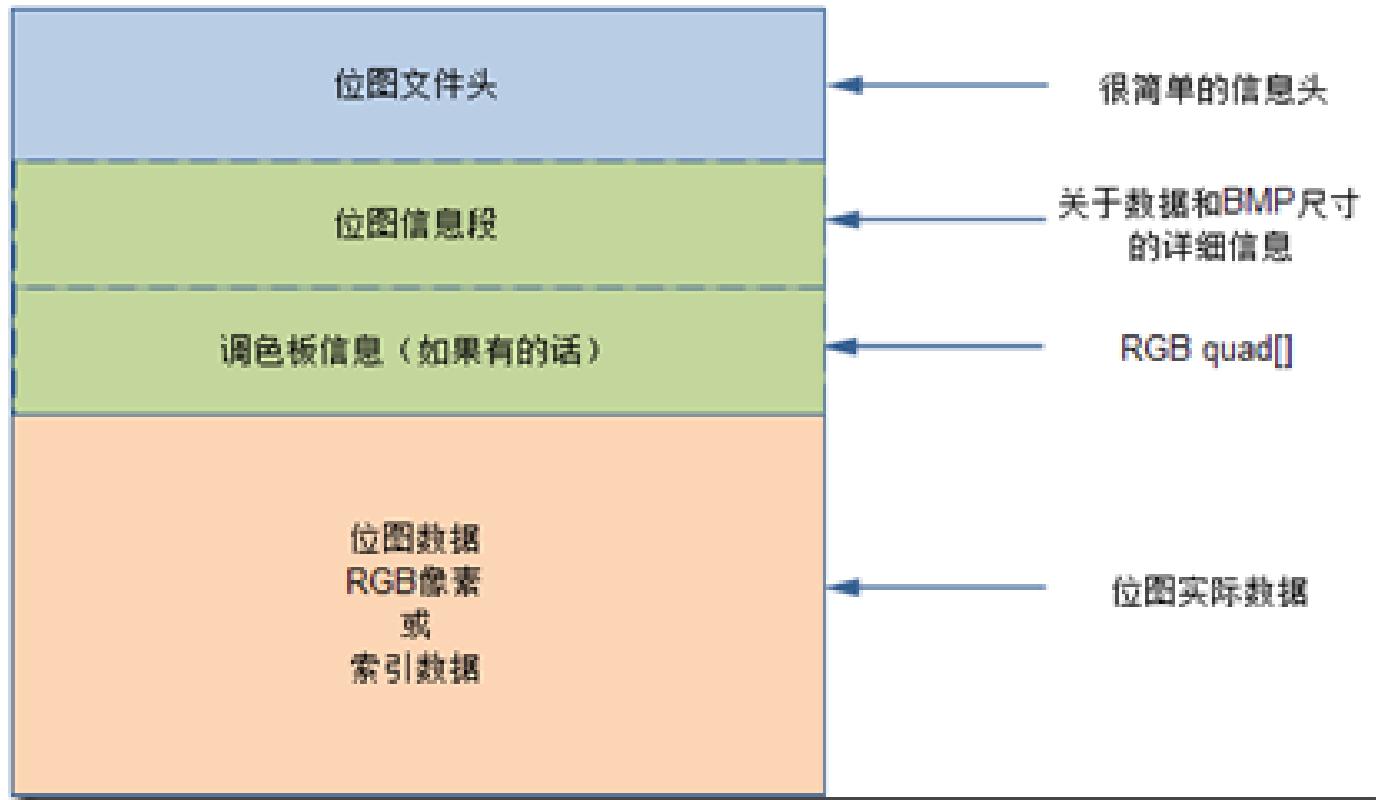
- ◆ 每一种图像文件均有一个文件头，在文件头之后才是图像数据。

- ◆ 文件头的内容一般包括文件类型、文件制作者、制作时间、版本号、文件大小等内容。

# 3.1 数字图像概念

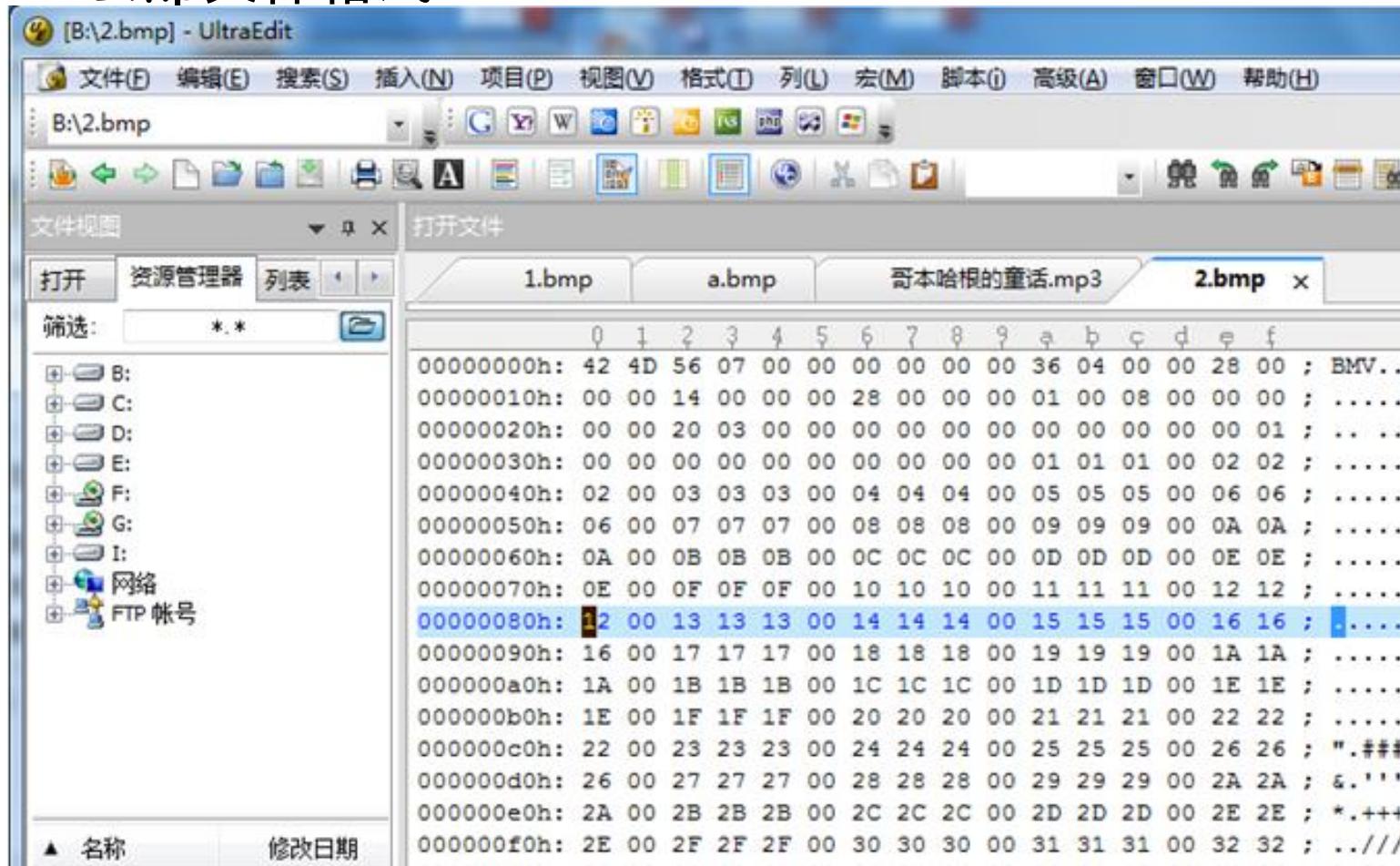
## ❖ 图像的存储

- Bmp文件格式



# 3.1 数字图像概念

- ❖ 图像的存储
  - Bmp文件格式

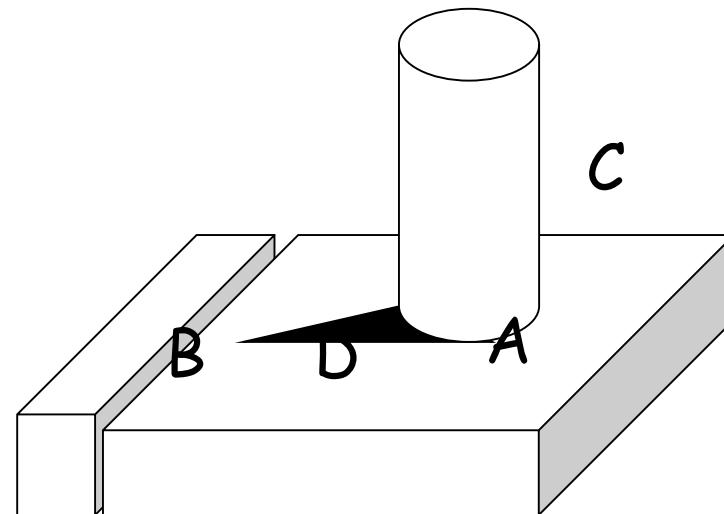


用Ultraedit打开一副Bmp文件，查看数值

## 3.1 数字图像概念

### ❖ 图像的信息

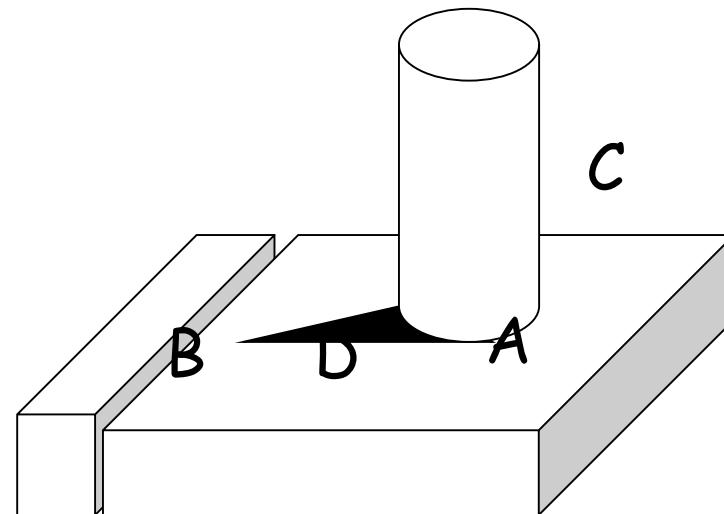
- A: 区域轮廓
- B: 立体视觉
- C: 图像语义(物体属性)
- D: 知识关联(联想)
- E: 行为主义的对象学习
- F: .....



## 3.1 数字图像概念

### ❖ 图像的边缘

- A: 空间曲面的不连续点
- B: 不同材料或不同颜色
- C: 物体与背景的分界线
- D: 阴影引起的边缘



边缘点:亮度显著变化的点

边缘段:边缘点坐标和方向的总和

边缘检测器:抽取边缘集合的算法

边缘跟踪:确定轮廓的搜索过程

## 3.1 数字图像概念

### ❖ 图像的边缘

- 阶跃边缘



- 线条边缘

## 3.1 数字图像概念

### ❖ 图像的边缘



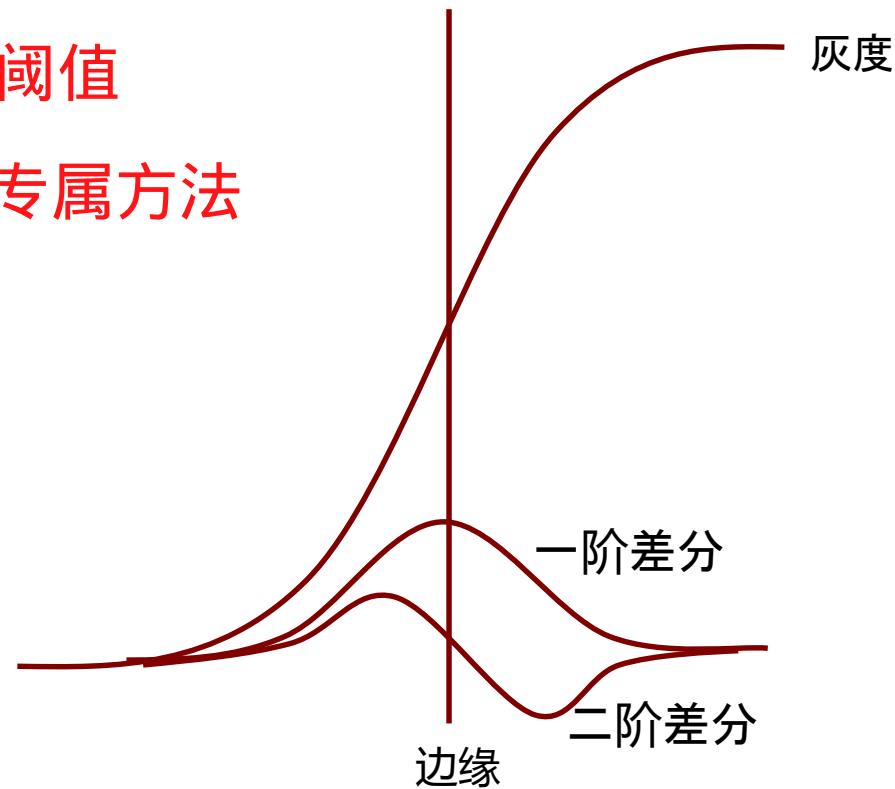
|    |     |     |     |     |     |     |    |    |    |    |    |
|----|-----|-----|-----|-----|-----|-----|----|----|----|----|----|
| 32 | 113 | 251 | 252 | 252 | 252 | 252 | 31 | 30 | 30 | 29 | 29 |
|----|-----|-----|-----|-----|-----|-----|----|----|----|----|----|

## 3.1 数字图像概念

### ❖ 图像的边缘

一阶差分极大值    需要阈值

二阶差分过零点    渐变专属方法



## 3.1 数字图像概念

❖ 视觉坐标系统：

像素坐标, 图像平面坐标:



像素坐标



像平面坐标

## 3.1 数字图像概念

❖ 视觉坐标系统：  
摄像机坐标, 场景坐标：



摄像机坐标



场景坐标

# 第三章 机器人的视觉分析

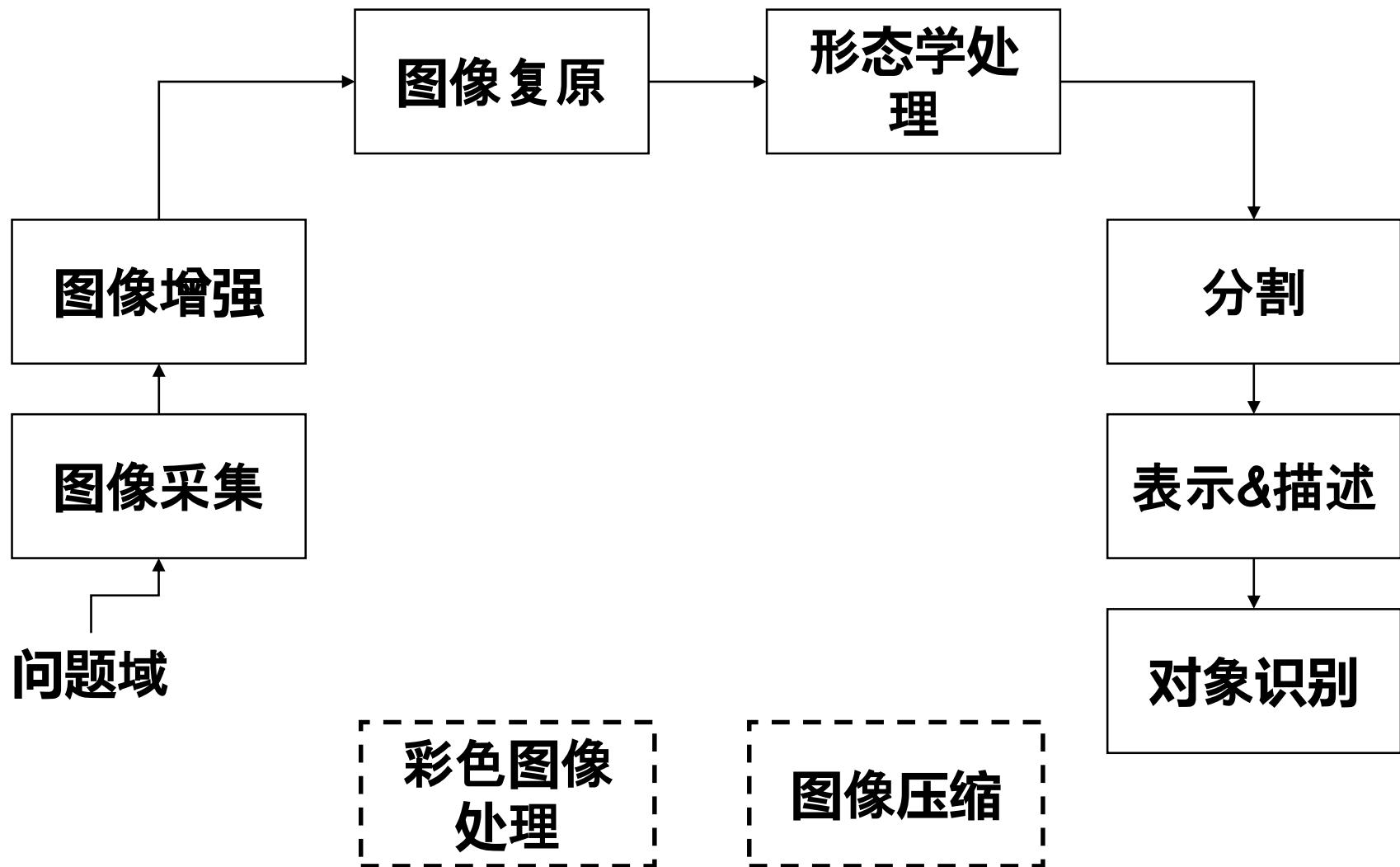
## 目标:

- ❖ 掌握图像处理的一般流程和属性
- ❖ 掌握图像滤波的原理和方法
- ❖ 了解特征分析的工具和方法

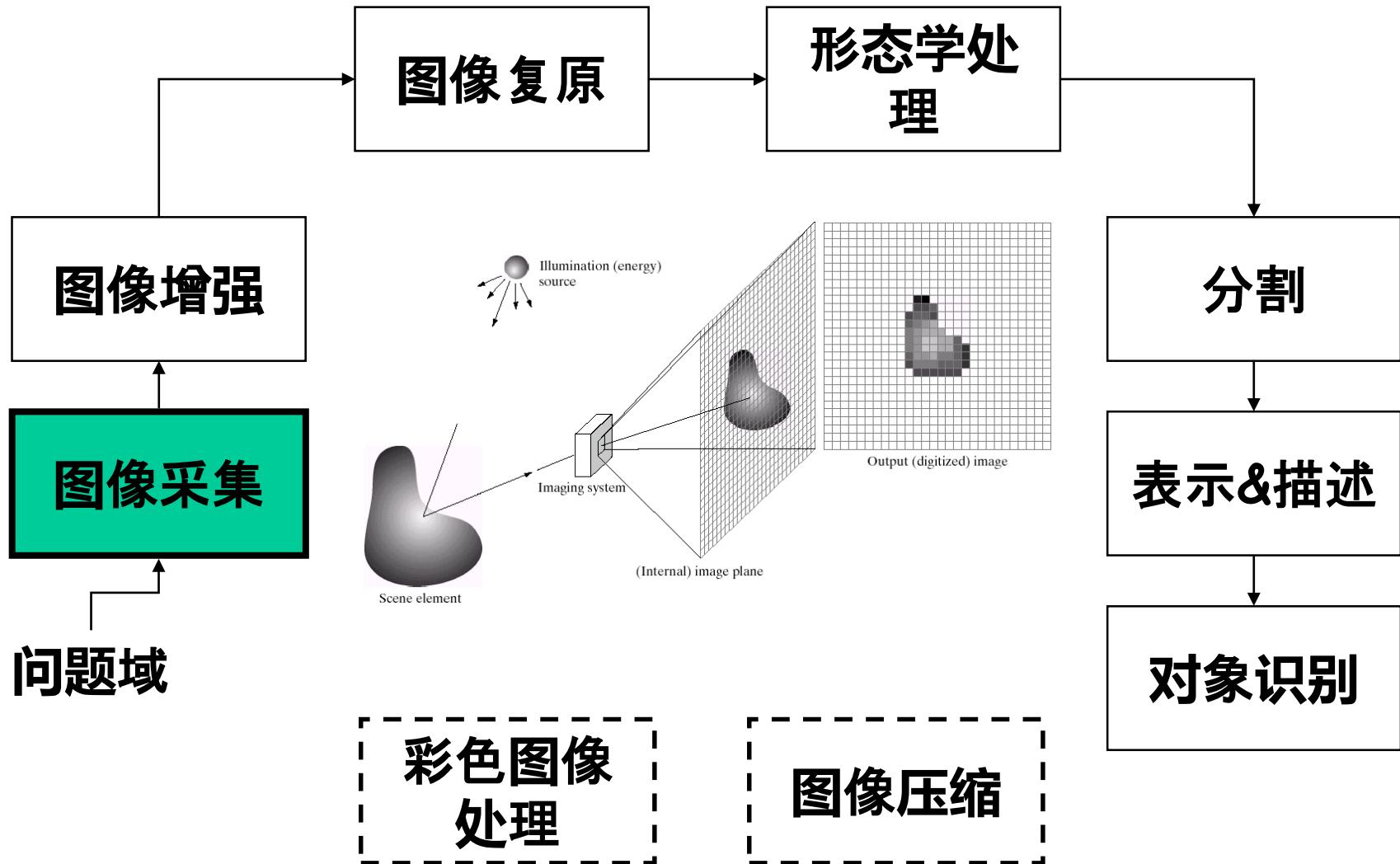
## 目录:

- ❖ 数字图像概念
- ❖ 数字图像处理基本步骤
- ❖ 图像预处理
- ❖ 图像区域表示
- ❖ 二值图像运算
- ❖ 图像特征比对
- ❖ OpenCV平台使用

