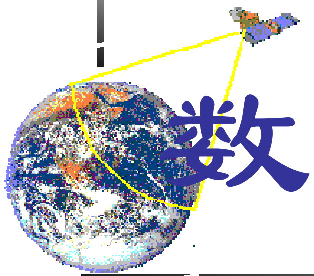


数字图像处理与分析

数字图像处理基本运算

数字图像处理基本运算

- 图像的像素级运算
  - 点运算 —— 线性点运算、非线性点运算
  - 代数运算 —— 加法、减法、乘法、除法
  - 逻辑运算 —— 求反、异或、或、与

注：变换像素灰度值

- 图像的空域变换

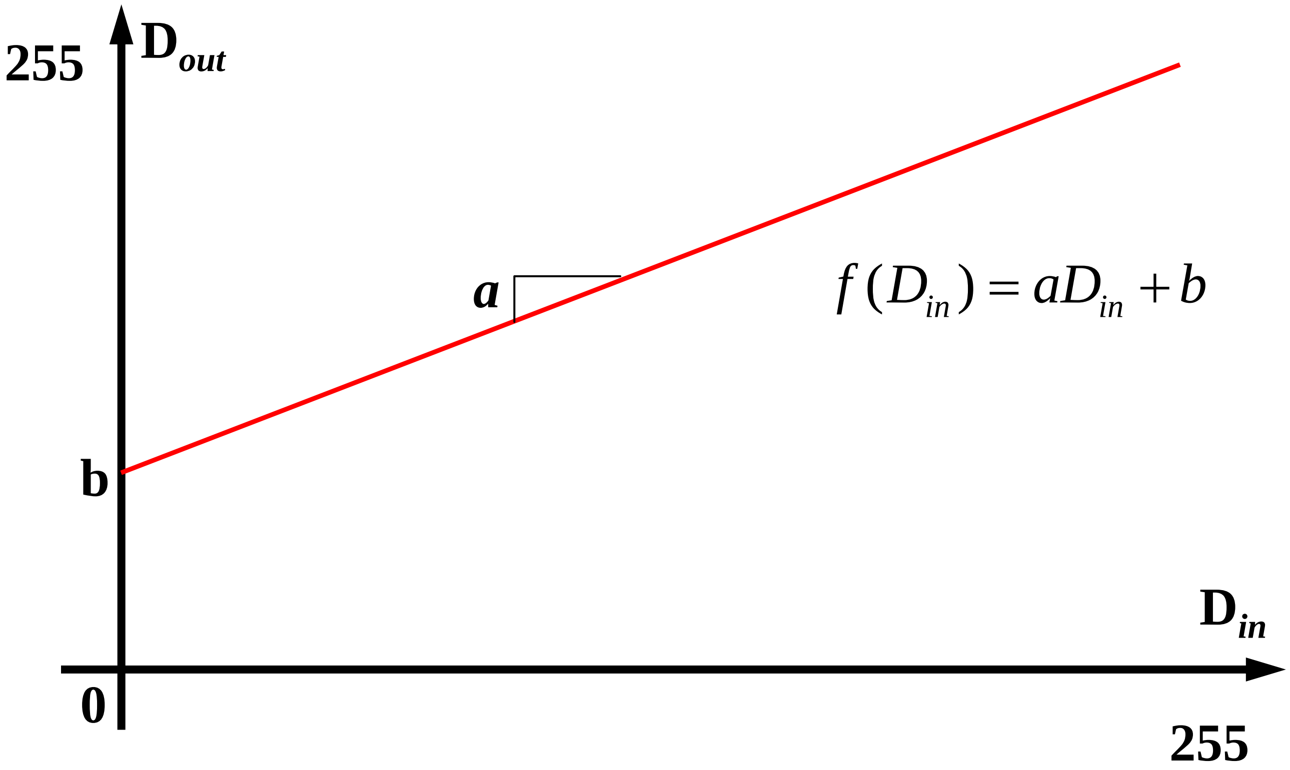
注：像素的坐标变换，其灰度值不变

  - 几何变换
  - 非几何变换
- 基本概念—灰度直方图
- 直方图变换
  - 基本理论
  - 直方图均衡

图像运算——点运算

- 线性点运算

$$D_{out} = f(D_{in}) = aD_{in} + b$$



图像运算——点运算

- 线性点运算

$$I_{out}(x,y) = a * I_{in}(x,y) + b$$

- a=1，b=0: 恒等
- a<0: 黑白反转
- |a|>1: 增加对比度
- |a|<1: 减小对比度
- b>0: 增加亮度
- b<0: 减小亮度

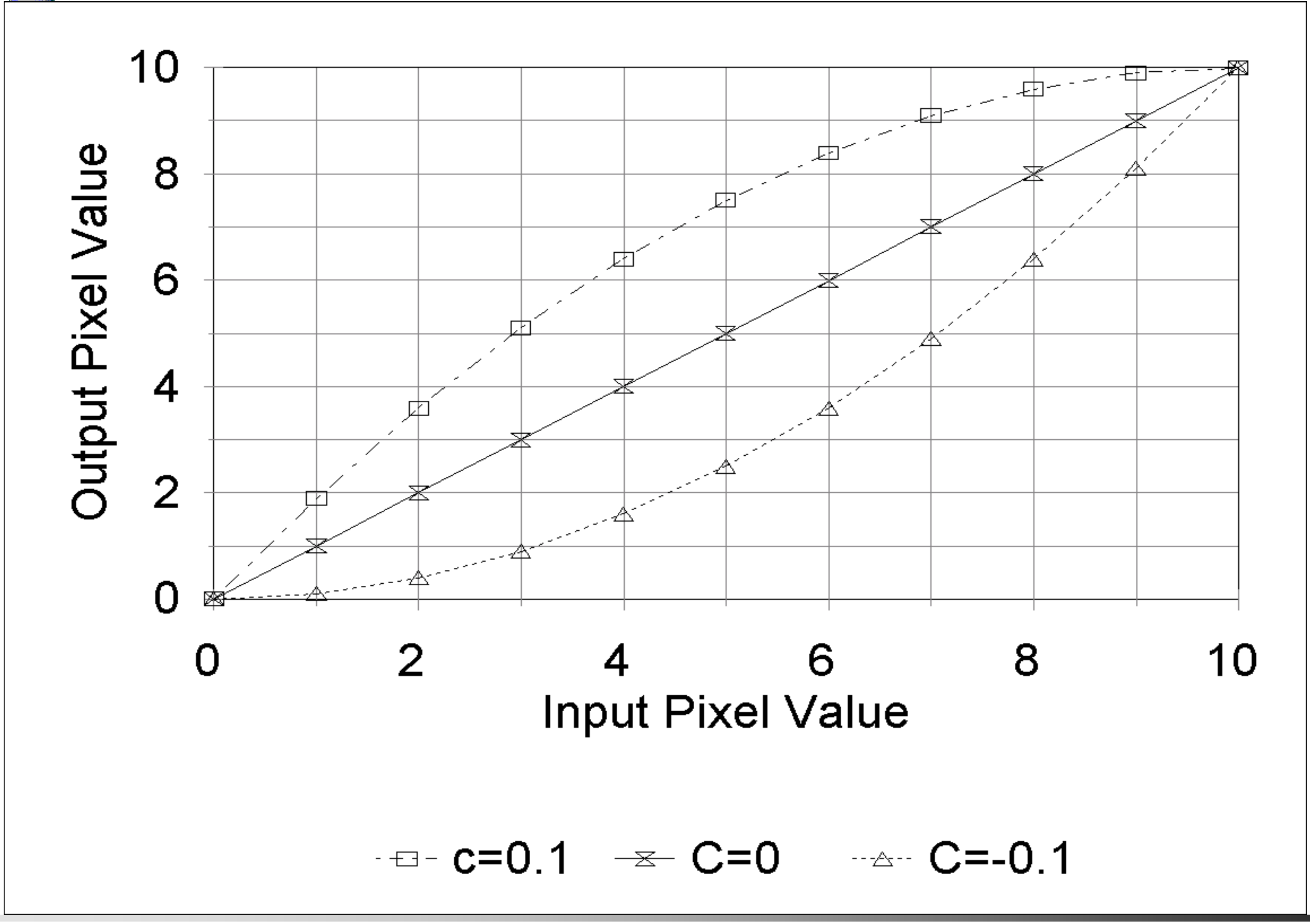
图像运算——点运算

- 非线性点运算

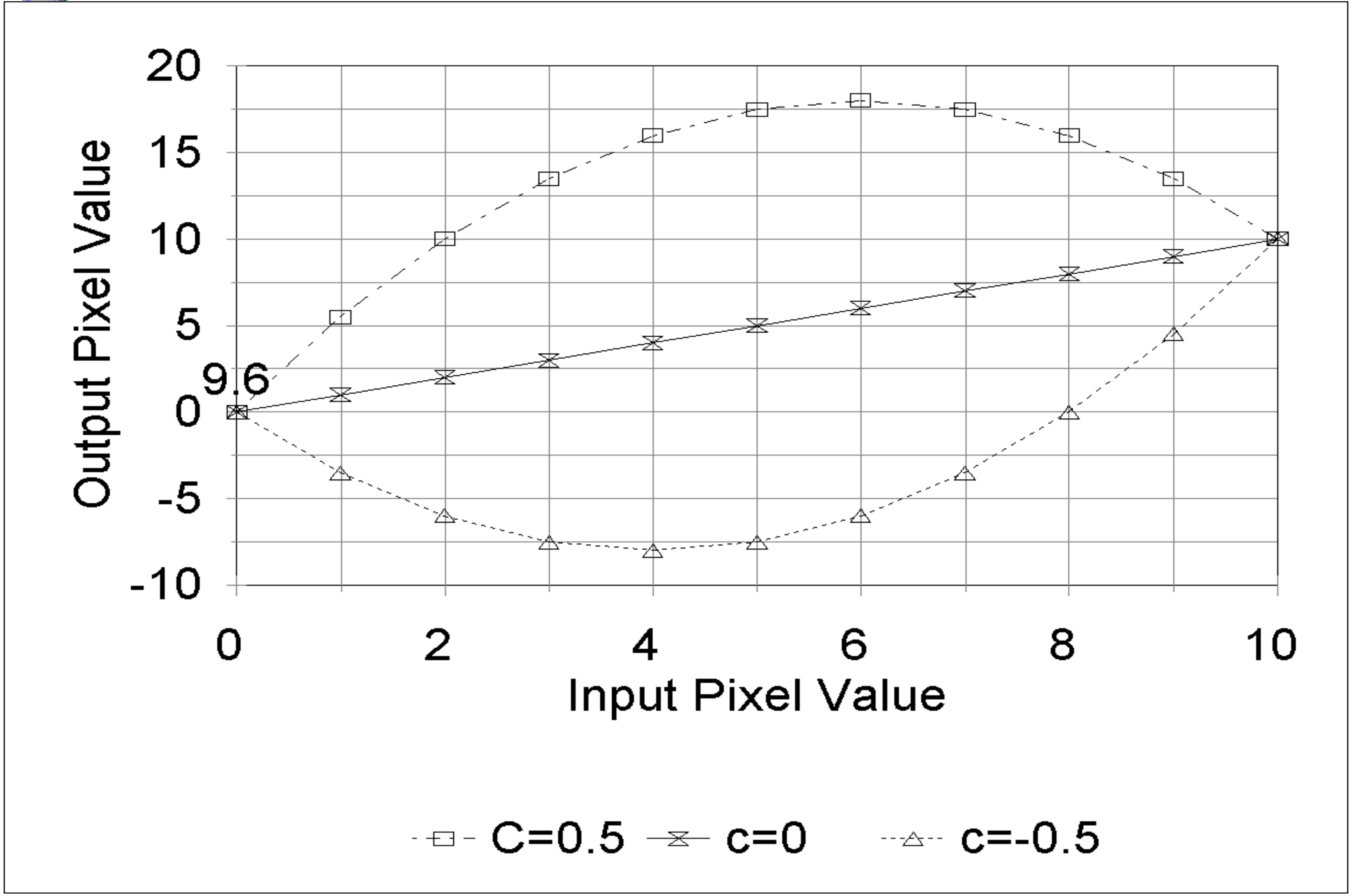
$$f\left(I\left(x,y\right)\right)=I\left(x,y\right)+C*I\left(x,y\right)*\left(I\left(x,y\right)_m-I\left(x,y\right)\right)$$

- **C>0**，增强中间部分亮度
- **C<0**，减小中间部分亮度

图像运算——点运算



图像运算——点运算



20
15
10
5
9.6
0
-5
-10
0
2
4
6
8
10
Output Pixel Value
Input Pixel Value
C=0.5
c=0
c=-0.5

图像运算——点运算

- 映射表点运算

输入像素值	0	1	2	3	4	5	6	7
输出像素值	4	6	7	3	2	0	1	5

图像运算——点运算

- 点运算特点
  - 点运算针对图像中的每一个像素灰度，独立地进行灰度值的改变
  - 输出图像中每个像素点的灰度值，仅取决于相应输入像素点的值
  - 点运算不改变图像内的空间关系
  - 从像素到像素的操作
  - 点运算可完全由灰度变换函数或灰度映射表确定
- 实例——“对比度增强、对比度拉伸、灰度变换”

图像运算—代数运算

- 加法运算的定义

