# 数字图像处理

中南大学自动化学院
谢斌
xiebin@csu.edu.cn

# 思考

1. 不同分辨率、不同量化等级对图像质量有何影响？

2. 图片存储大小与分辨率、量化等级有何关系？

# Sampling

- 图像的采样


图2.15图像的采样示例

# Quantization

- 图像的量化


图2.16图像的量化示例

# 第3章 图像基本运算

(Basic Operation in Digital Image Processing )

- 3.1 图像基本运算的概述(Introduction)
- 3.2 点运算 (Point Operation)
- 3.3 代数与逻辑运算(Algebra and Logical Operation)
- 3.4几何运算 (Geometric Operation)

# 3.1 图像基本运算的概述(Introduction)

## 图像基本运算的分类

按图像处理运算的数学特征，图像基本运算可分为：

- 图像基本运算
  - 点运算(Point Operation)
  - 代数运算(Algebra Operation)
  - 逻辑运算(Logical Operation)
  - 几何运算(Geometric Operation)

# 3.1 图像基本运算的概述(Introduction)

## 点运算

点运算是指对一幅图像中每个像素点的灰度值进行计算的方法。

## 代数运算、逻辑运算

代数运算或逻辑运算是指将两幅或多幅图像通过对应像素之间的加、减、乘、除运算或逻辑与、或、非运算得到输出图像的方法。

# 3.1 图像基本运算的概述(Introduction)

## 几何运算

几何运算就是改变图像中物体对象（像素）之间的空间关系。

从变换性质来分，几何变换可以分为图像的位置变换（平移、镜像、旋转）、形状变换（放大、缩小）以及图像的复合变换等。

# 3.2 点运算 (Point Operation)

## 1. 点运算的定义

设输入图像的灰度为$f(x,y)$，输出图像的灰度为$g(x,y)$，则点运算可以表示为：

$$g(x,y)=T[f(x,y)]$$

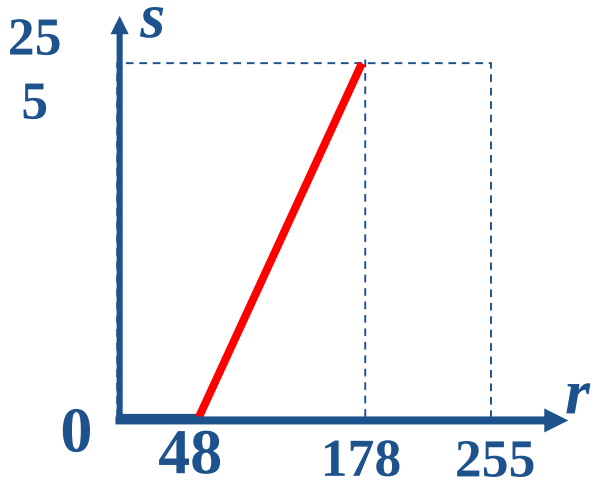
灰度变换函数

其中T[ ]是对$f$在（$x$，$y$）点值的一种数学运算，即点运算是一种像素的逐点运算，是灰度到灰度的映射过程，故称T[ ]为灰度变换函数。

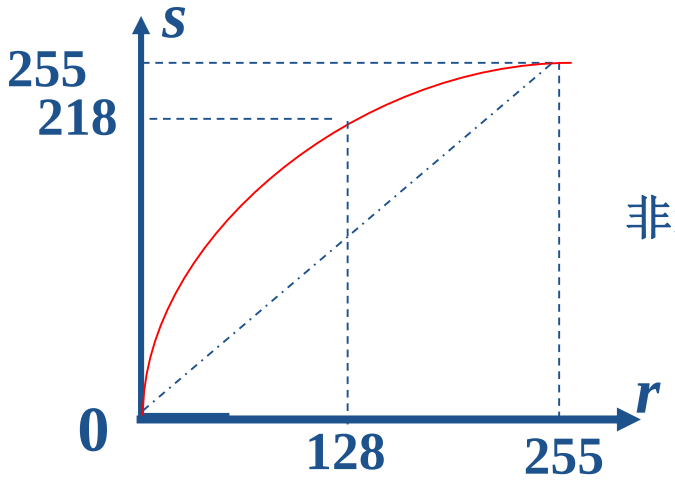
# 3.2 点运算 (Point Operation)

若令$f(x, y)$和$g(x, y)$在任意点（$x$，$y$）的灰度级分别为$r$和$s$，则灰度变换函数可简化表示为：

$$s = T[r]$$




3.1 对比度增大




3.2 加亮、减暗图像

非线性灰度变换

点运算可以改变图像数据所占据的灰度值范围，从而改善图像显示效果。

# 3.2 点运算 (Point Operation)

## 2. 点运算的分类

点运算又称为“对比度增强”、“对比度拉伸”、“灰度变换”等，按灰度变换函数T[ ]的性质，可将点运算分为：

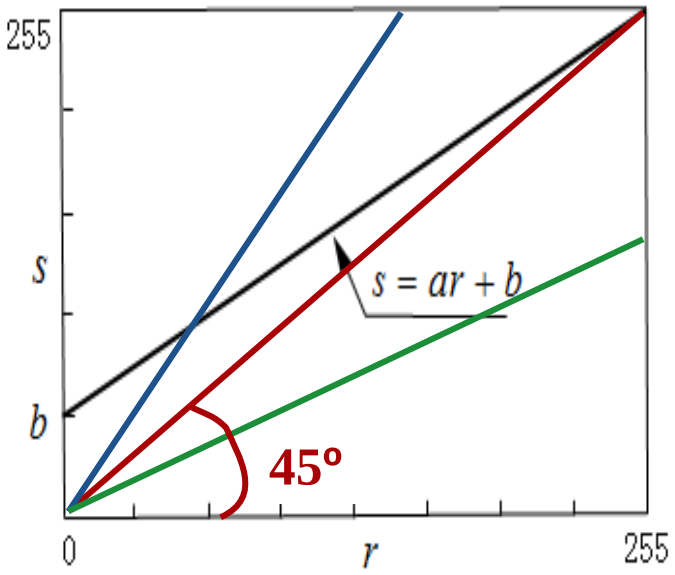
- 点运算
  - 灰度变换增强
    - 线性灰度变换（线性点运算）
    - 分段线性灰度变换（分段线性点运算）
    - 非线性灰度变换（非线性点运算）
  - 直方图增强(5.2.2 基于直方图处理的图像增强)