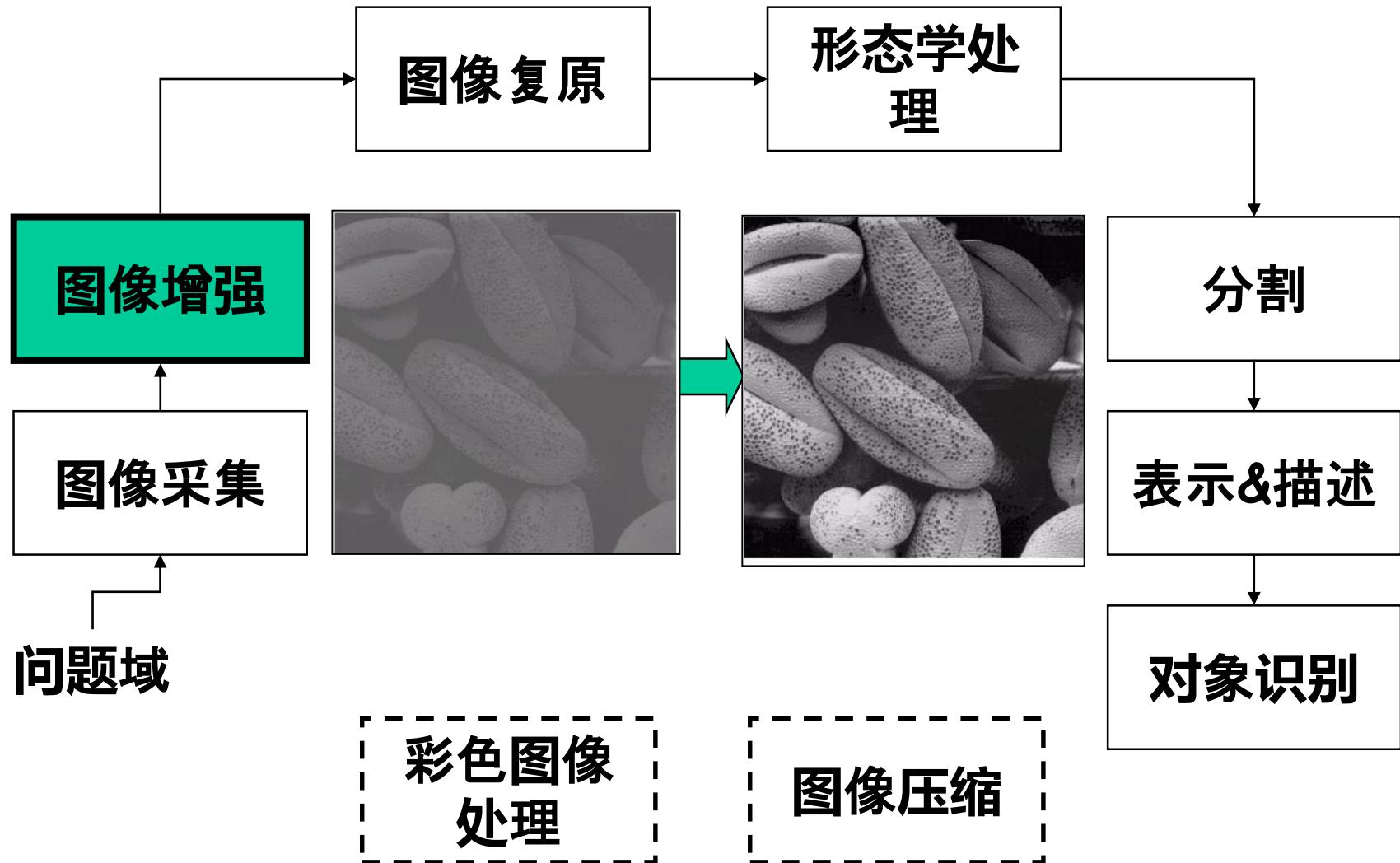
## 3.2 数字图像处理基本步骤



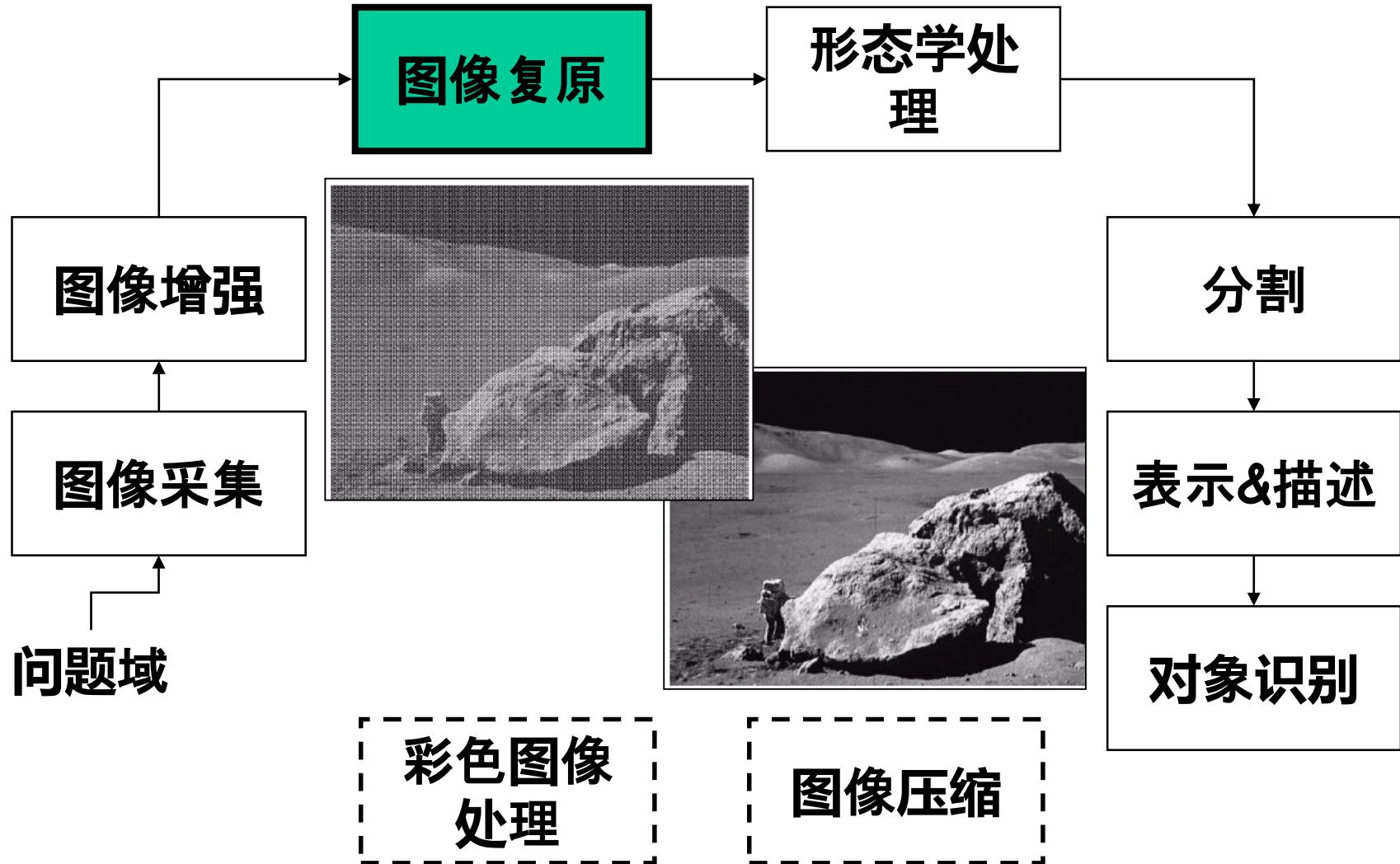
## 3.2 数字图像处理基本步骤



## 3.2 数字图像处理基本步骤



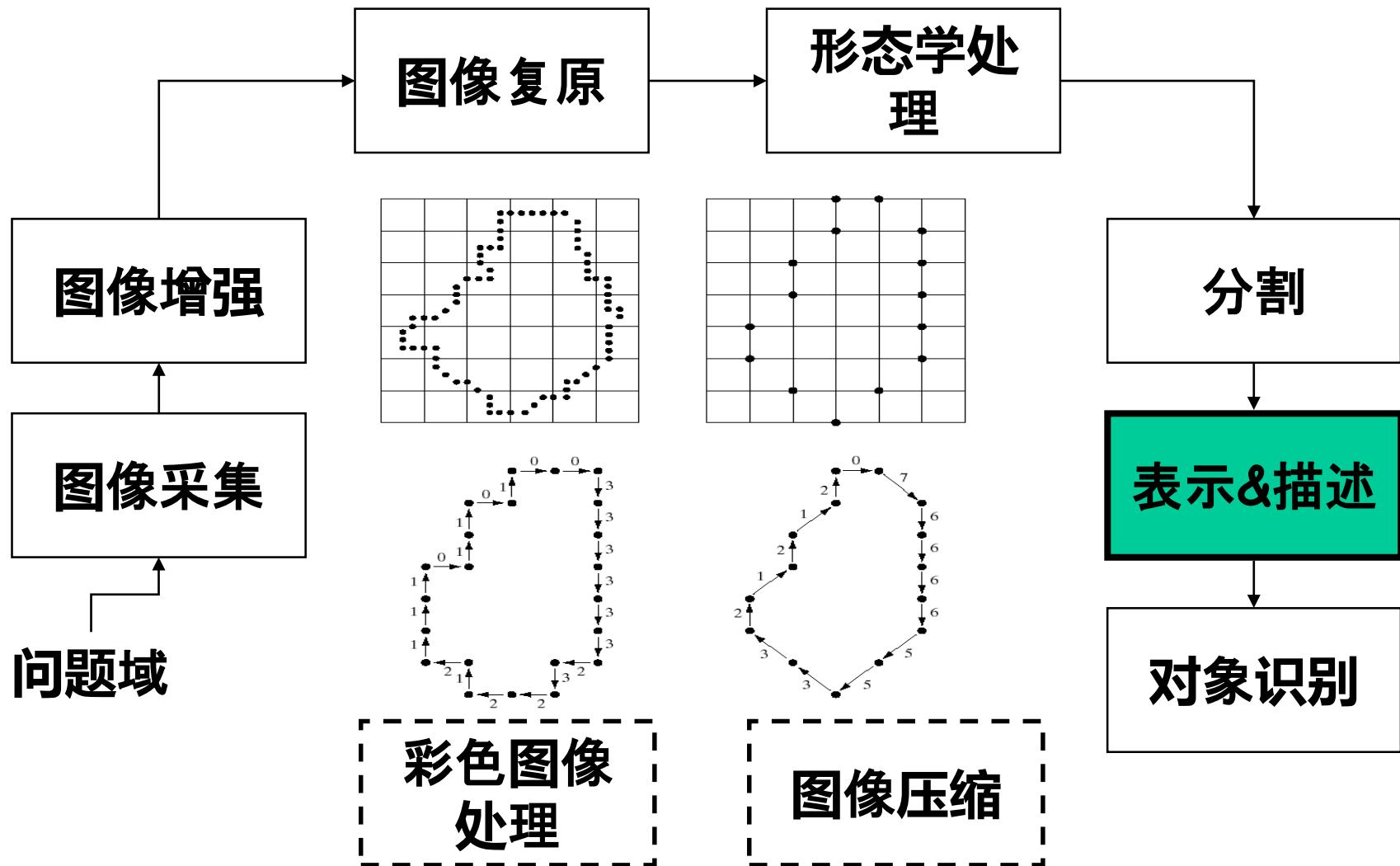
## 3.2 数字图像处理基本步骤



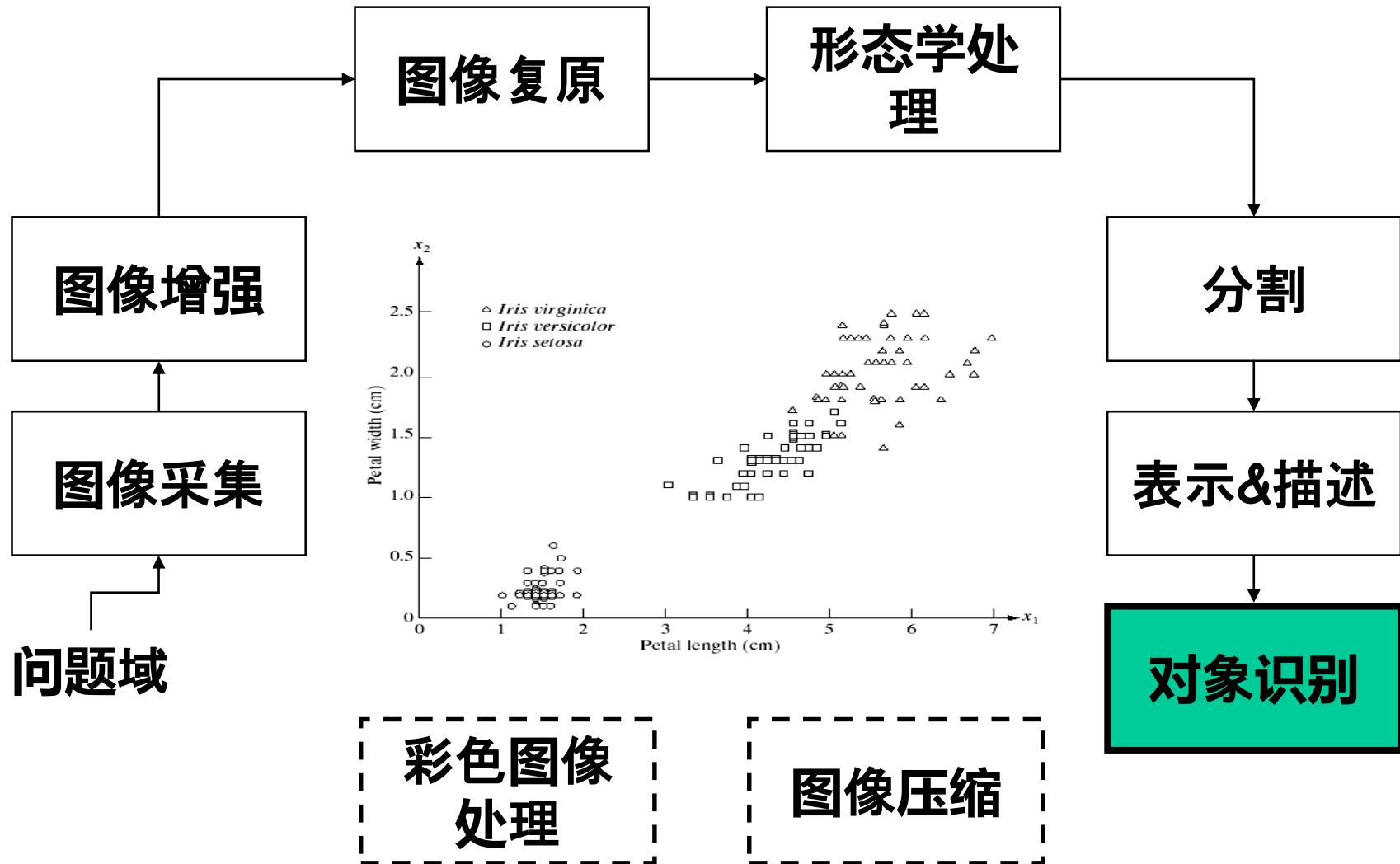
## 3.2 数字图像处理基本步骤



## 3.2 数字图像处理基本步骤



## 3.2 数字图像处理基本步骤



# Face Detection & Face Tracking

**Face Detection:**

在输入图象中确定所有人脸的位置、大小、姿态

**Face Tracking:**

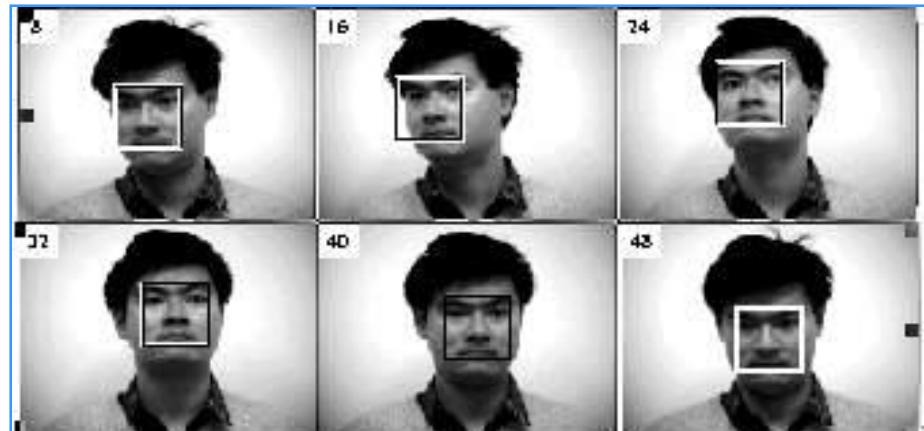
在输入图象序列中确定某个人脸的运动轨迹及大小

变化

多角度人脸检测



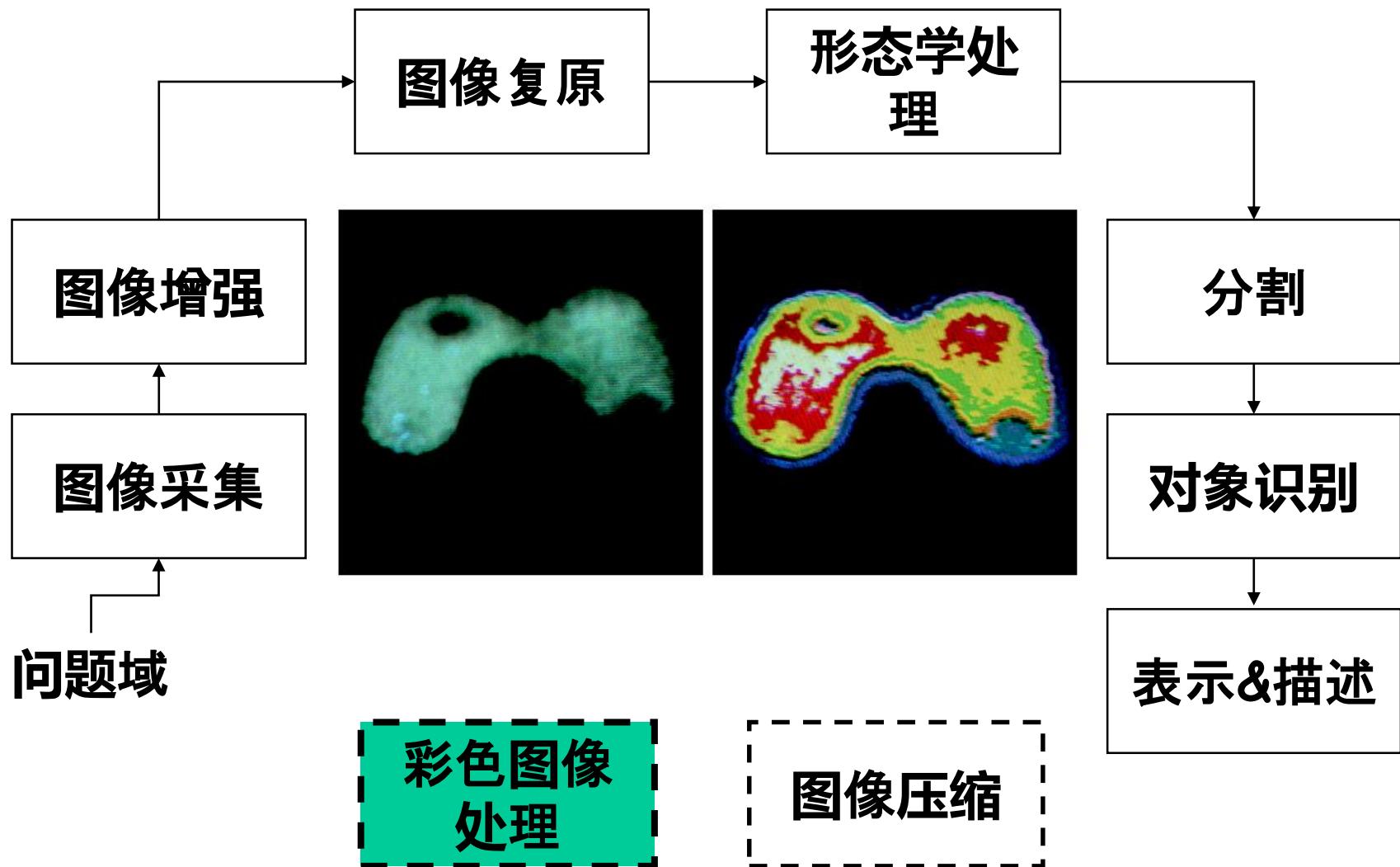
人脸跟踪



## 3.2 数字图像处理基本步骤



## 3.2 数字图像处理基本步骤



## 3.2 数字图像处理相关知识

---

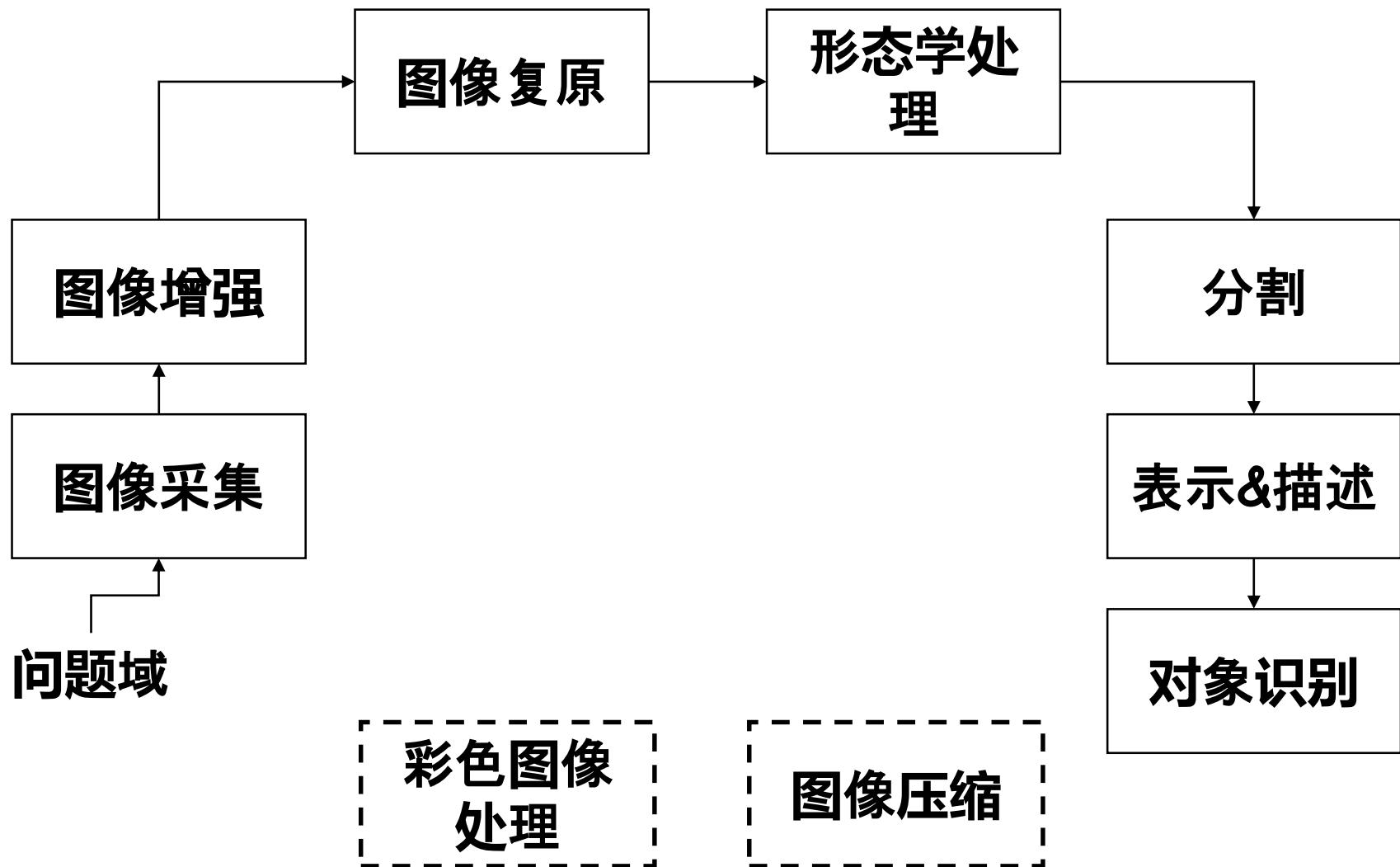
- ❖ 图像处理--> 图像分析--> 图像理解
  - ❖ 边缘、轮廓、纹理、颜色、二值图像、深度、二维、三维、运动、光流、跟踪、识别、生物特征识别.....
  - ❖ 直方图、彩色表示、数据结构、预处理、分割、特征表示、统计或结构模式识别、神经网络、模糊系统、人工智能、数学形态学、离散线性变换、图像数据压缩、最小均方误差滤波器
- .....

## 3.2 数字图像处理相关的工具

---

- ❖ 浏览图像
  - ACDSee
- ❖ 图像处理
  - Photoshop
- ❖ 应用与研究
  - MatLab OpenCV
    - ✓ 基本操作: 读写、显示、几何变换
    - ✓ 图像变换(傅立叶, 小波等)
    - ✓ 图像增强(直方图, 对比度, 平滑, 锐化)
    - ✓ 图像分割
    - ✓ 特征比对

## 3.2 数字图像处理基本步骤



# 3.4 数字图像预处理

---

- ❖ 3.4.1 概述
- ❖ 3.4.2 直方图修正
- ❖ 3.4.3 图像线性运算
- ❖ 3.4.4 线性滤波器
- ❖ 3.4.5 非线性滤波器

## 3.4.1 概述

### 1 为什么要对图像进行预处理？

- 一般情况下，获取图像是通过成像系统（光学镜头+CCD）得到；
- 成像系统本身电子干扰和外界环境产生的随机干扰，使得图像含有噪声；
- 成像系统所处的照明或其他环境，使得图像的灰度分辨率不高，甚至很难区分图像中的物体。
- 必须在视觉的初级阶段对原始图像进行灰度修正、噪声过滤等图像预处理。

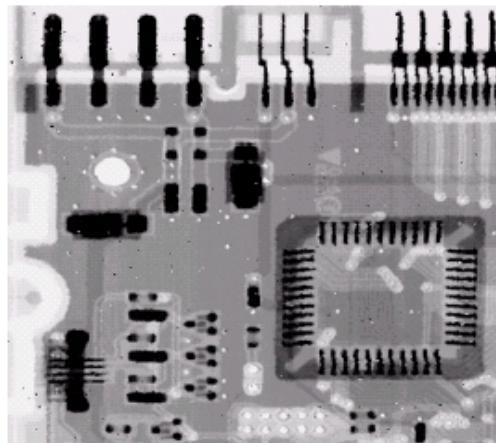
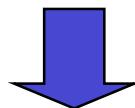
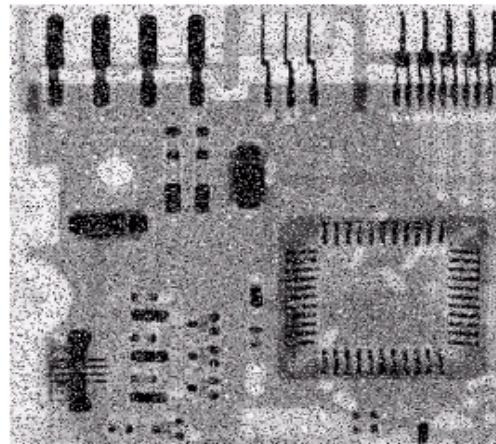
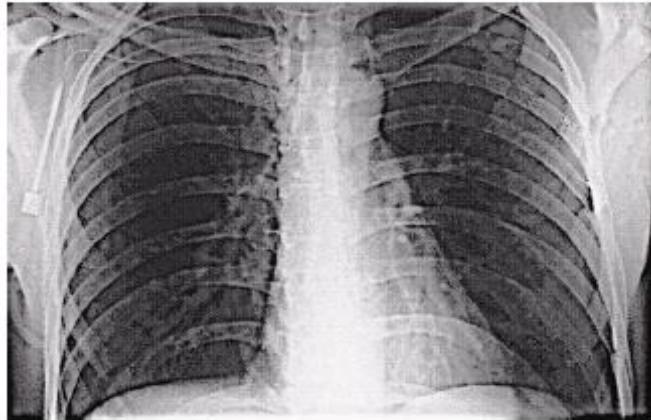
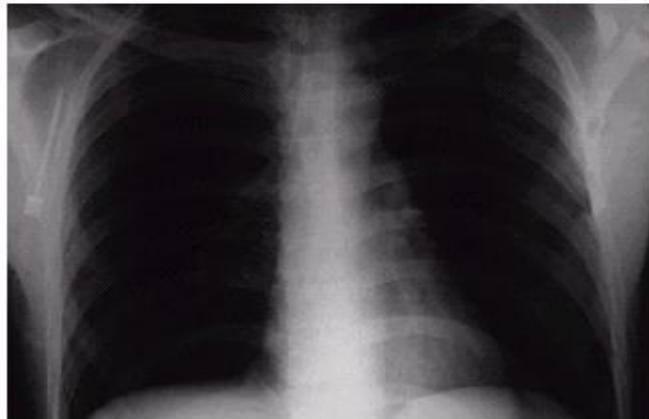
## 3.4.1 概述

### 2 图像预处理的功能是什么？

- 计算机视觉中的图像预处理并不考虑图像降质原因，**只将图像中感兴趣的特征有选择的突出，衰减其不需要的特征。**
- 图像预处理后的输出图像并不需要去逼近原始图像，这种图像预处理方法称为**图像增强**。

# 举例：图像增强

- ❖ 最常见的用处是：提高质量，消除噪音等等



## 3.4.1 概述

### 3 图像预处理有哪几种？

- 空间域法和频率域法

- ✓ 空间域法是直接在空间域对图像像素运算处理；

- ✓ 频率域法是先对图像做某种变换（DFT、DCT、DWT、K-L），然后在变换域对图像的变换值进行运算，最后将计算后的图像逆变换到空间域。

- 全局运算和局部运算

- 灰度图像和彩色图像

## 3.4.1 概述

- 直方图修正
- 图像线性运算模型
- 线性滤波器
  - ➡ 均值滤波器
  - ➡ 高斯平滑滤波器
- 非线性滤波器
  - ➡ 中值滤波器
  - ➡ 边缘保持滤波器

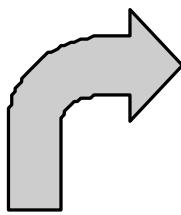
## 3.4.2 直方图修正

### 1 直方图概念

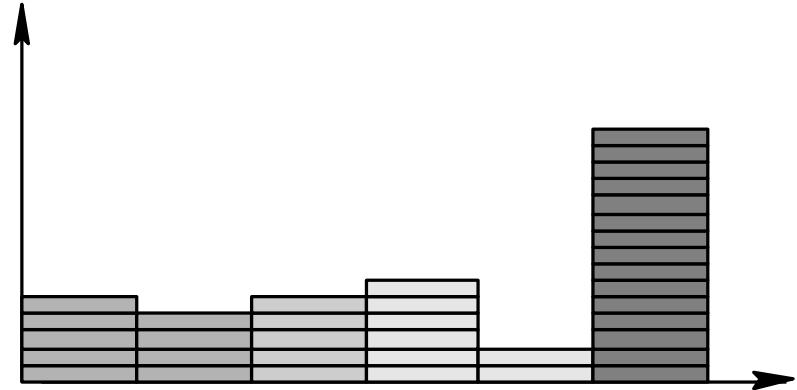
- 如果将图像中像素亮度（灰度级别）看成是一个随机变量，则其分布情况就反映了图像的统计特性，这可用Probability Density Function (PDF)来刻画和描述，表现为灰度直方图 (Histogram)。
- 灰度直方图是灰度级的函数，它表示图像中具有某种灰度级的像素的个数，反映了图像中每种灰度出现的频率，它是图像最基本的统计特征。

## 3.4.2 直方图修正

|   |   |   |   |   |   |
|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 |
| 6 | 4 | 3 | 2 | 2 | 1 |
| 1 | 6 | 6 | 4 | 6 | 6 |
| 3 | 4 | 5 | 6 | 6 | 6 |
| 1 | 4 | 6 | 6 | 2 | 3 |
| 1 | 3 | 6 | 4 | 6 | 6 |