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

- 主要应用举例
  - 去除“叠加性”噪音
  - 生成图像叠加效果

图像运算——代数运算

- 去除“叠加性”噪音

对于原图像f(x, y)，有一个噪音图像集

$$\{ g_i(x,y) \} \quad i = 1,2,\ldots M$$

其中：$g_i(x,y) = f(x,y) + h(x,y)_i$

M个图像的均值定义为:

$$g(x,y) = (g_0(x,y)+g_1(x,y)+\ldots+g_M(x,y))/M$$

当：噪音$h(x,y)_i$为互不相关，且均值为0时，

上述图像均值将降低噪音的影响。

定理：对M幅加性噪声图像进行平均，可以使图像的平方信噪比提高M倍。

信噪比 = 信号功率/噪声功率

图像运算——代数运算

- 生成图像叠加效果

对于两个图像**f(x,y)**和**h(x,y)**的均值有:

$$g(x,y)=\frac{1}{2}f(x,y)+\frac{1}{2}h(x,y)$$

会得到二次曝光的效果。推广这个公式为:

$\mathbf{g(x,y)} = \alpha\,\mathbf{f(x,y)} + \beta\,\mathbf{h(x,y)}$ 其中 $\alpha + \beta = 1$

我们可以得到各种<u>图像合成的效果</u>,也可以用于两张图片的衔接

图像运算——代数运算

- 减法的定义

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

- 主要应用举例
  - 去除不需要的叠加性图案
  - 检测同一场景两幅图像之间的变化

图像运算——代数运算

- 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<u>去除了背景的图像</u>。

<u>电视制作的蓝屏技术就基于此</u>

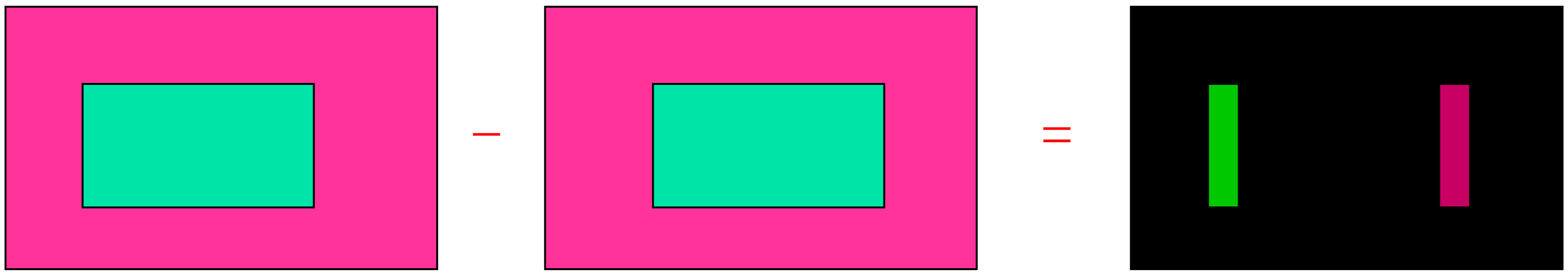
图像运算——代数运算

- 检测同一场景两幅图像之间的变化

设： 时间1的图像为$T_1(x,y)$，

时间2的图像为$T_2(x,y)$

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



图像运算——代数运算

- 乘法的定义

注意：逐点相乘

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

- 主要应用举例
  - 图像的局部显示
    - ✓用二值蒙板图像与原图像做乘法

模版中：要提取的局部区域“1”，要掩蔽的背景“0”

