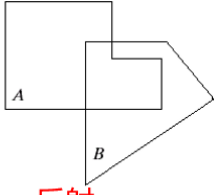
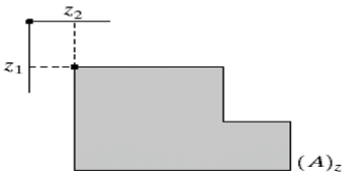
形态学中的附加定义



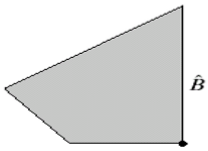
Translation 平移

$$(A)_z = \{c \mid c = a + z, \text{for } a \in A\}$$



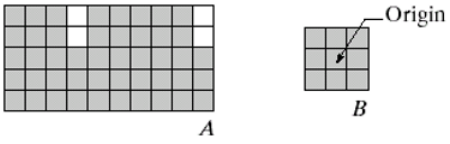
Reflection 反射

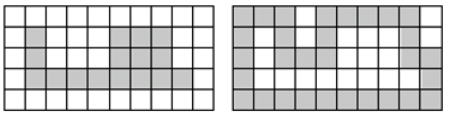
$$\hat{B} = \{w \mid w = -b, \text{ for } b \in B\}$$



应用: Boundary extraction 边界提取

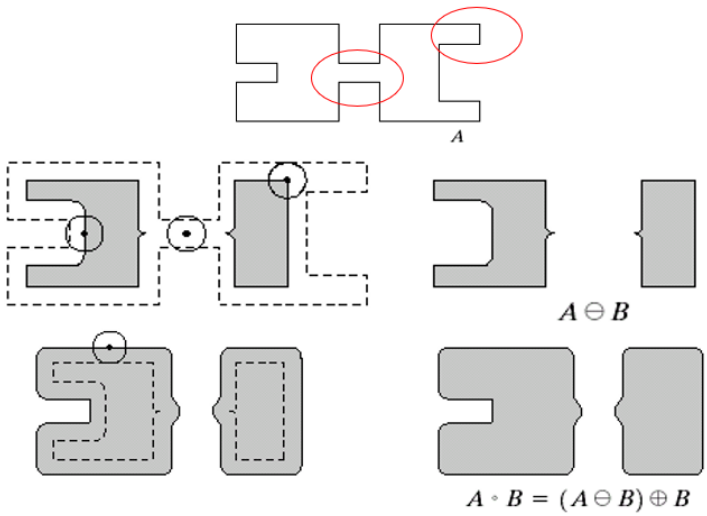
- Extract boundary of a set A:
  - First erode A (make A smaller)
  - A - erode(A)



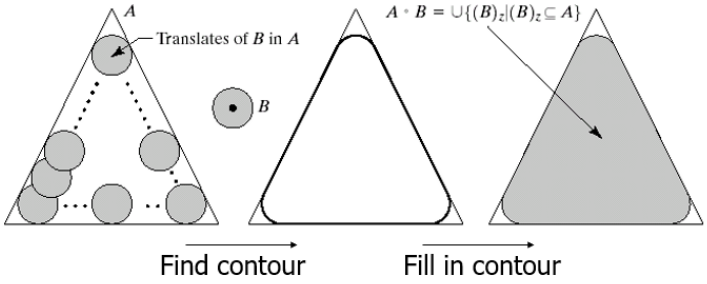


$$\beta(A) = A - (A \ominus B)$$

Opening (cont.)



Opening (cont.)

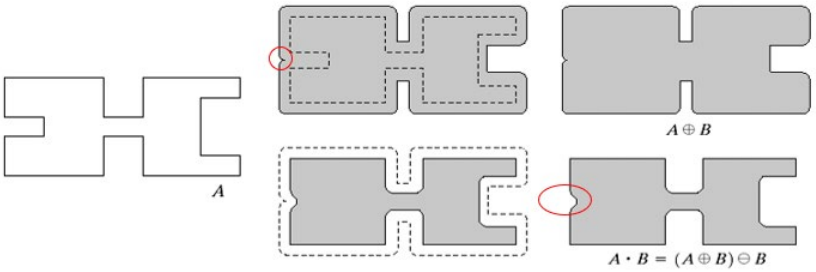


Smooth the contour of an image, breaks narrow isthmuses, eliminates thin protrusions

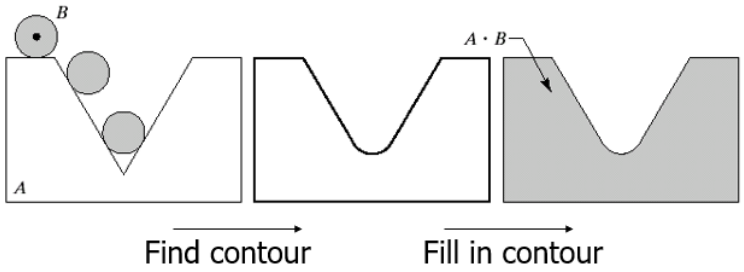
消去小凸起

断开窄接线

Closing



Closing (cont.)



Smooth the object contour, fuse narrow breaks and long thin gulfs, eliminate small holes, and fill in gaps

连接小断点，消除小空洞，填补空隙

Properties of opening and closing

Opening

(i) $A \circ B$ is a subset (subimage) of A Open后变小

(ii) If C is a subset of D, then $C \circ B$ is a subset of $D \circ B$

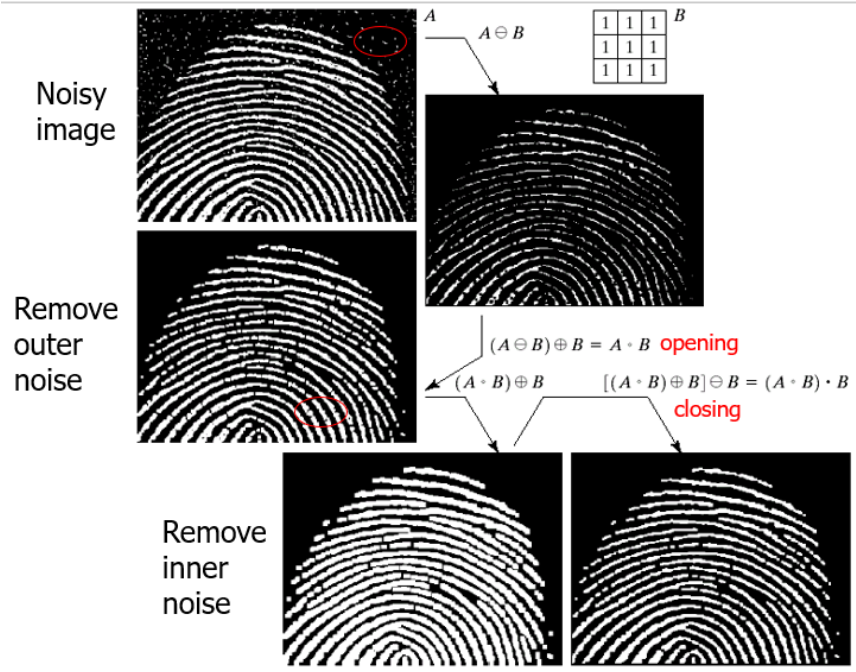
(iii) $(A \circ B) \circ B = A \circ B$ 重复做open等于做一次open

Closing

(i) A is a subset (subimage) of $A \bullet B$ Close后变大

(ii) If C is a subset of D, then $C \bullet B$ is a subset of $D \bullet B$

(iii) $(A \bullet B) \bullet B = A \bullet B$ 重复做close等于做一次close



二维离散傅立叶变换

因为数字图像信号是二维的数字信号，所以必须采用二维傅立叶变换才能够实现对图像的频域变换。

正变换

设图像大小为M*N，原图为f(x,y)，其频谱为F(u,v)，则：

$$F(\mu,\upsilon)=\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}f(x,y)\cdot e^{-j2\pi(\frac{x\mu}{M}+\frac{y\upsilon}{N})}$$

$$=\sum_{x=0}^{M-1}\{\sum_{y=0}^{N-1}f(x,y)\cdot e^{-j2\pi\frac{y\upsilon}{N}}\}e^{-j2\pi\frac{x\mu}{M}}$$

$$=fT_{行}(fT_{列}(f(x,y)))$$

$$=fT_{列}(fT_{行}(f(x,y)))$$

二维Fourier变换可以转化为两次一维Fourier变换。

反变换

$$f(x,y)=\frac{1}{MN}\sum_{\mu=0}^{M-1}\sum_{\upsilon=0}^{N-1}F(\mu,\upsilon)\cdot e^{j2\pi(\frac{x\mu}{M}+\frac{y\upsilon}{N})}$$

$$=\frac{1}{MN}fT^{-1}_{行}(fT^{-1}_{列}(f(x,y)))$$

$$=\frac{1}{MN}fT^{-1}_{列}(fT^{-1}_{行}(f(x,y)))$$

注：逆变换的系数不为1。

变换公式系数说明

- 因为Fourier变换是一种正交变换，所以其正、反变换的系数可以有几种表示形式。
- 按照严格意义上的正交变换，正、反变换的系数相等，为：$\frac{1}{\sqrt{MN}}$
- 按照计算方便的角度，正、反变换的系数可以按照前面的方式给出，并且正、反变换的系数可以互换。