# 3.2.1线性点运算（Linear Point Operation）

## 1、线性点运算

线性点运算的灰度变换函数形式可以采用线性方程描述，即

$$s=ar+b$$




黑线： $0<a<1,\quad b>0$ 输出灰度压缩

红线： $a=1,\quad b=0$ 输出灰度不变

蓝线： $a>1,\quad b=0$ 输出灰度扩展整体变亮

绿线： $0<a<1,\quad b=0$ 输出灰度压缩，整体变暗

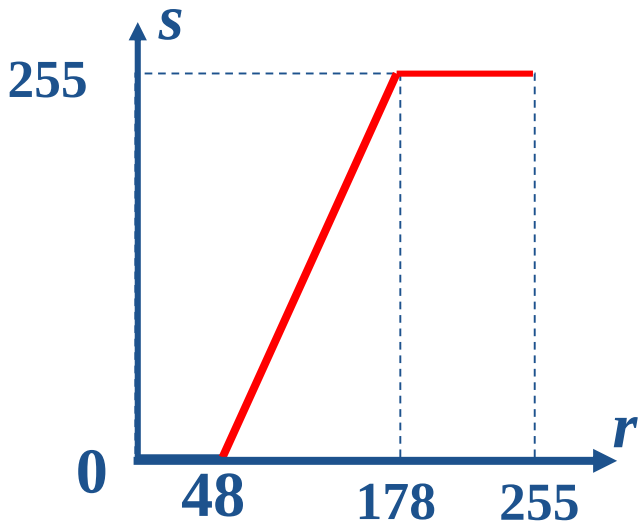
图 3.3线性点运算

# 3.2.1线性点运算（Linear Point Operation）

线性点运算的应用 $s = ar + b$

1）如果a>1，输出图像的对比度增大（灰度扩展）

变换前



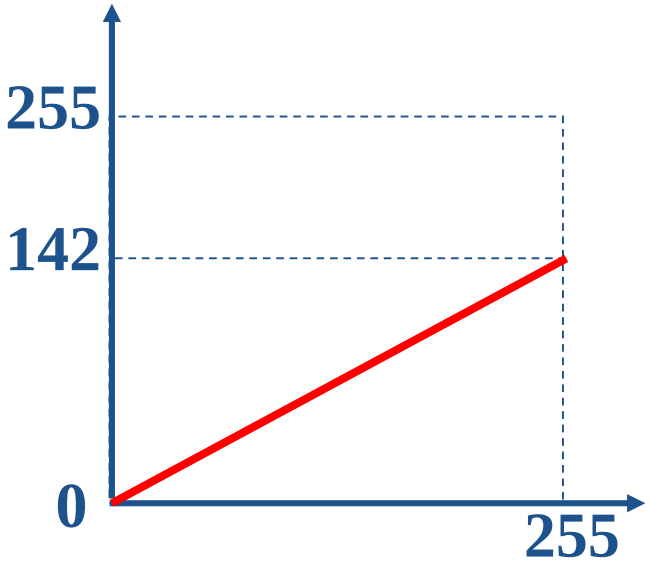

3.4 对比度增大

变换后

# 3.2.1线性点运算（Linear Point Operation）

2） 如果0<a<1，输出图像的对比度减小（灰度压缩）

变换前

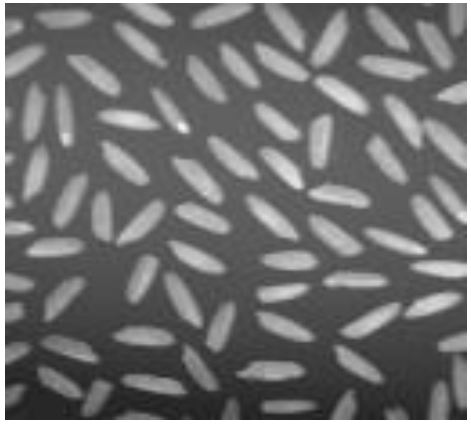



3.5 降低对比度

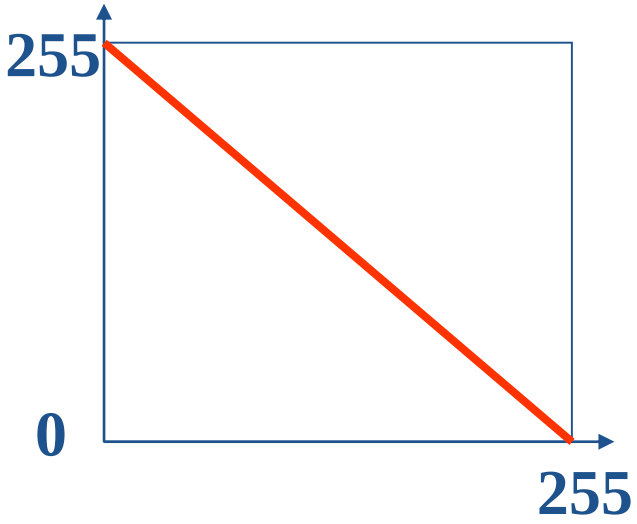
变换后

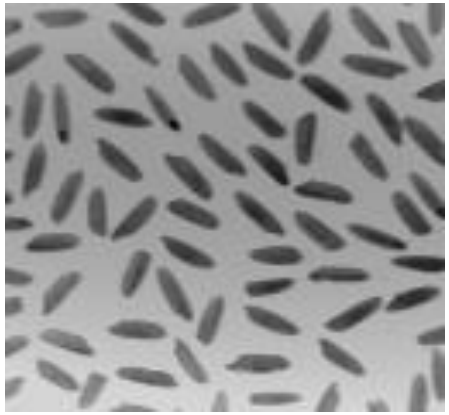
# 3.2.1线性点运算（Linear Point Operation）

3）如果a为负值，暗区域将变亮，亮区域将变暗



变换前



变换后

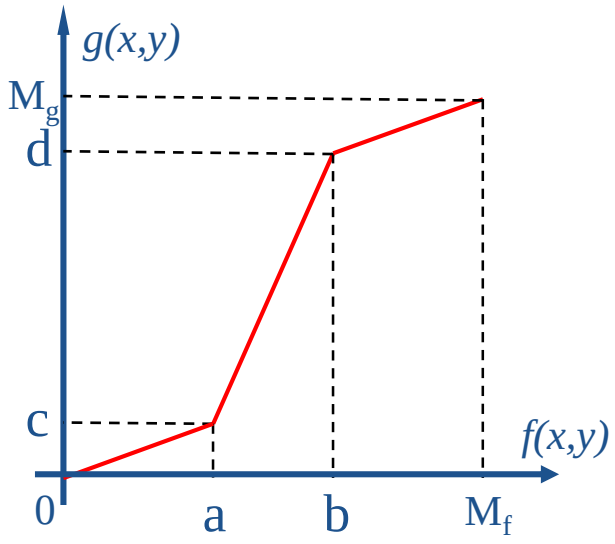
# 3.2.1线性点运算（Linear Point Operation）

## 2、分段线性点运算

将感兴趣的灰度范围线性扩展，相对抑制不感兴趣的灰度区域。

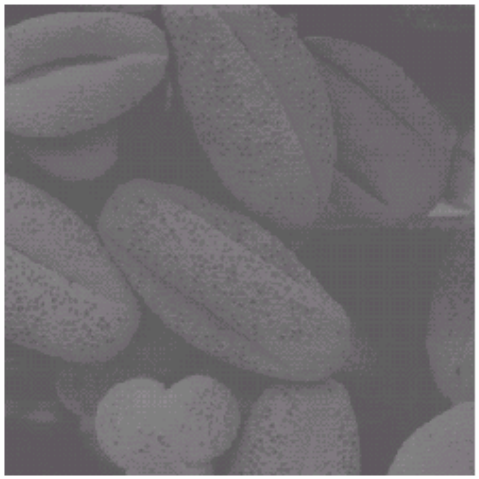
设$f(x,y)$灰度范围为$[0,M_f]$，$g(x,y)$灰度范围为$[0,M_g]$，

$$g(x,y)=\begin{cases}\dfrac{M_g-d}{M_f-b}[f(x,y)-b]+d & b\le f(x,y)\le M_f\\ \dfrac{d-c}{b-a}[f(x,y)-a]+c & a\le f(x,y)<b\\ \dfrac{c}{a}f(x,y) & 0\le f(x,y)<a\end{cases}$$