图像运算——逻辑运算

- 逻辑运算——求反、异或、或、与
  - 求反的定义

$$g(x,y) = R - f(x,y)$$

R为**f(x, y)**的灰度级，对于8bit灰度图，R=255。

  - 主要应用举例
    - ✓获得一个图像的负像
    - ✓获得一个子图像的补图像

图像运算——逻辑运算

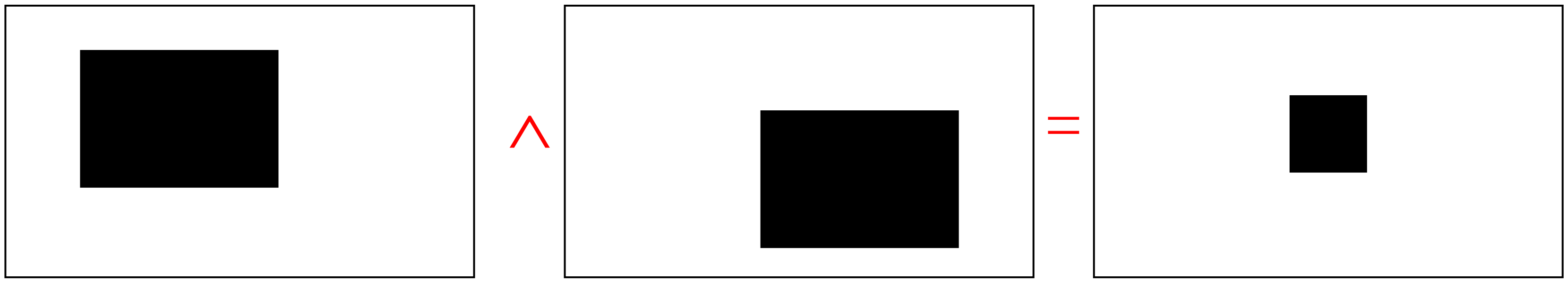
- 逻辑运算——求反、异或、或、与
  - 异或运算的定义，主要用于二值图像

$g(x,y) = f(x,y) \oplus h(x,y)$

  - 主要应用举例
    - ✓<u>获得不相交子图像</u>

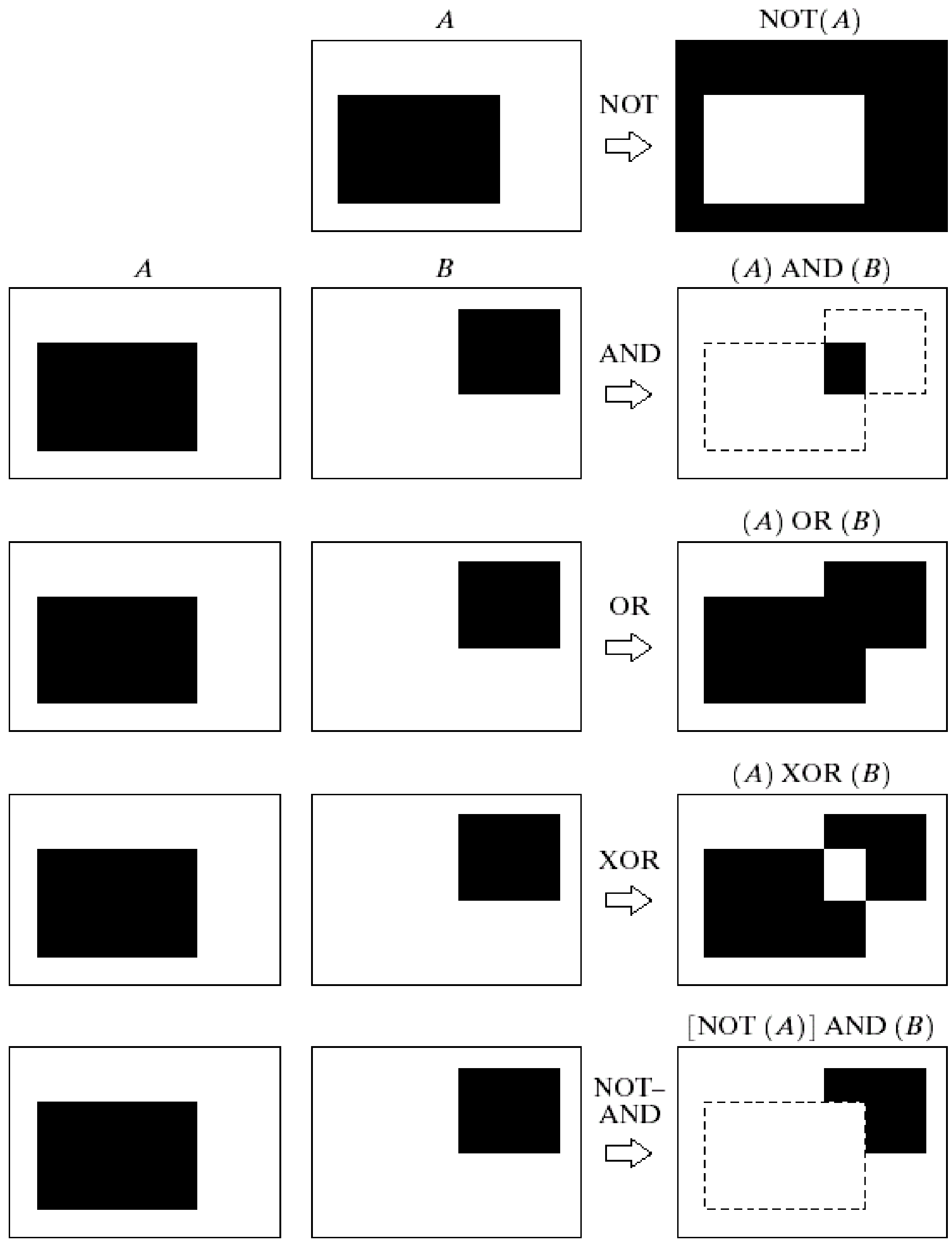
图像运算——逻辑运算

- 逻辑运算——求反、异或、或、与
- 与运算的定义
  - **$g(x,y) = f(x,y) \wedge h(x,y)$**
- 主要应用举例
  - 求两个子图像的相交子图



图像运

- 二值图像的基本逻辑运算



注意与和异或

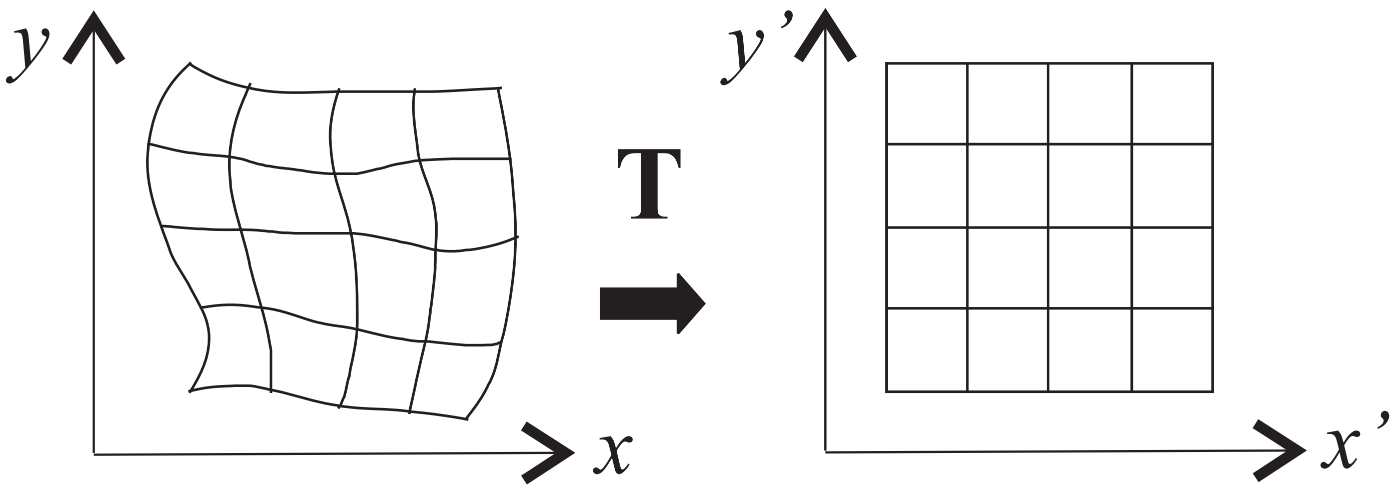
图像的空域变换

- 在图像空间，对图像的形状、像素值等进行变化、映射等处理
  - 几何变换
    - 改变图像的形状
      - 基本变换
      - 灰度插值
  - 非几何变换
    - 改变图像像素值
      - 模板运算
      - 灰度变换
      - 直方图变换

Geometrical Transformation

Transformation of spatial coordinates

基于几何变换的图像校正



Geometrical Transformation — What for?

- intentional image transformations (you know where to go)
  - resizing
  - rotation
  - shift
  - warping, texture mapping

将一幅图像映射到任意形状的物体表面，作为图像渲染

- correction of distortions (you know how it should look)
  - projective skew
  - non-linear distortion (fish-eyes)

Techniques shared by Image processing, Computer graphics, even Robotics or Mechanics.

Example of texture mapping

Realization — Rotation and shift

$$
\begin{aligned}
x' &= \cos(\alpha)x + \sin(\alpha)y + t_x \\
y' &= -\sin(\alpha)x + \cos(\alpha)y + t_y
\end{aligned}
$$

$\boxed{\text{顺时针旋转+平移}}$

More elegant and efficient

$$
\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(\alpha) & \sin(\alpha) & t_x \\ -\sin(\alpha) & \cos(\alpha) & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}
$$