作用

1）可以得出信号在各个频率点上的强度。

2）可以将卷积运算化为乘积运算。

快速Fourier变换（FFT）

- 快速Fourier变换的提出，是为了减少计算量。
- 基本思想是，找出Fourier变换中的数据变化规律，按照其规律整理出 *适合计算机运算* 的逻辑结构。

◆ FFT的数据变换规律之一是：

1）可以不断分成奇数项与偶数项之加权和。

2）奇数项、偶数项可分层分类。

摄像单元：

CCD图像（光电）传感器

工作原理是：将光能量转换为电荷，并将转换得到的电荷进行存储。

CCD传感器分为线阵式和阵列式两种，具有代表性的产品分别有扫描仪和数码相机。

r校正方法：

1）r 值的确定

通常CCD的r 值在0.4~0.8之间， r值越小，画面的效果越差。根据画面对比度的分析，可以大致得到该设备的r值（或依据设备的参考值）。

1	3	9	9	8
2	1	3	7	3
3	6	0	6	4
6	8	2	0	5
2	9	2	6	0

原始信息L

r=0.4

$I=3.8\cdot L^{0.4}$

4	6	9	9	9
5	4	6	8	6
6	8	0	8	7
8	9	5	0	7
5	9	5	8	0

CCD的输出信息

2）对输入信息进行r 校正

$$\because I=C\cdot L^{\gamma}$$

$$\therefore L=(I/C)^{1/\gamma}=\overline{C}\cdot I^{1/\gamma}$$

4	6	9	9	9
5	4	6	8	6
6	8	0	8	7
8	9	5	0	7
5	9	5	8	0

CCD的输出信息

r=0.4

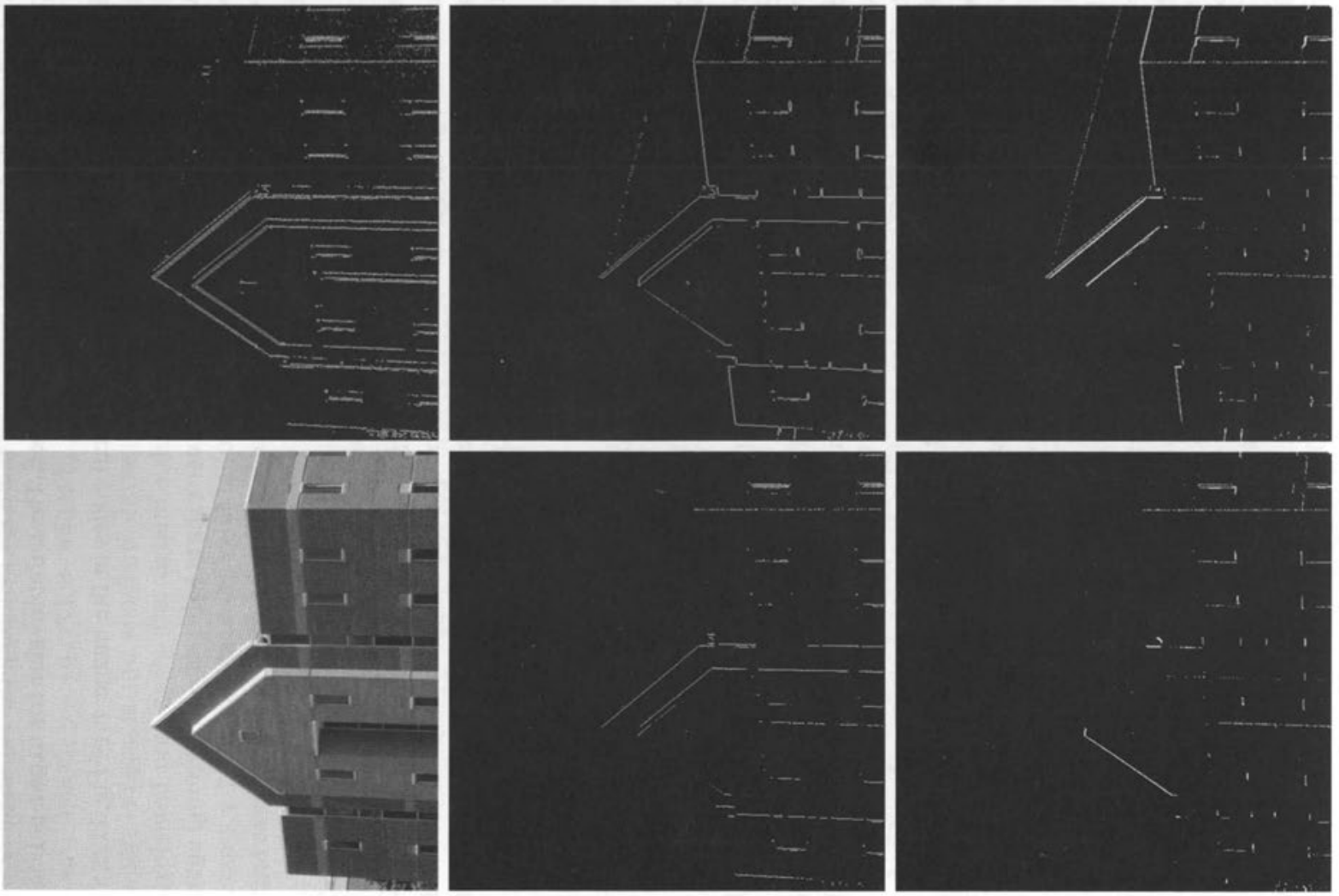
$L'=(I/3.8)^{1/0.4}$

1	3	9	9	9
2	1	3	6	3
3	6	0	6	5
6	9	2	0	5
2	9	2	6	0

r 校正后的信息

1	3	9	9	8
2	1	3	7	3
3	6	0	6	4
6	8	2	0	5
2	9	2	6	0

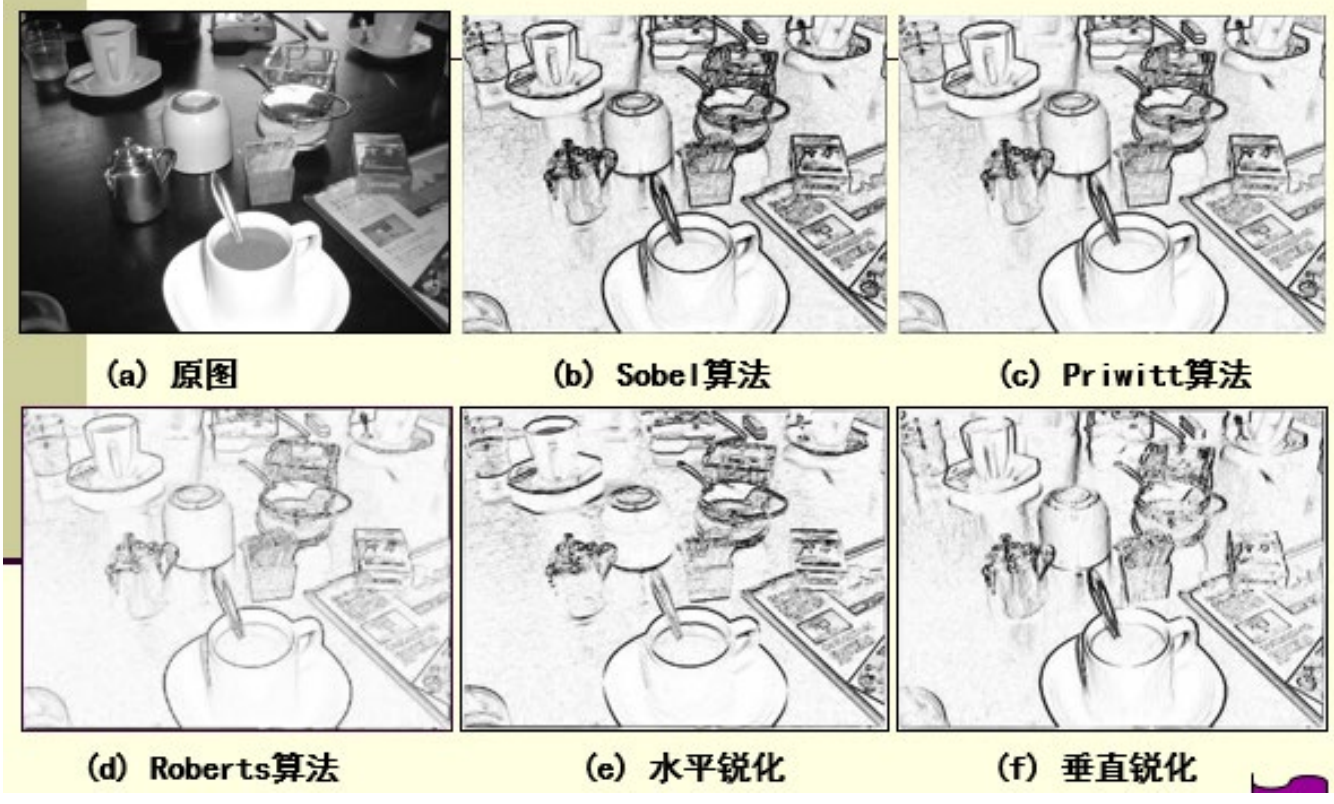
原始信息



a b
c d
e f

FIGURE 11.6
(a) Original image. (b) Result of function edge using a vertical Sobel mask with the threshold determined automatically. (c) Result using a specified threshold. (d) Result of determining both vertical and horizontal edges with a specified threshold. (e) Result of computing edges at −45° with `imfilter` using a specified mask and a specified threshold. (f) Result of computing edges at +45° with `imfilter` using a specified mask and a specified threshold.

一阶锐化方法的效果比较



YCbCr表色系 —— 基本设计思想

与YUV表色系统不同的是它充分考虑了色彩组成时RGB三色的重要因素。

YUV考虑的是色系转换的简单；

YCbCr考虑的是压缩时可以充分取出冗余量。

- RGB到YCbCr的转换

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

$$C_b = 2(1-0.114)(B-Y)$$

$$Cr = 2(1-0.299)(R-Y)$$

- YCbCr到RGB的转换

$k_r = \frac{1}{2(1-0.299)}$　　$k_b = \frac{1}{2(1-0.114)}$

$$R = Y + k_r \cdot C_r$$

$$B = Y + k_b \cdot C_b$$

$$G = Y - 0.299 / 0.587 * k_r C_r - 0.114 / 0.587 * k_b C_b$$

彩色补偿 —— 问题的提出

- 在某些应用中，目标是分离出主要或完全是颜色不同的各种类型的物体。
- 由于常用的彩色图像设备具有较宽而且相互覆盖的光谱敏感区，加上待拍摄图像的染色是变化的，所以很难在三个分量图中将物体分离出来。这种现象称为*颜色扩散*。

彩色补偿 —— 基本设计思想

通过数学运算，将扩散进来的颜色分量补偿掉。由此，使不同的目标在不同的颜色分量中信号最强。

彩色补偿 —— 算法思路

分析颜色扩散的现象，是“你中有我，我中有你”。

所以补偿的算法思路是：

将原本应该是纯红、纯绿、纯蓝色的像素点转换成理想的颜色，由此获得原图与补偿图之间的影射关系，最后用此影射关系处理所有的像素点。

彩色补偿 —— 算法步骤

1. 读入拍摄到的具有颜色扩散的图像设其三个颜色分量分别为R,G,B。
2. 分别求出某个颜色分量与其他两个颜色分量之间的强度差，即：