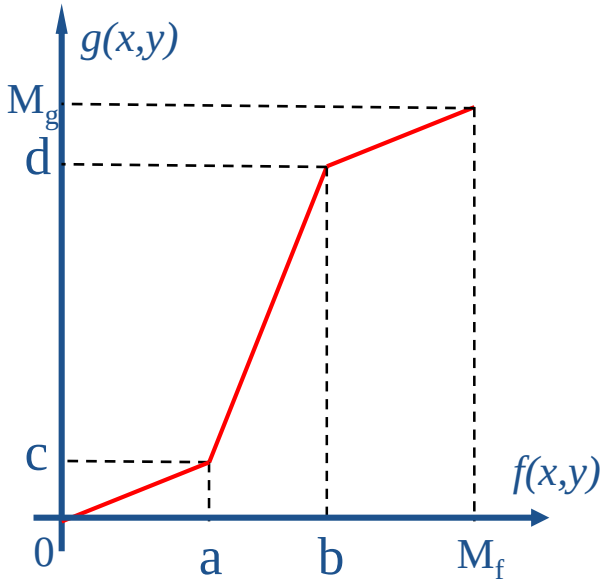


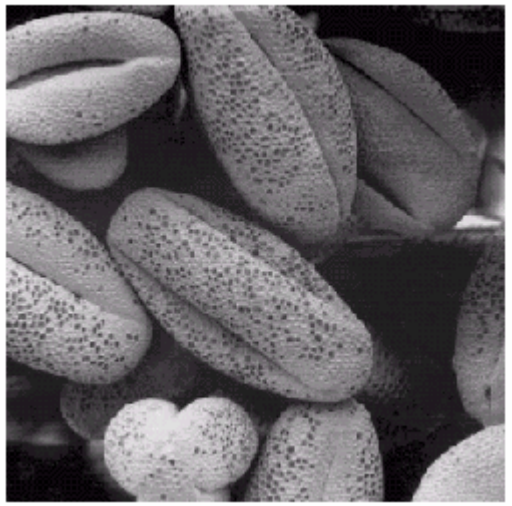
# 3.2.1线性点运算（Linear Point Operation）

## 分段线性点运算的应用



变换前



变换后

# 3.2.2非线性点运算(Non-Linear Point Operation)

## 1、非线性点运算

非线性点运算的输出灰度级与输入灰度级呈非线性关系，常见的非线性灰度变换为对数变换和幂次变换。

## 1）、对数变换

对数变换的一般表达式为： $s = c \log(1 + r)$

其中C是一个常数。

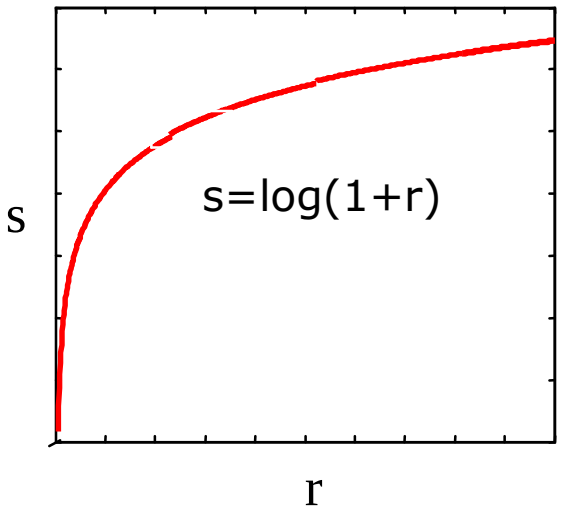


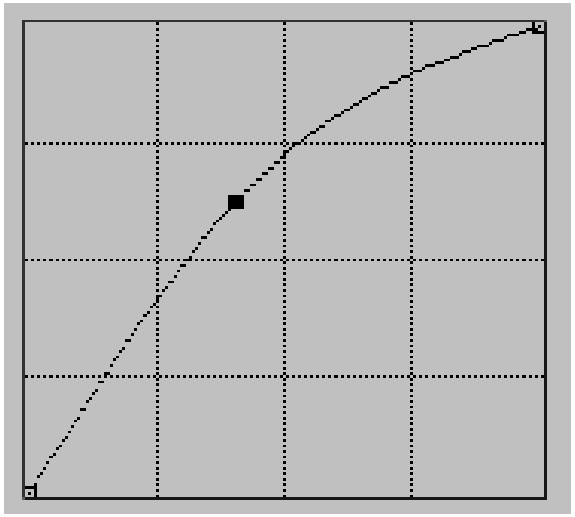

图3.9 对数曲线图

低灰度区扩展，高灰度区压缩。
图像加亮、减暗。

非线性拉伸不是对图像的整个灰度范围进行扩展，而是有选择地对某一灰度值范围进行扩展，其他范围的灰度值则有可能被压缩。

# 3.2.2非线性点运算(Non-Linear Point Operation)

## 非线性点运算应用实例1

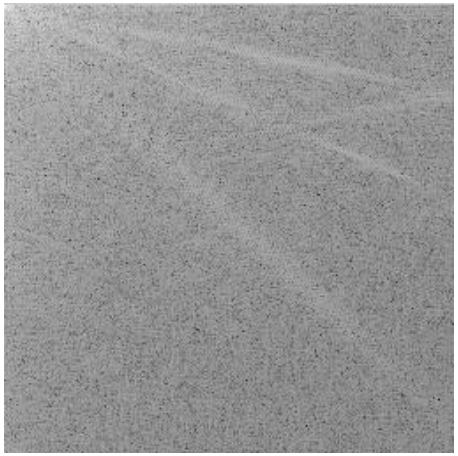


对比度拉伸效果：图像加亮、减暗

# 3.2.2非线性点运算(Non-Linear Point Operation)

## 非线性点运算应用实例2：傅里叶频谱的显示

原始图像

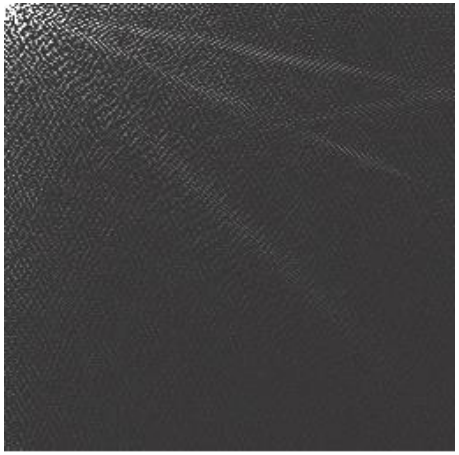


原始图像的傅里叶谱

$s = c \log(1 + r)$

此时，C=1

经对数灰度变换后的频谱图



# 3.2.2非线性点运算(Non-Linear Point Operation)

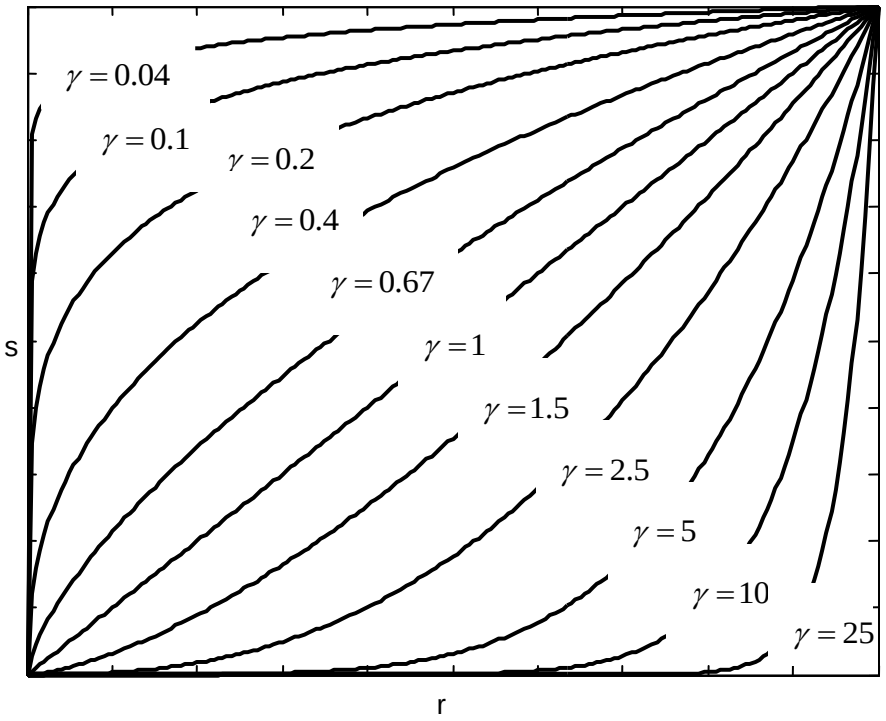
## 2）、幂次变换

幂次变换的一般形式为：$s = cr^{\gamma}$

其中C和$\gamma$ 为正常数。

$0 < \gamma < 1$

加亮、减暗图像

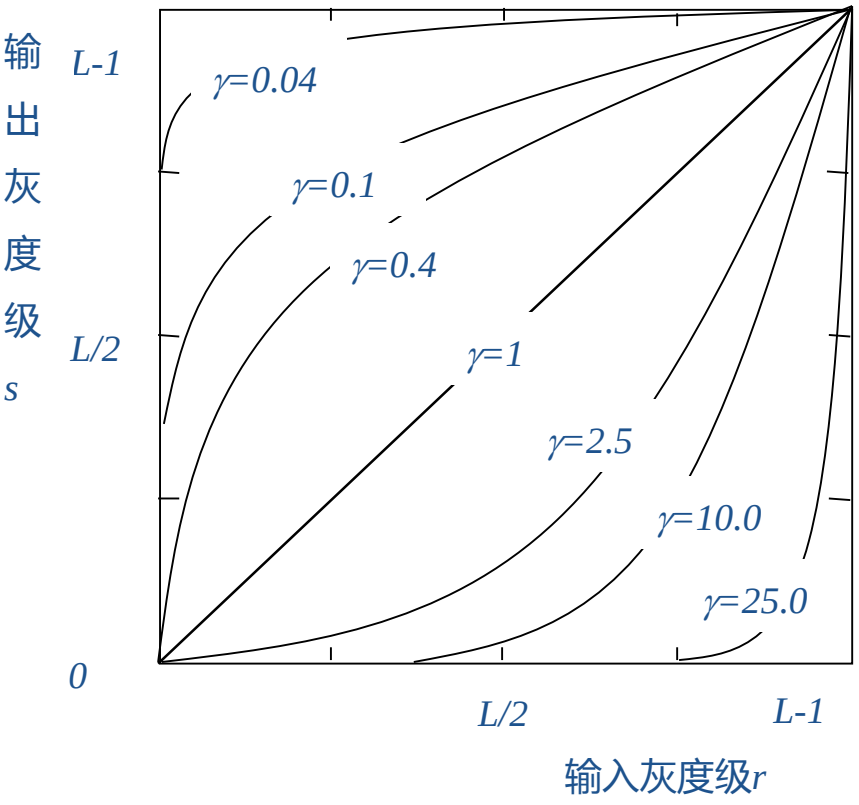



$\gamma > 1$

加暗、减亮图像

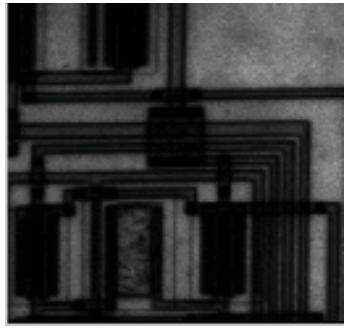
# 3.2.2非线性点运算(Non-Linear Point Operation)

## 非线性点运算应用实例3

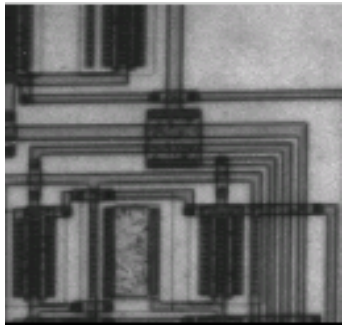



不同$\gamma$的$s=cr^{\gamma}$曲线及图像变换结果

加暗、减亮图像

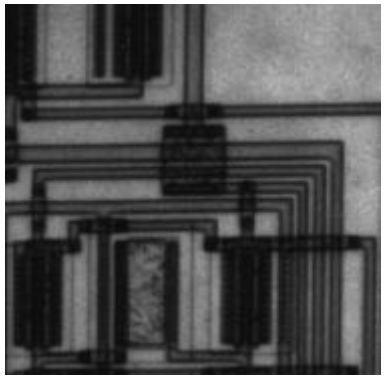


$\gamma=1.5$



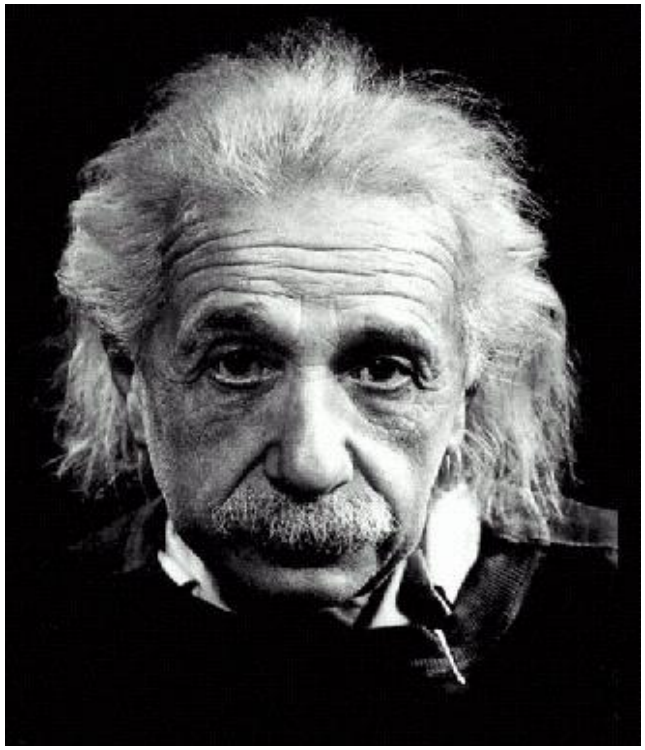
$\gamma=0.66$

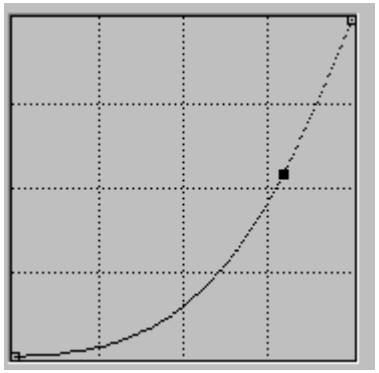
加亮、减暗图像

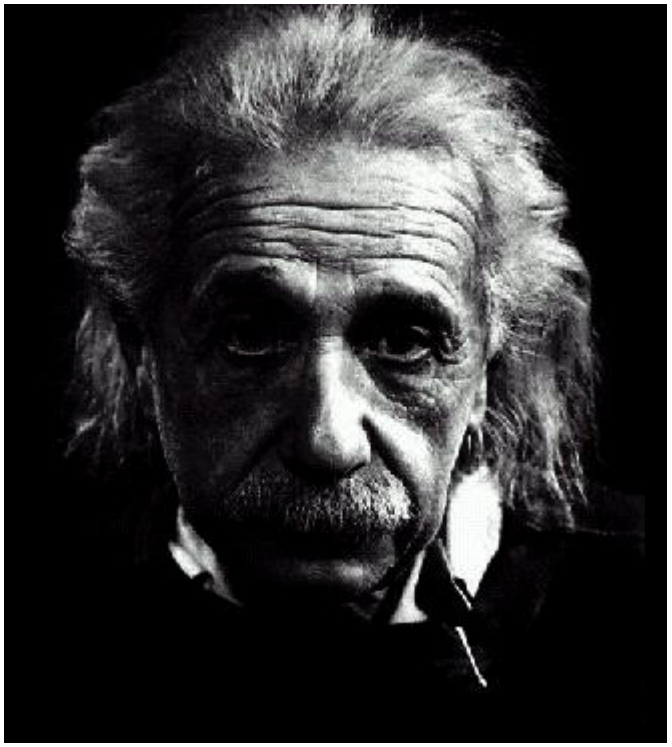


原始图像

# 3.2.2非线性点运算(Non-Linear Point Operation)







加暗、减亮图像

# 3.2.2非线性点运算(Non-Linear Point Operation)

思考问题：

1、点运算是否会改变图像内像素点之间的空间位置关系？

点运算是一种像素的逐点运算，它与相邻的像素之间没有运算关系，点运算不会改变图像内像素点之间的空间位置关系。

2、对图像灰度的拉伸，非线性拉伸与分段线性拉伸的区别？

非线性拉伸不是通过在不同灰度值区间选择不同的线性方程来实现对不同灰度值区间的扩展与压缩，而是在整个灰度值范围内采用统一的非线性变换函数，利用函数的数学性质实现对不同灰度值区间的扩展与压缩。

# 3.3代数运算与逻辑运算
# (Algebra and Logical Operation)

## 1.概念

代数运算是指两幅或多幅输入图像之间进行点对点的加、减、乘、除运算得到输出图像的过程。如果记输入图像为A(x,y)和B(x,y)，输出图像为C(x,y)，则有如下四种形式：

代数运算的四种基本形式

$$C(x,y)=A(x,y)+B(x,y)$$

$$C(x,y)=A(x,y)-B(x,y)$$

$$C(x,y)=A(x,y)\times B(x,y)$$

$$C(x,y)=A(x,y)\div B(x,y)$$

# 3.3代数运算与逻辑运算
# (Algebra and Logical Operation)

## 逻辑运算

逻辑运算是指将两幅或多幅图像通过对应像素之间的与、或、非逻辑运算得到输出图像的方法。

在进行图像理解与分析领域比较有用。运用这种方法可以为图像提供模板，与其他运算方法结合起来可以获得某种特殊的效果。

# 3.3.1加法运算(Addition)

## 1、加法运算

$$C(x,y) = A(x,y) + B(x,y)$$

主要应用举例：

去除“叠加性”随机噪音

生成图像叠加效果

# 3.3.1加法运算(Addition)

## 去除“叠加性”噪音

对于原图象f(x,y),有一个噪音图像集 { $g_i(x,y)$ }　　i =1,2,...M

其中: $g_i(x,y) = f(x,y) + e_i(x,y)$

$$\underbrace{g(x,y)}_{\text{混入噪声的图像}} = \underbrace{f(x,y)}_{\text{原始图像}} + \underbrace{e(x,y)}_{\text{随机噪声}}$$