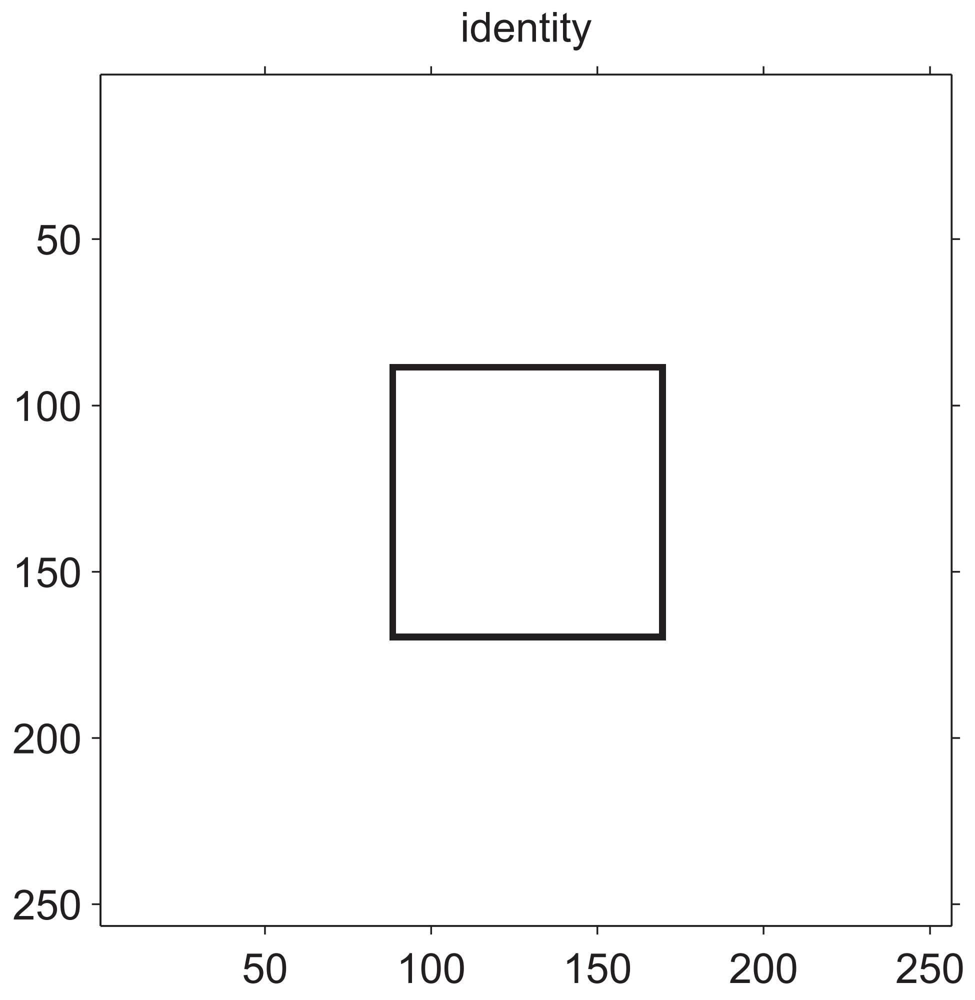
in a matrix form

$$\mathbf{x}' = T\mathbf{x}$$

where $\mathbf{x}'$ and $\mathbf{x}$ are homogeneous coordinates: $\mathbf{x} = [\lambda x, \lambda y, \lambda]^T$, $\lambda \neq 0$.

Identity

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

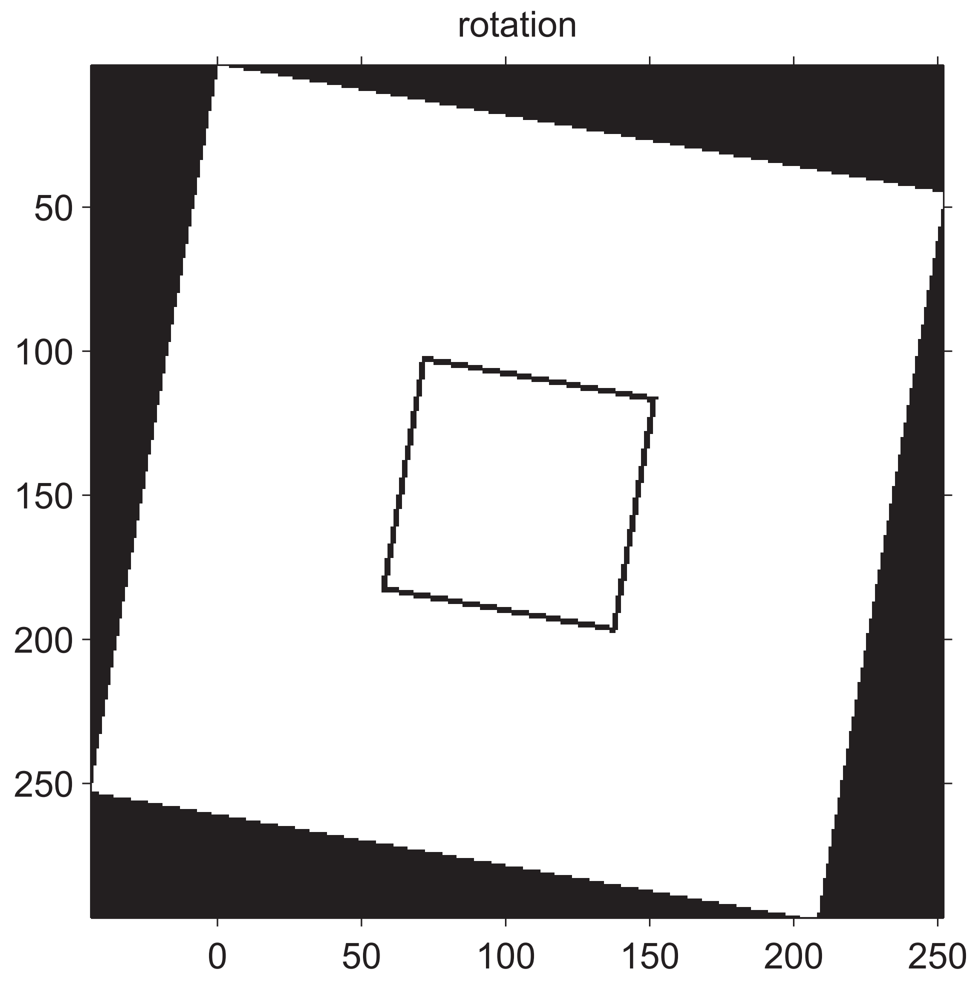


Rotation

$$T = \begin{bmatrix} \cos(\alpha) & \sin(\alpha) & 0 \\ -\sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

for $\alpha = 10°$

$$T = \begin{bmatrix} 0.9848 & 0.1736 & 0 \\ -0.1736 & 0.9848 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

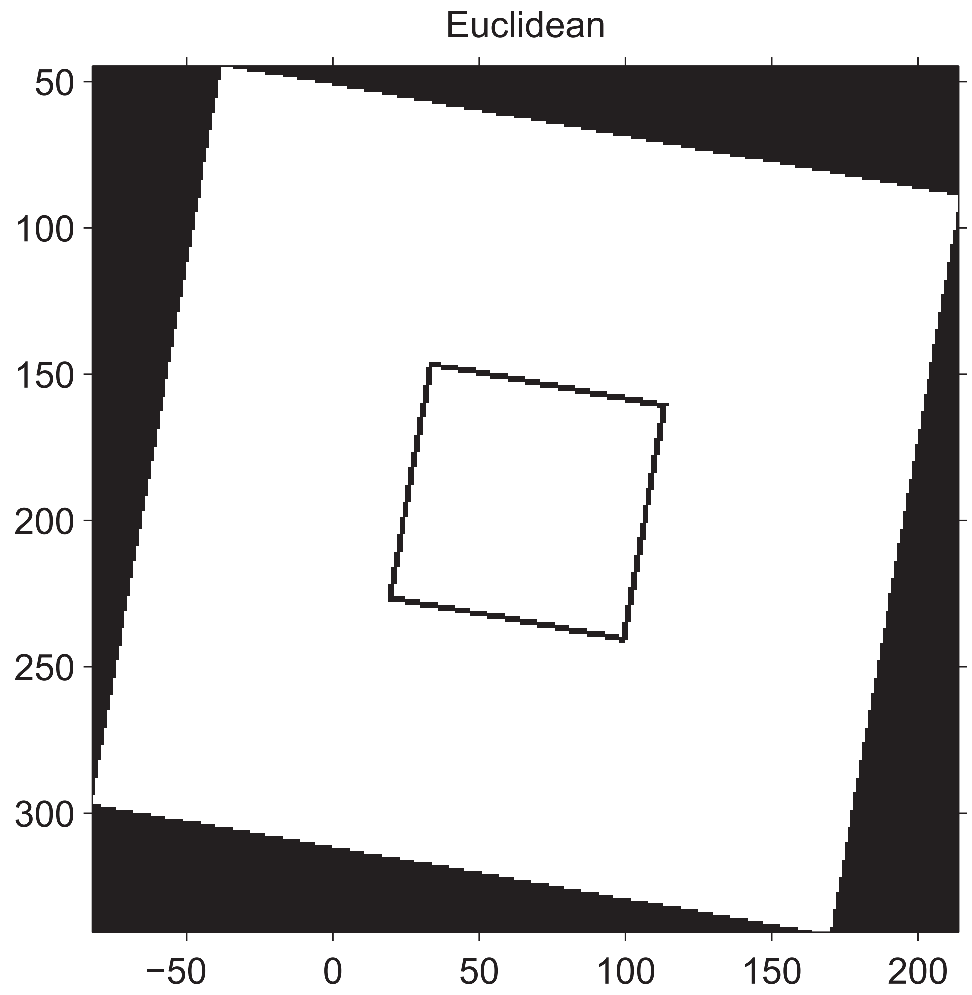


Rotation + translation

$$T = \begin{bmatrix} \cos(\alpha) & \sin(\alpha) & t_x \\ -\sin(\alpha) & \cos(\alpha) & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