$$\begin{cases} e_R = (R-B)+(R-G) \\ e_G = (G-B)+(G-R) \\ R_B = (B-R)+(B-G) \end{cases}$$

3. 分别求出强度差的最大值（即寻找画面中应该为纯红、纯绿、纯蓝色的点。可能是单个点，也可能是多个点）如下：

$$\begin{cases} e_R^{\max} = \max(e_R) & \text{（红色的可能点）} \\ e_G^{\max} = \max(e_G) & \text{（绿色的可能点）} \\ e_B^{\max} = \max(e_B) & \text{（蓝色的可能点）} \end{cases}$$

4. 在三个强度差分量eR，eG，eB中，分别找出等于其最大值的像素，并分别求出其像素的均值如下：

$$\begin{cases} \bar{r}_1 = \{\bar{r} \mid e_R = e_R^{\max}\} \\ \bar{g}_1 = \{\bar{g} \mid e_R = e_R^{\max}\} \\ \bar{b}_1 = \{\bar{b} \mid e_R = e_R^{\max}\} \end{cases} \quad \begin{cases} \bar{r}_2 = \{\bar{r} \mid e_G = e_G^{\max}\} \\ \bar{g}_2 = \{\bar{g} \mid e_G = e_G^{\max}\} \\ \bar{b}_2 = \{\bar{b} \mid e_G = e_G^{\max}\} \end{cases} \quad \begin{cases} \bar{r}_3 = \{\bar{r} \mid e_B = e_B^{\max}\} \\ \bar{g}_3 = \{\bar{g} \mid e_B = e_B^{\max}\} \\ \bar{b}_3 = \{\bar{b} \mid e_B = e_B^{\max}\} \end{cases}$$

5. 将该三组点设为在没有颜色扩散的情况下，应该是纯红、纯绿、纯蓝色的点，即：

$(\bar{r}_1, \bar{g}_1, \bar{b}_1)$　　$(\bar{r}_2, \bar{g}_2, \bar{b}_2)$　　$(\bar{r}_3, \bar{g}_3, \bar{b}_3)$

理想值应该为 ↓　　理想值应该为 ↓　　理想值应该为 ↓

$(r^*,0,0)$　　$(0,g^*,0)$　　$(0,0,b^*)$

6. 所以可构造彩色补偿变换矩阵如下：

$$A_1 = \begin{bmatrix} \bar{r}_1 & \bar{r}_2 & \bar{r}_3 \\ \bar{g}_1 & \bar{g}_2 & \bar{g}_3 \\ \bar{b}_1 & \bar{b}_2 & \bar{b}_3 \end{bmatrix} \quad A_2 = \begin{bmatrix} r^* & 0 & 0 \\ 0 & g^* & 0 \\ 0 & 0 & b^* \end{bmatrix}$$

7. 设原图为 $f = \begin{bmatrix} r_f(i,j) \\ g_f(i,j) \\ b_f(i,j) \end{bmatrix}_{m \times n}$ 补偿后的图像为 $g = \begin{bmatrix} r_g(i,j) \\ g_g(i,j) \\ b_g(i,j) \end{bmatrix}_{m \times n}$

则：$g = C \cdot f + b$

8. 将计算得到的A1，A2代入，有：$A_2 = C \cdot A_1 + b$ 先忽略b，最后再将结果图标准化，有 $C = A_2 \cdot A_1^{-1}$

9. 进行彩色补偿有：$g(i,j) = C \cdot f(i,j)$

彩色平衡 —— 问题的提出

当一幅彩色图像数字化后，在显示时颜色经常看起来有些不正常。这是因为色通道的不同敏感度、增光因子、偏移量等原因导致。称之为三基色不平衡。将其校正的过程就是彩色平衡。

基本设计思想：在画面中，寻找不同亮暗的中性色的像素点，这些点应该是满足 R=G=B 的，但是因为色偏的缘故不相等，于是通过将其影射为相等值获得彩色平衡的作用矩阵，就可进行彩色平衡处理。

白平衡算法步骤：

1. 计算输入的具有色偏的原图亮度，即：
   $Y = 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B$
2. 根据计算出的亮度值来寻找图像中的白色点。考虑到实际中，白色的点不一定是理想状态下的白点，因此在这里只是将白色定义为亮度值为最大的点，即：Y_max = max{Y}
3. 考虑到对环境光照具有一定的适应性，寻找出原图中所有亮度值不小于 0.95 倍最大亮度值的点。令这些点构成白色点集合，即：$\Omega_{white} = \{Y \mid Y \geq 0.95 \cdot Y_{\max}\}$
4. 计算白色点集Ωwhite 中所有像素的 R,G,B 三个颜色分量的均值
5. 按照下面的公式计算颜色均衡化的调整参数：