M个图像的均值为:

$$\bar{g}(x,y) = \frac{1}{M}\sum_{i=1}^{M}[f_i(x,y)+e_i(x,y)]$$

$$= f(x,y) + \frac{1}{M}\sum_{i=1}^{M}e_i(x,y)$$

当: 噪音$e_i(x,y)$为互不相关，且均值为0时，上述图象均值将降低噪音的影响。

# 3.3.1加法运算(Addition)

$$\bar{g}(x,y)=\frac{1}{M}\sum_{i=1}^{M}[f_i(x,y)+e_i(x,y)]$$

$$=f(x,y)+\frac{1}{M}\sum_{i=1}^{M}e_i(x,y)$$

则 $\bar{g}(x,y)$ 是 $f(x,y)$ 的无偏估计

$$\because E\{\bar{g}(x,y)\}=E\left\{\frac{1}{M}\sum_{i=1}^{M}g_i(x,y)\right\}=\frac{1}{M}\sum_{i=1}^{M}E\{g_i(x,y)\}$$

$$=\frac{1}{M}\sum_{i=1}^{M}\{E[f_i(x,y)]+E[e_i(x,y)]\}$$

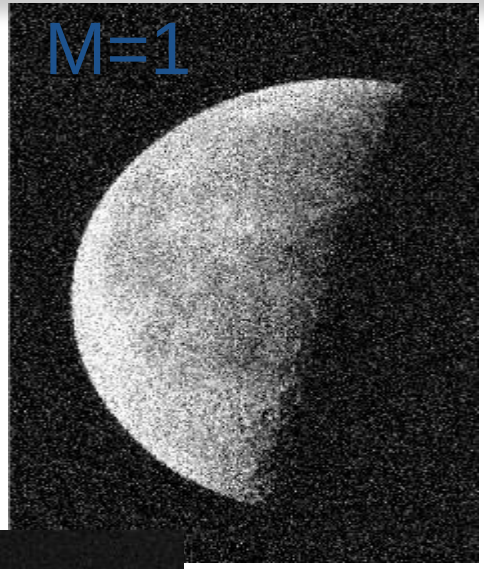
$$=\frac{1}{M}\sum_{i=1}^{M}f_i(x,y)=f(x,y)$$

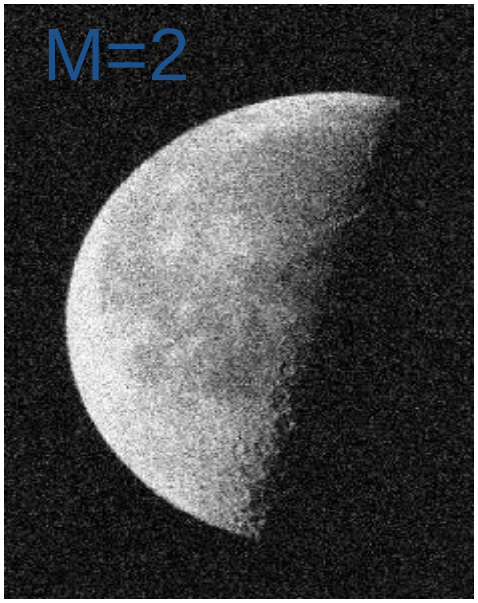
利用同一景物的多幅图像取平均、消除噪声。取M个图像相加求平均得到1幅新图像，一般选8幅取平均。

# 3.3.1加法运算(Addition)

相加

- Addition:
  - averaging for noise reduction





# 3.3.1加法运算(Addition)

生成图象叠加效果：可以得到各种图像合成的效果，也可以用于两张图片的衔接。

# 3.3.2减法运算 (Subtraction )

## 减法运算

将同一景物在不同时间拍摄的图像或同一景物在不同波段的图像相减，这就是图像的减法运算。实际中常称为差影法。

$$C(x,y) = A(x,y) - B(x,y)$$

差值图像提供了图像间的差值信息，能用于指导动态监测、运动目标的检测和跟踪、图像背景的消除及目标识别等。

主要应用举例：

差影法(检测同一场景两幅图像之间的变化)

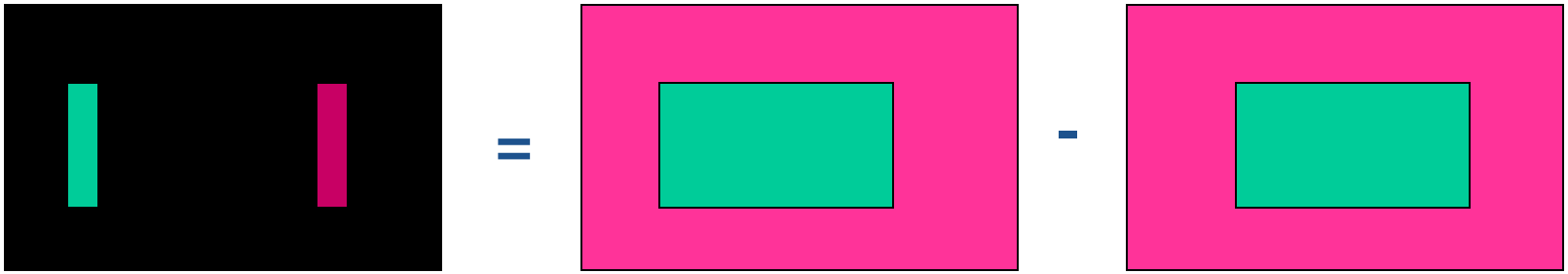
混合图像的分离

# 3.3.2减法运算 (Subtraction )

检测同一场景两幅图像之间的变化

设：时刻1的图像为 $T_1(x, y)$，
时刻2的图像为 $T_2(x, y)$

$$g(x, y) = T_2(x, y) - T_1(x, y)$$




$g(x,y)$ $T_1(x,y)$ $T_2(x,y)$

# 3.3.2减法运算（Subtraction ）

## 差影法在自动现场监测中的应用

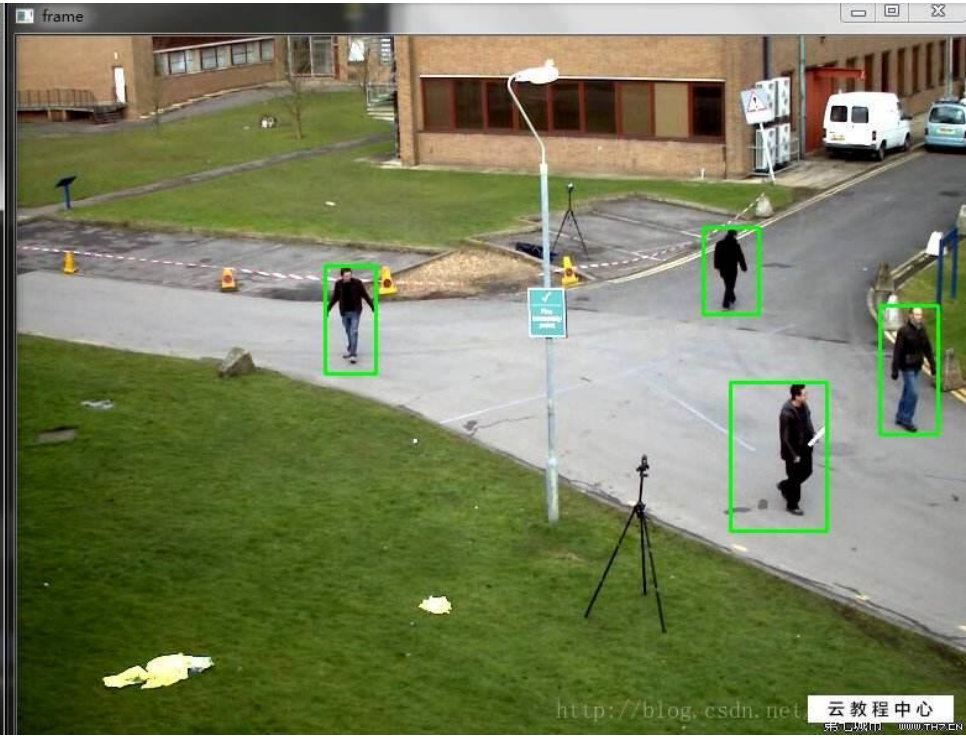
1、在银行金库内，摄像头每隔一固定时间拍摄一幅图像，并与上一幅图像做差影，如果图像差别超过了预先设置的阈值，则表明可能有异常情况发生，应自动或以某种方式报警；

2、用于遥感图像的动态监测，差值图像可以发现森林火灾、洪水泛滥，监测灾情变化等；

3、也可用于监测河口、海岸的泥沙淤积及监视江河、湖泊、海岸等的污染；

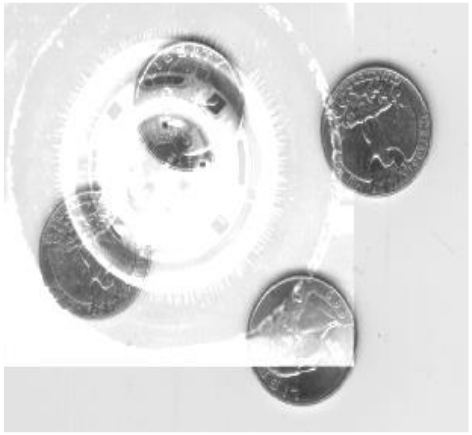
4、利用差值图像还能鉴别出耕地及不同的作物覆盖情况。

# 背景减除法检测移动目标

# 3.3.2减法运算 (Subtraction )

## 混合图像的分离



（a）混合图像

（b）被减图像

（c）差影图像

图3.6　差影法进行混合图像的分离

# 3.3.2减法运算（Subtraction ）

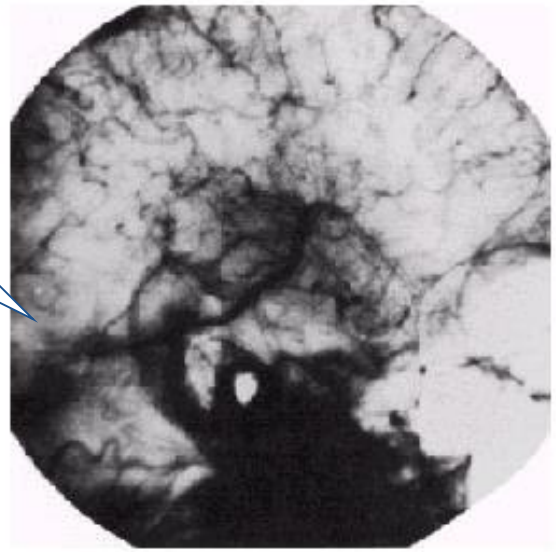
## 消除背景影响

即去除不需要的叠加性图案

设：背景图像b(x ，y)，前景背景混合图像f(x ，y)