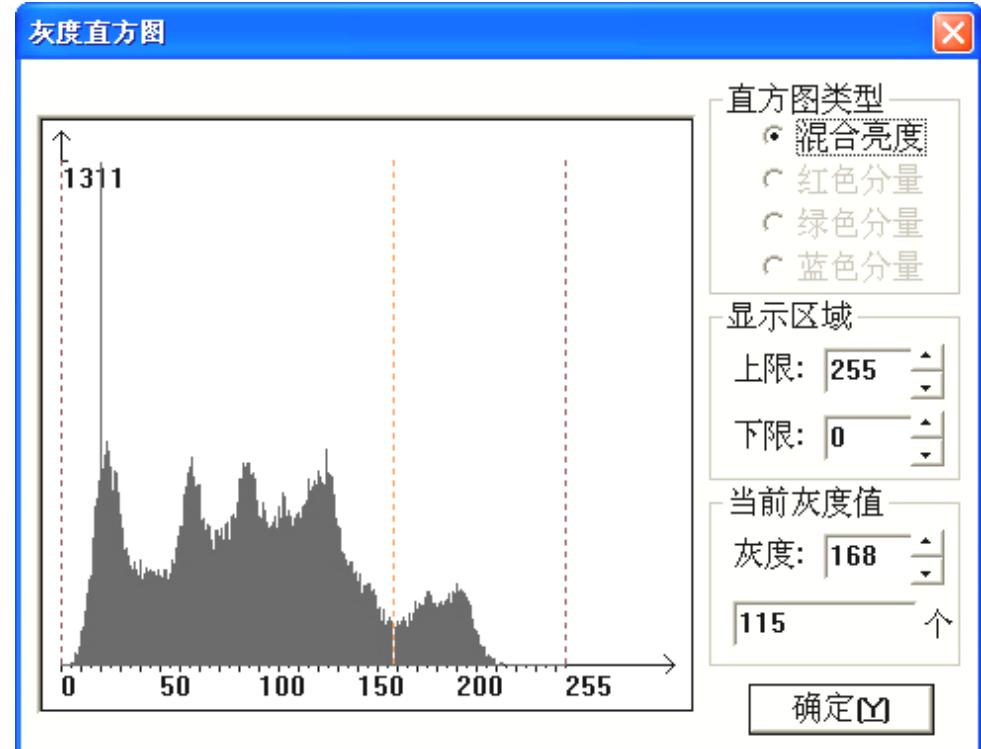


|   |   |   |   |   |    |
|---|---|---|---|---|----|
| 1 | 2 | 3 | 4 | 5 | 6  |
| 5 | 4 | 5 | 6 | 2 | 14 |



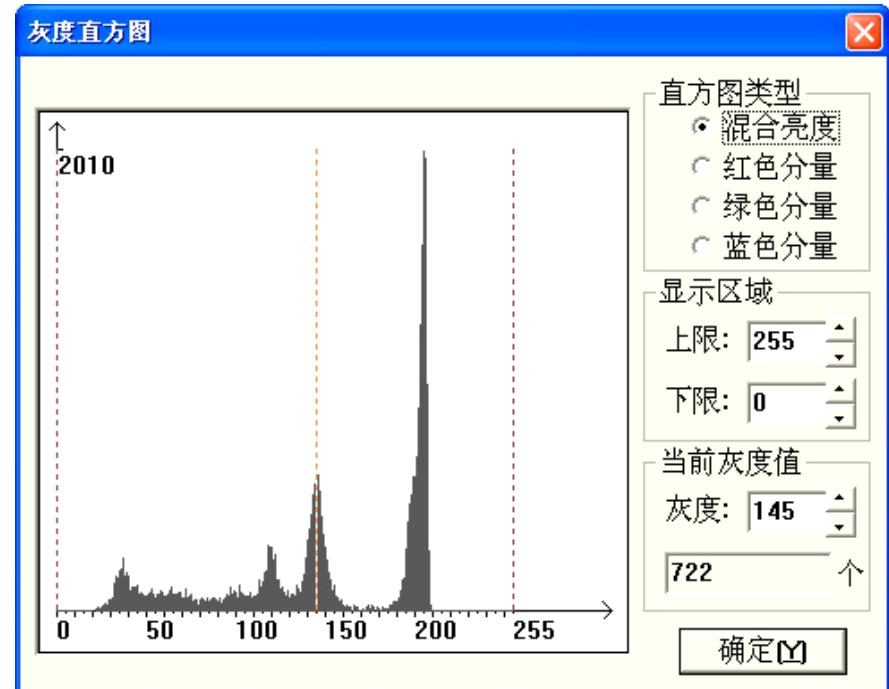
图像灰度直方图

## 3.4.2 直方图修正



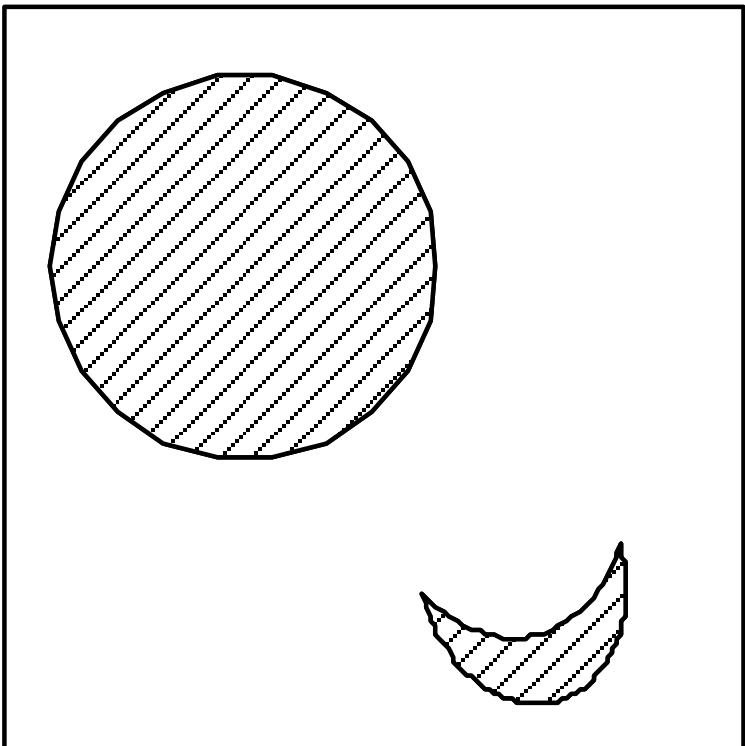
Lena图像及灰度直方图

## 3.4.2 直方图修正

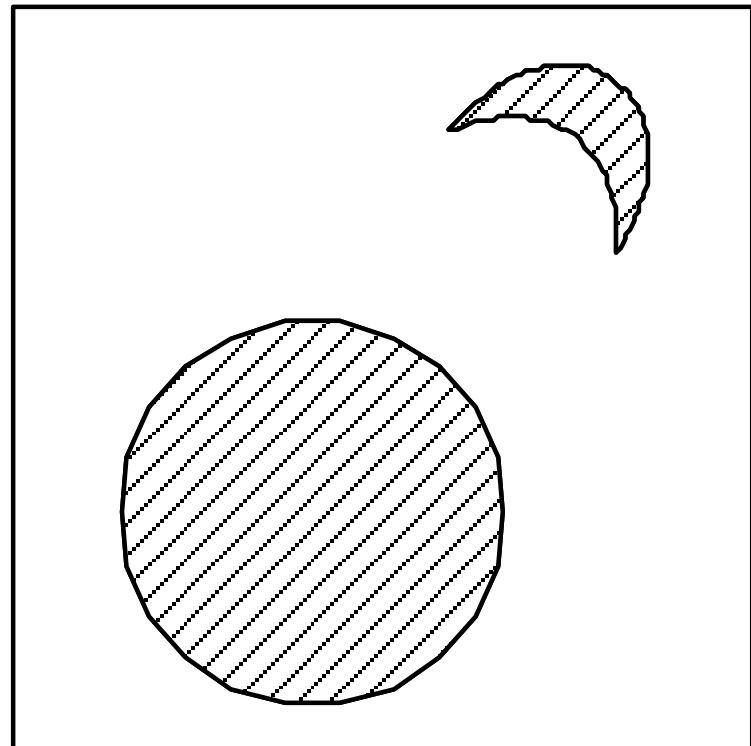


钟楼图像及灰度直方图  
(彩色)

## 3.4.2 直方图修正



(a)



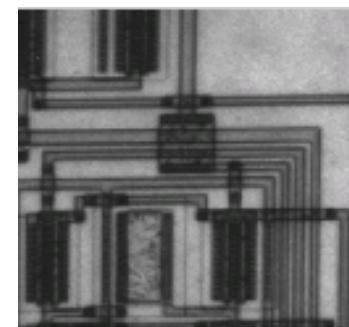
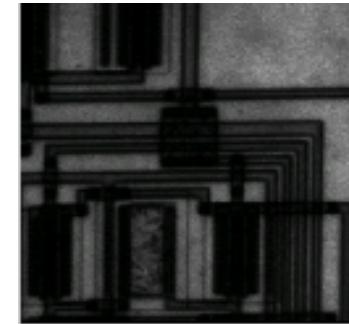
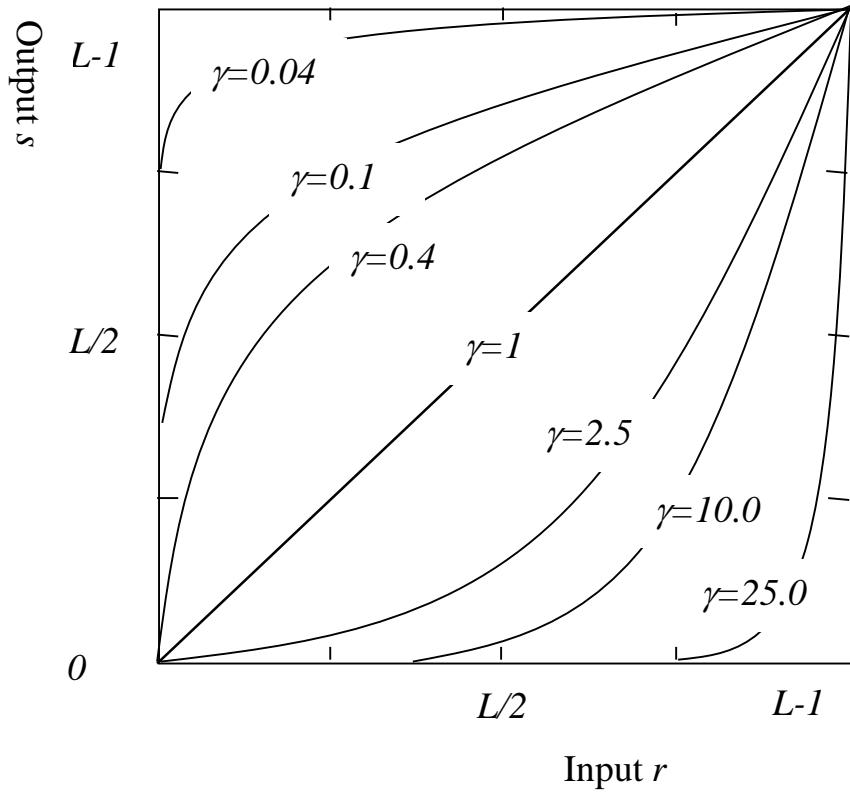
(b)

**图像与直方图间的多对一关系**

## 3.4.2 直方图修正

- 问题：图像的灰度值非均匀分布，其中有很多图像的灰度值集中在一个个小的区间内，导致图像的对比度降低。
- 解决：**直方图均衡化**可以通过重新均匀分布各个灰度值来增强图像的对比度。经过直方图均衡化后的图像对做二值化处理中的阈值选择很有帮助。

# $\gamma$ Calibration $s = c r^\gamma$ 幂函数



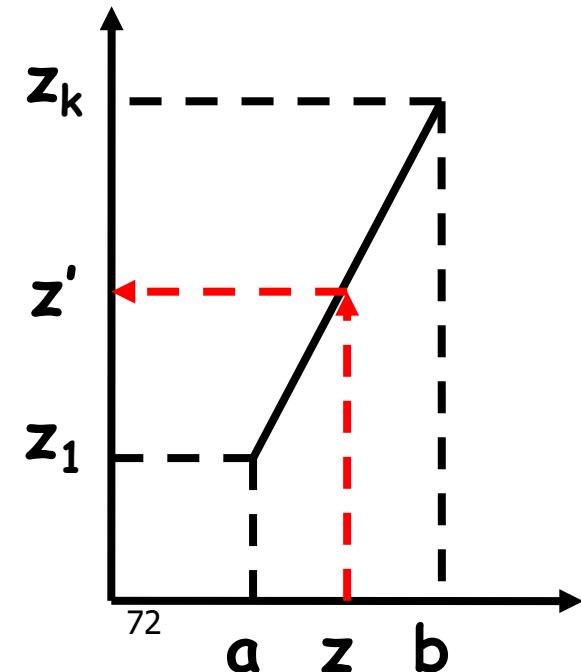
Some  $\gamma s = cr^\gamma$  curve and trans. result

## 3.4.2 直方图修正

- 图像灰度尺度变换：
  - ✓ 把图像灰度区间在 $[a, b]$ 内的像素点值映射到 $[z_1, z_k]$ 的区间内。
  - ✓ 一般情况下，由于光照原因，原始图像灰度区间 $[a, b]$ 常常是 $[z_1, z_k]$ 的子空间。

$$z' = \frac{z_k - z_1}{b - a} (z - a) + z_1$$

- ✓ 使曝光不充分的图像黑的更黑白的更白。

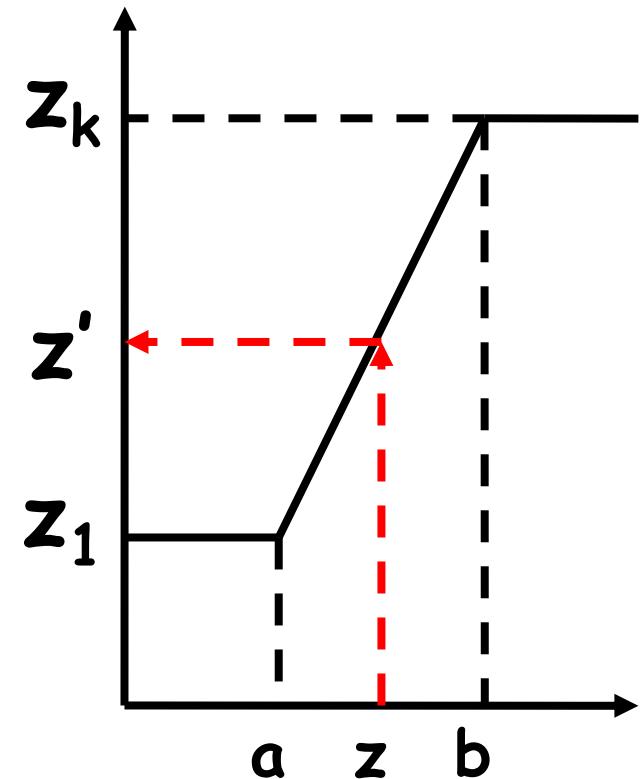


## 3.4.2 直方图修正

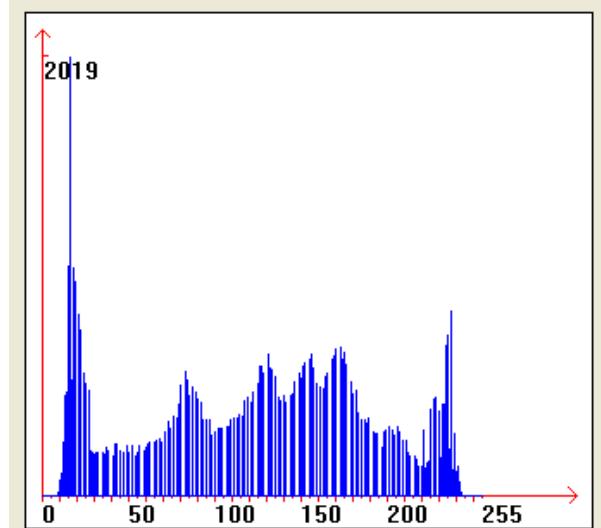
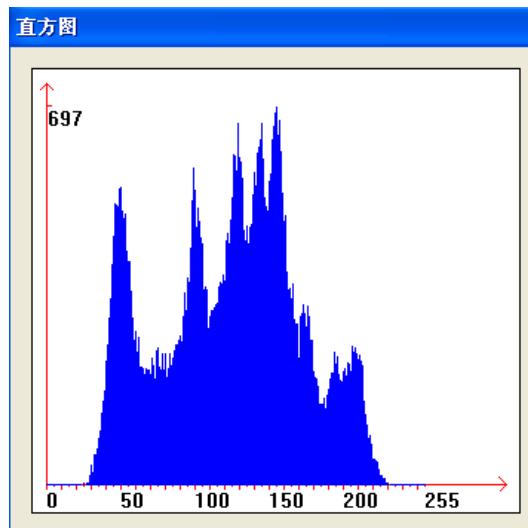
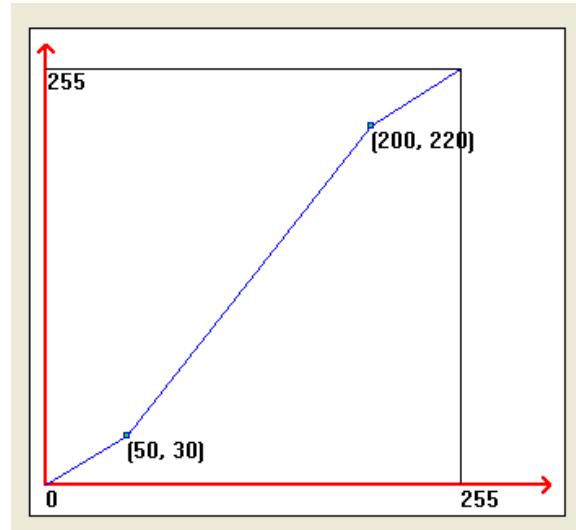
- 图像灰度尺度变换：**上下封顶**

如果图像的大多数像素灰度值分布在区间 $[a, b]$ ，一小部分分布在此区间外，可以使用以下映射函数：

$$z' = \begin{cases} \frac{z_k - z_1}{b - a}(z - a) + z_1 & a \leq z \leq b \\ z_1 & z < a \\ z_k & z > b \end{cases}$$



# GST result and relevant histogram



# Histogram equalization algorithm

1. List the gray level  $f_j, j=0, 1, \dots, k, \dots, L-1$ ;
2. Calculate the num of pixels with  $f_j, n_j, j=0, 1, \dots, k, \dots, L-1$ ;
3. Calculate  $P_f(f_j)=n_j/n$ ,  $n$  is the total num of pixels;
4. Get the Cumulative Distribution Function  $c(f)$ ;

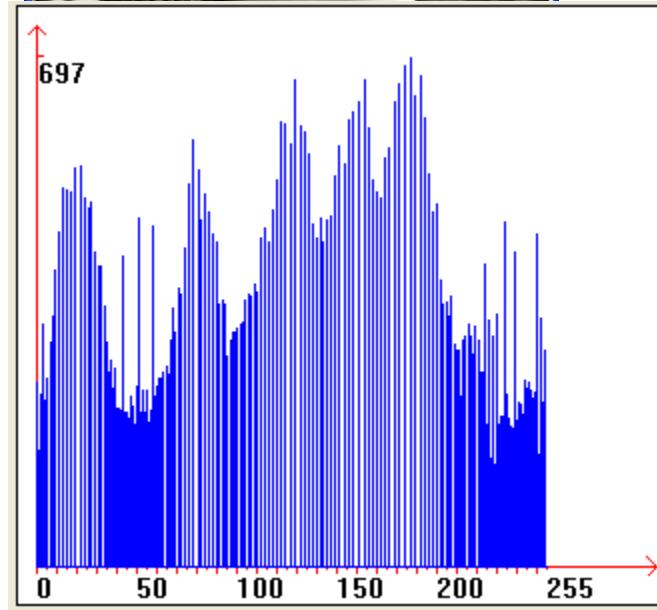
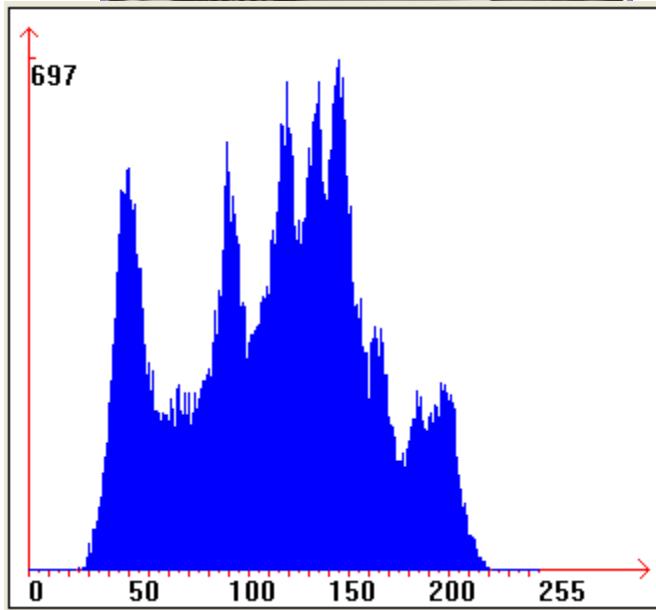
$$c(f) = \sum_{j=0}^m P_g(g_i), i = 0, 1, \dots, m, \dots, L-1$$

5. Apply trans. , calculate the target gray level,

$$g_i = INT[(g_{max}-g_{min})c(f)+g_{min}+0.5]$$

6. Get **the num of pixels with  $g_i$**   $n_i, i=0, 1, \dots, p-1$ ;
7. Calculate the output histogram  $P_g(g_i)=n_i/n, i=0, 1, \dots, p-1$ ;
8. With the relationship between  $f_j$  and  $g_i$ , modify the gray of original image, get the result.

# Histogram equalization example



# 针对bmp图像的直方图均衡化

```
// 直方图
int nCount[256];

// 图象的高度和宽度
CSize sizeImage;
sizeImage = pDib->GetDimensions();

// 获得图象数据存储的高度和宽度
CSize SizeSaveImage;
SizeSaveImage = pDib->GetDibSaveDim();

// 重置计数为0
for (i = 0; i < 256; i++)
{
    // 清零
    nCount[i] = 0;
}

// 计算各个灰度值的计数，即得到直方图
for (i = 0; i < sizeImage.cy; i++)
{
    for (j = 0; j < sizeImage.cx; j++)
    {
        lpSrc = (unsigned char *)pDib->m_lpImage + SizeSaveImage.cx * i + j;
        // 计数加1
        nCount[*lpSrc]++;
    }
}

// 计算累积直方图
for (i = 0; i < 256; i++)
{
    // 初始为0
    nTemp = 0;

    for (j = 0; j <= i ; j++)
    {
        nTemp += nCount[j];
    }

    // 计算对应的新灰度值
    byMap[i] = (BYTE) (nTemp * 255 / sizeImage.cy / sizeImage.cx);
}

// 每行
for(i = 0; i < sizeImage.cy; i++)
{
    // 每列
    for(j = 0; j < sizeImage.cx; j++)
    {
        // 指向DIB第i行，第j个象素的指针
        lpSrc = (unsigned char *)pDib->m_lpImage + pDib->GetPixelOffset(i,j);

        // 计算新的灰度值
        *lpSrc = byMap[*lpSrc];
    }
}
```

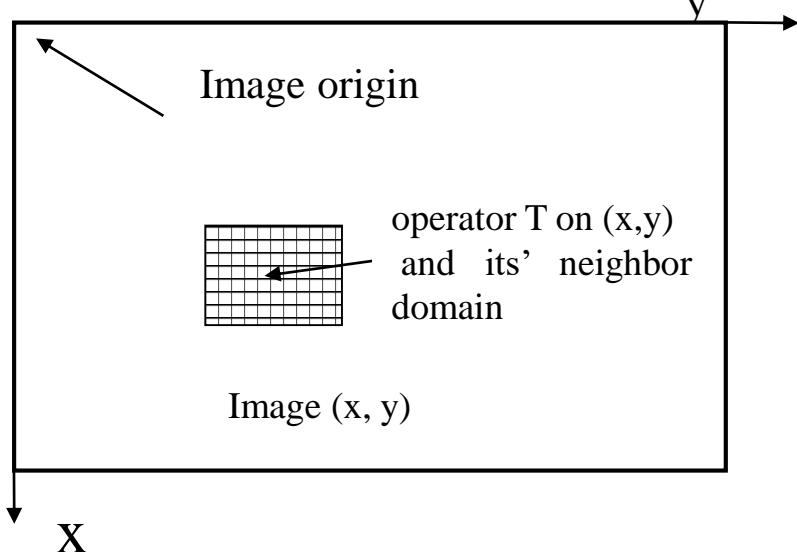
## 3.4.1 概述

- 直方图修正
- 图像线性运算模型
- 线性滤波器
  - ➡ 均值滤波器
  - ➡ 高斯平滑滤波器
- 非线性滤波器
  - ➡ 中值滤波器
  - ➡ 边缘保持滤波器

### 3.4.3 图像线性运算

✓ 线性卷积：

$$f(x, y) \rightarrow \begin{matrix} LSI \text{ 系统} \\ g(x, y) \end{matrix} \rightarrow h(x, y)$$



输出  $h(x, y)$  用输入  $f(x, y)$  与脉冲响应  $g(x, y)$

✓ 卷积定义：

$$h(x, y) = f(x, y) * g(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x', y') g(x - x', y - y') dx' dy'$$

✓ 图像离散系统可写成：

$$h[i, j] = f[i, j] * g[i, j] = \sum_{k=0}^{n-1} \sum_{l=0}^{m-1} f[k, l] g[i - k, j - l]$$

# Mask format

$$R = w(-1,-1)f(x-1, y-1) + w(-1,0)f(x-1, y) + w(-1,1)f(x-1, y+1) \\ + w(0,-1)f(x, y-1) + w(0,0)f(x, y) + w(0,1)f(x, y+1) \\ + w(1,-1)f(x+1, y-1) + w(1,0)f(x+1, y) + w(1,1)f(x+1, y+1)$$

|   |              |            | y            |
|---|--------------|------------|--------------|
|   |              |            | origin       |
| x |              |            |              |
|   | $f(x-1,y-1)$ | $f(x-1,y)$ | $f(x-1,y+1)$ |
|   | $f(x,y-1)$   | $f(x,y)$   | $f(x,y+1)$   |
|   | $f(x+1,y-1)$ | $f(x+1,y)$ | $f(x+1,y+1)$ |

|            |           |            |
|------------|-----------|------------|
| $w(-1,-1)$ | $w(-1,0)$ | $w(-1, 1)$ |
| $w(0,-1)$  | $w(0,0)$  | $w(0,1)$   |
| $w(1,-1)$  | $w(1,0)$  | $w(1, 1)$  |