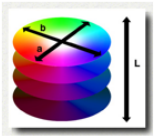
$k_R = \bar{Y} / \bar{R}$　　$k_G = \bar{Y} / \bar{G}$　　$k_B = \bar{Y} / \bar{B}$

6. 对整幅图像的R,G,B三个颜色分量，进行彩色平衡调整如下：$R^* = k_R \cdot R$　$G^* = k_G \cdot G$　$B^* = k_B \cdot B$

得到的图像就是彩色平衡后的图像。

❖Lab颜色空间是在1976年制定的等色空间，以克服在x，y色度图上相等的距离并不相当于我们所觉察到的相等色差的问题。

L: 明亮度
a: 从绿色到红色
b: 从蓝色到黄色



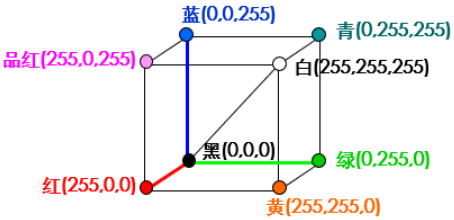
Lab 的概念图

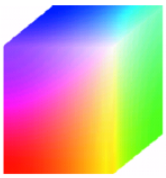
■ 在比较色差时，设A为基准色，B为试料色，A'为与B相同亮度的基准色，ΔE*ab*为色差程度，ΔE*ab*大小决定了色差程度的大小，具体见表。

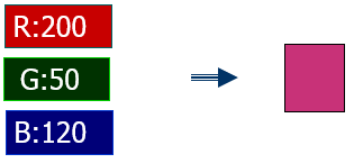
色差程度的鉴定ΔE*ab	
微量	0 ~ 0.5
轻微	0.5 ~ 1.5
能感觉到	1.5 ~ 3.0
明显	3.0 ~ 6.0
很大	6.0 ~ 12.0
截然不同	12.0以上

RGB色系 —— 基本概念

■ CIE规定了以700nm(红)、546.1nm (绿)、435.8nm (蓝)三个色光为三基色。又称为物理三基色。自然界的所有颜色都可以过选用这三基色按不同比例混合而成。







RGB色系 —— 应用场合

目前包括计算机显示器、彩色电视机在内的绝大部分图形显示器中。

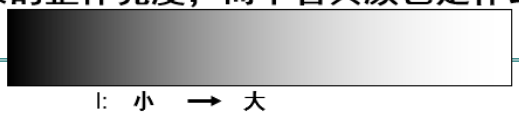
如果采用其他色系进行了处理，最终一定要转换到RGB色系，才能正常显示结果。

HSI色系 —— 问题的提出

~~RGB色系虽然是目前各类显示器使用的色系，但~~颜色的构成与人对颜色的理解方式不同，所以在进行处理与调整时，比较不容易获得准确的参数。

HSI色系 —— 亮度分量I

I 表示光照强度或称为亮度，它确定了像素的整体亮度，而不管其颜色是什么。



H: 表示色度，由角度表示。反映了该颜色最接近什么样的光谱波长。0°为红色，120°为绿色，240°为蓝色。

S: 表示饱和度，饱和度参数是色环的原点到彩色点的半径长度。

在环的外围圆周是纯的或称饱和的颜色，其饱和度值为1。在中心是中性（灰）色，即饱和度为0。

CMYK色系—— 基本概念

- 这种表色系用于印刷行业。
- 是一种减色系统，将从白光中滤出三种原色之后获得的颜色作为其表色系的三原色CMY。
- K为黑色，为了印刷时对黑色可用黑色墨来印刷。

- C: 青色，从白色中滤去红色。
- M: 品红，从白色中滤去绿色。
- Y: 黄色，从白色中滤去蓝色。

CMYK色系—— 着色原理

既然是减色系统，其着色原理是基于光吸收的，这有别于RGB的光射入的方式。

C与M叠加：同时吸收了R与G，则为蓝色；

C与Y叠加：同时吸收了R与B，则为绿色；

M与Y叠加：同时吸收了G与B，则为红色。

■ CMYK 色系转换到RGB色系

$$\begin{cases} R = W - C = 0.5 \cdot [M + Y - C] \\ G = W - M = 0.5 \cdot [Y + C - M] \\ B = W - Y = 0.5 \cdot [M + C - Y] \end{cases}$$ 其中，W表示白色。

■ RGB色系转换到CMYK色系

$$\begin{cases} C = W - R = G + B \\ M = W - G = R + B \\ Y = W - B = R + G \\ K = \min\{C, M, Y\} \end{cases}$$ 其中，W表示白色。

■ RGB到YUV的转换

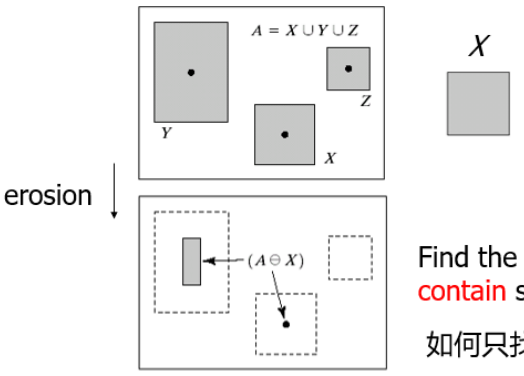
$$\begin{cases} Y = 0.299R + 0.587G + 0.114B \\ U = B - Y \\ V = R - Y \end{cases}$$

■ YUV到RGB的转换

$$\begin{cases} R = Y + V \\ G = Y - 0.192U - 0.509V \\ B = Y + U \end{cases}$$

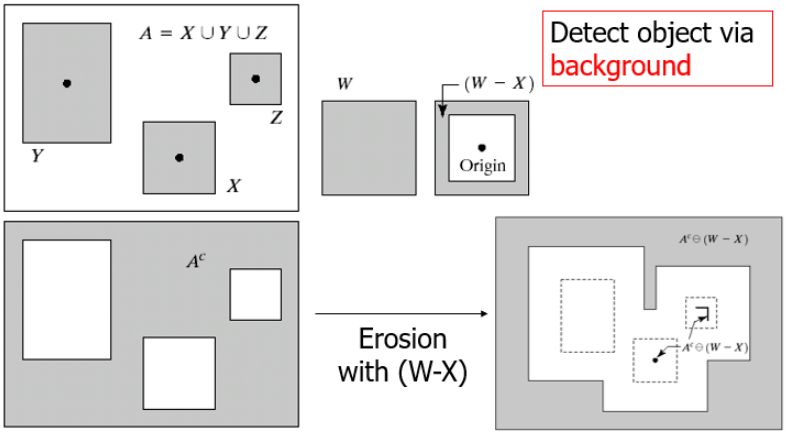
特点：这两个色系的转换非常简单，所以可满足转换的快速性要求。

- Find the location of certain shape

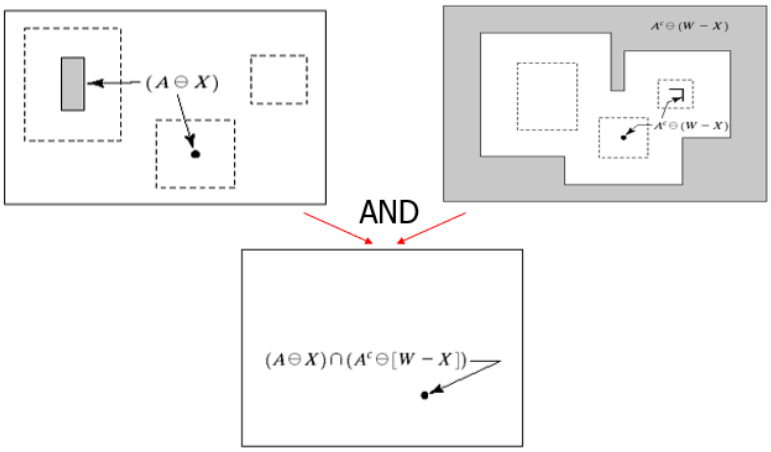


Find the set of pixels that contain shape X

如何只找到相符形狀中心點?