g(x, y)=f(x, y)−b(x, y)

g(x, y) 为去除了背景图像



背景图像

差值图像

(a) 从病人头顶向下拍摄的X光照片

（b）碘元素注入后拍摄的X光照片与背景图像的差值

# 3.3.3乘法运算(Multiplication)

## 乘法运算

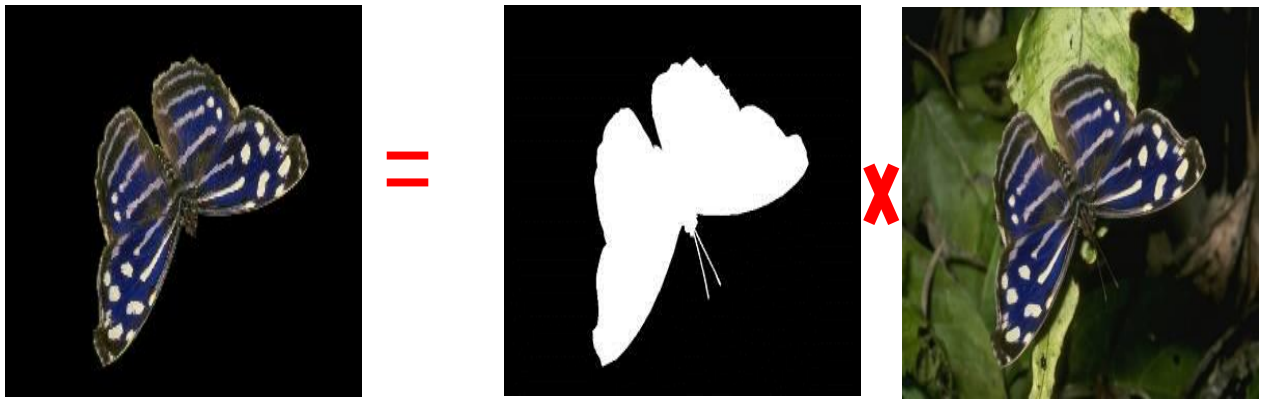
$$C(x,y) = A(x,y) \times B(x,y)$$

主要应用举例：

图像的局部显示

改变图像的灰度级

图像的局部显示



# 3.3.3乘法运算(Multiplication)

## 改变图像的灰度级

(a) 原图

(b) 乘以1.2

(c) 乘以2

图3.8 乘法运算结果

# 3.3.4除法运算(Division)

- **除法运算** $C(x,y)=A(x,y)\div B(x,y)$

简单的除法运算可用于改变图像的灰度级，常用于遥感图像处理中。

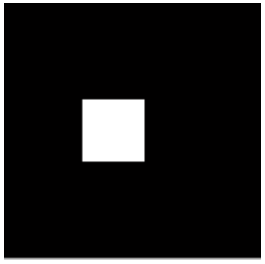
在四种算术运算中，减法与加法在图像增强处理中最为有用。

# 3.3.5逻辑运算(Logical Operation)

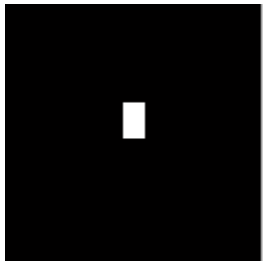
## “与”、“或”，“非”逻辑运算

逻辑运算主要以像素对像素为基础在两幅或多幅图像间进行。

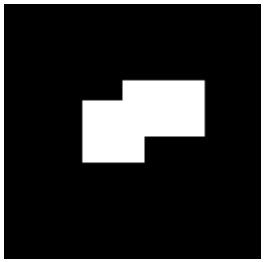
（a）A图



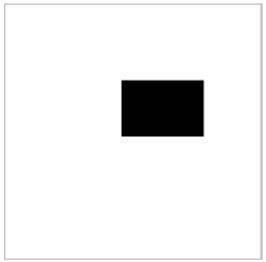
（b）B图



(c) A、B相与结果图



(d) A、B相或结果图



(e) A取反结果图

**图3.7　图像的逻辑运算**

# 3.3.5逻辑运算(Logical Operation)

“与”、“或”逻辑运算可以从一幅图像中提取子图像

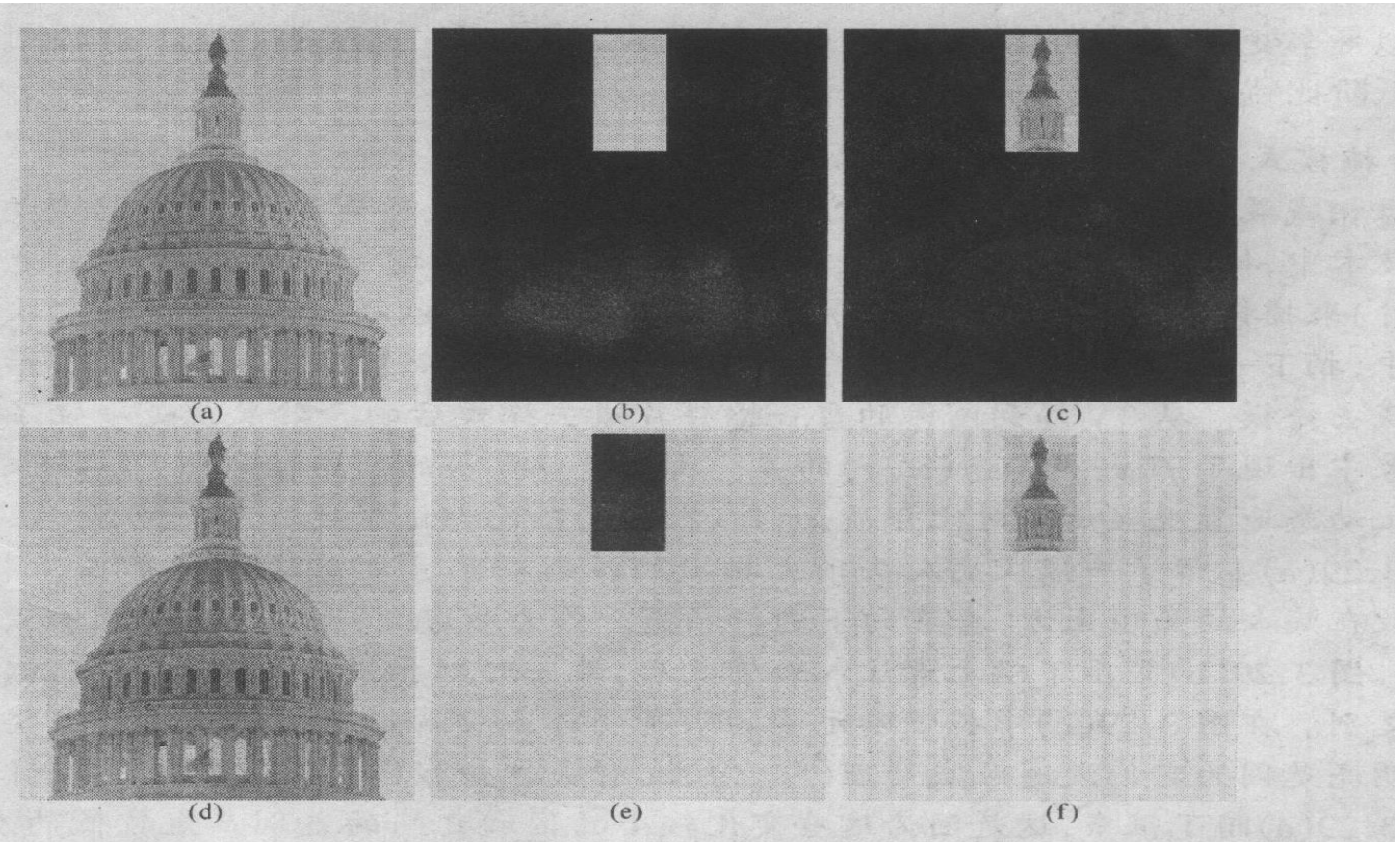



图 3.27 (a)原图像,(b)“与”图像模板,(c)对图像(a)和(b)进行“与”操作的结果,(d)原图像,(e)进行“或”图像模板,(f)对图像(d)和(e)进行“或”操作的结果

# 3.4 几何运算（*Geometric Operation*）

- 几何运算
- 几何运算就是改变图像中物体对象（像素）之间的空间关系。
- 从变换性质来分，几何变换可以分为图像的位置变换（平移、镜像、旋转）、形状变换（放大、缩小）以及图像的复合变换等。

# 3.4 几何运算(Geometric Operation)

- **几何运算**

图像几何运算的一般定义为：

$$g(x,y) = f(u,v) = f(p(x,y), q(x,y))$$

式中，$u = p(x,y)$，$v = q(x,y)$ 唯一的描述了空间变换，即将输入 图像 $f(u,v)$ 从 $u-v$ 坐标系变换为 $x-y$ 坐标系的输出图像 $g(x,y)$。