(a) pixels under mask

(b) position and coefficient of mask

### 3.4.3 图像线性运算

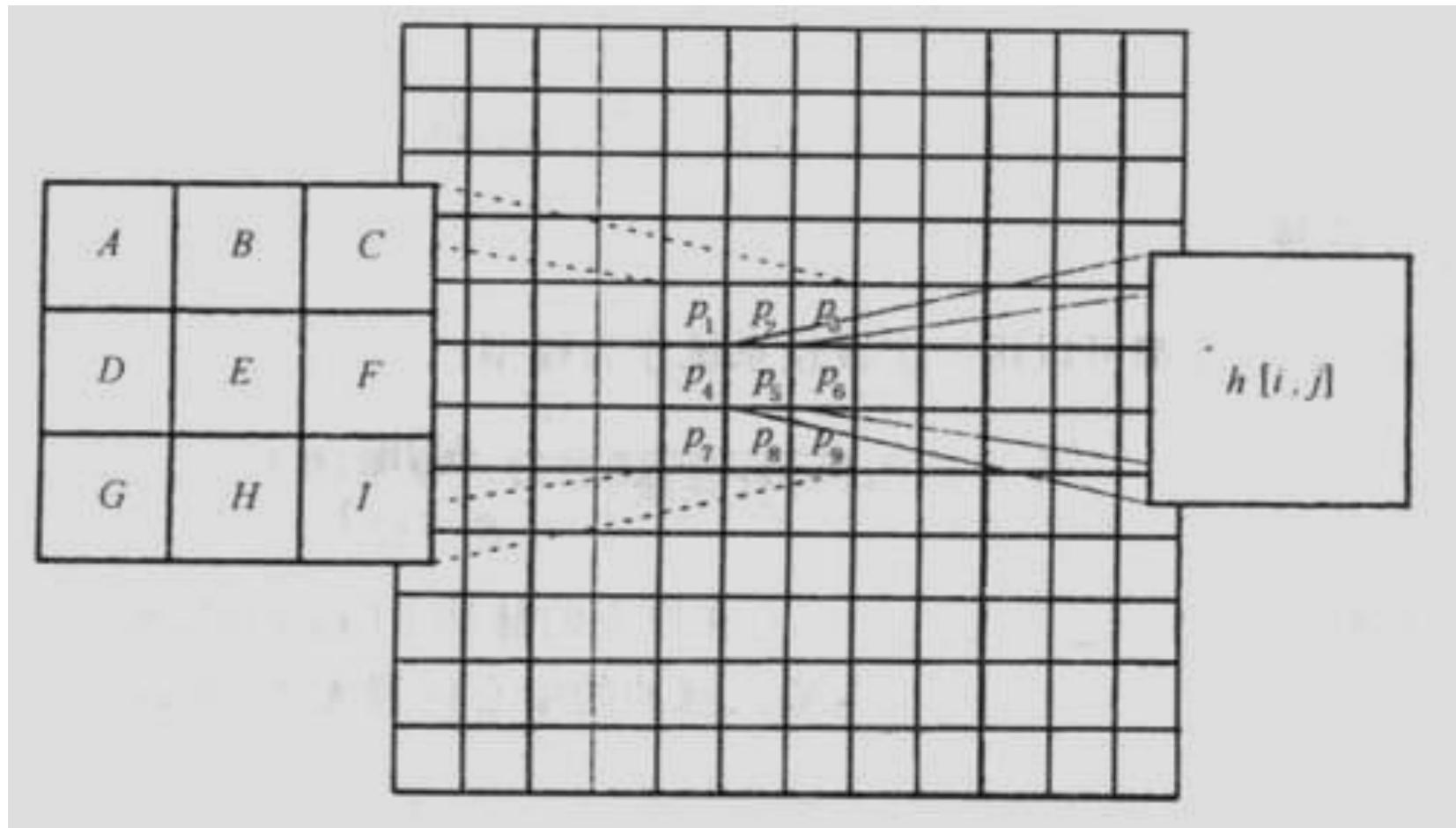
- 如果 $f[i, j]$ 和 $h[i, j]$ 表示图像，则卷积就变成了对像素点的加权计算，脉冲响应 $g[i, j]$ 就是一个卷积模板。
- 对图像中每一个像素点 $[i, j]$ ，输出响应值 $h[i, j]$ 是通过平移卷积模板到像素点 $[i, j]$ 处，计算模板与像素点 $[i, j]$ 邻域加权得到，其中加权值对应着卷积模板的各个对应值。

$$g[i, j] * \{a_1 f_1[i, j] + a_2 f_2[i, j]\} = a_1 \{g[i, j] * f_1[i, j]\} + a_2 \{g[i, j] * f_2[i, j]\}$$

- 和的卷积等于卷积的和，尺度变换后的图像卷积等于卷积后作相应的尺度变换。

### 3.4.3 图像线性运算

**线性卷积:**  $h[i, j] = f[i, j] * g[i, j] = \sum_{k=0}^{n-1} \sum_{l=0}^{m-1} f[k, l]g[i - k, j - l]$



### 3.4.4 线性滤波器

使用线性滤波器的原因----图像噪声：

**椒盐噪声 (Salt & Pepper)**：含有随机出现的黑白亮度值。

- 脉冲噪声：只含有随机的正脉冲和负脉冲噪声。
- 高斯噪声：含有**亮度服从高斯或正态分布的噪声**。  
高斯噪声是很多传感器噪声的模型，如摄像机的电子干扰噪声。



(a) 原始图像



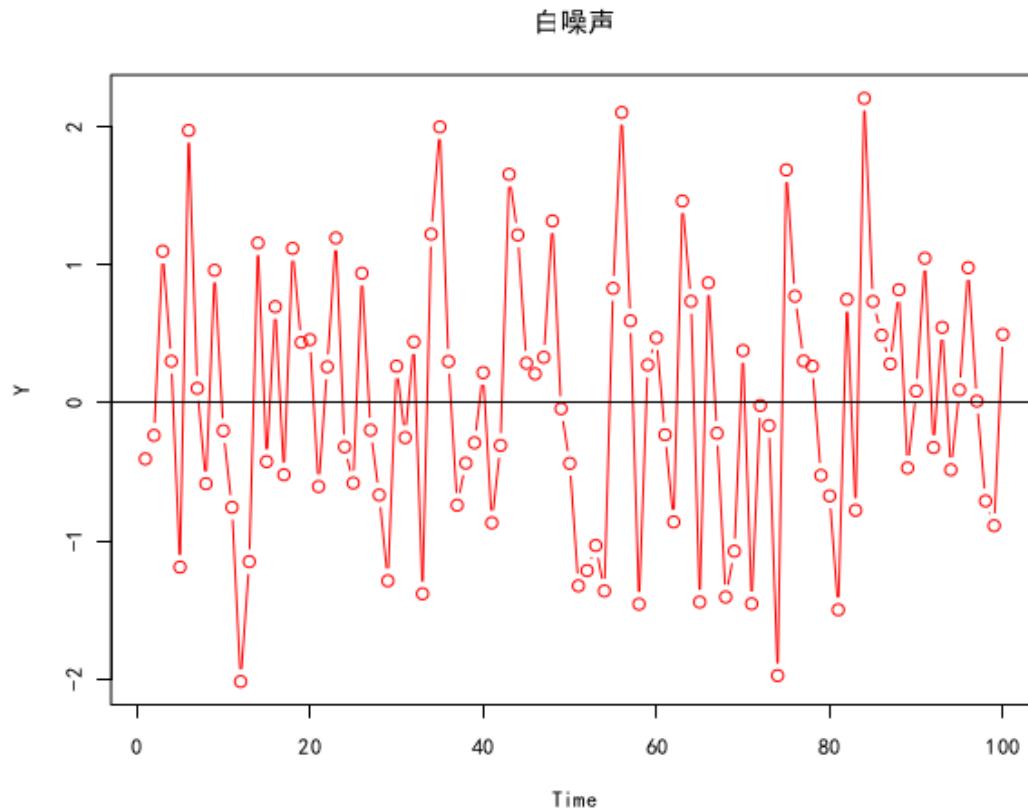
(b) 加入高斯  
噪声图像



(c) 加入椒盐  
噪声图像

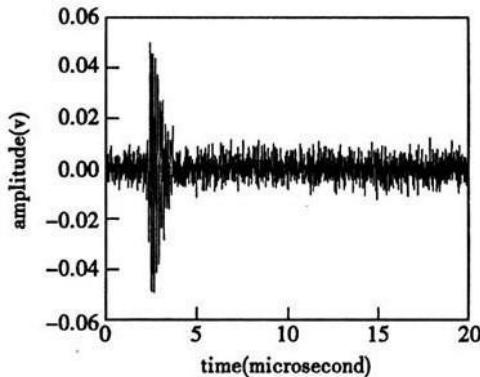
### 3.4.4 线性滤波器

■ 椒盐噪声 (Salt & Pepper) : 含有随机出现的黑白亮度值。如下图共100个元素，每个元素都独立服从标准正态分布 $N(0,1)$ ，可以从图中看出白噪声基本是在均值附近较为平均的随机震荡

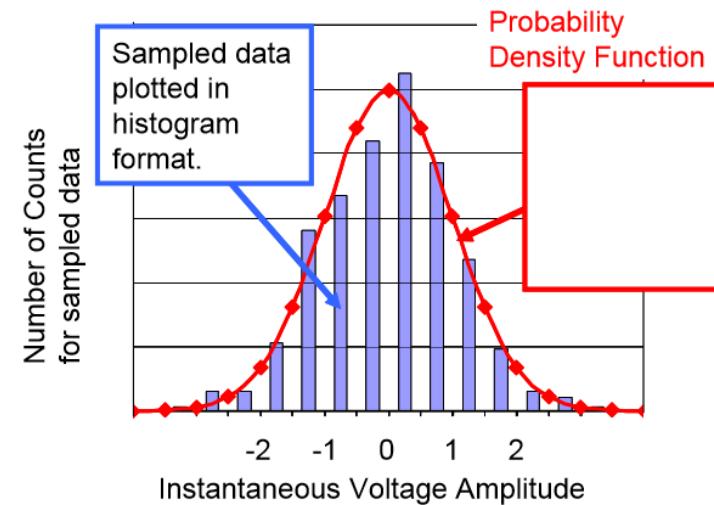
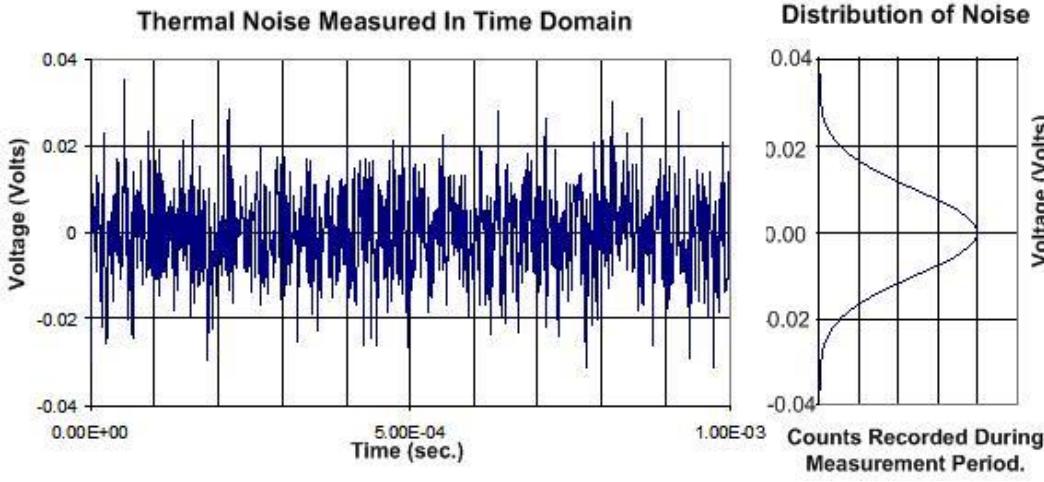


## 3.4.4 线性滤波器

- 脉冲噪声：只含有随机的正脉冲和负脉冲噪声。



- 高斯噪声：含有亮度服从高斯或正态分布的噪声。高斯噪声是很多传感器噪声的模型，如摄像机的电子干扰噪声。



### 3.4.4 线性滤波器

- 线性滤波器使用**连续窗函数内像素加权和**来实现滤波，**同一模式的权重因子**可以作用在每一个窗口内，即线性滤波器是**空间不变的**
- 如果图像的不同部分使用**不同的滤波权重因子**，线性滤波器是**空间可变的**。因此可以使用卷积模板来实现滤波。
- 线性滤波器对去除高斯噪声有很好的效果。

常用的线性滤波器：

- 均值滤波器
- 高斯平滑滤波器

### 3.4.4 线性滤波器

#### 均值滤波器

- 最简单均值滤波器是局部均值运算，即每一个像素只用其局部邻域内所有值的平均值来置换：

$$h[i, j] = \frac{1}{M} \sum_{(k, l) \in N} f[k, l]$$

- 一种只有一个峰值，并且在水平和垂直方向上对称的典型权值模板：

|                |               |                |
|----------------|---------------|----------------|
| $\frac{1}{16}$ | $\frac{1}{8}$ | $\frac{1}{16}$ |
| $\frac{1}{8}$  | $\frac{1}{4}$ | $\frac{1}{8}$  |
| $\frac{1}{16}$ | $\frac{1}{8}$ | $\frac{1}{16}$ |

### 3.4.4 线性滤波器

#### 高斯平滑滤波器

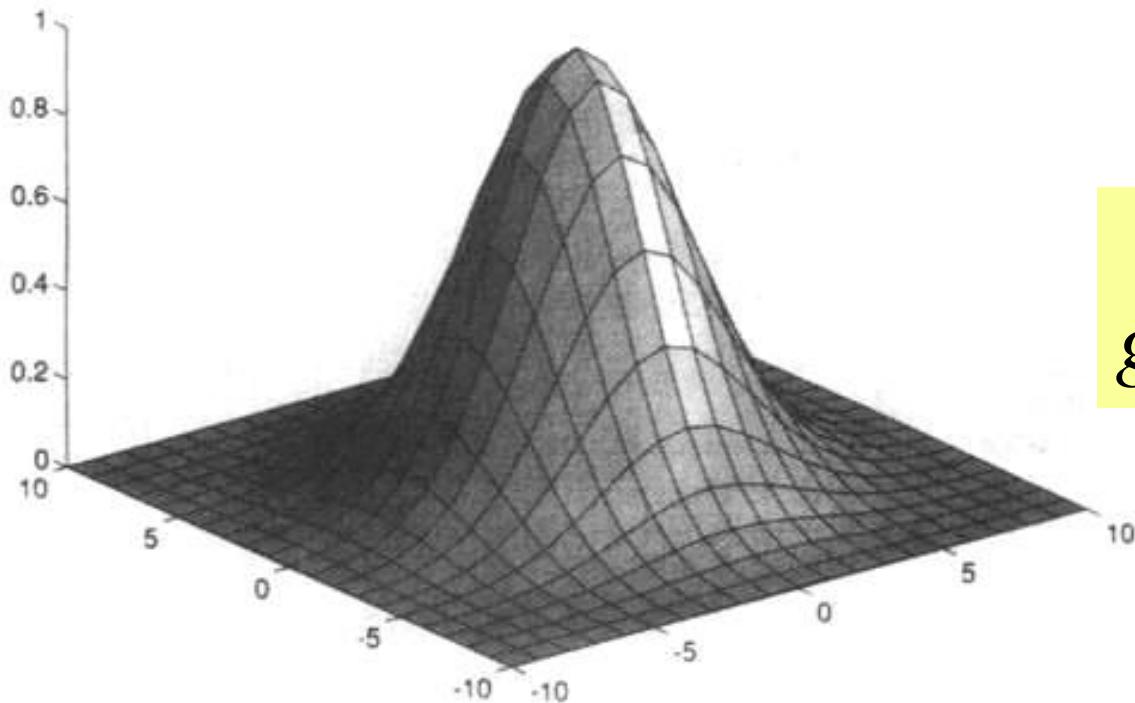
- 高斯平滑滤波器是一类根据高斯函数的形状来选择权值的线性滤波器。
- 高斯平滑滤波器对去除服从正态分布的噪声是很有效的。
- 二维零均值离散高斯平滑滤波器函数表达式为：

$$g[i, j] = e^{-\frac{(i^2 + j^2)}{2\sigma^2}}$$

## 3.4.4 线性滤波器

### 高斯平滑滤波器

根据高斯函数选择邻域内各像素的权值



$$g[i, j] = e^{-\frac{(i^2 + j^2)}{2\sigma^2}}$$

# 数字图像处理中常用的高斯滤波器模板

|   |   |   |    |   |   |   |
|---|---|---|----|---|---|---|
| 1 | 1 | 2 | 2  | 2 | 1 | 1 |
| 1 | 2 | 2 | 4  | 2 | 2 | 1 |
| 2 | 2 | 4 | 8  | 4 | 2 | 2 |
| 2 | 4 | 8 | 16 | 8 | 4 | 2 |
| 2 | 2 | 4 | 8  | 4 | 2 | 2 |
| 1 | 2 | 2 | 4  | 2 | 2 | 1 |
| 1 | 1 | 2 | 2  | 2 | 1 | 1 |

7×7 高斯滤波模板

|   |   |    |    |    |    |    |    |    |    |    |    |    |   |   |
|---|---|----|----|----|----|----|----|----|----|----|----|----|---|---|
| 2 | 2 | 3  | 4  | 5  | 5  | 6  | 6  | 6  | 5  | 5  | 4  | 3  | 2 | 2 |
| 2 | 3 | 4  | 5  | 7  | 7  | 8  | 8  | 8  | 7  | 7  | 5  | 4  | 3 | 2 |
| 3 | 4 | 6  | 7  | 9  | 10 | 10 | 11 | 10 | 10 | 9  | 7  | 6  | 4 | 3 |
| 4 | 5 | 7  | 9  | 10 | 12 | 13 | 13 | 13 | 12 | 10 | 9  | 7  | 5 | 4 |
| 5 | 7 | 9  | 11 | 13 | 14 | 15 | 16 | 15 | 14 | 13 | 11 | 9  | 7 | 5 |
| 5 | 7 | 10 | 12 | 14 | 16 | 17 | 18 | 17 | 16 | 14 | 12 | 10 | 7 | 5 |
| 6 | 8 | 10 | 13 | 15 | 17 | 19 | 19 | 19 | 17 | 15 | 13 | 10 | 8 | 6 |
| 6 | 8 | 11 | 13 | 16 | 18 | 19 | 20 | 19 | 18 | 16 | 13 | 11 | 8 | 6 |
| 6 | 8 | 10 | 13 | 15 | 17 | 19 | 19 | 19 | 17 | 15 | 13 | 10 | 8 | 6 |
| 5 | 7 | 10 | 12 | 14 | 16 | 17 | 18 | 17 | 16 | 14 | 12 | 10 | 7 | 5 |
| 5 | 7 | 9  | 11 | 13 | 14 | 15 | 16 | 15 | 14 | 13 | 11 | 9  | 7 | 5 |
| 4 | 5 | 7  | 9  | 10 | 12 | 13 | 13 | 13 | 12 | 10 | 9  | 7  | 5 | 4 |
| 3 | 4 | 6  | 7  | 9  | 10 | 10 | 11 | 10 | 10 | 9  | 7  | 6  | 4 | 3 |
| 2 | 3 | 4  | 5  | 7  | 7  | 8  | 8  | 8  | 7  | 7  | 5  | 4  | 3 | 2 |
| 2 | 2 | 3  | 4  | 5  | 5  | 6  | 6  | 6  | 5  | 5  | 4  | 3  | 2 | 2 |

15×15高斯滤波模板

### 3.4.4 线性滤波器

#### 高斯平滑滤波器

- 高斯函数有5个重要性质，其性质决定了它在空间域和频率域都是很好的低通滤波器。
  - 二维高斯函数具有旋转对称性：滤波器在各个方向平滑程度一致。旋转对称性意味着高斯平滑滤波器在后续边缘检测中不会偏向任一方向。
  - 高斯是单值函数：高斯滤波器用像素邻域的加权值来代替该点的像素值，每一邻域像素点权值是随该点与中心点的距离单调增减的。平滑运算会对离算子中心越远，作用越小。

### 3.4.4 线性滤波器

高斯平滑滤波器性质

• 旋转对称性

✓ 将高斯函数从直角坐标变换到极坐标得：

$$g[i, j] = e^{-\frac{i^2 + j^2}{2\sigma^2}} \quad r^2 = i^2 + j^2 \quad g(r, \theta) = e^{-\frac{r^2}{2\sigma^2}}$$

✓ 从高斯函数的极坐标表达式可以看出，函数值不依赖于角度 $\theta$ ，所以是旋转对称的。

### 3.4.4 线性滤波器

#### 高斯平滑滤波器

- **高斯函数的傅立叶变换频谱是单瓣的：**图像常被不希望的高频信号所污染，平滑图像不会被不需要的高频噪声污染，可以保留大部分信号。
- **高斯滤波器宽度是由参数 表示的：**越大，高斯滤波器的时域越宽，频带越窄，平滑程度越好。可以调节，使图像特征在过分模糊（过平滑）与平滑图像中由于噪声和细纹理所引起的过多不希望突变量（欠平滑）之间取折中。
- **高斯函数有可分离性：**实现高斯滤波时可分步进行

### 3.4.5 非线性滤波器

#### 中值滤波器

- 均值滤波和高斯滤波运算主要问题是有可能模糊图像中尖锐不连续的部分。
- 中值滤波器的基本思想使用像素点邻域灰度值的中值来代替该像素点的灰度值，它可以去除脉冲噪声、椒盐噪声同时保留图像边缘细节。
- 中值滤波不依赖于邻域内与典型值差别很大的值，处理过程不进行加权运算。

### 3.4.5 非线性滤波器

中位数

#### 中值滤波器

- 运算步骤：
  - 按亮度值大小排列像素点
  - 选择排序像素集的中间值作为点 $[i, j]$ 的新值
  - 一般采用奇数点的邻域计算中值，如果像素点数为偶数，则中值就取排序像素中间两点的平均值。
  - 中值滤波在一定条件下可以克服线性滤波器所造成的图像细节模糊，而对滤除脉冲干扰很有效。

## 3.4.5 非线性滤波器

### 边缘保持滤波器

#### Why?

- 均值滤波：平滑图像外还可能导致图像边缘模糊；
- 中值滤波：去除脉冲噪声的同时可能将图像中的线条细节滤除。

#### So

- 边缘保持滤波器是在综合考虑了均值滤波器和中值滤波器的优缺点后发展起来的，它的特点是：
  - 滤波器在除噪声脉冲的同时，又不至于使图像边缘十分模糊。

### 3.4.5 非线性滤波器

#### 边缘保持滤波器

##### ■ 边缘保持算法过程：

- 对灰度图象的每一个像素点[i, j]取适当大小的一个邻域（ $3 \times 3$ 邻域）；
- 分别计算[i, j]的左上角子邻域、左下角子邻域、右上角子邻域、右下角子邻域的灰度分布均匀度V；
- 然后取最小均匀度对应区域的均值作为该像素点的新灰度值。

$$V = \sum f^2[i, j] - (\sum f[i, j])^2 / N$$

## 3.4.5 非线性滤波器

### 边缘保持滤波器

- 上述算法过程中的灰度分布均匀度  $V$ , 除了可以用,

$$V = \sum f^2[i, j] - (\sum f[i, j])^2 / N$$

还可以用,

$$V = \sum (f[i, j] - \bar{f}[i, j])^2$$

## 3.4.5 非线性滤波器

### 边缘保持滤波器

例：计算一个邻域的均匀度分布

|   |   |   |
|---|---|---|
| 0 | 1 | 1 |
| 0 | 0 | 1 |
| 0 | 0 | 1 |

$$V = \sum (f[i,j] - \bar{f}[i,j])^2$$

|   |   |
|---|---|
| 0 | 1 |
| 0 | 0 |

v=3/4

|   |   |
|---|---|
| 1 | 1 |
| 0 | 1 |

v=3/4

|   |   |
|---|---|
| 0 | 0 |
| 0 | 0 |

v=0

|   |   |
|---|---|
| 0 | 1 |
| 0 | 1 |

v=1

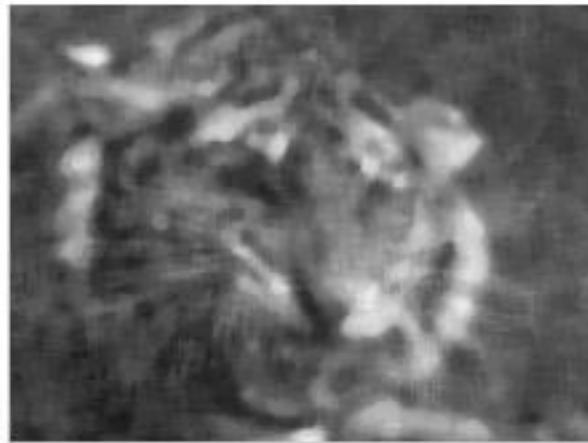
- 从计算可知，分布越均匀，V值越小。
- 左下角邻域全为零，因此用V=0最小值代替。



3×3模板



保持边缘



中值

7×7模板

模板过大，造成边缘模糊

## 3.5.6 二值图像运算

### 目录:

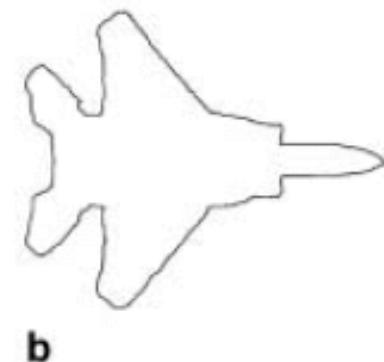
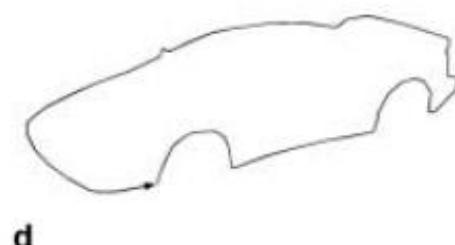
- ❖ 视觉的计算理论
- ❖ 数字图像概念
- ❖ 数字图像处理基本步骤
- ❖ 图像预处理
- ❖ 图像区域表示
- ❖ 二值图像运算
- ❖ 图像特征比对
- ❖ OpenCV平台使用

## 3.5 图像区域表示

- 3.5.1 区域检测中的一些基本术语
- 3.5.2 梯度
- 3.5.3 一阶边缘检测算子
- 3.5.4 二阶边缘检测算子
- 3.5.5 Canny 检测算法
- 3.5.6 轮廓表示
- 3.5.7 Hough 变换

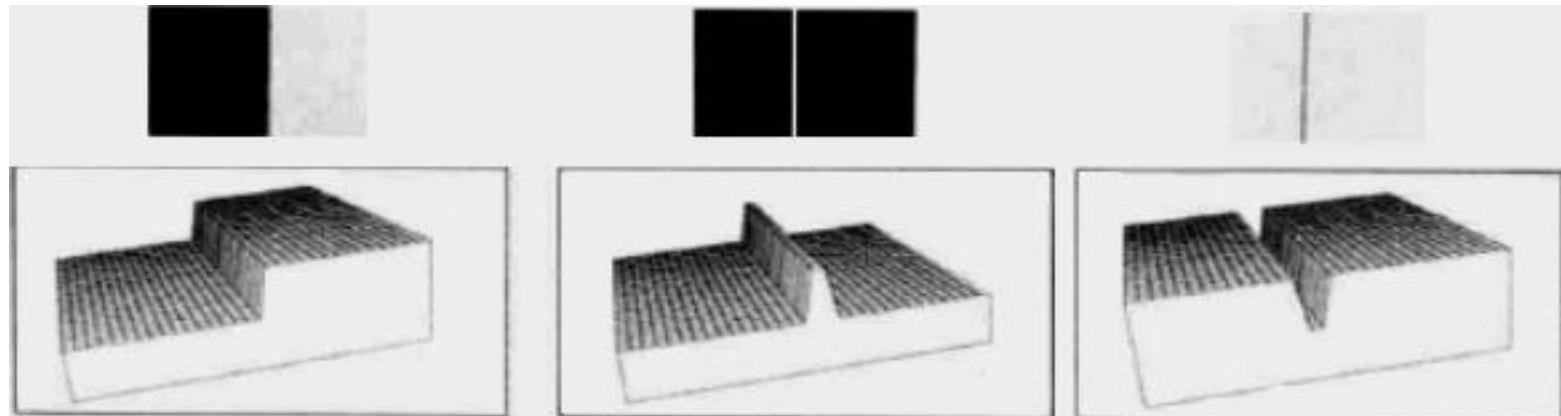
# 边缘与轮廓 (contour)

- 轮廓是物体在场景中的完整边界.
- 边缘的连接构成轮廓.