- Eliminate un-necessary parts



Region filling 区域填充

- How?
- Idea: place a point inside the region, then dilate that point iteratively

$$X_0 = p$$
$$X_k = (X_{k-1} \oplus B) \cap A^c, k = 1,2,3,...$$
Until $X_k = X_{k-1}$

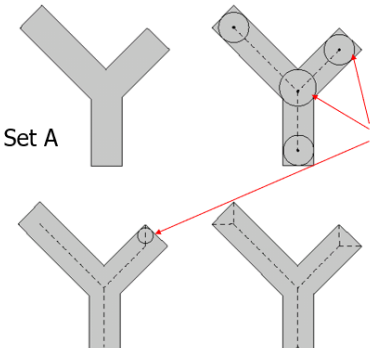
Bound the growth

Extraction of connected components 连通分量的提取

- 找到连通部分
- Idea: start from a point in the connected component, and dilate it iteratively

$$X_0 = p$$
$$X_k = (X_{k-1} \oplus B) \cap A, \ k = 1,2,3,...$$
Until $X_k = X_{k-1}$

Skeletons 骨架



How to define a Skeletons?

Maximum disk

1. The largest disk Centered at a pixel
2. Touch the boundary of A at two or more places

Recall: Balls of erosion!

Skeleton

- Idea: 不断的 erosion

Erosions	Openings	Set differences
A	$A \circ B$	$A - (A \circ B)$
$A \ominus B$	$(A \ominus B) \circ B$	$(A \ominus B) - ((A \ominus B) \circ B)$
$A \ominus 2B$	$(A \ominus 2B) \circ B$	$(A \ominus 2B) - ((A \ominus 2B) \circ B)$
$A \ominus 3B$	$(A \ominus 3B) \circ B$	$(A \ominus 3B) - ((A \ominus 3B) \circ B)$
⋮	⋮	⋮
$A \ominus kB$	$(A \ominus kB) \circ B$	$(A \ominus kB) - ((A \ominus kB) \circ B)$

Erosion k 次直到空集合

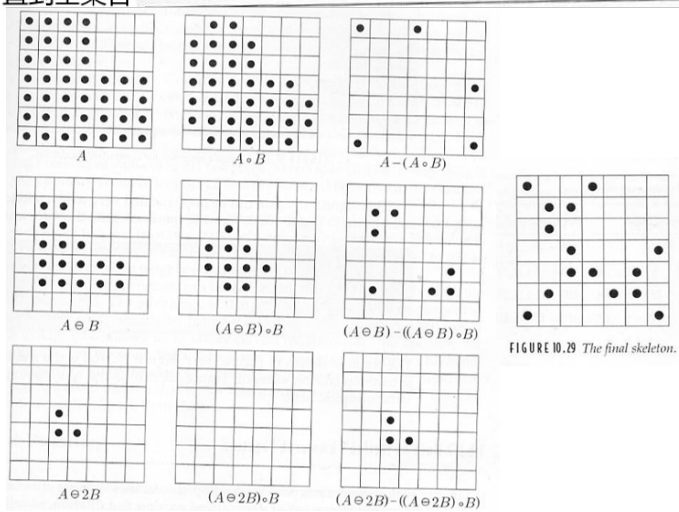


FIGURE 10.29 The final skeleton.

Basic morphological algorithms

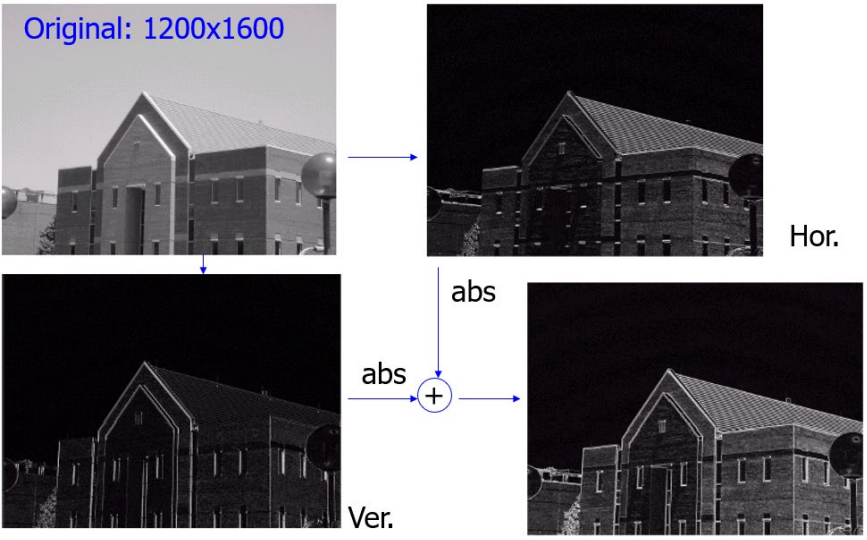
- Extract image components that are useful in the representation and description of shape
- Boundary extraction 边界提取
- Region filling 区域填充
- Extract of connected components
- Convex hull
- Thinning
- Thickening
- Skeleton 骨架
- Pruning

- Why use color in image processing?
  - Color is a powerful descriptor
    - Object identification and extraction
    - eg. Face detection using skin colors
  - Humans can discern thousands of color shades and intensities
    - c.f. Human discern only two dozen shades of grays

How to deal with color vector?

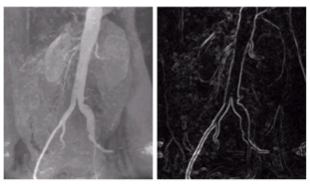
- Per-color-component processing
  - Process each color component
- Vector-based processing
  - Process the color vector of each pixel
- When can the above methods be equivalent?
  - Process can be applied to both scalars and vectors
  - Operation on each component of a vector must be independent of the other component

Example: Sobel filter



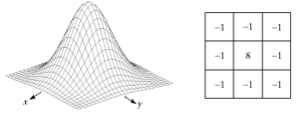
Example: 2nd derivative

original

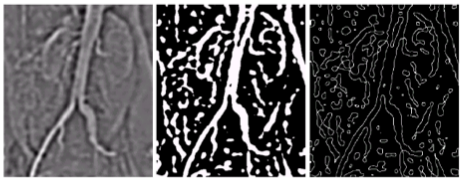


Sobel

Gaussian Smoothing



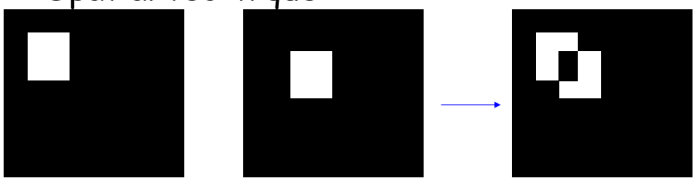
Laplacian (2nd derivative)



LoG Threshold(LoG) Zero-crossing

Motion as a clue to extract object

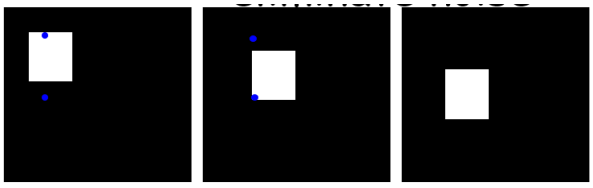
- Spatial technique



Reference image f(x,y,1) — next image f(x,y,2) — Thresholded difference image

time index

1 if |d(x,y)| > T
0 otherwise

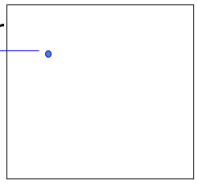


Reference image R(x,y) — Image f(x,y,2) — Image f(x,y,3) ...