# 3.4.1图像的平移(Image Translation)

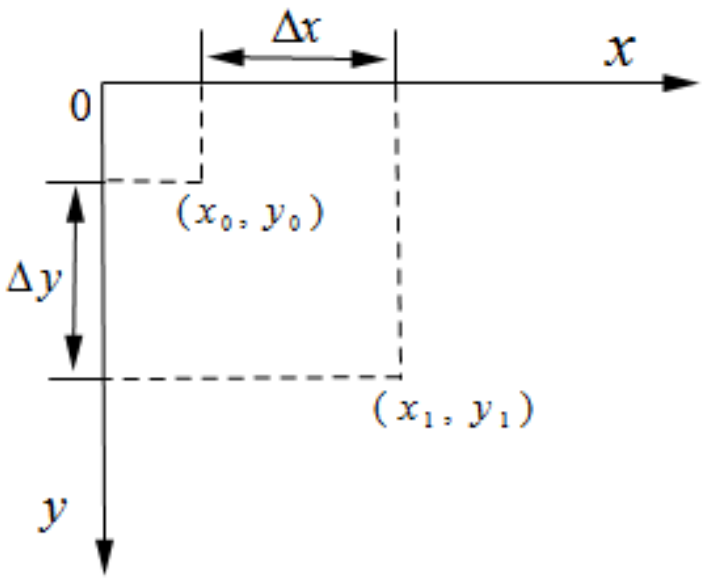



图3.8 像素点的平移

两点之间存在如下关系：

$$\begin{cases} x_1 = x_0 + \Delta x \\ y_1 = y_0 + \Delta y \end{cases}$$

# 齐次坐标

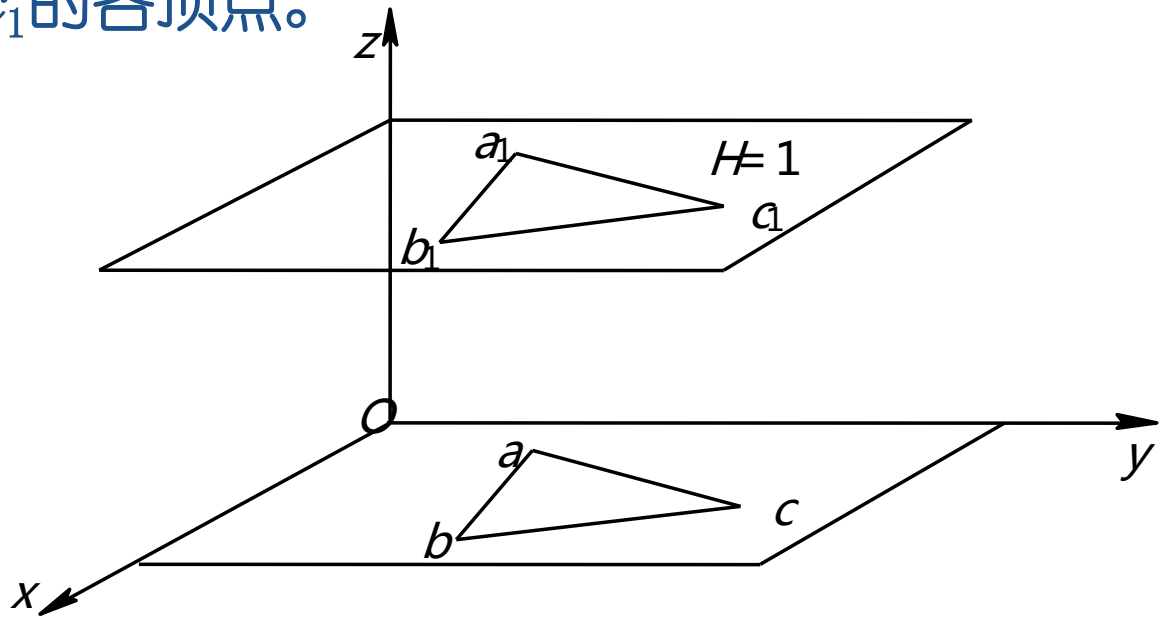
2D图像中的点坐标$(x,\ y)$表示成齐次坐标$(H_x,\ H_y,\ H)$，当$H=1$时，则$(x,\ y,\ 1)$就称为点$(x,\ y)$的规范化齐次坐标。规范化齐次坐标的前两个数是相应二维点的坐标，没有变化，仅在原坐标中增加了$H=1$的附加坐标。

由点的齐次坐标$(H_x,\ H_y,\ H)$求点的规范化齐次坐标$(x,\ y,\ 1)$，可按如下公式进行：

$$x=\frac{H_x}{H} \qquad y=\frac{H_y}{H}$$

# 齐次坐标

齐次坐标的几何意义相当于点(x，y)落在3D空间H＝1的平面上，如果将XOY 平面内的三角形abc 的各顶点表示成齐次坐标($x_i$，$y_i$，1)($i$=1，2，3)的形式，就变成$H$=1平面内的三角形$a_1b_1c_1$的各顶点。



# 3.4.1图像的平移(*Image Translation*)

$$\begin{cases} x_1 = x_0 + \Delta x \\ y_1 = y_0 + \Delta y \end{cases}$$

以矩阵形式表示平移前后的像素关系为：

$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & \Delta x \\ 0 & 1 & \Delta y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \\ 1 \end{bmatrix}$$

# 3.4.1图像的平移(Image Translation)

（a）原始图像

（b）平移后的图像

图3.9 图像的平移

# 3.4.2图像的镜像(Image Mirror)

图像的镜像（Mirror）是指原始图像相对于某一参照面旋转180°的图像

设原始图像的宽为 $w$ 高为 $h$ 原始图像中的点为 $(x_0, y_0)$，对称变换后的点为 $(x_1, y_1)$

- （1）水平镜像（相对于 $y$ 轴）

水平镜像的变换公式 如下：

$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & w \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \\ 1 \end{bmatrix}$$

# 3.4.2图像的镜像(Image Mirror)

(a)原始图像

(b)水平镜像

**图3.10　图像水平镜像变换**

# 3.4.2图像的镜像(Image Mirror)

- （2）垂直镜像（相对于 $x$ 轴）

  垂直镜像的变换公式为如下：

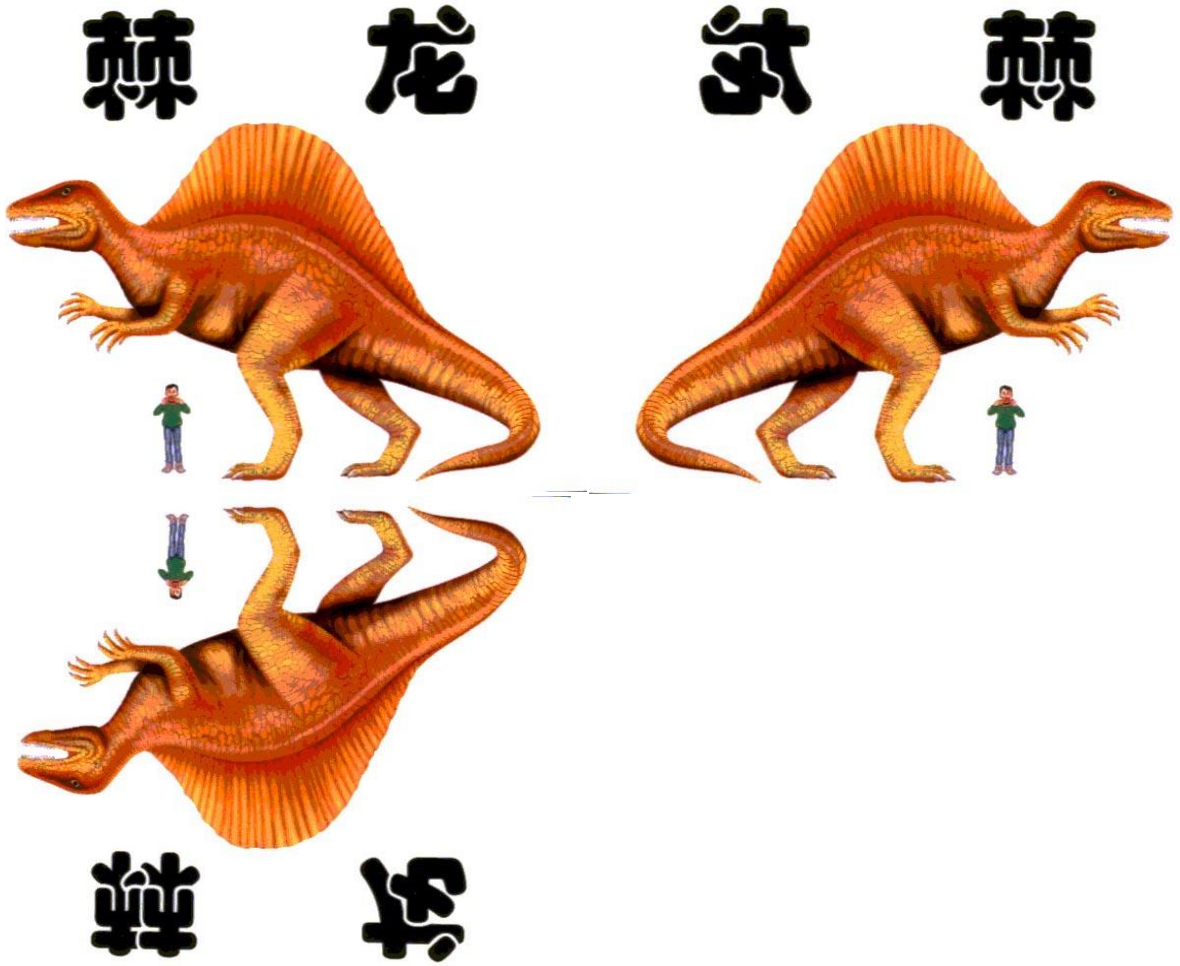
$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & h \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \\ 1 \end{bmatrix}$$

# 3.4.2图像的镜像(Image Mirror)

(a)原始图像

(b)垂直镜像

**图3.11　图像垂直镜像变换**

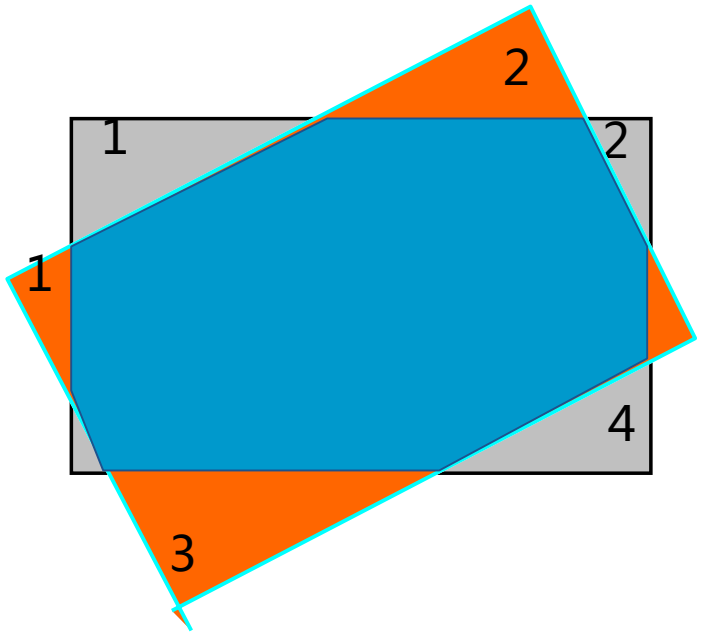


水平镜像

垂直镜像

# 3.4.3图像的旋转(*Image Rotation*)

一般图像的旋转是以图像的中心为原点，旋转一定的角度，即将图像上的所有像素都旋转一个相同的角度。



# 3.4.3图像的旋转(Image Rotation)

- 设原始图像的任意点 $A_0(x_0, y_0)$经旋转角度 $\beta$ 以后到新的位置 $A(x, y)$，为表示方便，采用极坐标形式表示，原始的角度为 $\alpha$ ，如下图所示：