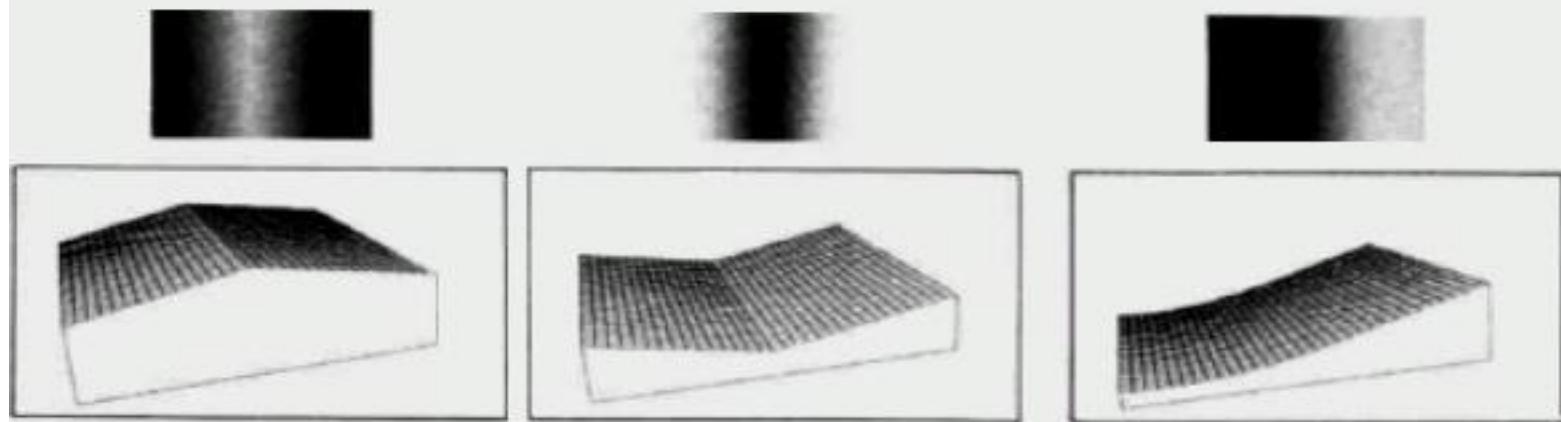
## 3.5.1 边缘检测

- 边缘是指图像局部亮度变化最显著的部分
- 图像亮度变化有几种：
- **阶跃不连续：** 图像亮度在不连续处两边的像素灰度值有着显著的差异；
- **线条不连续：** 图像亮度突然从一个值变化到另一个值，保持一个较小的行程后又返回到原来的值。
- 以上是理想状态，实际中由于大多数传感器元件具有低频特性，使得阶跃边缘变成斜坡形边缘，线条边缘变成屋顶形边缘。**渐变**



(a)

(b)



(c)

(d)

图像中不同类型的边界 形象！！！

(a) 阶跃变化；

(c) 折线变化；

(b) 线条变化；

(d) 缓慢的平滑变化

### 3.5.1 边缘检测

- 一些术语：
  - **边缘点**：图像中亮度显著变化的点。
  - **边缘段**：边缘点坐标 $[i,j]$ 及其方向 $\theta$ 的总和，边缘的方向可以是梯度值。
  - **边缘检测器**：从图像中抽取边缘（边缘点或边缘线段）集合的算法。
  - **轮廓**：边缘列表，或是一条边缘列表的曲线模型。
  - **边缘连接**：从无序边缘表形成有序边缘表的过程，习惯上边缘的表示用顺时针方向来排序。
  - **边缘跟踪**：一个用来确定轮廓图像的搜索过程。

# 梯 度

- 一维情况下，阶跃边缘同图像的一阶导数局部峰值有关。图像灰度值的显著变化可用梯度的离散逼近函数来检测，梯度是一阶导数的二维等效式：

$$\mathbf{G}(x, y) = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

- 梯度的幅度值： $|G(x, y)| = \sqrt{G_x^2 + G_y^2}$
- 梯度的幅度值也可是： $|G(x, y)| = |G_x| + |G_y|$
- 梯度的幅度值是与方向无关的，算子是各向同性算子。
- 数字图像中，导数用差分表示：

$$G_x = f[i, j+1] - f[i, j]$$

$$G_y = f[i, j] - f[i+1, j]$$

# Gradient operator

$$G[f(x,y)] = [\partial f / \partial x, \partial f / \partial y]^T$$

character: 1) the *largest increased orientation*;  
2)  $|G[f(x,y)]| = [(\partial f / \partial x)^2 + (\partial f / \partial y)^2]^T$

## Differences instead of division

forward differences definition:

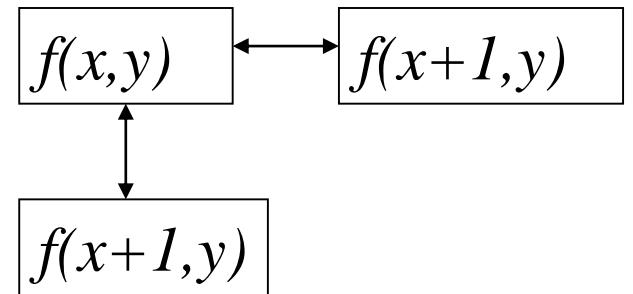
$$\Delta f_i = f_{i+1} - f_i,$$

$$\Delta_n f_i = \Delta_{n-1} f_{i+1} - \Delta_{n-1} f_i,$$

Backward differences definition :

$$\nabla f_i = f_i - f_{i-1}, \quad \nabla_n f_i = \nabla_{n-1} f_i - \nabla_{n-1} f_{i-1},$$

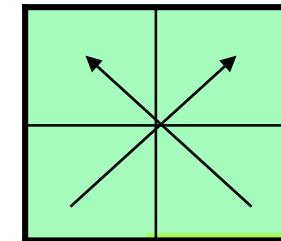
$$\delta f_i = f_{i+1/2} - f_{i-1/2}, \quad \delta_n f_i = \delta_{n-1} f_{i+1/2} - \delta_{n-1} f_{i-1/2},$$



## 3.5.2 一阶边缘检测

### ■ Roberts交叉算子

Roberts算子为梯度幅度值计算提供了一种简单的近似：



$$G[i, j] = |f[i, j] - f[i + 1, j + 1]| + |f[i + 1, j] - f[i, j + 1]|$$

模板卷积：  $G[i, j] = |G_x| + |G_y|$

$G_x$  和  $G_y$  模板是：

$$(i, j) \rightarrow G_x = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

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

通过卷积来实现算子，叫做卷积模板

## 3.5.2 一阶边缘检测

### ■ Sobel算子

Sobel算子为梯度幅度值计算：

$$M = \sqrt{s_x^2 + s_y^2}$$

模板卷积：

$$s_x = (a_2 + ca_3 + a_4) - (a_0 + ca_7 + a_6)$$

$$s_y = (a_0 + ca_1 + a_2) - (a_6 + ca_5 + a_4)$$

c=2

$G_x$ 和 $G_y$ 模板是：

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

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

## 3.5.2 一阶边缘检测

### ■ Prewitt算子

Prewitt算子和Sobel算子方程一样：

$$M = \sqrt{s_x^2 + s_y^2}$$

模板卷积：

$$s_x = (a_2 + ca_3 + a_4) - (a_0 + ca_7 + a_6)$$

$$s_y = (a_0 + ca_1 + a_2) - (a_6 + ca_5 + a_4)$$

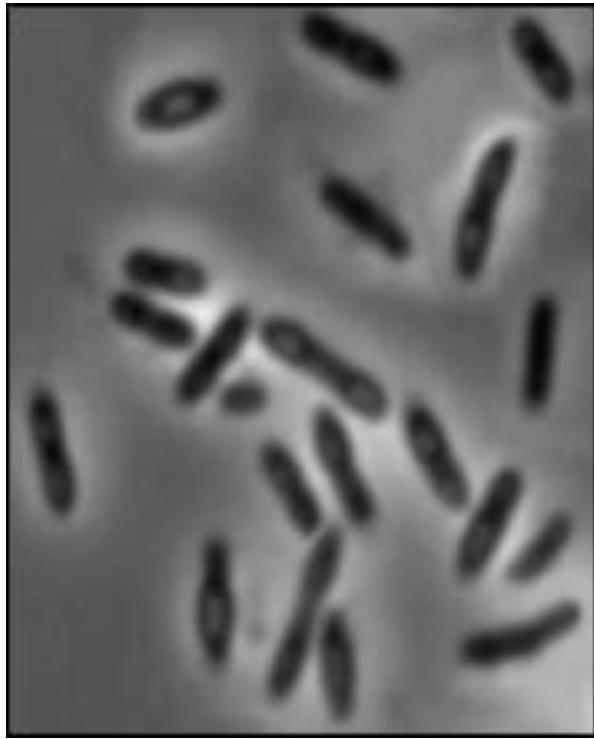
c=1

$G_x$ 和 $G_y$ 模板是：

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

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

$$\begin{array}{ccc} a_0 & a_1 & a_2 \\ a_7 & [i,j] & a_3 \\ a_6 & a_5 & a_4 \end{array}$$



用Prewitt算子进行边缘检测的结果

# 小结

- 直方图修正
- 图像线性运算模型
- 线性滤波器
  - ↳ 均值滤波器
  - ↳ 高斯平滑滤波器
- 非线性滤波器
  - ↳ 中值滤波器
  - ↳ 边缘保持滤波器

## 3.5.5 图像区域表示

### 目录:

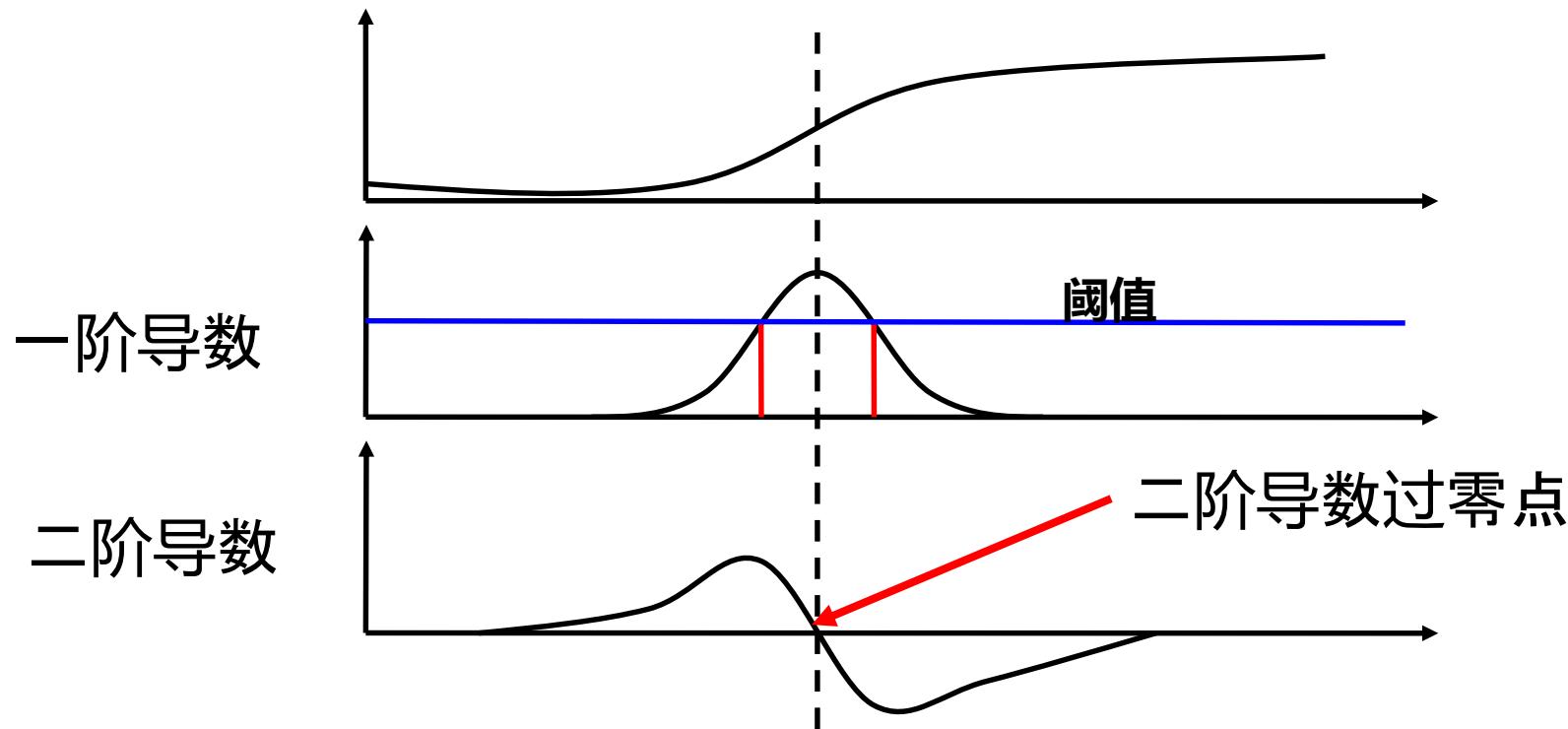
- ❖ 数字图像概念
- ❖ 数字图像处理基本步骤
- ❖ 图像预处理
- ❖ 图像区域表示
- ❖ 二值图像运算
- ❖ 图像特征比对
- ❖ OpenCV平台使用

## 3.5 图像区域表示

- 3.5.1 区域检测中的一些基本术语
- 3.5.2 梯度
- 3.5.3 一阶边缘检测算子
- 3.5.4 二阶边缘检测算子
- 3.5.5 Canny 检测算法
- 3.5.6 轮廓表示
- 3.5.7 Hough 变换

### 3.5.3 二阶边缘检测

- 一阶导数的边缘检测器会导致边缘点检测太多。
- 一种解决办法是：使用梯度值的局部最大值对应点作为边缘点，对应着二阶导数零交叉点。



### 3.5.3 二阶边缘检测

■ 拉普拉斯算子

● 拉普拉斯算子是二阶导数的二维等效式：

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

● 差分表示：

$$\begin{aligned}\nabla^2 f(x, y) &= L(x, y) \\&= \{[f(x+1, y) - f(x, y)] - [f(x, y) - f(x-1, y)]\} \\&\quad + \{[f(x, y+1) - f(x, y)] - [f(x, y) - f(x, y-1)]\} \\&= \underline{f(x+1, y)} + \underline{f(x-1, y)} + \underline{f(x, y+1)} + \underline{f(x, y-1)} - \underline{4f(x, y)}\end{aligned}$$

$$\nabla^2 \approx \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

### 3.5.3 二阶边缘检测

■拉普拉斯算子

•拉普拉斯算子是二阶导数的二维等效式：

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

•差分表示：

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

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

$$\nabla^2 \approx \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

### 3.5.3 二阶边缘检测

■拉普拉斯算子

•拉普拉斯算子的卷积模板：

$$\nabla^2 \approx \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

•有时为了突出中心点，可以使用更大的权值：

$$\nabla^2 \approx \begin{bmatrix} 0 & 4 & 0 \\ 4 & -20 & 4 \\ 0 & 4 & 0 \end{bmatrix}$$

# Typical Laplacian mask

|    |    |    |
|----|----|----|
| 1  | -2 | 1  |
| -2 | 4  | -2 |
| 1  | -2 | 1  |

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

|    |    |    |
|----|----|----|
| -1 | -1 | -1 |
| -1 | 8  | -1 |
| -1 | -1 | -1 |

# example



### 3.5.3 二阶边缘检测

- 二阶方向导数的推广形式：
- 一张图像曲面 $f(x,y)$ ,其 $\theta$ 方向（与y轴夹角）的方向导数在 $f(x,y)$ 点的值为

$$\frac{\partial f}{\partial \theta} = \frac{\partial f(x, y)}{\partial x} \sin \theta + \frac{\partial f(x, y)}{\partial y} \cos \theta$$

- 二阶方向导数为

$$\frac{\partial^2 f}{\partial \theta^2} = \frac{\partial^2 f(x, y)}{\partial x^2} \sin^2 \theta + 2 \frac{\partial^2 f(x, y)}{\partial x \partial y} \sin \theta \cos \theta + \frac{\partial^2 f(x, y)}{\partial y^2} \cos^2 \theta$$

- 一般二阶方向导数不常见，因为很容易受到噪声影响，故使用之前必须经过特别有效的滤波处理。

### 3.5.3 二阶边缘检测

Canny边缘检测算法：

- 用高斯平滑滤波器平滑图像
- 用一阶偏导数的有限差分来计算图像的梯度的幅度和方向
- 对梯度幅度值进行非极大值抑制 (NMS)
- 用双阈值算法检测和连接边缘
- Matlab: `BW = EDGE(I, 'canny', THRESH, SIGMA)`
- OpenCV: `cvCanny(OriginalImage, CannyImage, Threshold, Threshold*3, 3);`

### 3.5.3 二阶边缘检测

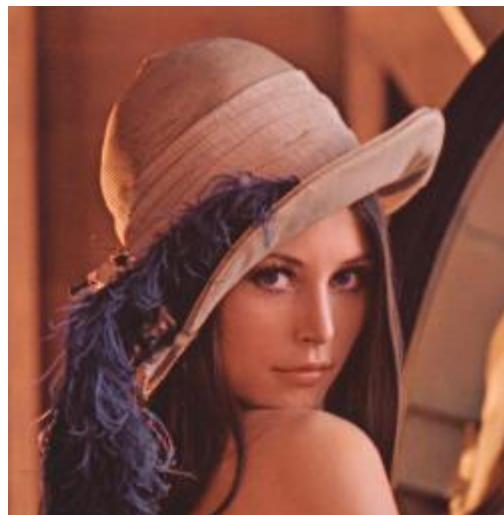
#### Canny边缘检测算法：

■  $I[i,j]$  表示图像，使用可分离滤波方法求取图像与高斯平滑滤波器的卷积，得到平滑后的结果：

$$S[i,j] = G[i,j;\sigma] * I[i,j]$$



原图



平滑后图像

### 3.5.3 二阶边缘检测

#### Canny边缘检测算法：

■  $I[i,j]$  表示图像，使用可分离滤波方法求取图像与高斯平滑滤波器的卷积，得到平滑后的结果：

$$S[i,j] = G[i,j;\sigma] * I[i,j]$$

■ 平滑后的图像阵列  $S[i,j]$  的梯度可以用  $2 \times 2$  的一阶有限差分近似计算， $x$  和  $y$  方向的偏导数  $P[i,j]$  与  $Q[i,j]$ ：

$$P[i,j] \approx (S[i,j+1] - S[i,j] + S[i+1,j+1] - S[i+1,j]) / 2$$

$$Q[i,j] \approx (S[i,j] - S[i+1,j] + S[i,j+1] - S[i+1,j+1]) / 2$$

$P[i,j]$  与  $Q[i,j]$  是  $2 \times 2$  正方形内求得的有限差分的均值

### 3.5.3 二阶边缘检测

Canny边缘检测算法：

■ 梯度幅度值和方向角分别是：

$$M[i, j] = \sqrt{P^2[i, j] + Q^2[i, j]}$$

$$\theta[i, j] = \arctan(Q[i, j], P[i, j])$$

其中反正切函数中有两个参数，其表示一个角度，此角度的取值范围是整个圆周范围。

两个图像等空间数组：

像素梯度幅度值

像素梯度角度值

### 3.5.3 二阶边缘检测

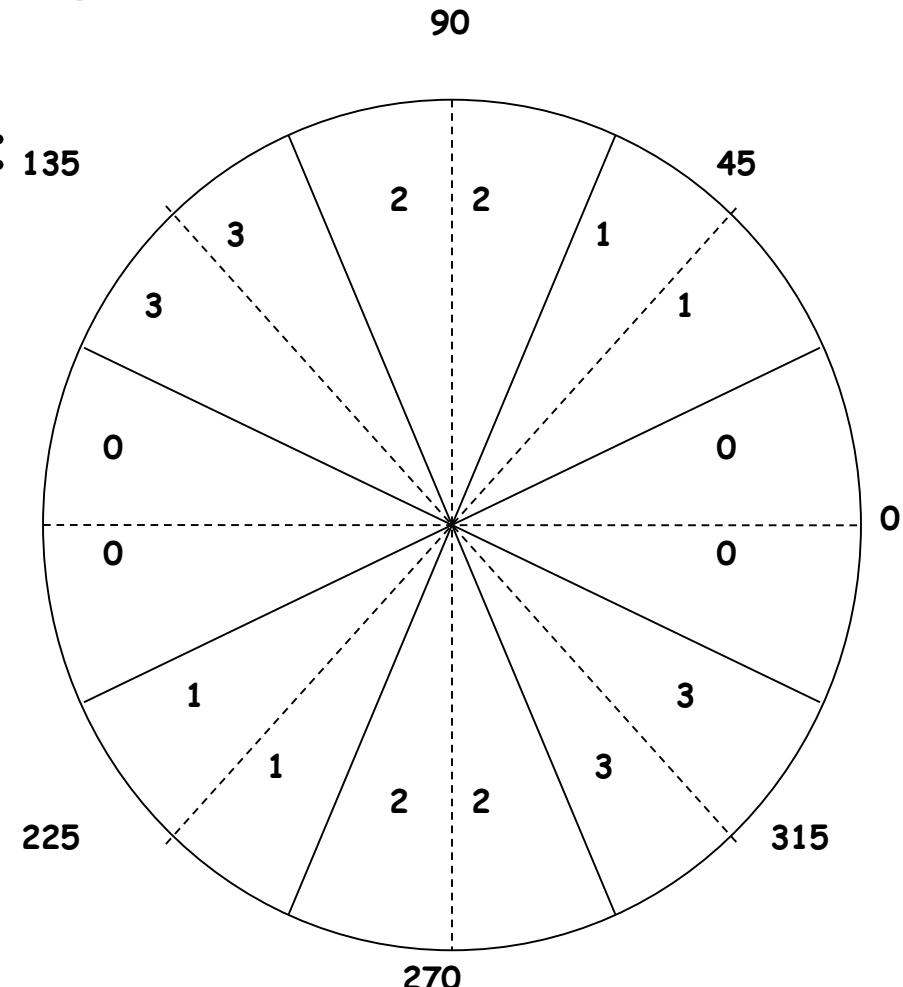
Canny边缘检测算法：

■ 梯度幅度值和方向角分别是：

$$M[i, j] = \sqrt{P^2[i, j] + Q^2[i, j]}$$

$$\theta[i, j] = \arctan(Q[i, j], P[i, j]) \bmod 180$$

其中反正切函数中有两个参数，其表示一个角度，此角度的取值范围是整个圆周范围。



梯度角度 $\theta$ 范围从弧度 $-\pi$ 到 $\pi$ ，然后把它近似到四个方向，分别代表水平，垂直和两个对角线方向

### 3.5.3 二阶边缘检测

#### Canny边缘检测算法：

■**非极大值抑制**：把混子赶走（非极值）

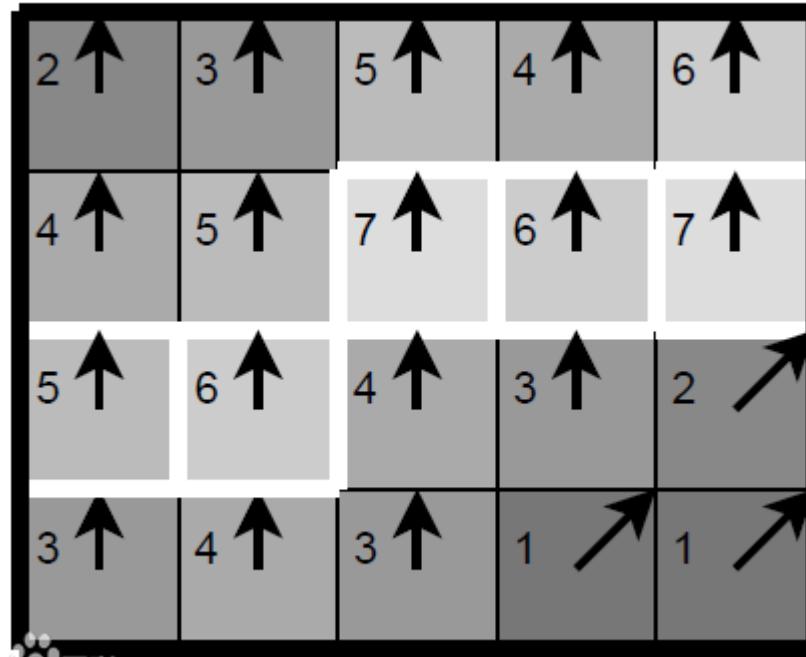
难点

- 算法使用一个 $3 \times 3$ 邻域作用在幅值阵列 $M[i, j]$ 的所有点上；
- 每一个点上，邻域的中心像素 $M[i, j]$ 与沿着梯度线的两个元素进行比较，其中梯度线是由邻域的中心点处的扇区值 $\zeta[i, j]$ 给出。
- 如果在邻域中心点处的幅值 $M[i, j]$ 不比梯度线方向上的两个相邻点幅值大，则 $M[i, j]$ 赋值为零，否则维持原值；
- 此过程可以把 $M[i, j]$ 宽屋脊带细化成只有一个像素点宽，即保留屋脊的高度值。

### 3.5.3 二阶边缘检测

Canny边缘检测算法：

■ 非极大值抑制



Baidu 百度

以第二排第三个像素点为例，由于梯度方向向上，则将这一点的强度(7)与其上下两个像素点的强度(5和4)比较，由于这一点强度最大，则保留

### 3.5.3 二阶边缘检测

Canny边缘检测算法：

■非极大值抑制



### 3.5.3 二阶边缘检测

Canny边缘检测算法：

■ 双阈值(Double Thresholding)