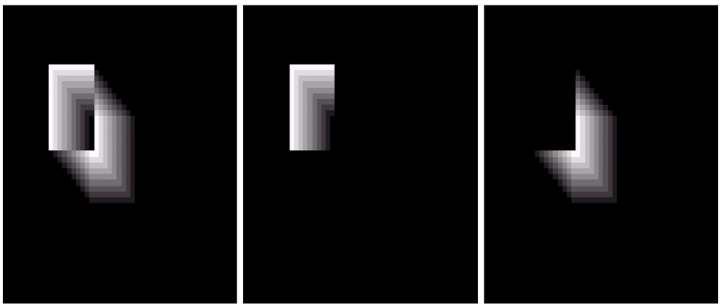
d(x,y)=R(x,y)-f(x,y,t)

counter + 1,
a. if d(x,y) > T positive ADI
b. if d(x,y) < -T negative ADI
c. if |d(x,y)| > T absolute ADI

counter



Accumulative difference image

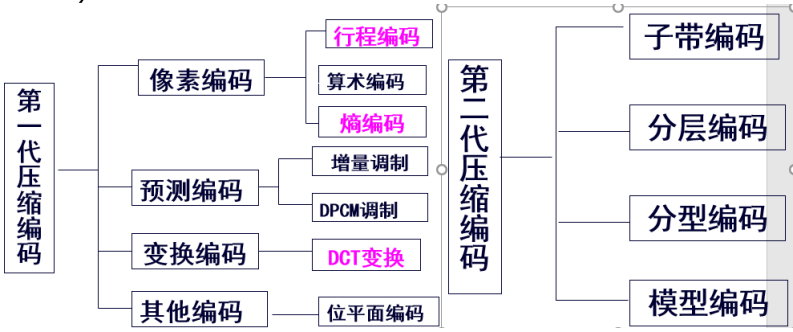


Absolute ADI — Positive ADI — Negative ADI

* Object shape
* Location in ref. image

图像压缩原理

- 由于一幅图像存在数据冗余和主观视觉冗余，所以压缩方式就可以从这两方面着手开展。
- 改变图像信息的描述方式，以压缩掉图像中的数据冗余。
- 忽略一些视觉不太明显的微小差异，以压缩掉图像中的视觉冗余。



Huffman编码 —— 算法

首先求出图像中灰度分布的灰度直方图；

根据该直方图，对其按照分布概率从小到大的顺序进行排列；

每一次从中选择出两个概率为最小的节点相加，形成一个新的节点，构造一个称为“Huffman树”的二叉树；

对这个二叉树进行编码，就获得了Huffman编码码字。

正变换：

$$F_c(\mu,\upsilon)=\frac{2}{\sqrt{MN}}c(\mu)c(\upsilon)\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}f(x,y)\cos[\frac{\pi}{2N}(2x+1)\mu]\cos[\frac{\pi}{2M}(2y+1)\upsilon]$$

逆变换：

$$f(x,y)=\frac{2}{\sqrt{MN}}\sum_{\mu=0}^{M-1}\sum_{\upsilon=0}^{N-1}c(\mu)c(\upsilon)F_c(\mu,\upsilon)\cos[\frac{\pi}{2N}(2x+1)\mu]\cos[\frac{\pi}{2M}(2y+1)\upsilon]$$

其中：
$$c(x)=\begin{cases}\frac{1}{\sqrt{2}} & x=0\\ 1 & x=1,2,\ldots,N-1\end{cases}$$

Huffman编码 —— 例

□ 例：对数据序列

aaaa bbb cc d eeeee ffffff

其概率分布为：

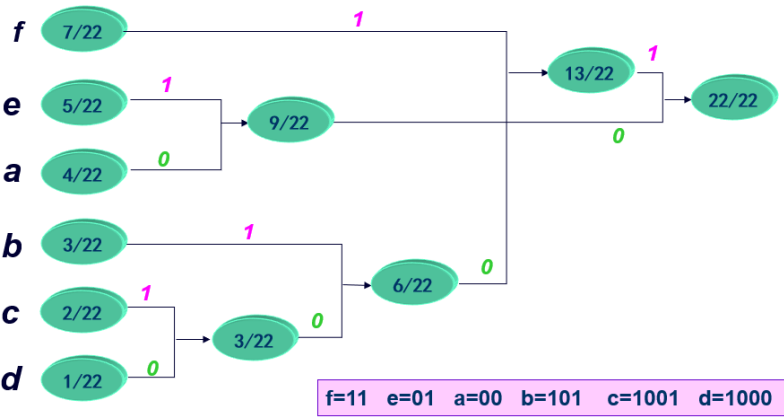
a:4/22　b:3/22　c:2/22

d:1/22　e:5/22　f:7/22

概率大小的排序为：

d,　c,　b,　a,　e,　f

1/22　2/22　3/22　4/22　5/22　7/22



f=11　e=01　a=00　b=101　c=1001　d=1000

1）编码过程：

原图像 → DCT变换 → 除以量化系数 → 取整 → 压缩图像

2）解码过程：

压缩图像 → 乘以量化系数 → DCT逆变换 → 取整 → 解压图像

混合编码 —— 设计思想

- 每一种编码方式都有其擅长的一点，以及局限的一点，混合编码的思想就是将两种以上的编码方式的优点进行综合，达到提高编码效率的目的。

混合编码 —— 可能性及有效性分析。

- 行程编码：擅长于重复数字的压缩。
- Huffman 编码：擅长于像素个数分布不均匀情况下的编码。
- DCT 变换：　擅长分离视觉敏感与不敏感的部分。

例: aaaa bbb cc d eeeee ffffff　（共22*8=176 bits

4　3　2　1　5　7

行程编码：4a3b2c1d5e7f

(共6*（8+3）= 66Bits)

Huffman编码 —— 压缩效率

对这个例子，计算出经过Huffman编码后的数据为：

1010101010001001001000100010000111111111

101010101010101

共 7*2+5*2+4*2+3*3+2*4+1*4=53 bit

压缩比为176：53=3.32:1

aaaa bbb cc d eeeee ffffff　（共22*8=176 bits)

4　3　2　1　5　7

Huffman编码：

f=01　e=11　a=10　b=001　c=0001　d=0000

1010101010001001001000100010000111111111101010101010101

(共 7*2+5*2+4*2+3*3+2*4+1*4=53 bits)

- 我们知道，对一幅图像进行编码时，如果图像的大小大于256 时，这幅图像的不同的码字就有可能是很大，例如极限为 256 个不同的码字。

这时如果采用全局 Huffman 编码则压缩效率不高。甚至有可能与原来的等长编码的数据量相同。

常用的且有效的方法是：

□ 将图像分割成若干的小块，对每块进行独立的 Huffman 编码。例如：分成 8*8 的子块，就可以大大降低不同灰度值的个数（最多是 64 而不是 256）。

aaaa bbb cc d eeeee ffffff　（共22*8=176 bits)

4　3　2　1　5　7

Hufman与行程编码混合：

41030012000110000511701

(共：3+2+3+3+3+4+3+4+3+2+3+2=35 bits)

1 一次小波变换　DCT变换. 行程编码　Huffman编码

2 一次小波变换　差值编码　变字长行程编码　Huffman编

DCT 变换编码 —— 设计思想

- 1. DCT 变换是希望在接收方不产生误解的前提下进行一定的信息丢失。
- 由前面所讲到的频域变换得到的启示，就是将低频与高频部分的信息，分别按照不同的数据承载方式进行表述。

行程编码(RLE编码)——基本概念

行程编码是一种最简单的，在某些场合是非常有效的一种无损压缩编码方法。

虽然这种编码方式的应用范围非常有限，但是因为这种方法中所体现出的编码设计思想非常明确，所以在图像编码方法中都会将其作为一种典型的方法来介绍。

行程编码 —— 基本原理

通过改变图像的描述方式，来实现图像的压缩。

将一行中灰度值相同的相邻像素，用一个计数值和该灰度值来代替。

行程编码 —— 实现方法

- 举例说明：a=100,b=1,c=23,d=254

aaaa bbb cc d eeeee fffffff

4 3 2 1 5 7

(共22*8=176 bits)

→ 4a3b2c1d5e7f

(共12*8=96 bits)

压缩比为：176：96=1.83:1

行程编码——传真中的应用方法

对于： 500W 3b 470w 12b 4w 3b 3000w

编码为： 500, 3, 570, 12, 4, 3, 3000

编码位数为:12, 12, 12, 12, 12,12,12

需要的数据量为： 12*7=84 bit

压缩比为： 176：84=2.1：1

因为只有白或黑，而且排版中一定要留出页边距，因此，一般情况下，可以只传输计数值即可。

- 现在，根据传真件的特点，对其进行改进。
- 既然已经可以预知白色多黑色少，所以可对白色和黑色的计数值采用不同的位数。
- 以这个例子，可以定义：

白色：12 bit，黑色：4 bit

编码为： 500,3,570,12,4,3,3000

编码位数为：12,4,12, 4,12,4,12

所需字节数为：4*12+3*4=60bit

压缩比为： 176：60=2.93：1

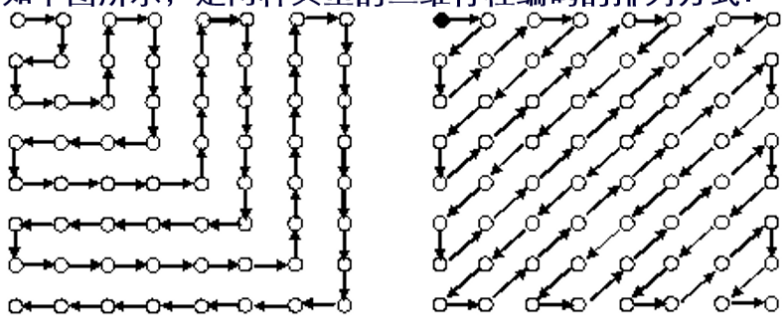
比原来的RLE方式120bit减少了60bit，相当于又提高了压缩比为 120/60=2：1 。