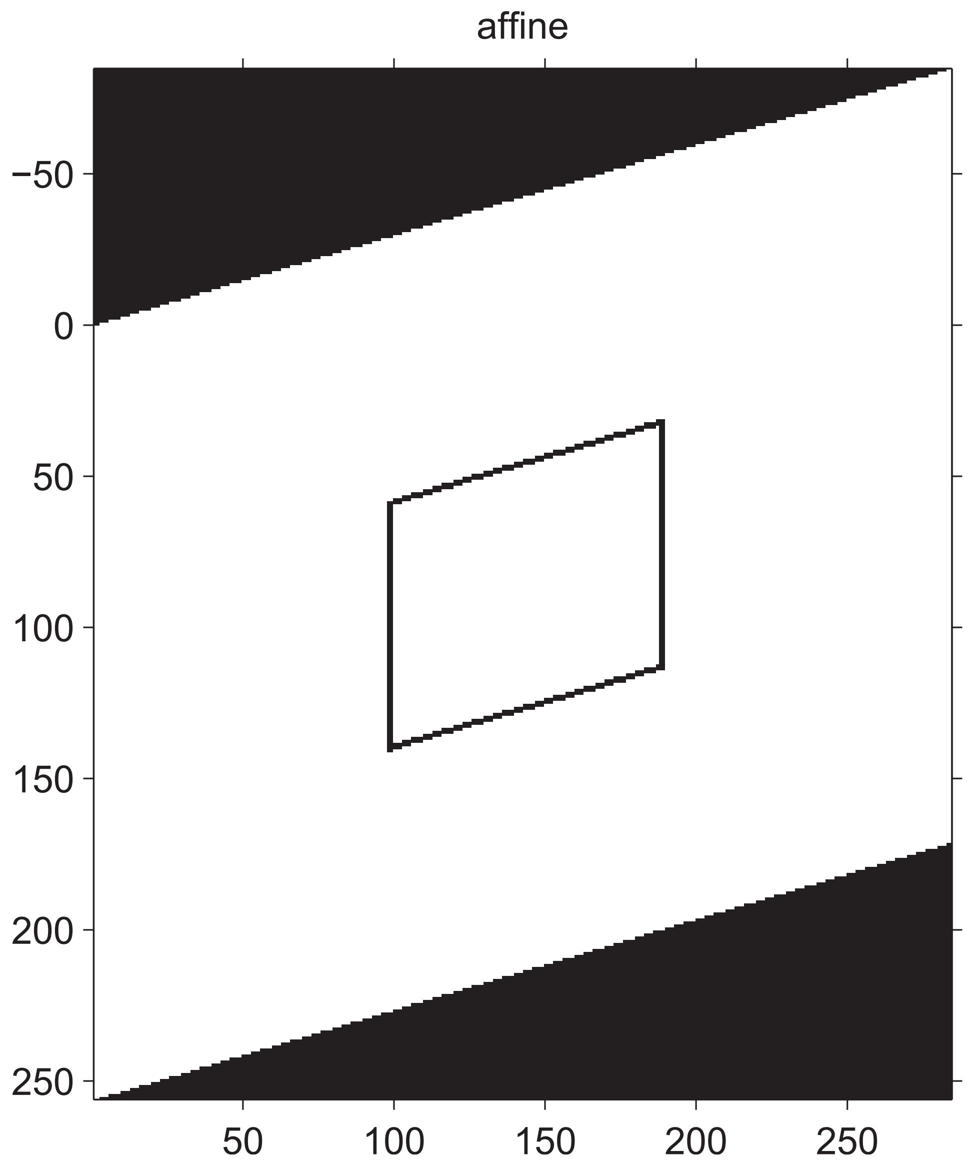
for $\alpha = 10^\circ$ and $\mathbf{t} = [-50, 30]^T$

$$T = \begin{bmatrix} 0.9848 & 0.1736 & -50 \\ -0.1736 & 0.9848 & 30 \\ 0 & 0 & 1 \end{bmatrix}$$



Affine （仿射变换）

$$T = \begin{bmatrix} 0.9000 & 0 & 0 \\ 0.3000 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



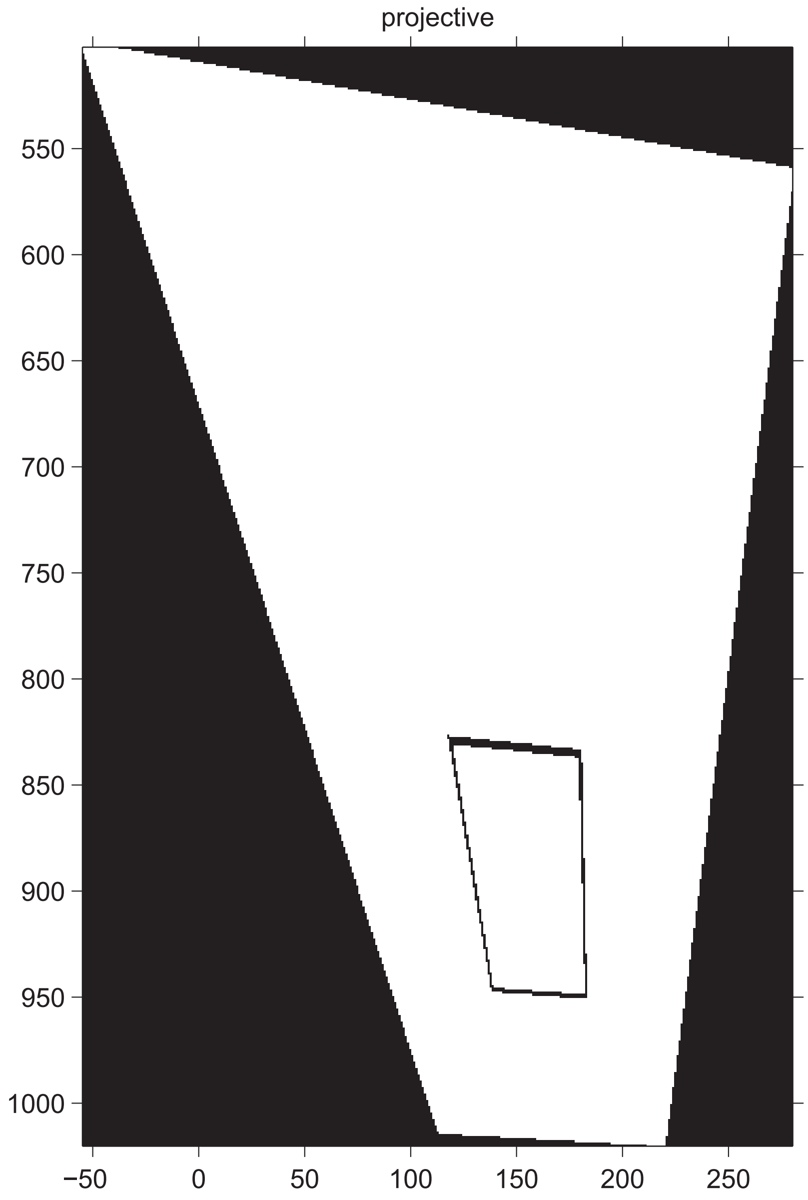
in general

$$T = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix}$$

6 degrees of freedom

Projective

$$T = \begin{bmatrix} 0.445 & -0.147 & 98.400 \\ -0.018 & 0.099 & -50.000 \\ -0.000 & -0.001 & 1 \end{bmatrix}$$