Canny算法应用双阈值，即一个高阈值和一个低阈值来区分边缘像素。

如果边缘像素点梯度值大于高阈值，则被认为是强边缘点。

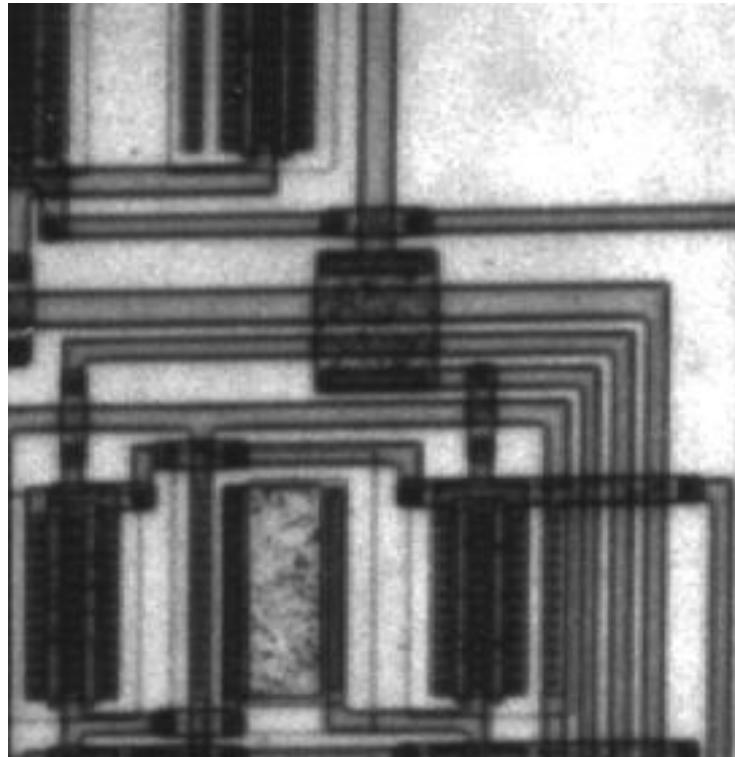
如果边缘梯度值小于高阈值，大于低阈值，则标记为弱边缘点，小于低阈值的点则被抑制掉。



### 3.5.3 二阶边缘检测

Canny边缘检测算法结果对比：

印制电路图



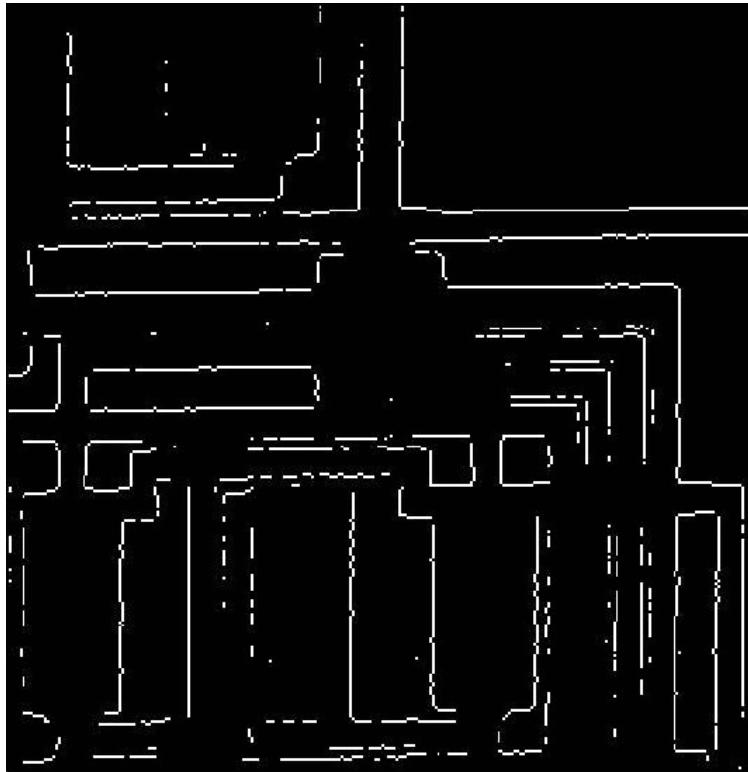
Roberts



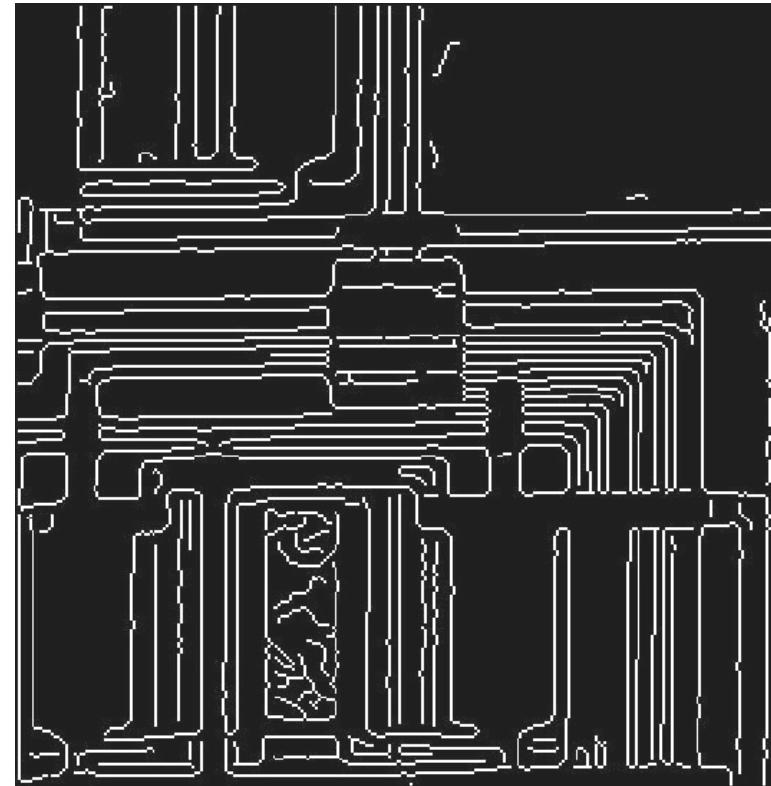
### 3.5.3 二阶边缘检测

Canny边缘检测算法结果对比：

Prewitt



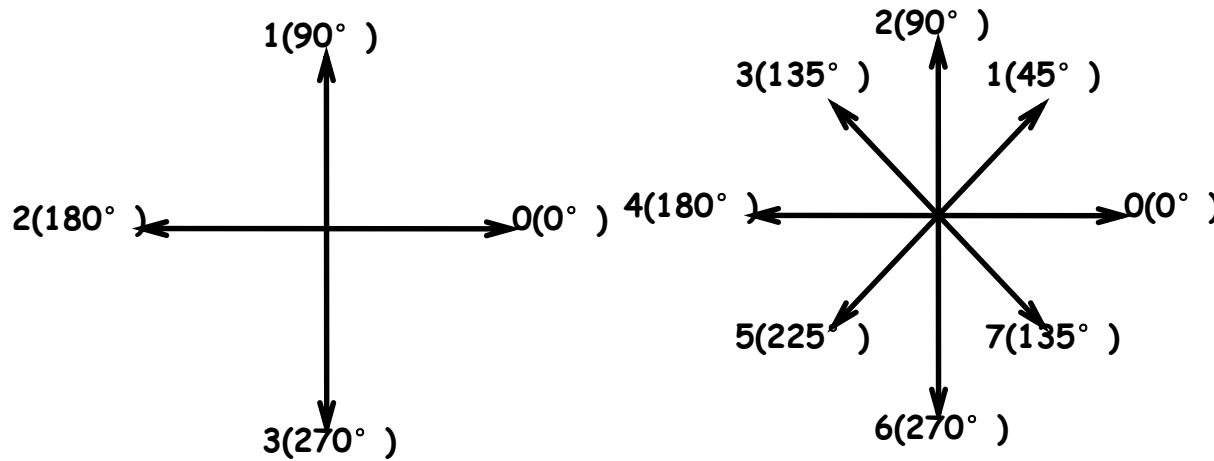
Canny



## 3.5.4 轮廓表示

### ■ 链码轮廓表示：

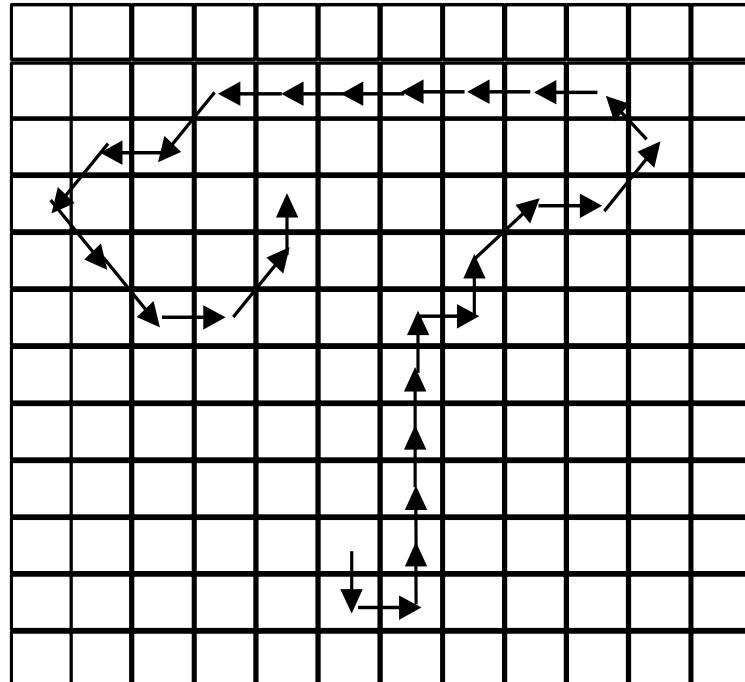
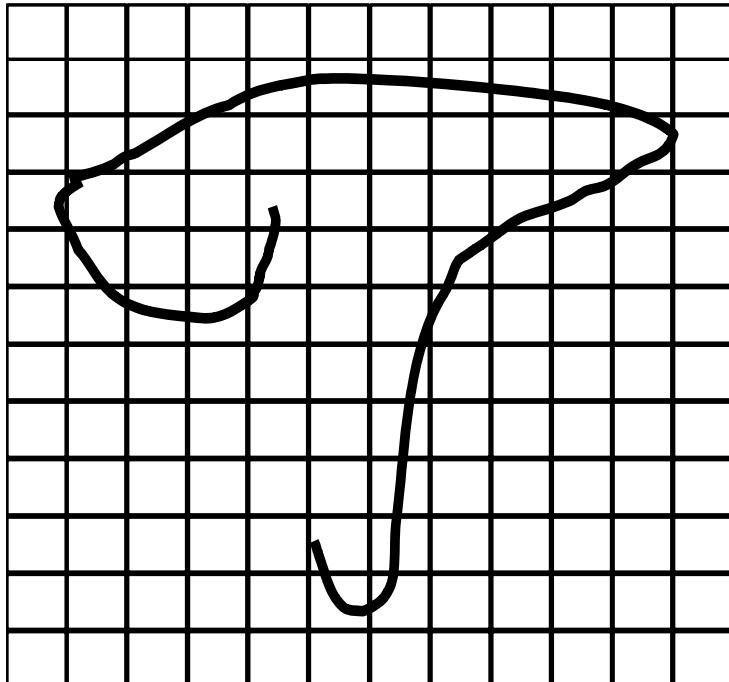
- 链码：沿着轮廓记录边缘表的一种表示方法。
- 链码：规定了边缘表中每一个边缘点所对应的轮廓方向，轮廓方向被量化为4邻点链码或8邻点链码



数字代表方向

## 3.5.4 轮廓表示

链码轮廓表示：逆时针，以8邻点链码表示得：

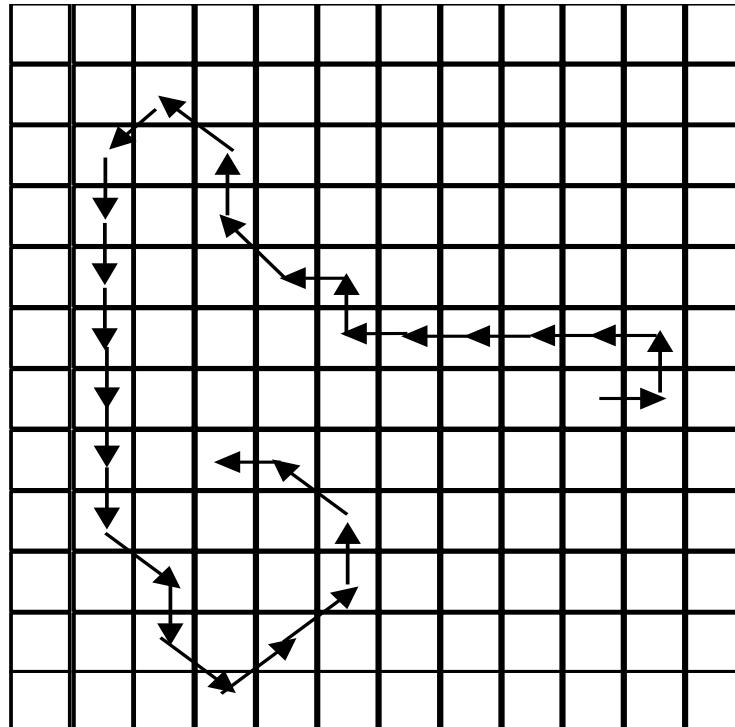
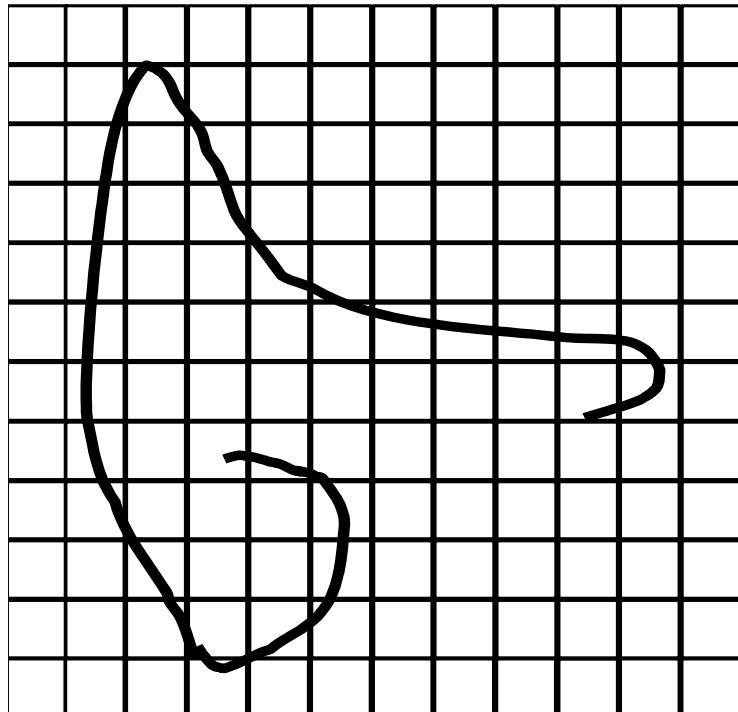


曲线链码：

6 0 2 2 2 2 2 0 2 1 0 1 3 4 4 4 4 4 4 4 5 4 5 7 7 0 1 2

## 3.5.4 轮廓表示

链码轮廓表示：逆时针，以8邻点链码表示得：



曲线链码：

0 2 4 4 4 4 4 2 4 3 2 3 5 6 6 6 6 6 6 7 6 7 1 1 2 3 4

## 3.5.4 轮廓表示

### ■ 链码性质：

- 链码的微分（差分链码）可由原链码的一阶差分求得。差分链码具有旋转不变性，可用于手写体识别。

曲线链码A:

6 0 2 2 2 2 2 0 2 1 0 1 3 4 4 4 4 4 4 4 5 4 5 7 7 0 1 2

差分链码A:

2 2 0 0 0 0 6 2 7 7 1 2 1 0 0 0 0 0 1 7 1 2 0 1 1 1

曲线链码B:

0 2 4 4 4 4 4 2 4 3 2 3 5 6 6 6 6 6 6 7 6 7 1 1 2 3 4

差分链码B:

2 2 0 0 0 0 6 2 7 7 1 2 1 0 0 0 0 0 1 7 1 2 0 1 1 1

# 如何对边缘结果进行兴趣区域搜索？



(a).原图像。



(a).Canny 算子进行边缘检测结果。



(b).Hough 变换检测直线。

### 3.5.5 Hough变换

- Hough变换概述
- Hough变换是利用图像全局特性而将边缘像素连接起来组成区域封闭边界的一种方法。
- 在预先知道区域形状的条件下，利用Hough变换可以方便的得到边界曲线而将不连续的边缘像素点连接起来。
- 主要优点是受噪声和曲线间断的影响较小。
- Hough变换的主要思想就是在已知形状的条件下，将曲线从图像坐标系变换到参数坐标系，通过研究参数坐标系下的一些特征，得到图像坐标系下的表示。

## 3.5.5 Hough变换

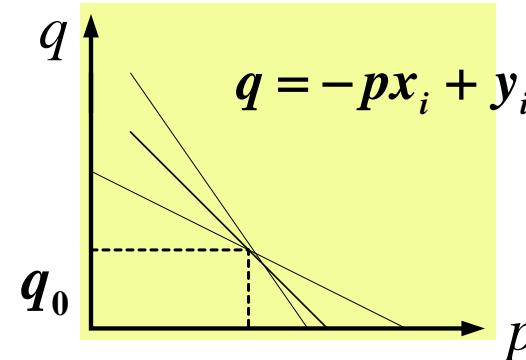
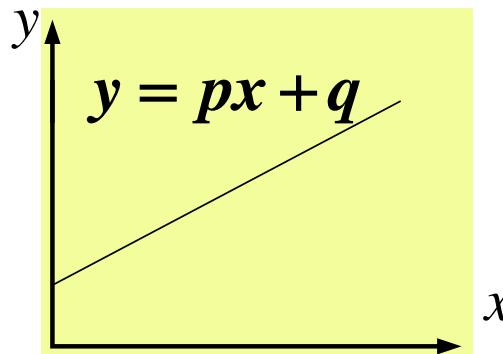
### ■ 直线的Hough 变换

#### ▪ 直线的斜率和截距表达式

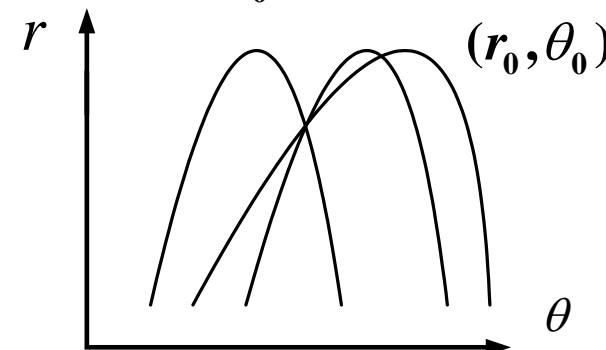
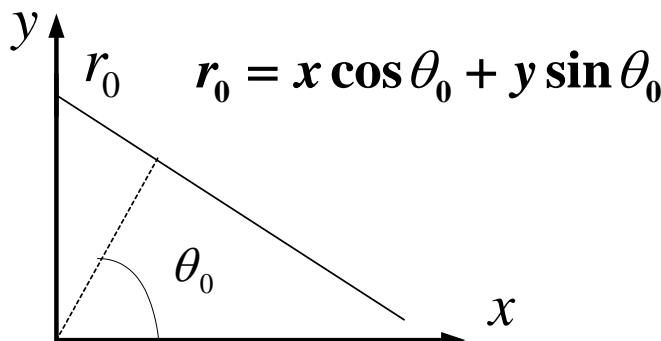
$$y = px + q \quad \text{所以} \quad q = -px_i + y_i$$

#### ▪ 直线的极坐标系表示式

$$r = x_i \cos \theta + y_i \sin \theta$$



直角坐标



极坐标

### 3.5.5 Hough变换

#### ■ 直线Hough变换

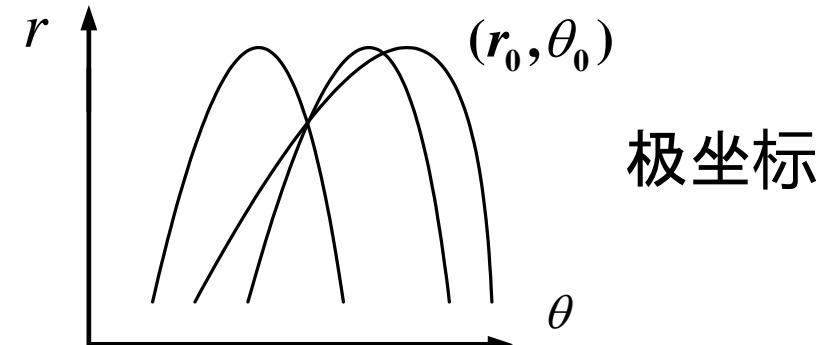
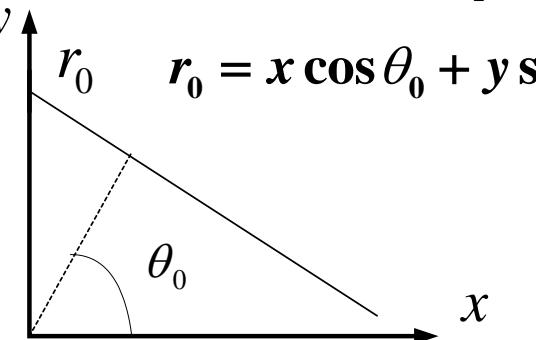
- 在图像空间中共线的点在参数空间中里相交的线。反过来，在参数空间中相交于一个点的所有直线在图像空间中都是共线的点与之对应；
- 点—线的对偶性；
- Hough变换通过这种对偶性把在图像空间中的检测问题转换到参数空间里进行，通过参数空间里的简单累加统计完成检测。
- 直线Hough变换具体累加统计过程是使用一个2D累加数组进行。

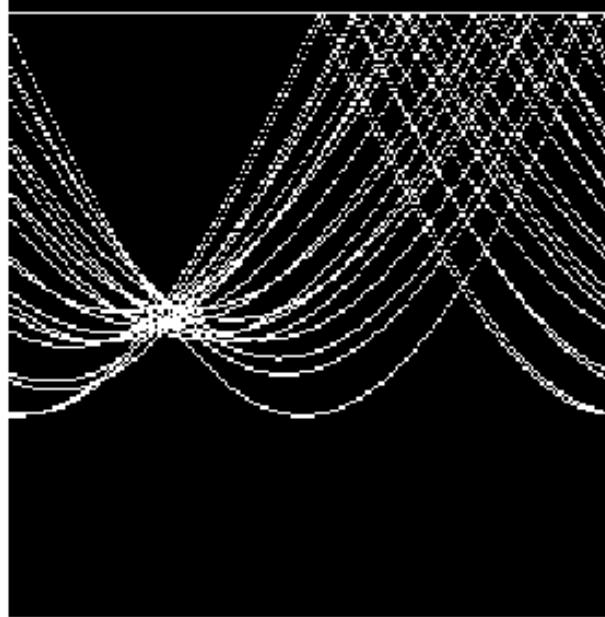
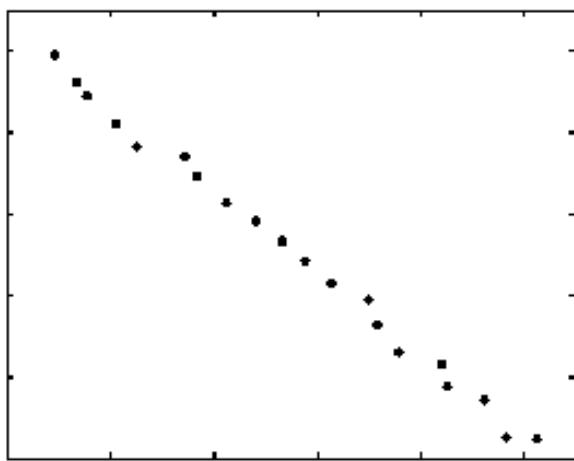
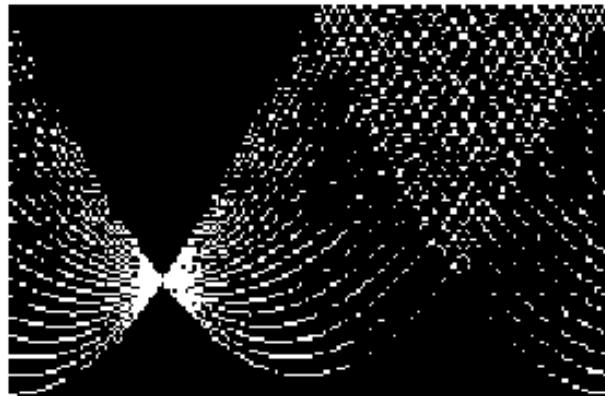
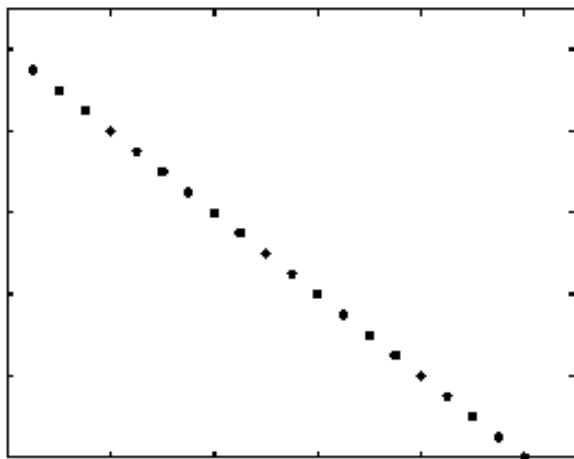
### 3.5.5 Hough变换

- 累加统计:
- 在参数空间PQ中建立一个2D的累加数组A(p,q), 其中p,q的变化范围是 $[p_{\min}, p_{\max}]$ ,  $[q_{\min}, q_{\max}]$ , 分别是斜率和截距的取值范围。
- 开始时置数组A为零, 然后对每一个图像空间中的给定点, 让p取遍P轴上的所有可能值, 根据公式 $q = -px_i + y_i$ 计算对应的q。
- 再根据p和q的值对A累加 $A(p,q) = A(p,q) + 1$ ;
- 累加结束后, 根据A(p,q)的值可以判断有多少点共线, 即A(p,q)的值就是在(p,q)处共线点的个数。