二维行程编码 —— 基本概念

二维行程编码要解决的核心问题是:将二维排列的像素，采用某种方式转化成一维排列的方式。之后按照一维行程编码方式进行编码。

二维行程编码 —— 数据排序

如下图所示，是两种典型的二维行程编码的排列方式：



(a) (b)

一维行程编码为：

(7, 130) , (2, 130) , (4, 129) , (2, 130) , (1, 129) ;
(1, 127) , (1, 128) , (1, 127) , (1, 129) , (1, 131) ,
(1, 130) , (1, 132) , (2, 134) , (2, 133) , (1, 132) ,
(1, 130) , (1, 129) , (1, 128) , (1, 127) , (1, 128) ,
(1, 127) , (1, 128) , (1, 127) , (1, 125) , (1, 126) ,
(2, 129) , (1, 127) , (1, 129) , (1, 133) , (1, 132) ,
(1, 131) , (1, 129) , (2, 130) , (1, 129) , (3, 130) ,
(1, 129) , (1, 130) , (2, 132) , (2, 131) , (1, 130) ,
(1, 126) , (2, 128) , (2, 127)

数据量为:43*（3+8）=473(bit)

压缩比为：512：473=1.08:1

Huffman 编码（熵编码）

行程编码要获得好的压缩率的前提是，有比较长的相邻像素的值是相同的。

熵是指数据中承载的信息量。

所谓的熵编码是指在完全不损失信息量前提下最小数据量的编码。

Huffman编码 —— 基本原理

为了达到大的压缩率，提出了一种方法就是将在图像中出现频度大的像素值，给一个比较短的编码，将出现频度小的像数值，给一个比较长的编码。

aaaa bbb cc d eeeee fffffff

4 3 2 1 5 7

- 按照熵编码的原理进行编码：

f=0 e=10 a=110 b=1111 c=11100 d=11101

这里的编码规则是长短不一的异字头码

f=0 e=10 a=110 b=1111 c=11100 d=11101

aaaa bbb cc d eeeee fffffff →

10110110110111111111111111001110011101101
01010100000000

数据量：7*1+5*2+4*3+3*4+2*5+1*5=56 bit

压缩比为：176：56=3.14：1

均值滤波器的改进—— 加权均值滤波

均值滤波器的缺点是，会使图像变的模糊，原因是它对所有的点都是同等对待，在将噪声点分摊的同时，将景物的边界点也分摊了。

为了改善效果，就可采用加权平均的方式来构造滤波器。

中值滤波器—— 问题的提出

- 虽然均值滤波器对噪声有抑制作用，但同时会使图像变得模糊。即使是加权均值滤波，改善的效果也是有限的。
- 为了有效地改善这一状况，必须改换滤波器的设计思路，中值滤波就是一种有效的方法。

中值滤波器—— 设计思想

因为噪声（如椒盐噪声）的出现，使该点像素比周围的像素亮（暗）许多。

如果在某个模板中，对像素进行由小到大排列的重新排列，那么最亮的或者是最暗的点一定被排在两侧。

取模板中排在中间位置上的像素的灰度值替代待处理像素的值，就可以达到滤除噪声的目的。

边界保持类平滑滤波器—— 问题的提出

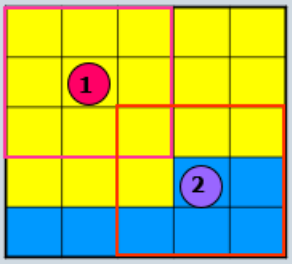
- 经过平滑滤波处理之后，图像就会变得模糊。
- 分析原因，在图像上的景物之所以可以辨认清楚是因为目标物之间存在边界。
- 而边界点与噪声点有一个共同的特点是，都具有灰度的跃变特性。所以平滑处理会同时将边界也处理了。

边界保持类平滑滤波器—— 设计思想

- 为了解决图像模糊问题，一个自然的想法就是，在进行平滑处理时，首先判别*当前像素是否为边界上的点*，如果是，则不进行平滑处理；如果不是，则进行平滑处理。

K近邻(KNN)平滑滤波器—— 原理分析

- 边界保持滤波器的核心是确定边界点与非边界点。
- 如图所示，点1是黄色区域的非边界点，点2是蓝色区域的边界点。
- 点1模板中的像素全部是同一区域的；点2模板中的像素则包括了两个区域。



- 在模板中，分别选出5个与点1或点2灰度值最相近的点进行计算，则不会出现两个区域信息的混叠平均。
- 这样，就达到了边界保持的目的。

K近邻(KNN)平滑滤波器—— 实现算法

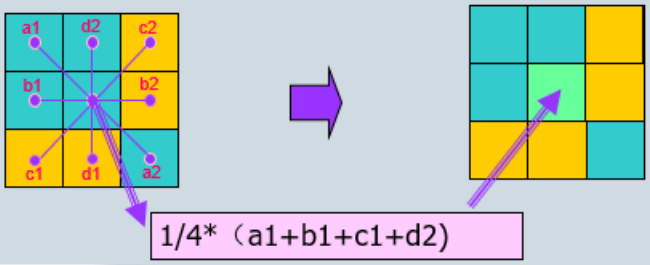
1) 以待处理像素为中心，作一个m*m的作用模板。
2) 在模板中，选择K个与待处理像素的灰度差为最小的像素。
3) 将这K个像素的灰度均值替换掉原来的像素值。

K近邻(KNN)平滑滤波器—— 效果分析

- 首先来看一下KNN平滑滤波的效果。
- KNN滤波器因为有了边界保持的作用，所以在去除椒盐以及高斯噪声时，对图像景物的清晰度保持方面的效果非常明显。
- 当然，所付出的代价是：算法的复杂度增加了。

对称近邻平滑滤波器—— 基本原理

- 算法示意图如下，从模板中的对称点对寻找与待处理像素相同区域的点。然后对选出的点做均值运算。



最小方差平滑滤波器—— 基本原理

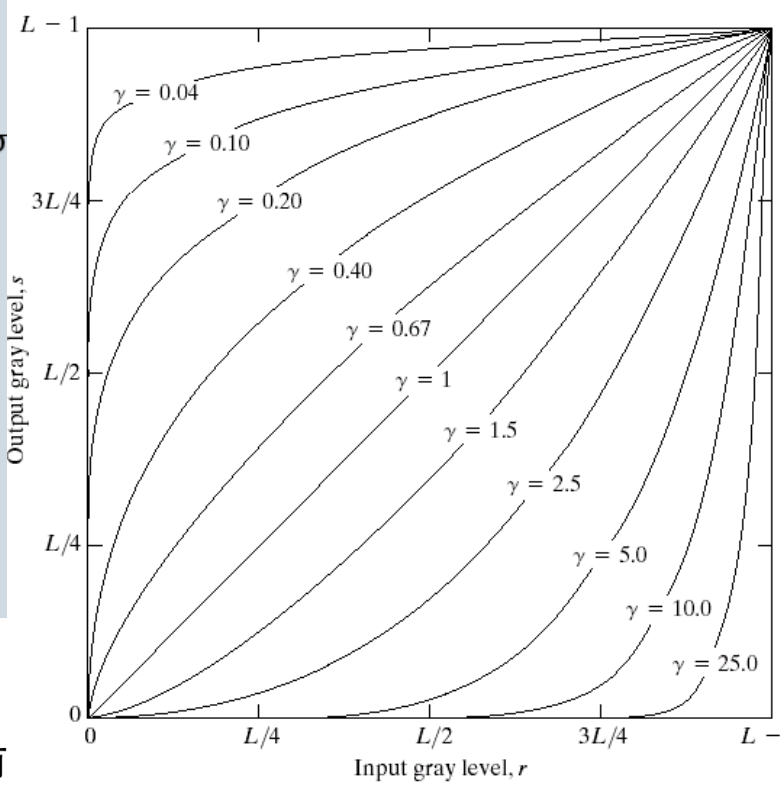
- 将属于同一个区域的可能的相邻关系以**9**种模板表示出来，然后计算每个模板中的灰度分布方差，以方差最小的那个模板的均值替代原像素值。