in general

$$T = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix}$$

8 degrees of freedom

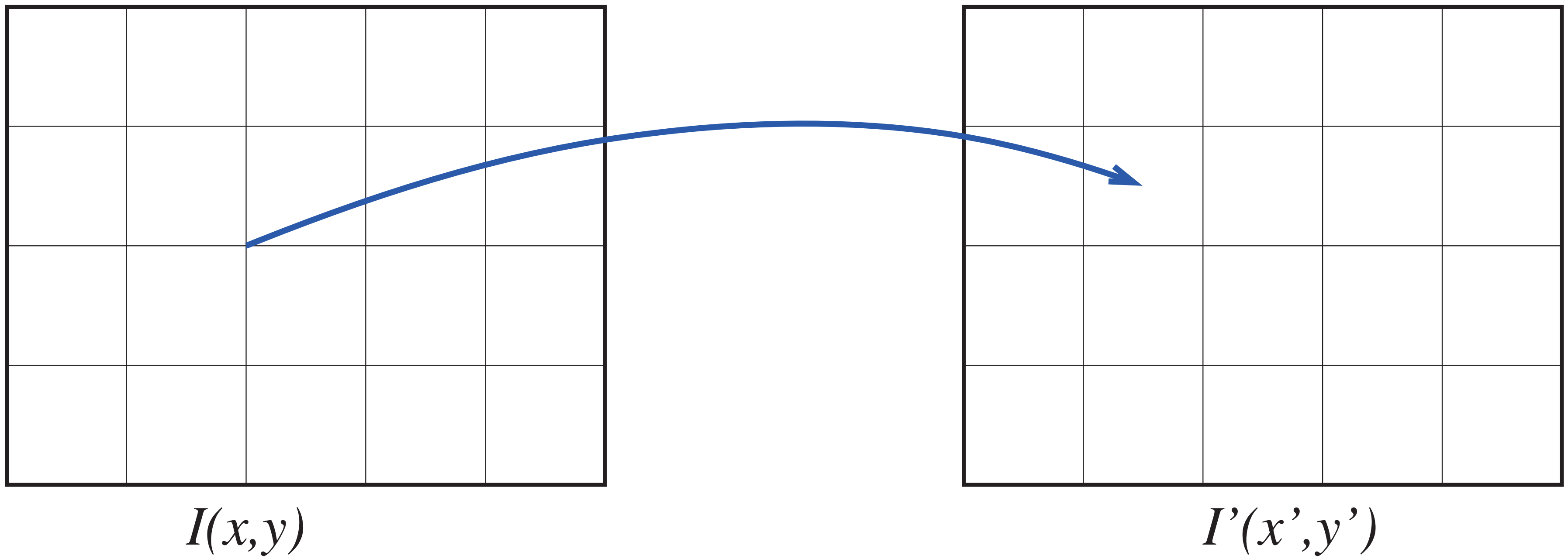
Correction of converging lines I

Correction of converging lines II

Correction of converging lines III

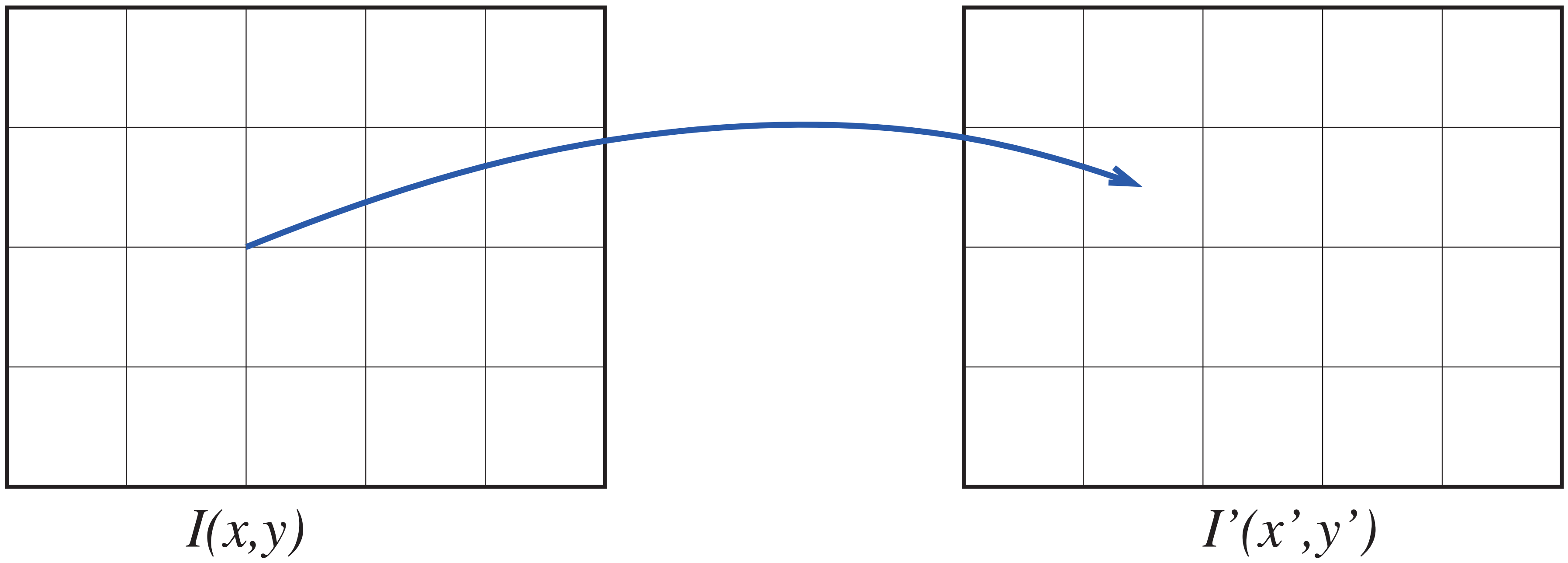
Forward mapping

$\mathbf{x}' = T\mathbf{x}$;