### 3.5.5 Hough变换

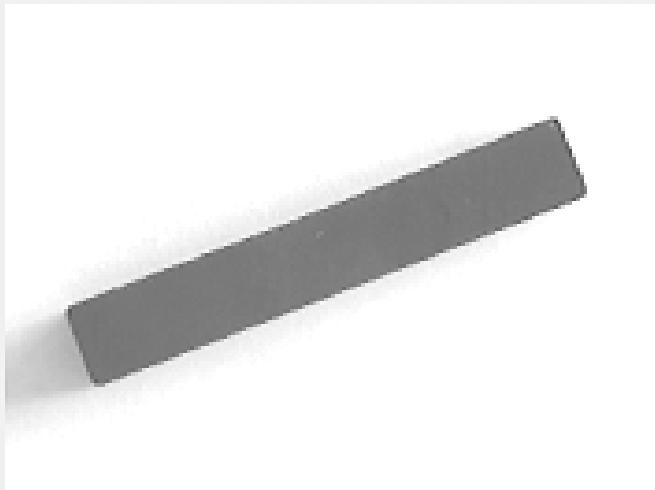
- 这种方法的准确性取决于累加数组的尺寸。
- 如果把P轴划分成K份，那么对每一个点 $(x_k, y_k)$ 计算可得q的K个值。如果图像中有n个点，则需要 $nK$ 次运算，小于 $n^2$ 次。
- 如果直线接近于垂直方向，则由于p和q的值都接近无穷而使计算量陡增，所以使用直线的极坐标方程表示  $r = x_i \cos \theta + y_i \sin \theta$
- 则原图像空间中的点对应着新参数空间中的曲线，那么原来的点—直线对偶变成了点—曲线对偶。
- 累加器数组是 $A(\theta, \lambda)$



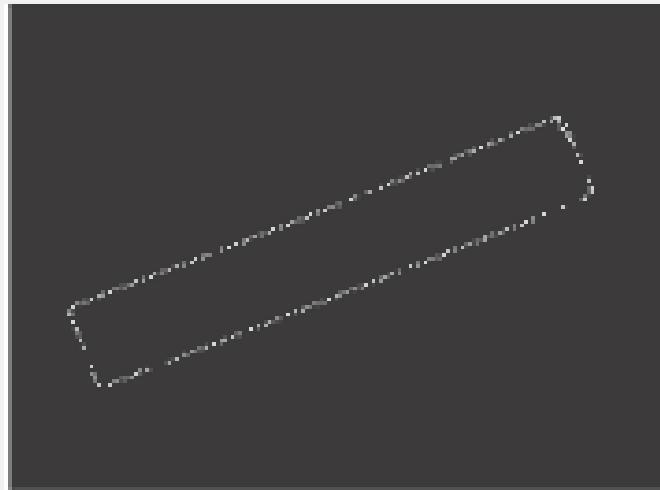


# Hough变换应用于尺寸测量

原始图像

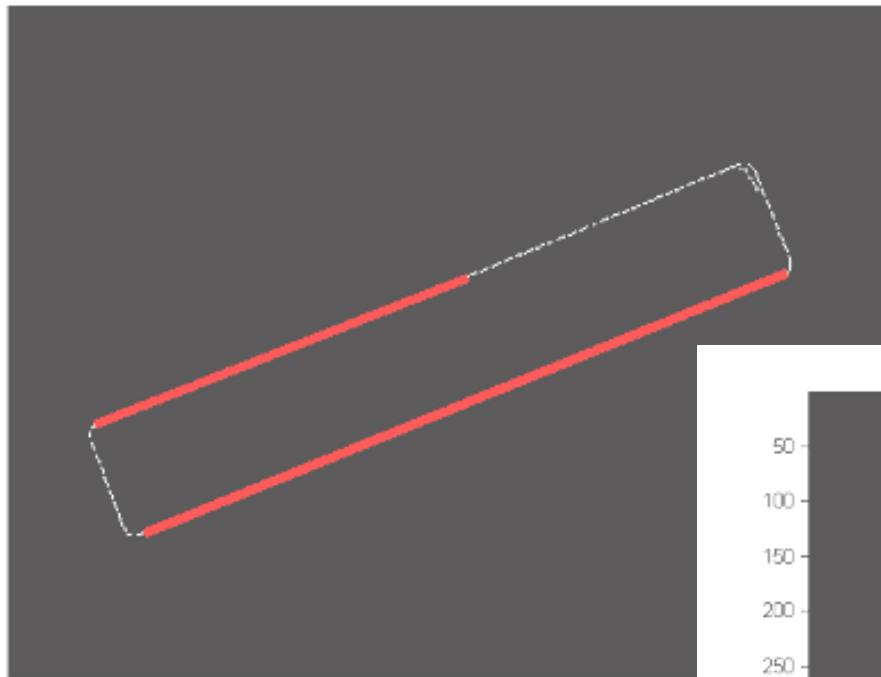


Canny算子边缘提取结果

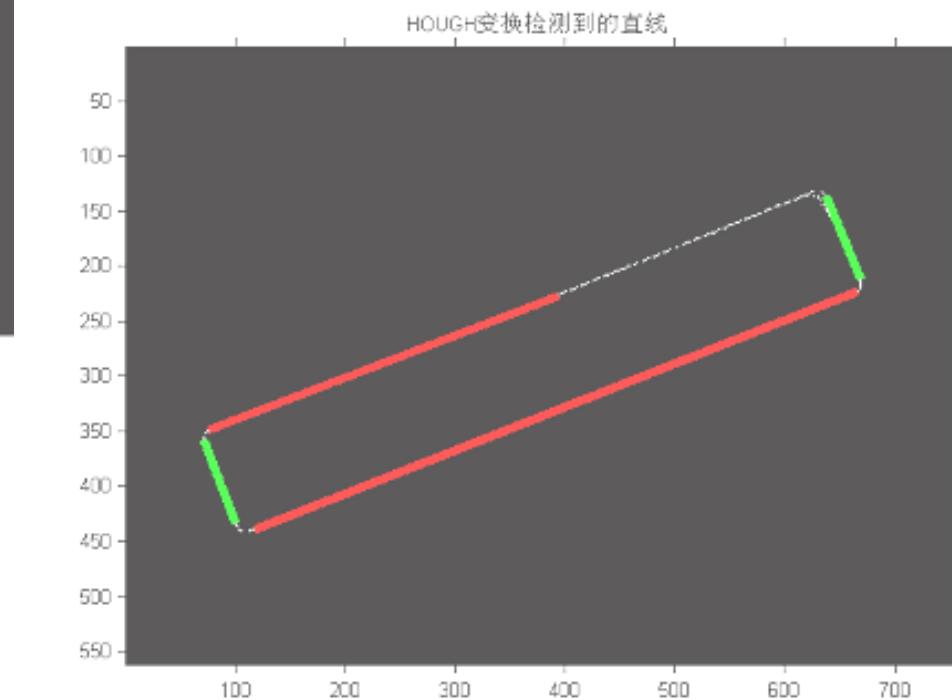


# Hough变换应用于尺寸测量

HOUGH变换检测到的直线



HOUGH变换检测到的直线



## 3.5.6 二值图像运算

### 目录:

- ❖ 数字图像概念
- ❖ 数字图像处理基本步骤
- ❖ 图像预处理
- ❖ 图像区域表示
- ❖ 二值图像运算
- ❖ 图像特征比对
- ❖ OpenCV平台使用

# 1 什么是二值图像，为什么使用二值图像？

**原因：**

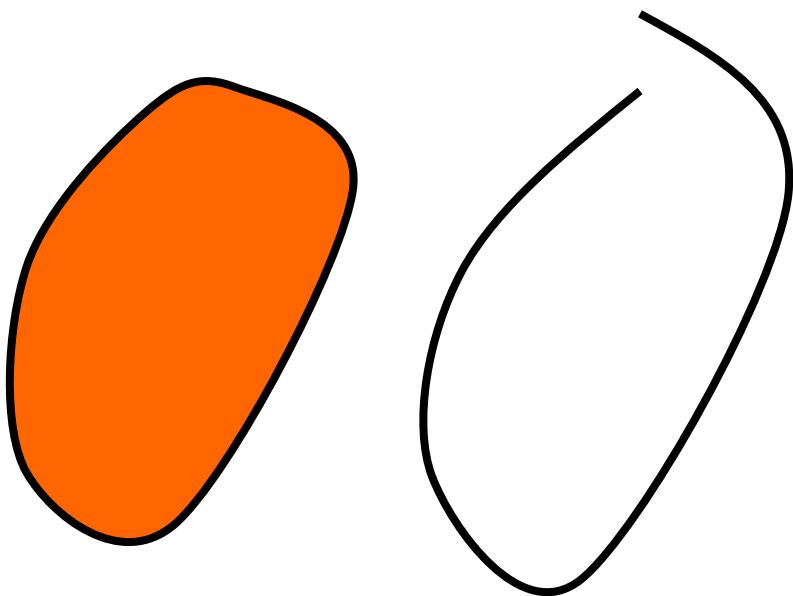
- 人类在理解只有两个灰度组成的线条、轮廓影像或其它图像是没有任何困难。
- 二值图像计算算法简单，容易理解和实现，计算速度快。
- 二值图像所需内存小，对计算机设备要求低  
**256灰度级的视觉系统内存是二值图像的8倍**
- 许多二值视觉系统的技术也可以用于灰度视觉系统

## 2 如何得到二值图像？

- 最简单的方法是通过**阈值分割**的方法得到二值图像，阈值取决于照明条件和物体的反射特性。
- **算法过程：**比较图像灰度与某一灰度值的关系可以分成两部分，第一部分小于阈值，将此部分的灰度值设定为0；第二部分大于阈值，将此部分的灰度值设定为1。
- 由此得到的经过阈值处理图像因为只有两个灰度值故称为二值图像。

# 二值化的方法

- ✓ 基于区域的方法
- ✓ 基于边缘检测的轮廓方法



### 3 如何表示二值图像？

#### 几何特性

- 尺寸
- 位置
- 方向
- 密度

## 4 基本定义与如何计算二值图像？

- 基本定义：**近邻、路径、前景、连通性、连通成分。
- 计算：**连通成分标记、边界跟踪、距离测量、中轴、细化、扩展与收缩
- 形态学算子：**膨胀、腐蚀、开闭运算、应用算法等

## 基本算法过程

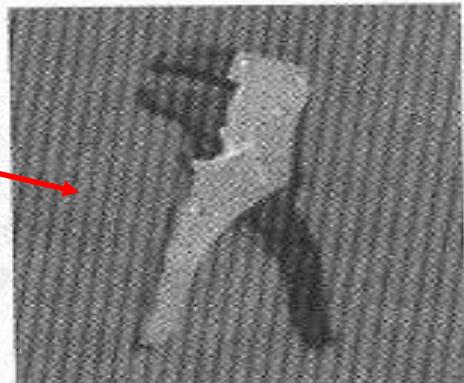
- 设 $(x,y)$ 是二维图像的平面坐标，图像灰度级的取值范围是 $G = \{0, 1, 2, \dots, L-1\}$ （习惯上0表示最暗的像素点， $L-1$ 代表最亮的像素点），位于坐标点 $(x,y)$ 上的像素点的灰度级表示 $f(x,y)$ 。
- 设 $t$ 为分割阈值，于是图像函数在阈值 $t$ 上的分割结果可以有如下表示，

$$f_t(x, y) = \begin{cases} b_0 & f(x, y) < t \\ b_1 & f(x, y) \geq t \end{cases}$$

- 阈值分割算法实际就是按某个准则函数求最优阈值的过程

# 阈值分割图示：

原始图像



(a)

$T=100$



(b)

阈值  $T=128$



(c)

$T_1=100$   
 $T_2=128$



(d)

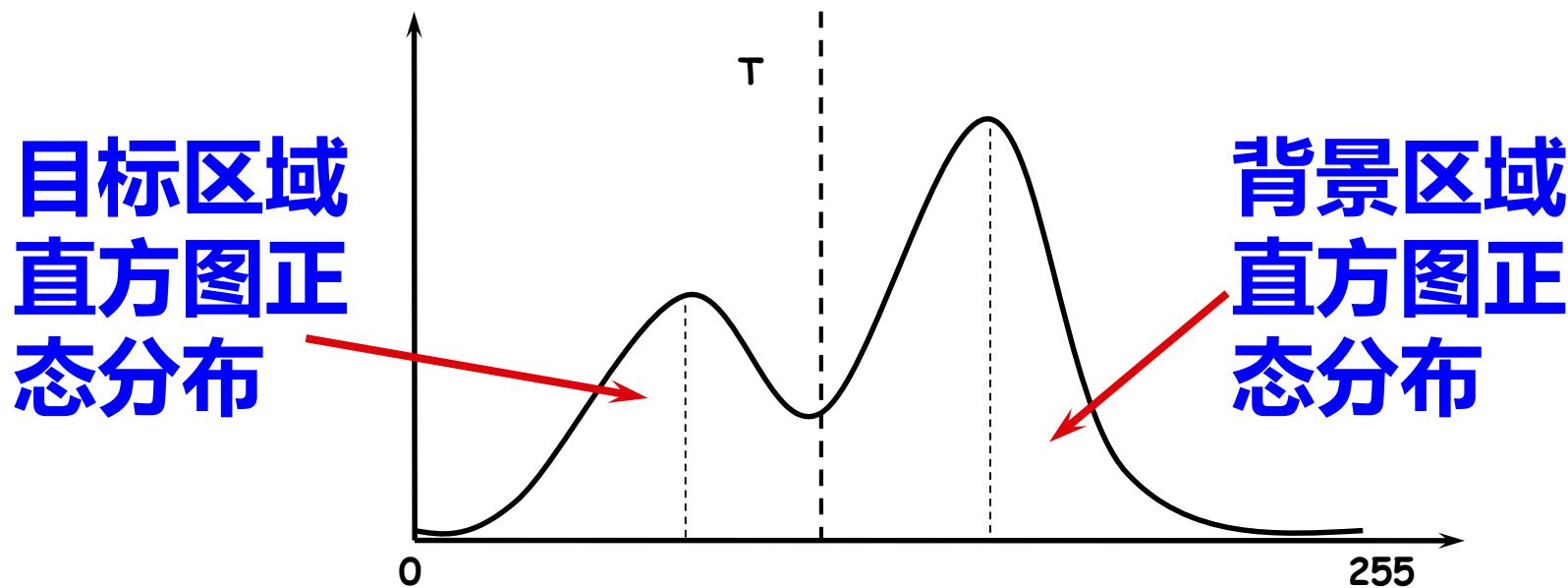
图 3.1 一幅灰度图象和使用不同阈值得到的二值图象结果

(a) 原始灰度图象; (b) 阈值  $T = 100$ ; (c)  $T = 128$ ; (d)  $T_1 = 100$ ;  $T_2 = 128$

# 自动阈值分割

## 直方图方法和直方图变换法

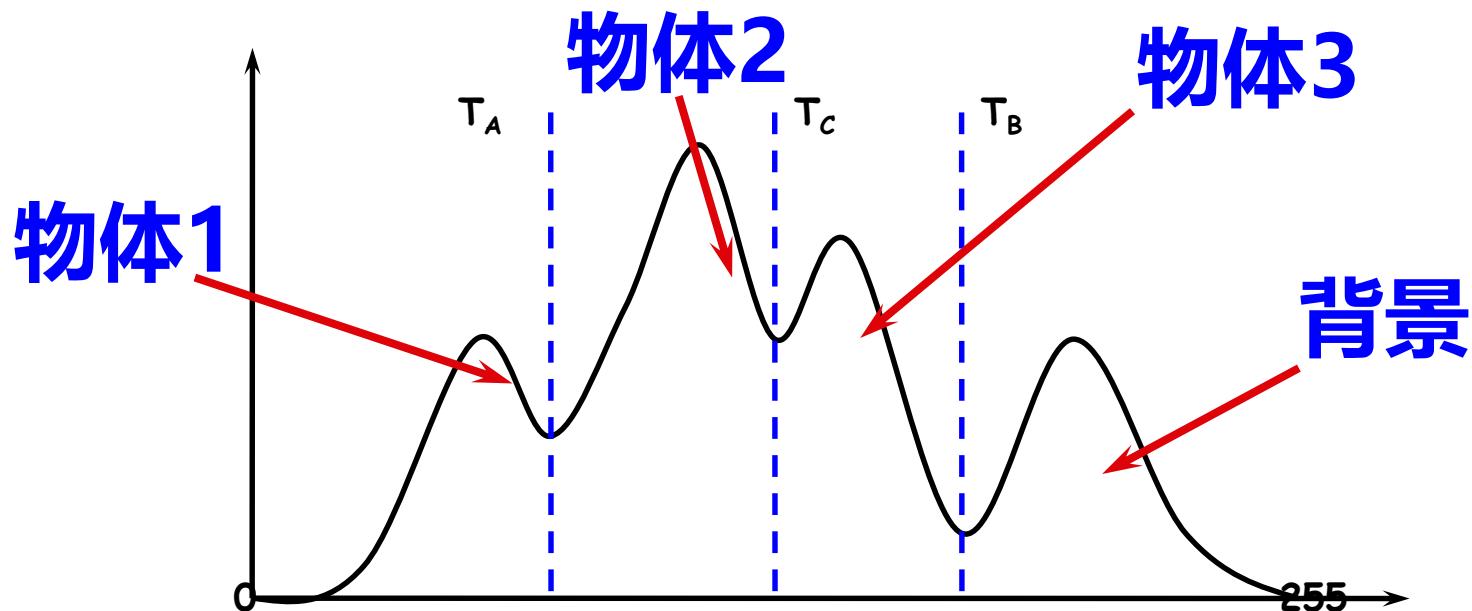
- 图像直方图会出现两个分离峰值，那么直方图中分割阈值的选择在两个波峰之间的波谷。



# 自动阈值分割

## 直方图方法和直方图变换法

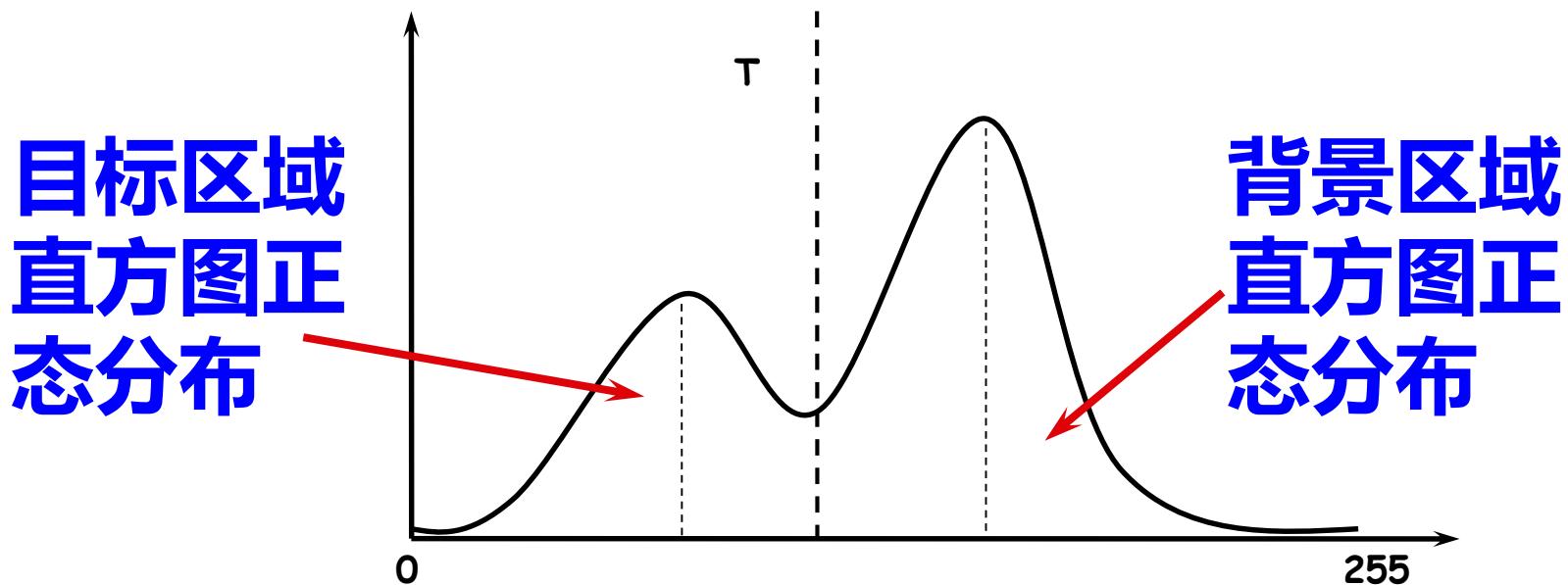
- 双峰直方图可以推广到具有不同灰度值的多个物体图像中。
- 图像中有多个物体且有不同的灰度分布，如图：



# 自动阈值分割

## 直方图方法和直方图变换法

- 直方图变换：如果目标区域和背景区域之间的谷不够深，以至于无法选择一个合适的阈值，则就要对直方图进行变换。
- 常用的方法是将像素的梯度值作为直方图加权的方法



# 自动阈值分割

## 直方图方法的局限性

- 如果场景中不同部分有不同的照明，那么即使图像中仅包含有一个物体也无法用一个阈值来分割图像。
- 基于直方图的阈值分割方法没有利用图像强度的空间信息，在本质上存在局限性。

因此，需要通过边缘检测的方式得到跳变像素点，进而做二值化处理。

## 3.5.6 二值图像运算

### ■ 二值图像中的一些基本定义

近邻

连通性

前景与背景

边界

### ■ 连通成分标记算法

### ■ 区域边界跟踪

### ■ 中轴、细化和扩展与收缩

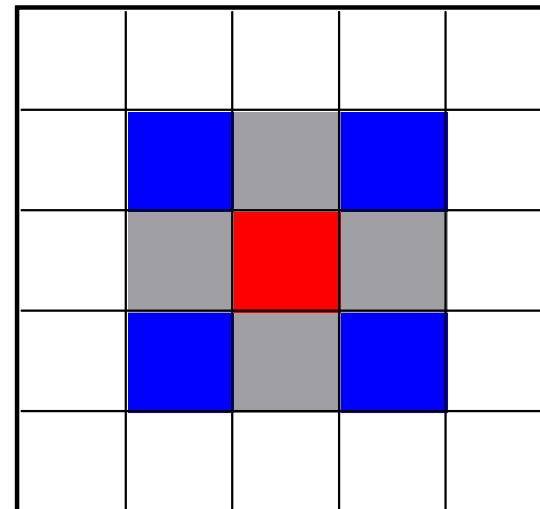
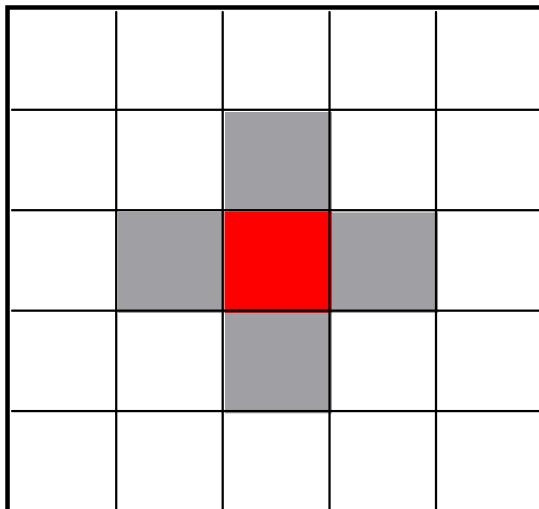
### ■ 形态学算子

# 二值图像运算

## 基本定义

### 1 近邻 (neighbor)

- 如果两个像素只有公共边界，则称它们互为4邻点；
- 如果两个像素至少共享一个顶角，则称它们互为8邻点；

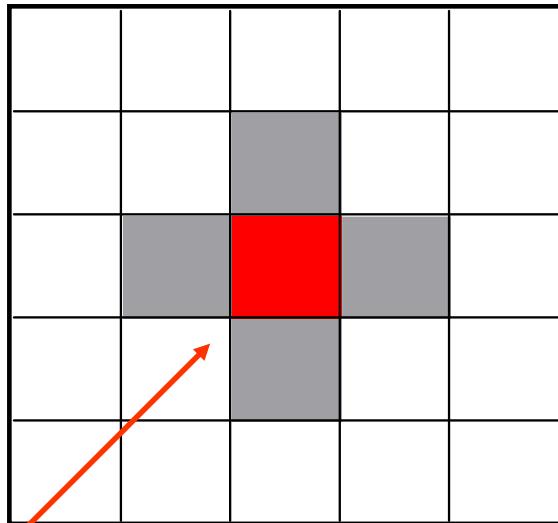


# 二值图像运算

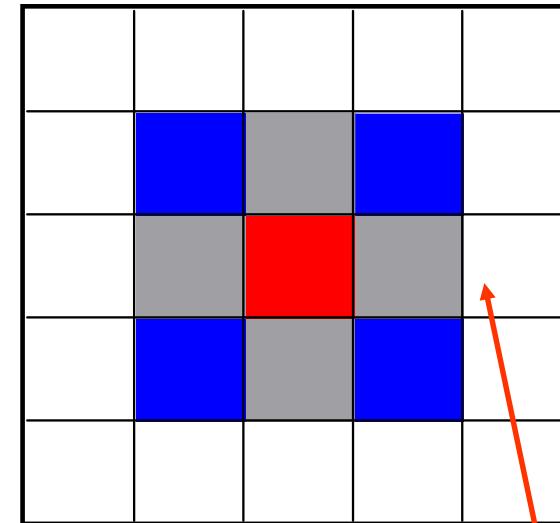
## 基本定义

### 1 近邻 (neighbor)

- 一个像素被认为与它的4邻点是4连通关系；
- 一个像素被认为与它的8邻点是8连通关系；



4邻点与4连通



8邻点与8连通

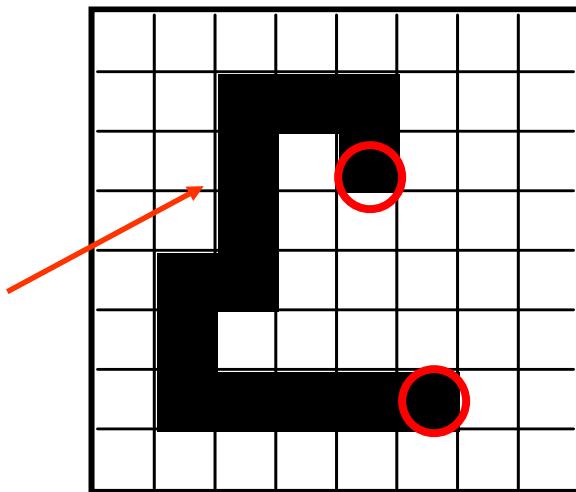
# 二值图像运算

## 基本定义

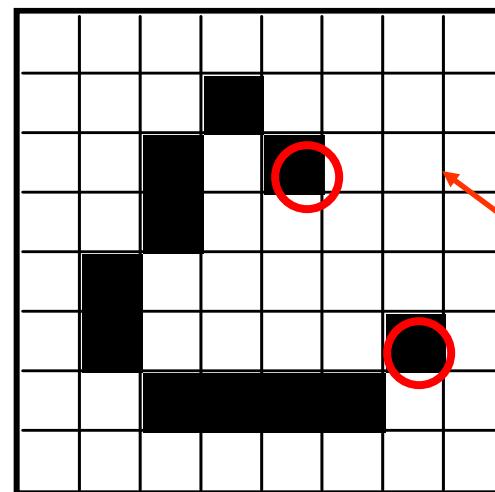
### 2 路径 (path)