Sigma平滑滤波器—— 基本原理

- 根据统计数学的原理，属于同一类别的元素的置信区间，落在均值附近±2σ 范围之内。
- Sigma滤波器是构造一个模板，计算模板的标准差σ置信区间为当前像素值的±2σ范围。
- 将模板中落在置信范围内的像素的均值替换原来的像素值。

边界保持类平滑滤波器—— 总结

- **边界保持类平滑滤波器的核心是：尽可能地将平滑处理避开两个或多个不同区域进行计算。可以采用不同形状结构判别，也可以采用同类相似的概念进行判别。**

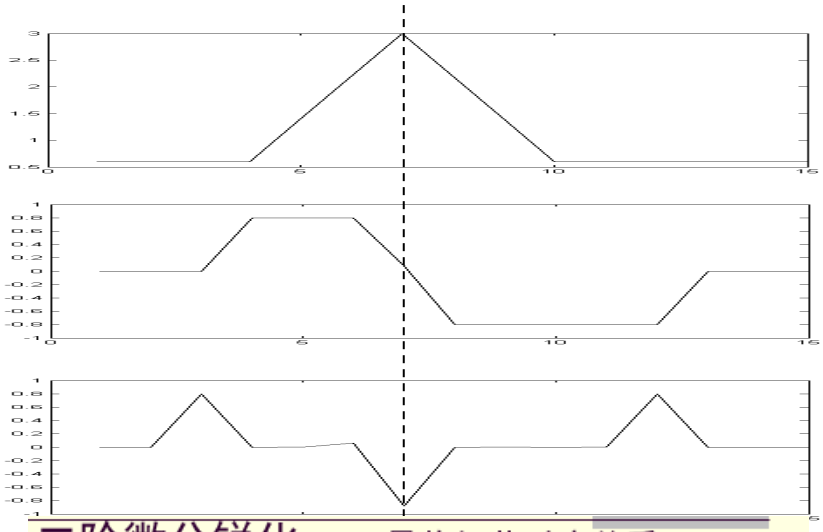


单方向锐化

- 前面的锐化处理结果对于人工设计制造的具有矩形特征物体（例如：楼房、汉字等）的边缘的提取很有效。但是，对于不规则形状（如：人物）的边缘提取，则存在信息的缺损。

无方向一阶锐化—— 设计思想

- 为了解决上面的问题，就希望提出对任何方向上的边缘信息均敏感的锐化算法。
- 因为这类锐化方法要求对边缘的方向没有选择，所有称为无方向的锐化算法。



二阶微分锐化—— 景物细节对应关系

1）对于突变形的细节，通过一阶微分的极大值点，二阶微分的过0点均可以检测出来。

2）对于细线形的细节，通过一阶微分的过0点，二阶微分的极小值点均可以检测出来。

3）对于渐变的细节，一般情况下很难检测，但二阶微分的信息比一阶微分的信息略多。

离散余弦变换（DCT）—— 问题的提出

- Fourier变换的一个最大的问题是：它的参数都是复数，在数据的描述上相当于实数的两倍。
- 为此，我们希望有一种能够达到相同功能但数据量又不大的变换。在此期望下，产生了DCT变换。

离散余弦变换（DCT）

正变换：

$$F_c(\mu,\upsilon)=\frac{2}{\sqrt{MN}}c(\mu)c(\upsilon)\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}f(x,y)\cos[\frac{\pi}{2N}(2x+1)\mu]\cos[\frac{\pi}{2M}(2(y+1)\upsilon]$$

逆变换：

$$f(x,y)=\frac{2}{\sqrt{MN}}\sum_{\mu=0}^{M-1}\sum_{\upsilon=0}^{N-1}c(\mu)c(\upsilon)F_c(\mu,\upsilon)\cos[\frac{\pi}{2N}(2x+1)\mu]\cos[\frac{\pi}{2M}(2(y+1)\upsilon]$$

其中：
$$c(x)=\begin{cases}\frac{1}{\sqrt{2}} & x=0\\ 1 & x=1,2,\ldots,N-1\end{cases}$$

- 余弦变换实际上是利用了Fourier变换的实数部分构成的变换。
- 余弦变换主要用于图像的压缩，如目前的国际压缩标准的JPEG格式中就用到了DCT变换。
- 具体的做法与DFT相似。即高频部分压缩多一些，低频部分压缩少一些。

贴标签 —— 算法步骤

1. 初始化：设标签号为Lab=0,已贴标签数N=0，标签矩阵g为全0阵，按照从上到下，从左到右的顺序寻找未贴标签的目标点；

例：

$$f=\begin{bmatrix}0&1&0&0&0&0\\0&1&0&1&1&0\\0&0&1&0&0&0\\0&0&0&0&0&0\\0&1&1&0&1&1\\0&1&0&1&0&0\end{bmatrix}\qquad g=\begin{bmatrix}0&0&0&0&0&0\\0&0&0&0&0&0\\0&0&0&0&0&0\\0&0&0&0&0&0\\0&0&0&0&0&0\\0&0&0&0&0&0\end{bmatrix}$$

2. 检查相邻像素的状态：根据模板中的相邻像素的状态进行相应的处理；

例：

$$f=\begin{bmatrix}0&1&0&0&0&0\\0&1&0&1&1&0\\0&0&1&0&0&0\\0&0&0&0&0&0\\0&1&1&0&1&1\\0&1&0&1&0&0\end{bmatrix}$$

模板

☻=“已经扫描过的像素”

*=“当前像素”

⊗=“未处理的像素”

- 如果扫描过的像素均为0，则Lab=Lab+1，g(i,j)=Lab,N=N+1；
- 如果扫描过的像素标签号相同，则g(i,j)=Lab;
- 如果扫描过的像素标签号不相同，例如：Lab2> Lab1, 则g(i,j)=Lab1，N=N-1，修改所有为Lab2的像素值，使之为Lab1;

3. 将全部的像素进行2.的处理，直到所有的像素全部处理完成;

腐蚀 —— 算法步骤

1）扫描原图，找到第一个像素值为1的目标点；

2）将预先设定好形状以及原点位置的结构元素的原点移到该点；

3）判断该结构元素所覆盖的像素值是否全部为1：

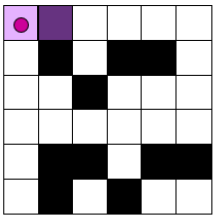
如果是，则腐蚀后图像中的相同位置上的像素值为1；

如果不是，则腐蚀后图像中的相同位置上的像素值为0；

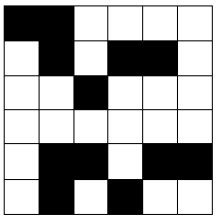
4）重复2）和3），直到所有原图中像素处理完成。

膨胀 —— 设计思想

- 设计一个结构元素，结构元素的原点定位在*背景像素*上，判断是否覆盖有目标点，来确定是否该点被膨胀为目标点。



结构元素



膨胀 —— 算法步骤

1）扫描原图，找到第一个像素值为0的背景点；

2）将预先设定好形状以及原点位置的结构元素的原点移到该点；

3）判断该结构元素所覆盖的像素值是否存在为1的目标点：

如果是，则膨胀后图像中的相同位置上的像素值为1；

如果不是，则膨胀后图像中的相同位置上的像素值为0；

4）重复2）和3），直到所有原图中像素处理完成

开运算与闭运算的提出背景

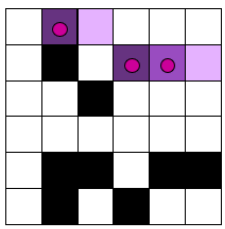
- 前面介绍的膨胀与腐蚀运算，对目标物的后处理有着非常好的作用。但是，腐蚀和膨胀运算的一个缺点是，改变了原目标物的大小。
- 为了解决这一问题，考虑到腐蚀与膨胀是一对逆运算，将膨胀与腐蚀运算同时进行。由此便构成了开运算与闭运算。

开运算 —— 算法原理

- 开运算是对原图先进行腐蚀处理，后再进行膨胀的处理。
- 开运算可以在分离粘连目标物的同时，基本保持原目标物的大小。
- 闭运算是对原图先进行膨胀处理，后再进行腐蚀的处理。
- 闭运算可以在合并断裂目标物的同时，基本保持原目标物的大小。

腐蚀 ——设计思想

- 设计一个结构元素，结构元素的原点定位在待处理的*目标像素*上，通过判断是否覆盖，来确定是否该点被腐蚀掉。



结构元素