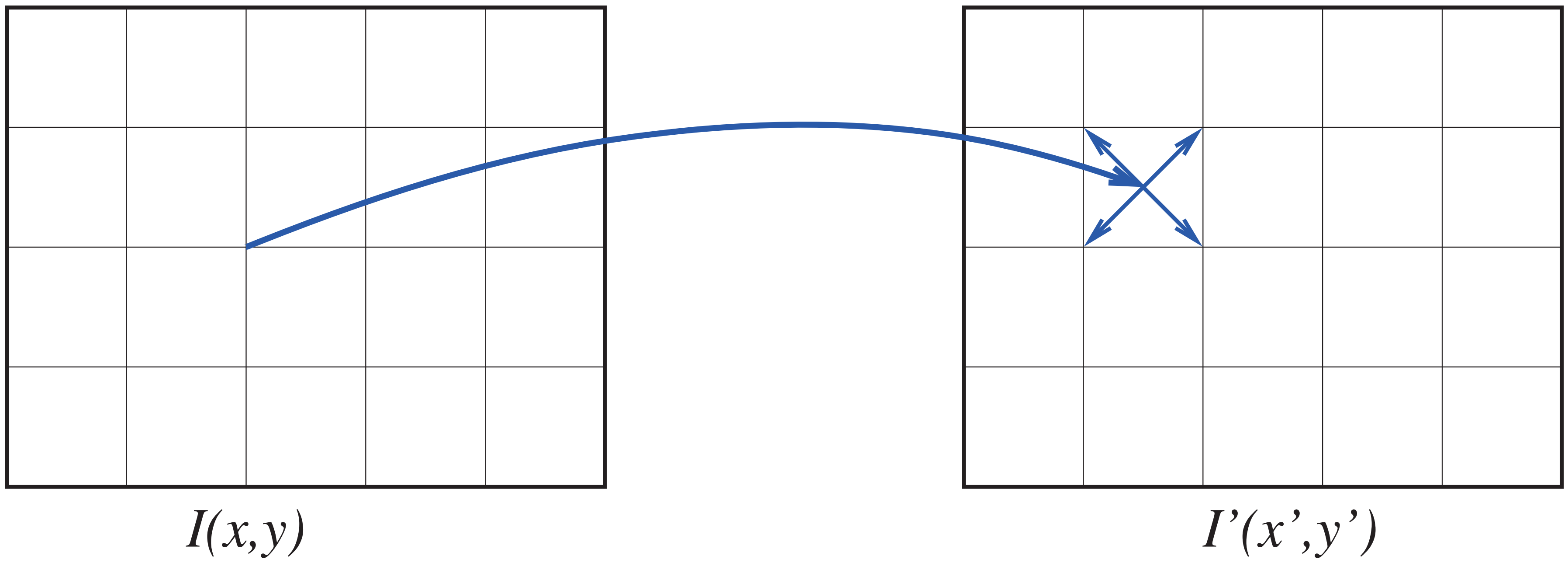
Forward mapping — problems

Maps outside the pixel locations. **(not in the display grid)**



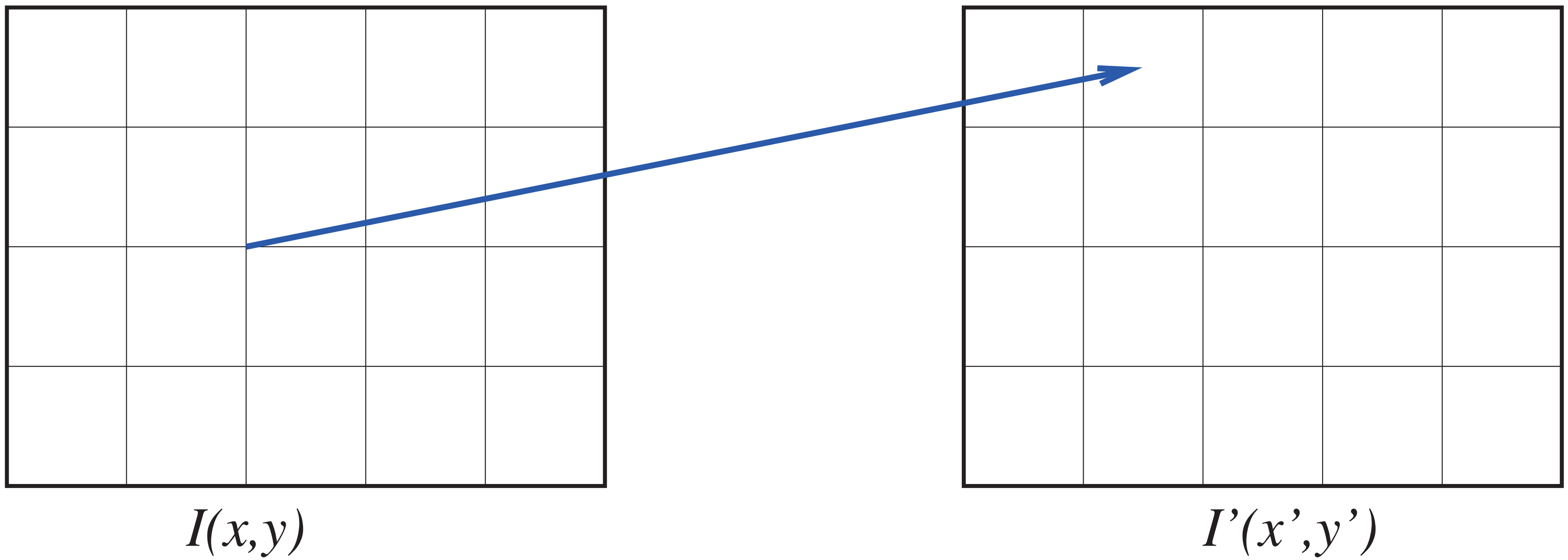
Forward mapping — problems

Solution: Spread out the effect of each pixel



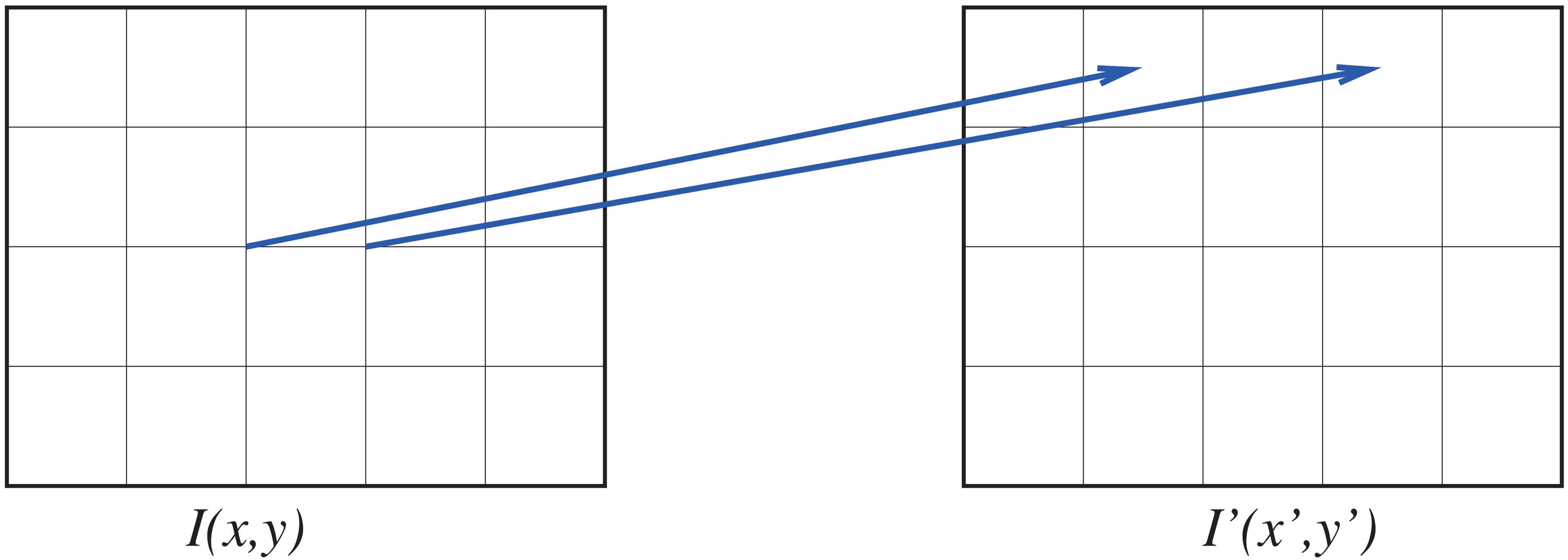
Forward mapping — problems

May produce holes in the output **（有些显示格点附近没有变换过来的像素，形成hole）**



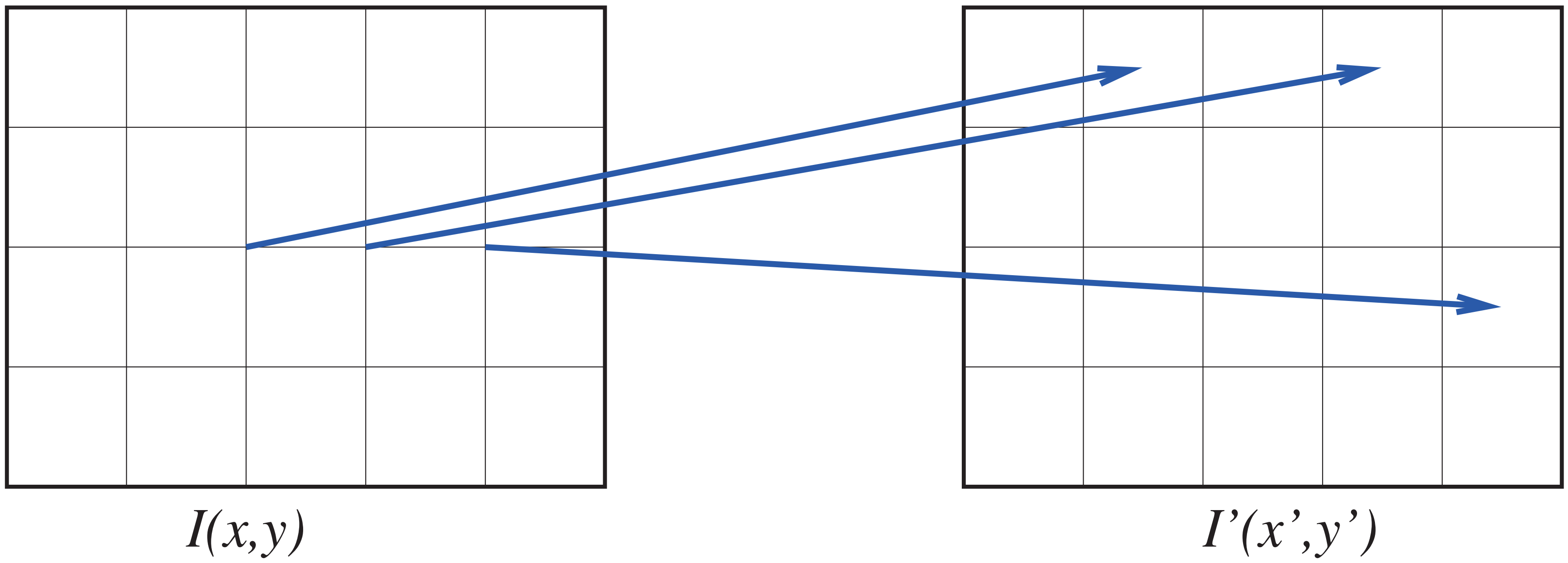
Forward mapping — problems

May produce holes in the output



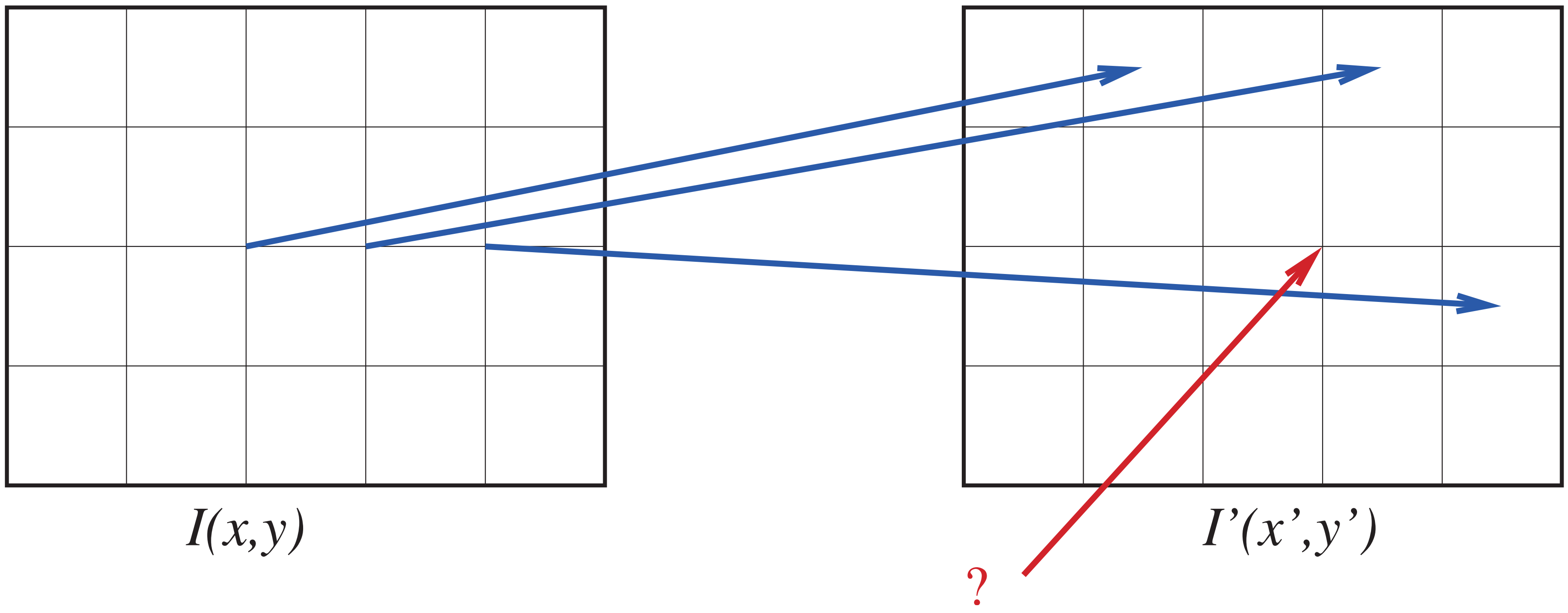