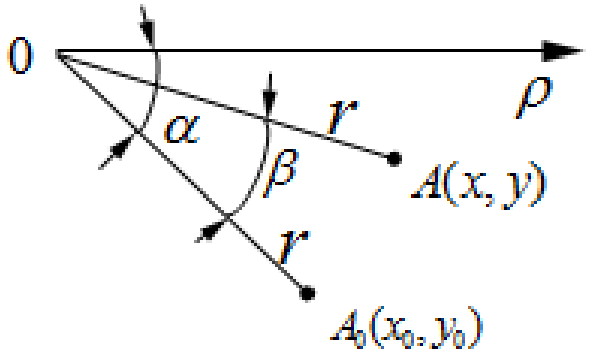



图3.12　图像的旋转

原始图像的点 $A_0(x_0, y_0)$ 的坐标如下：

$$\begin{cases} x_0 = r\cos\alpha \\ y_0 = r\sin\alpha \end{cases}$$

# 3.4.3图像的旋转(Image Rotation)

旋转到新位置以后点 $A(x,\ y)$ 的坐标如下：

$$\begin{cases} x = r\cos(\alpha-\beta) = r\cos\alpha\cos\beta + r\sin\alpha\sin\beta \\ y = r\sin(\alpha-\beta) = r\sin\alpha\cos\beta - r\cos\alpha\sin\beta \end{cases}$$

$$\begin{cases} x = x_0\cos\beta + y_0\sin\beta \\ y = -x_0\sin\beta + y_0\cos\beta \end{cases}$$

- 图像旋转用矩阵表示如下：

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\beta & \sin\beta & 0 \\ -\sin\beta & \cos\beta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \\ 1 \end{bmatrix}$$

# 3.4.3图像的旋转(Image Rotation)

（a）原图　　（b）旋转图　　（c）旋转图

**图3.13　图像的旋转**

# 3.4.3图像的旋转(Image Rotation)

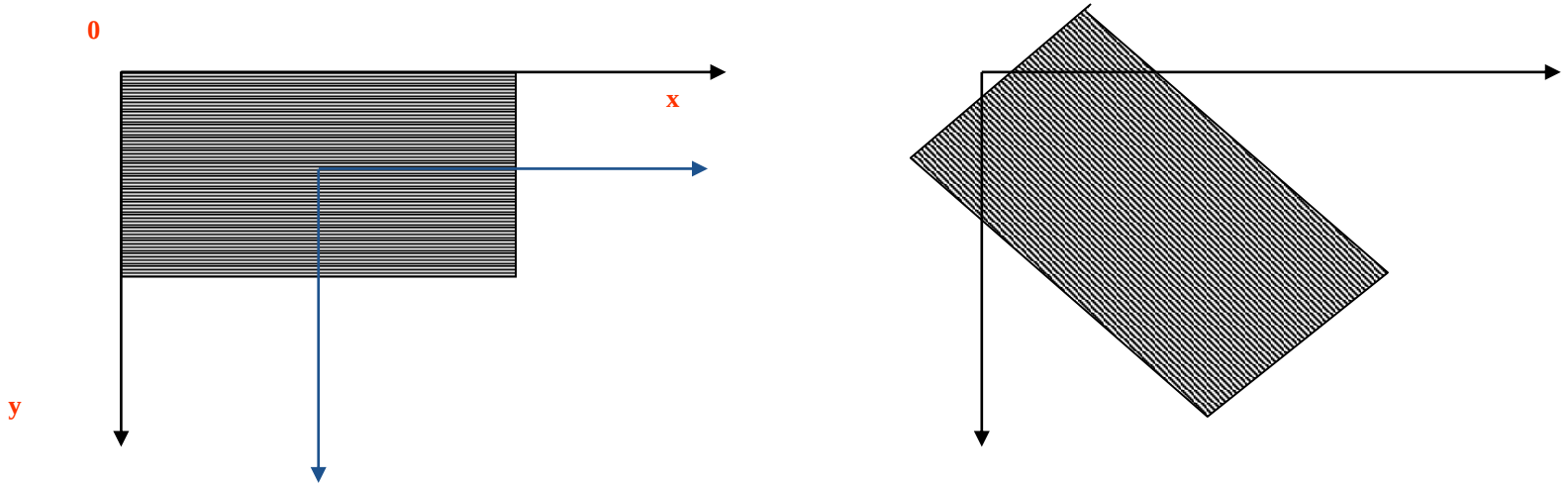
图像旋转之后，由于数字图像的坐标值必须是整数，因此，可能引起图像部分像素点的局部改变，因此，这时图像的大小也会发生一定的改变。

若图像旋转角 $\beta$ =45时，则变换关系如下：

$$\begin{cases} x = 0.707x_0 + 0.707y_0 \\ y = -0.707x_0 + 0.707y_0 \end{cases}$$

## 图像绕任意点旋转

上述的旋转是绕坐标轴原点（0，0）进行的，如果是绕某一个指定点（a，b）旋转，则先要将坐标系平移到该点，再进行旋转，然后将旋转后的图像平移回原坐标系。例如，我们这里以图像的中心为旋转中心：



利用公式进行图像旋转正变换时需要注意如下两点：

1、为了避免图像信息的丢失，图像旋转后必须进行平移变换。

2、图像旋转之后，会出现许多空洞点，我们需要对这些空洞点必须进行填充处理，否则图像旋转后的效果不好，一般也称这种操作为插值处理，可采用行或列插值方法。最简单的插值方法是，图像旋转前某一点（x，y）的像素点颜色，除了填充在旋转后坐标（x’，y’）上外，还要填充（x’+1，y’）和（x’，y’+1）。

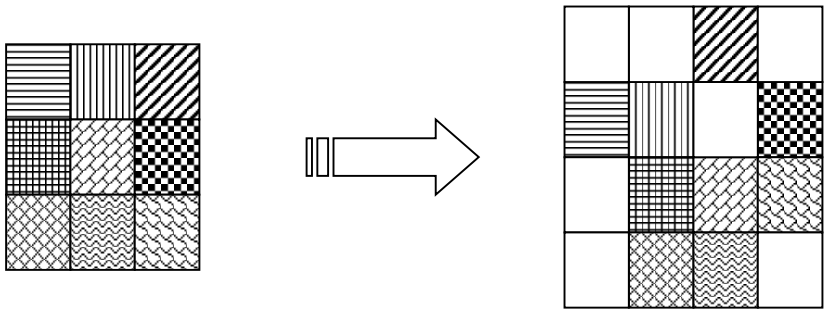


图7-9：图像的旋转

# 3.4.3图像的旋转(Image Rotation)

图像旋转角 $\beta$=45° 时，则变换关系如下：

$$\begin{cases} x = 0.707x_0 + 0.707y_0 \\ y = -0.707x_0 + 0.707y_0 \end{cases}$$

以原始图像的点（1，1）为例，旋转以后，均为小数，经舍入后为（1，0），产生了位置误差。因此，图像旋转之后，可能会出现一些空白点，需要对这些空白点进行灰度级的插值处理，否则影响旋转后的图像质量。

旋转前的图像

图旋转15°并进行插值处理的图像

# 3.4.4图像的缩放(Image Zoom)

图像全比例缩放变

数字图像的全比例缩放是指将给定的图像在 $x$方向和 $y$ 方向按相同的比例$a$ 缩放，从而获得一幅新的图像，

比例缩放前后两点$A_0(x_0, y_0)$、$A_1(x_1, y_1)$之间的关系用矩阵形式可以表示为：

$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = \begin{bmatrix} a & 0 & 0 \\ 0 & a & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \\ 1 \end{bmatrix}$$

即

$$\begin{cases} x_1 = ax_0 \\ y_1 = ay_0 \end{cases}$$

# 3.4.4图像的缩放(Image Zoom)

以$a=1/2$为例，即图像被缩小为原始图像的一半。图像被缩小一半以后根据目标图像和原始图像像素之间的关系，有如下两种缩小方法。

第一种方法是取原图像的偶数行列组成新图像；

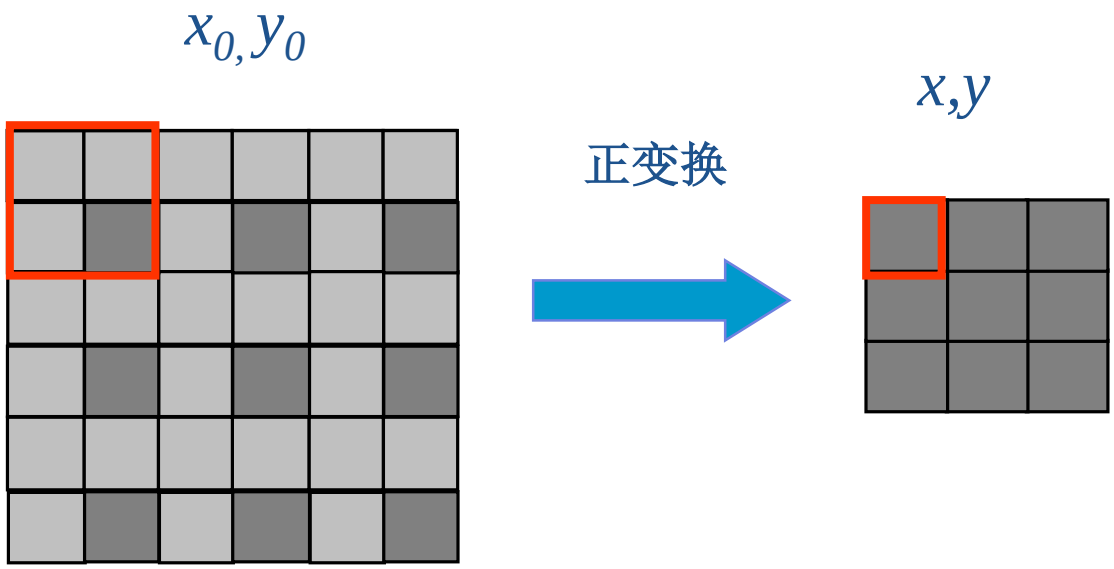
另一种方法是取原图像的奇数行列组成新图像。

# 3.4.4图像的缩放(Image Zoom)

**缩小**

$x=x_0/2$

$y=y_0/2$



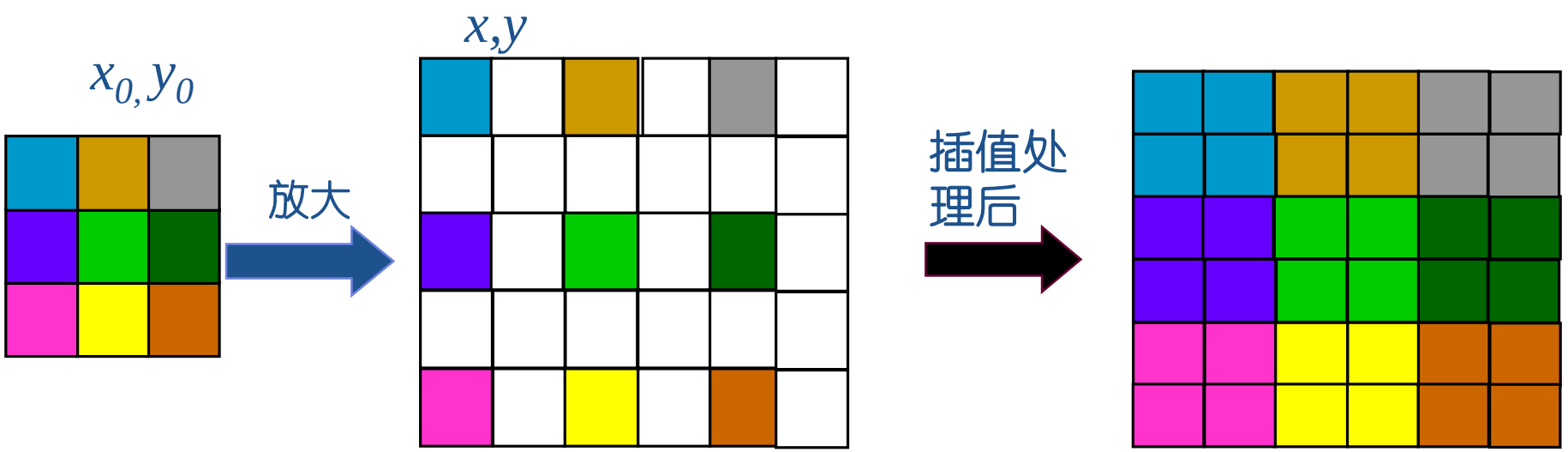
# 3.4.4图像的缩放(Image Zoom)