- 概念：一个像素到另一个像素的路径指一个像素序列，此序列中每一个像素互为邻点。

4路  
径



8路  
径



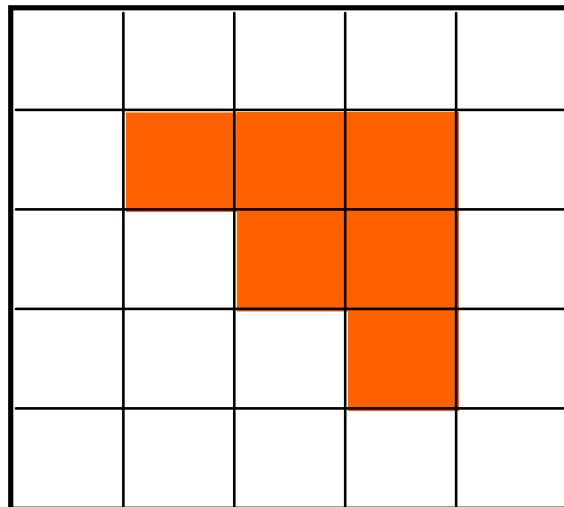
- 如果邻点关系是4连通的，则路径是4路径
- 如果邻点关系是8连通的，则路径是8路径

# 二值图像运算

## 基本定义

### 3 前景 (foreground)

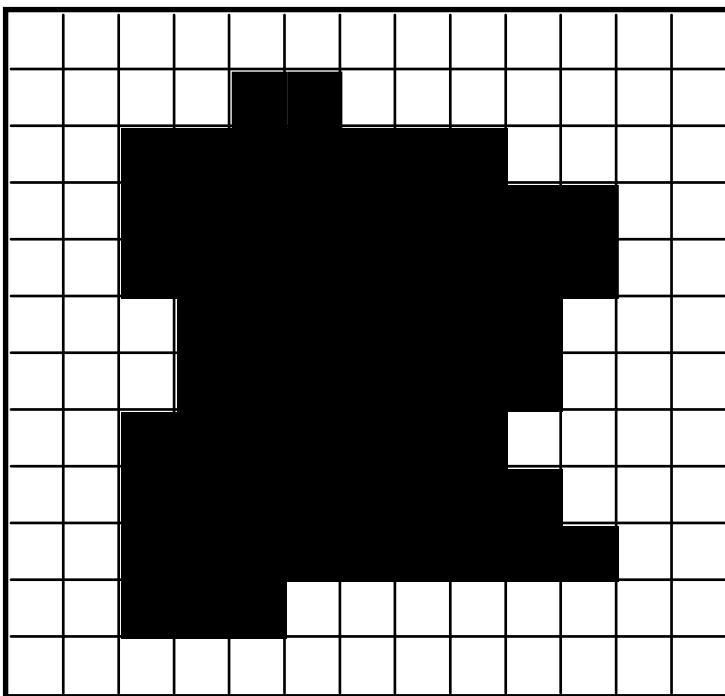
- 图像中值为1的全部像素的集合称为前景，前景目标检测中的前景可以认为是检测到的目标。
- 用S表示



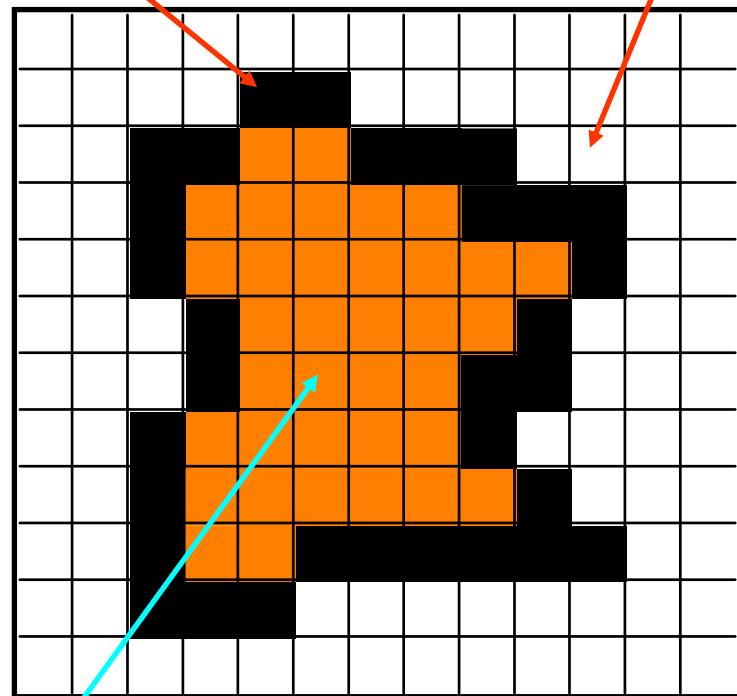
# 二值图像运算

- 7 **边界**:  $s$  的边界是  $s$  (前景) 中与 (背景) 有 4 连通关系的像素集合, 用  $s'$  表示
- 8 **内部**:  $s$  中不属于它的边界的像素集合, 用  $s - s'$  表示
- 9 **包围**: 如果从  $s$  中任意一点到 **图像边界** 的 4 路径必须与区域  $T$  相交, 则区域  $T$  包围区域  $s$  ( $s$  在  $T$  内)。

# 二值图像运算



边界



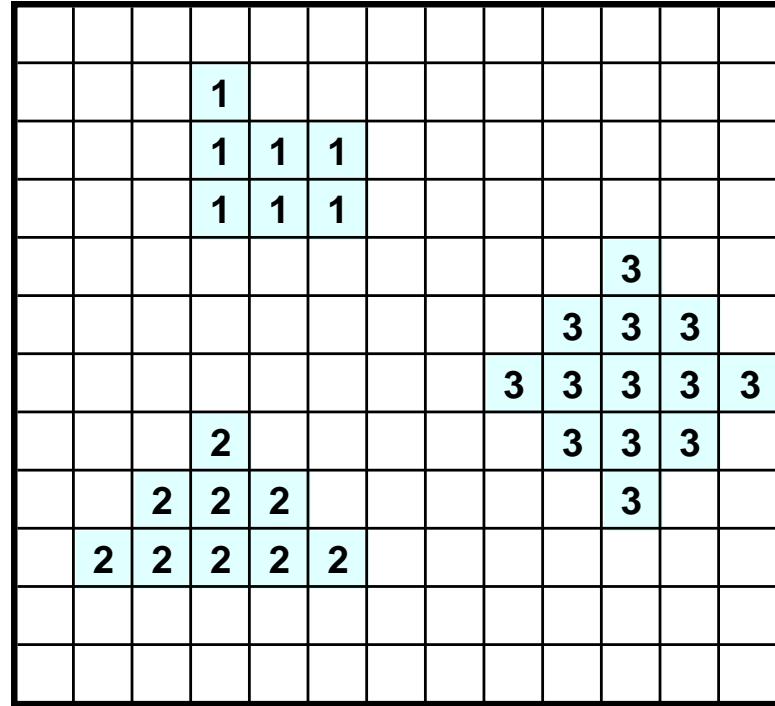
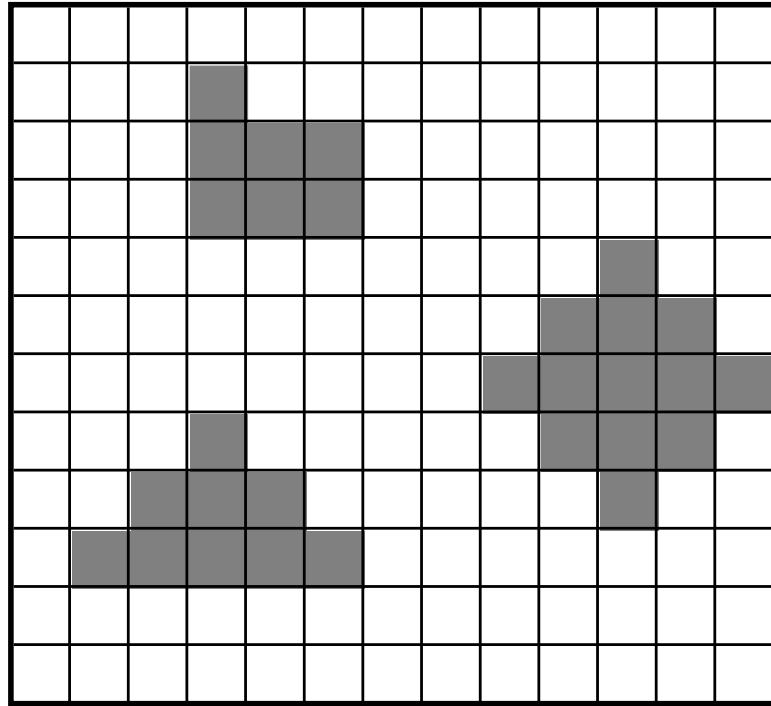
包围

内部

# 二值图像运算

## 连通成分标记

- **连通标记算法：**找出图像中的所有连通成分，并对同一连通成分中所有点分配同一标记。
- **常用连通成分标记算法：**递归算法，序贯算法



# 二值图像运算

## 连通成分标记

- 递归算法

- ✓ 扫描图像，找到没有标记的1点，给它分配一个新的标记L

- ✓ 递归分配标记L给1点的邻点（4邻点或8邻点）

- ✓ 如果不存在未标记的点，则停止

- ✓ 返回第一步

# 二值图像运算

## 中轴

- 定义：如果对  $S$  中像素  $[i, j]$  的所有邻点  $[u, v]$  有下式成立：

$$d([i, j], \bar{S}) \geq d([u, v], \bar{S})$$

则  $S$  中的像素  $[i, j]$  到  $\bar{S}$  的距离  $d([i, j], \bar{S})$  是局部最大值， $S$  中所有到  $\bar{S}$  的距离是局部最大值的像素点集合称为对称轴或中轴，通常记为  $S^*$ 。

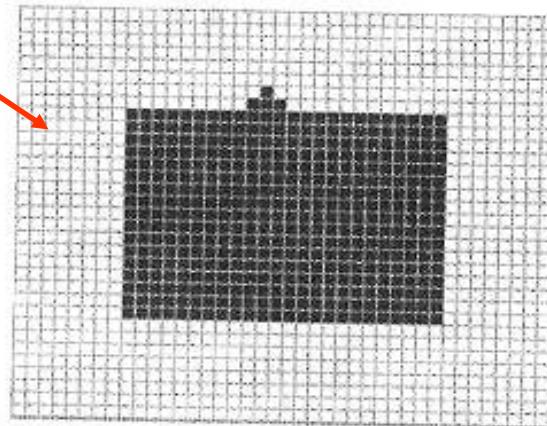
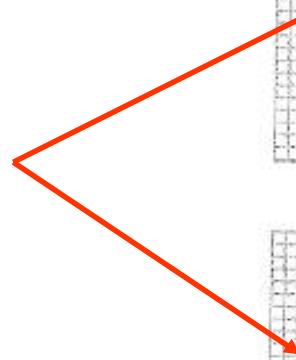
- 中轴可作为物体的一种简洁表示
- 由  $S^*$  和  $S^*$  中每一点到  $\bar{S}$  的距离能重构原始像素。
- 中轴可以判断某一像素是否在一区域内。

# 二值图像运算

## 中轴

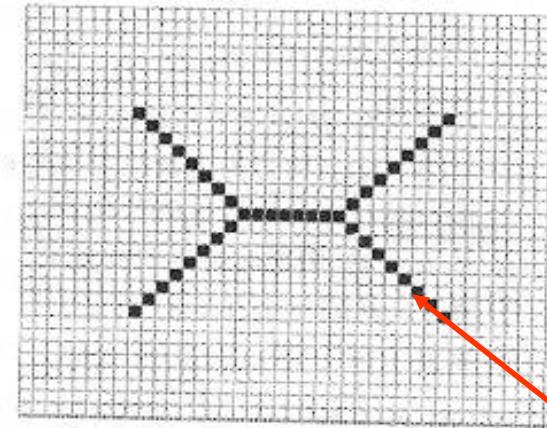
- 图示：

原始图像

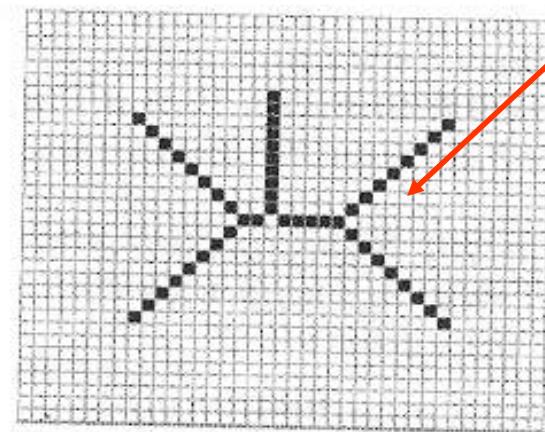


(b)

形象！！！



(a)



中轴

图 3.13 中轴变换举例

# 二值图像运算

## 细化

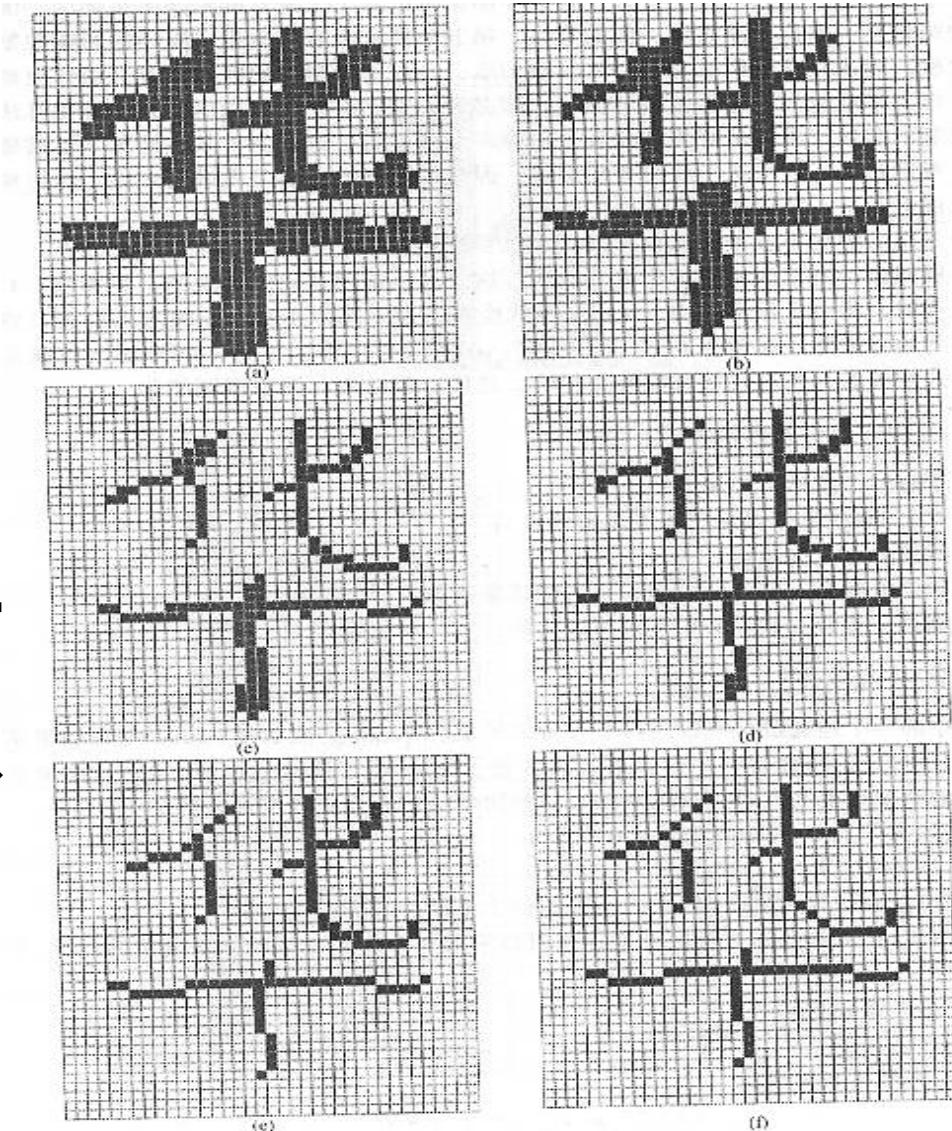
- **定义**: 是图像处理运算，可以把二值图像区域缩成线条，以逼近区域的中心线，称为骨架或核线。
- **目的**: 减少图像成分，只保留区域的最基本信息，以便进一步分析和识别。
- **对象**: 主要对细长形区域有效，主要对文本分析预处理，以便将文本图像中线条图画或字符笔画表示成单像素线条。

# 二值图像运算

## 细化

3×3邻域4邻点迭代算法：

- 对每一个像素，若：
  - 没有上（下\左\右）邻点
  - 不是孤立点或终止线；
  - 去除该像素点不会断开区域，则去除该像素点；
  - 重复这一步骤直到没有像素点可以去除。



# 二值图像运算

## 扩展与收缩

定义：

- 扩展：如果某一连通成分可以变化，使得一些背景像素变成1；
- 收缩：如果物体像素点全方位的消减或变为0；

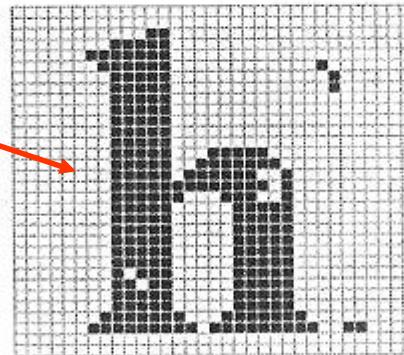
最简单的实现方法：

- 扩展：如果邻点是1，则将该点从0变为1；
- 收缩：如果邻点是0，则将该点从1变为0；

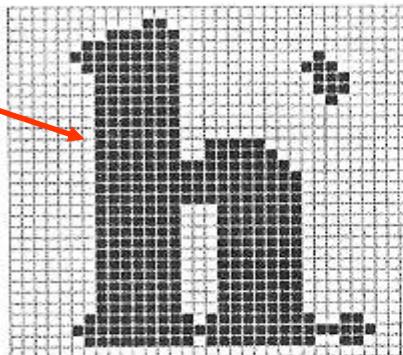
$S^{(k)}$ :  $S$ 扩展 $k$ 倍；     $S^{(-k)}$ :  $S$ 收缩 $k$ 倍

# 二值图像运算

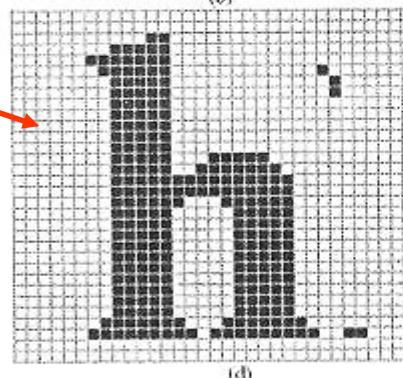
原始噪声图像



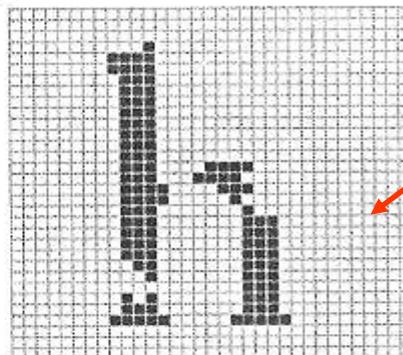
扩展



扩展后收缩

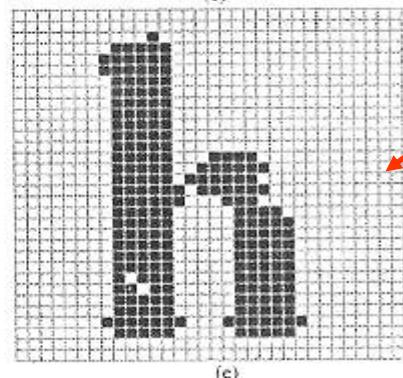


(a)



(c)

收缩后扩展



(d)

(e)

# 形态学算子

## 数学形态学 (Morphology)

- 一般认为是生物学中研究动物和植物的结构（研究生物机体外形和内部结构及其与功能相关的科学）一个分支。
- 数学形态学用具有一定形态的结构元素去度量和提取图像中的对应形状以达到对图像分析和识别的目的。
- 二值形态学运算对象是集合。X为图像集合，B为结构元素，运算过程是用B对X进行操作。

# 形态学算子

## 腐蚀 (Erosion)

- **腐蚀目的：**消除目标图像中的无用点（或孤立噪声点）的一个过程，其结果使得剩下的目标比处理前减少了一些像素。
- **腐蚀定义：**

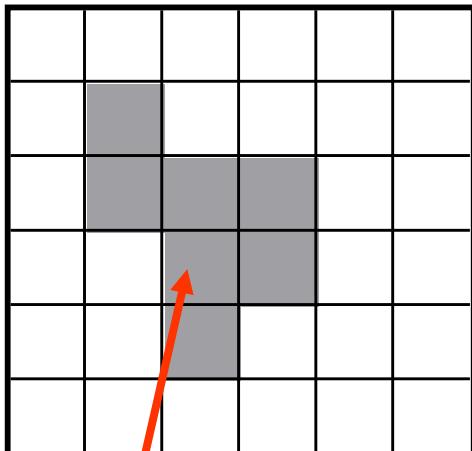
$X$ 用 $B$ 来腐蚀记为 $X \square B$  定义为

$$E = X \square B = \{(x, y) | \hat{B}_{x,y} \subseteq X\}$$

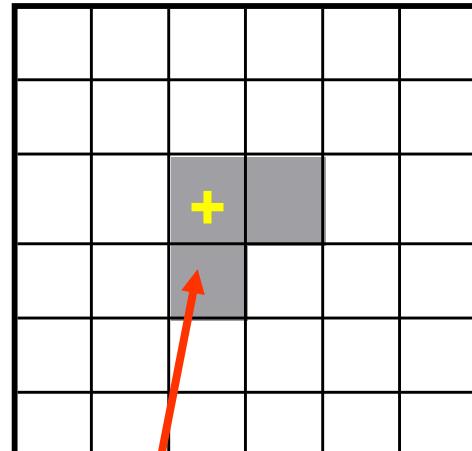
- **腐蚀过程：** $B$ 平移 $(x, y)$ 后仍在集合 $X$ 中的结构元素其参考点的集合。换句话说，用 $B$ 来腐蚀 $X$ 得到的集合是 $B$ 完全包括在集合 $X$ 中时 $B$ 的参考点位置的集合。

# 形态学算子

## 腐蚀

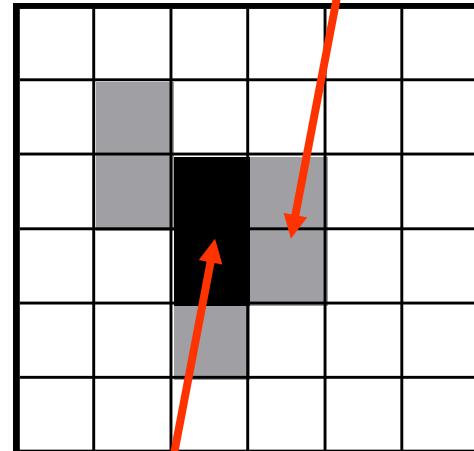


原始图像



腐蚀结构元素

灰色部分为原  
图像部分



黑色部分为腐蚀  
后结果；

# 形态学算子

## 膨胀 (Dilation)

• **膨胀目的：**是腐蚀运算的对偶运算，它是将与目标接触的所有点合并到该目标的过程，其结果是使目标的面积增加了相应数量的像素。膨胀在填补分割后目标中的空洞很有用。

• **膨胀定义：**

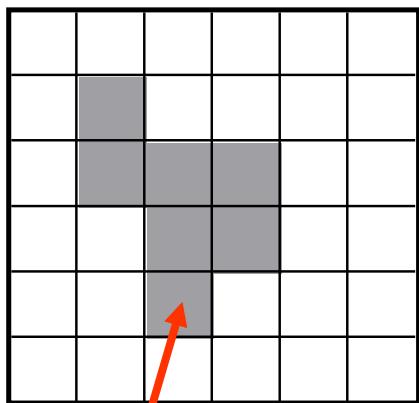
$X$ 用 $B$ 来膨胀记为 $X \oplus B$ 定义为

$$D = X \oplus B = \{(x, y) | [\hat{B}_{x,y} \cap X] \subseteq X\}$$

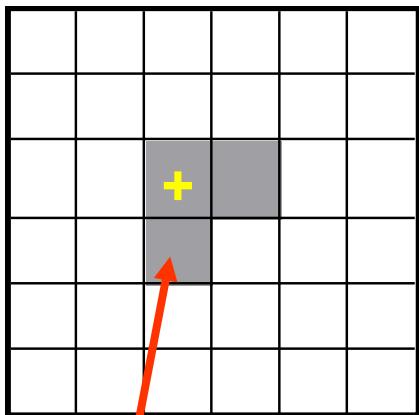
• **膨胀过程：**用 $B$ 来膨胀 $X$ 得到的集合的位移与集合 $X$ 至少有一个非零元素相交时结构元素 $B$ 的**参考点**位置的集合。

# 形态学算子 膨胀

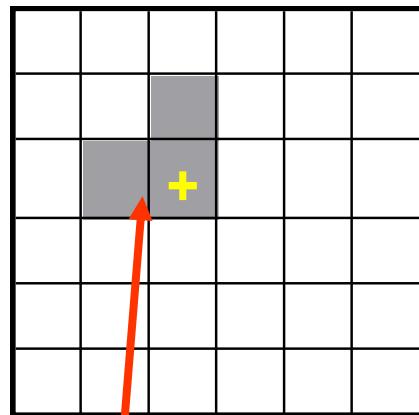
使用结构元素2  
膨胀出的点



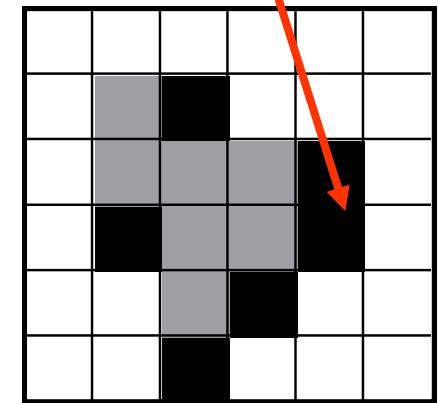
原始图像



结构元素1



结构元素2



# 形态学算子

## 开运算与闭运算

**开运算：**  $A \circ B$

- **过程：**对图像先进行腐蚀然后再膨胀运算；
- **目的：**除去比结构元小的所有区域像素点，而留下其余部分。

**闭运算：**  $A \bullet B$

- **过程：**对图像先进行膨胀然后再腐蚀运算。
- **目的：**填充比结构元小的孔洞和凹状区。

## 3.5.6 二值图像运算

### 目录:

- ❖ 数字图像概念
- ❖ 数字图像处理基本步骤
- ❖ 图像预处理
- ❖ 图像区域表示
- ❖ 二值图像运算
- ❖ 图像特征比对
- ❖ OpenCV平台使用

# 练习

---

- ❖ 实验1: 基于颜色特征的目标识别与跟踪

Thanks