**放大**

$x=2x_0$

$y=2y_0$

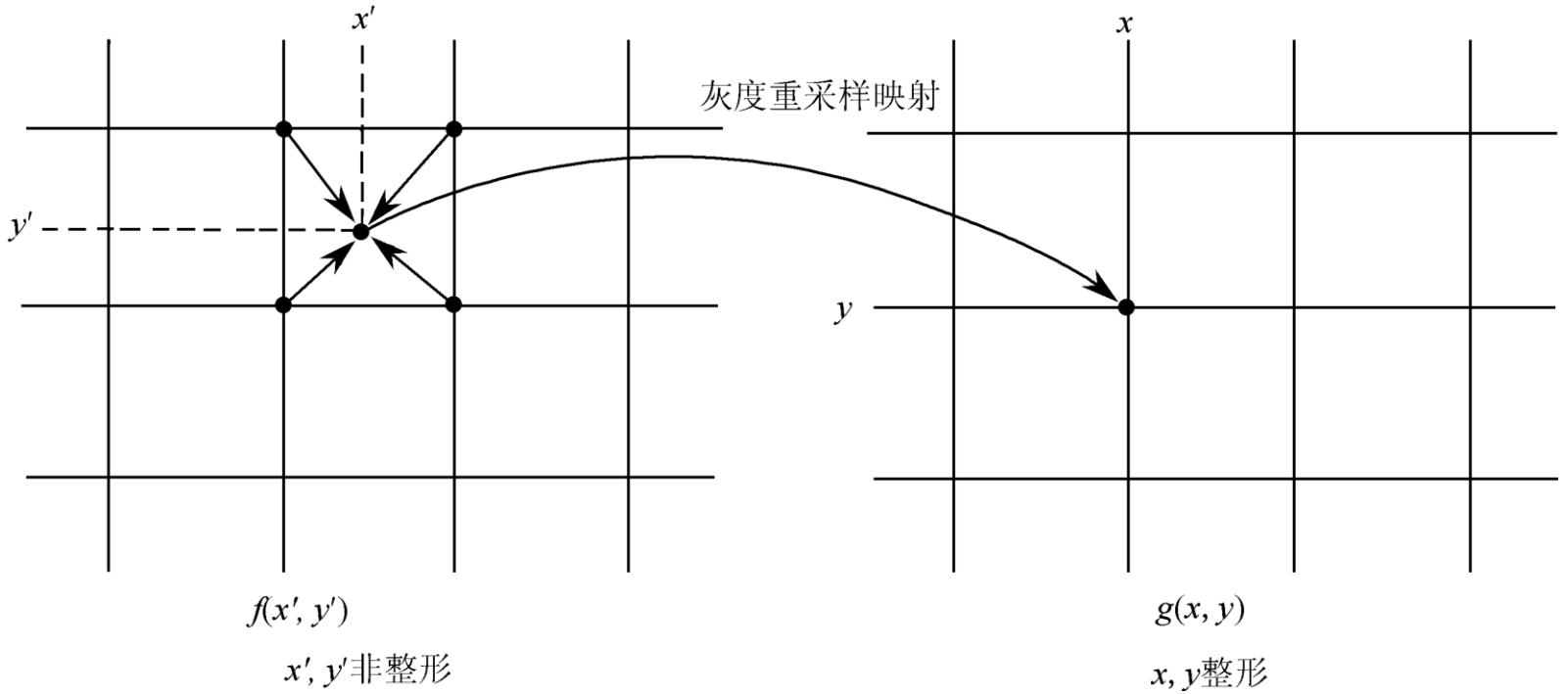
但放大后图像的像素点（0，1）对应于原始图中的像素点（0，0.5），（1，0）对应于原始图中的（0.5，0），原始图像中不存在这些像素点，那么放大图像如何处理这些问题呢？




在图像放大的正变换中，出现了很多的空格。因此，需要对放大后所多出来的一些空格填入适当的像素值。一般采用最邻近插值和线性插值法。

# 3.4. 5灰度重采样（Gray Resampling）

几何运算还需要一个算法用于灰度级的重采样。如果一个输出像素映射到四个输入像素之间，则其灰度值由灰度插值算法决定，如图3.24所示。



# 3.4. 5灰度重采样（Gray Resampling）

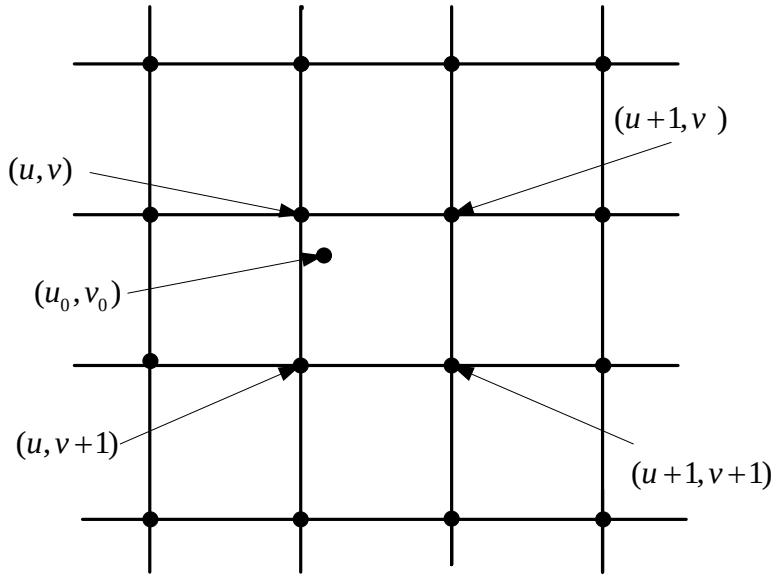



**图3.20　最近邻法**

# 3.4. 5灰度重采样（Gray Resampling）

- 最近邻法：

最近邻法是将 $(u_0,v_0)$点最近的整数坐标 $(u,v)$ 点的灰度值取为 $(u_0,v_0)$点的灰度值。在 $(u_0,v_0)$ 点各相邻像素间灰度变化较小时，这种方法是一种简单快捷的方法，但当 $(u_0,v_0)$ 点相邻像素间灰度差很大时，这种灰度估值方法会产生较大的误差。

# 3.4. 5灰度重采样（Gray Resampling）

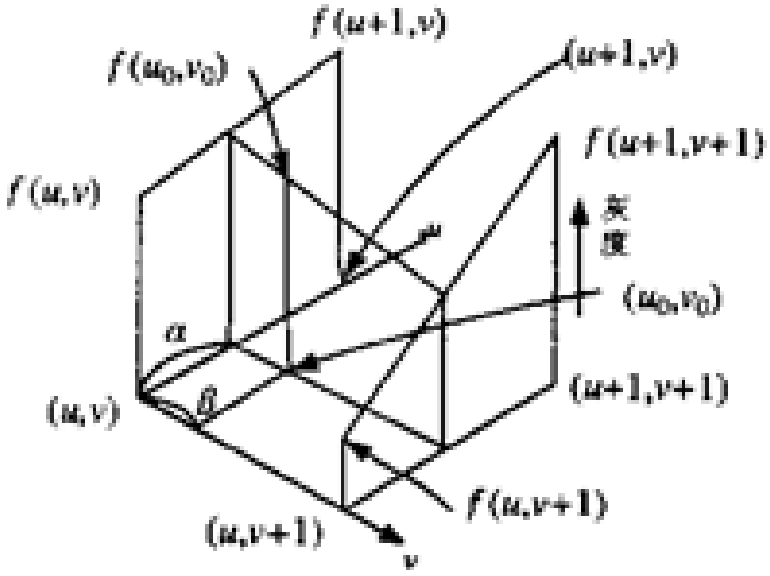



**图3.20　线性内插法**

# 3.4. 5灰度重采样（Gray Resampling）

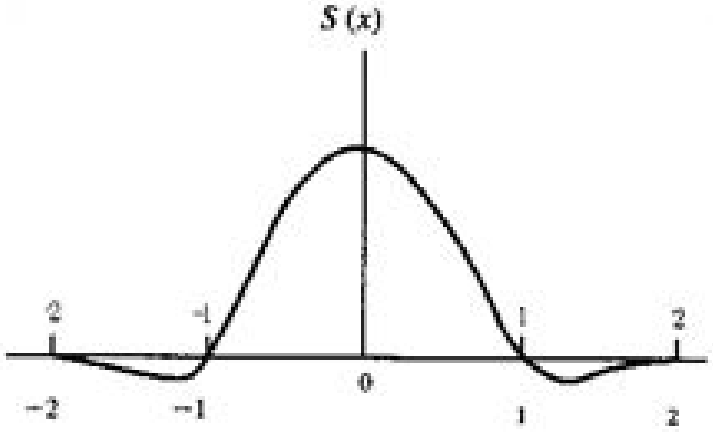



**图3.20　$\sin(\pi x)/(\pi x)$ 的三次多项式近似**

# 3.4. 5灰度重采样（Gray Resampling）

- 三次内插法

三次内插法不仅考虑 $(u_0, v_0)$ 点的直接邻点对它的影响，还考虑到该点周围16个邻点的灰度值对它的影响。由连续信号采样定理可知，若对采样值用插值函数 $S(x) = \sin(\pi x)/(\pi x)$ 插值，则可精确地恢复原函数，当然也就可精确得到采样点间任意点的值。此方法计算量很大，但精度高，能保持较好的图像边缘。

# 小结（Summary）

- 本章主要介绍了图像的基本运算，包括点运算、代数运算、逻辑运算和几何运算，举了相应的Matlab实例，并对其相应的应用做了介绍。比如说代数运算可用于去除图像的噪声，进行混合图像的分离等等。其中的几何运算包括两个步骤，一个是空间变换，一个是重采样。然后简单介绍了下常用的三种灰度插值方法—最近邻法、双线性插值法和三次内插法，比较了优缺点。


Thank You !