Forward mapping — problems

May produce holes in the output



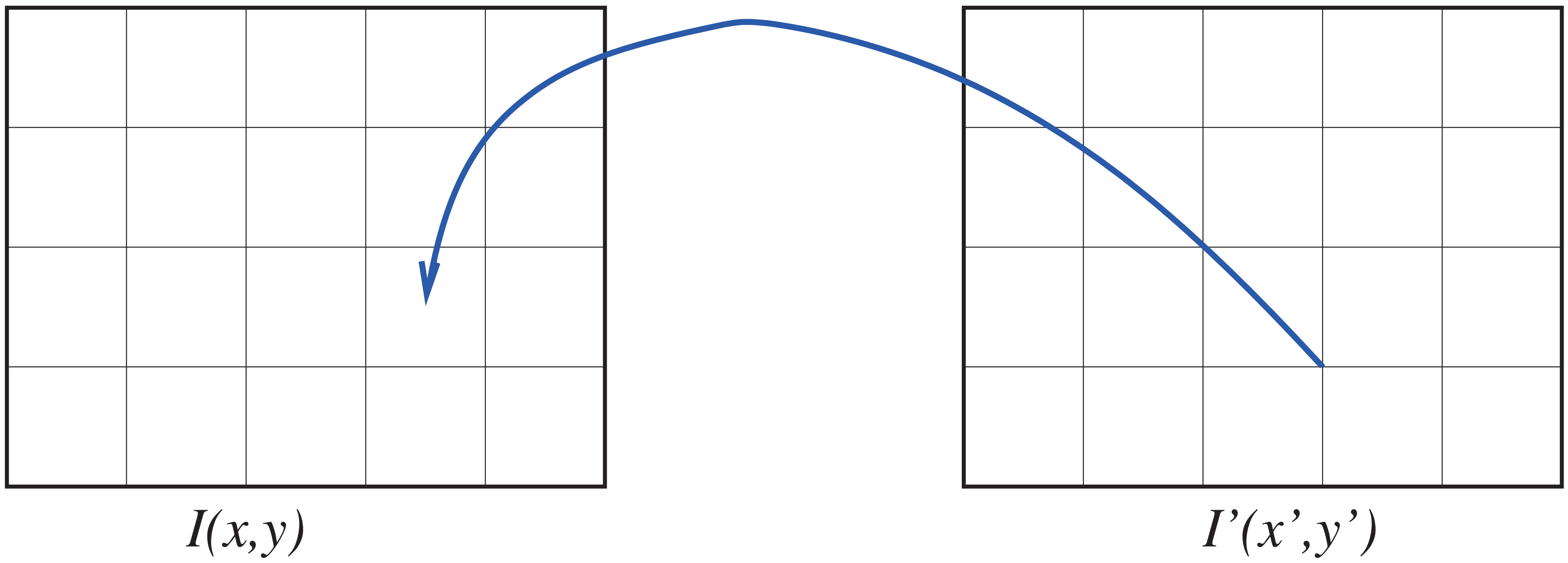
Forward mapping — problems

May produce holes in the output



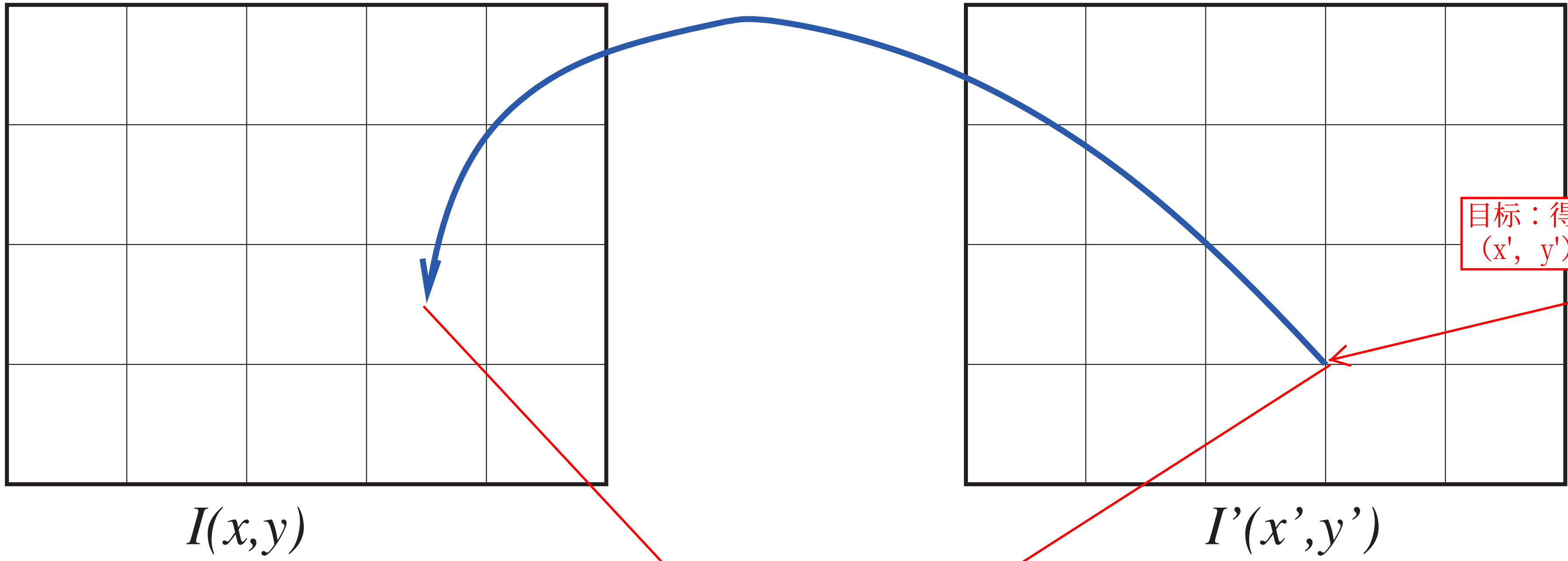
Backward mapping

$\mathbf{x} = T^{-1}\mathbf{x}'$



Backward mapping — problems

Does not always map from a pixel

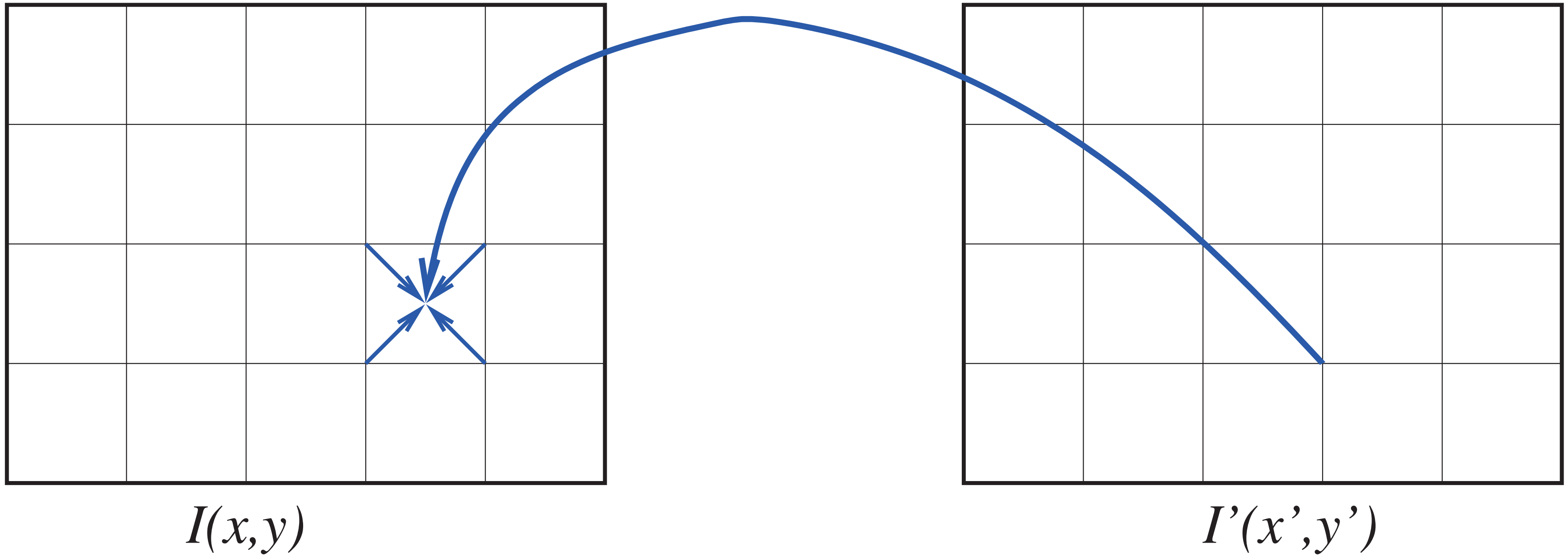


目标：得到（x'，y'）的灰度值

这2点的灰度值相同，原坐标的显示格点上灰度值已知，利用插值

Backward mapping — problems

Solution: Interpolate between pixels



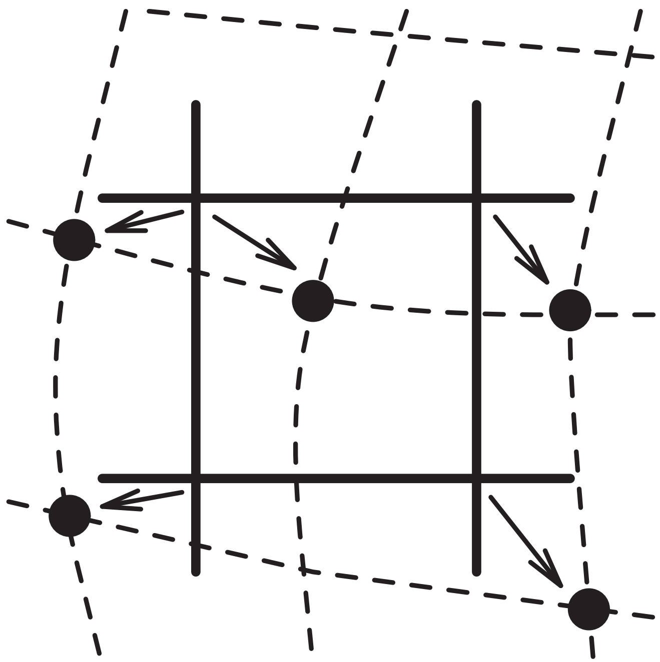
Interpolation as 2D convolution

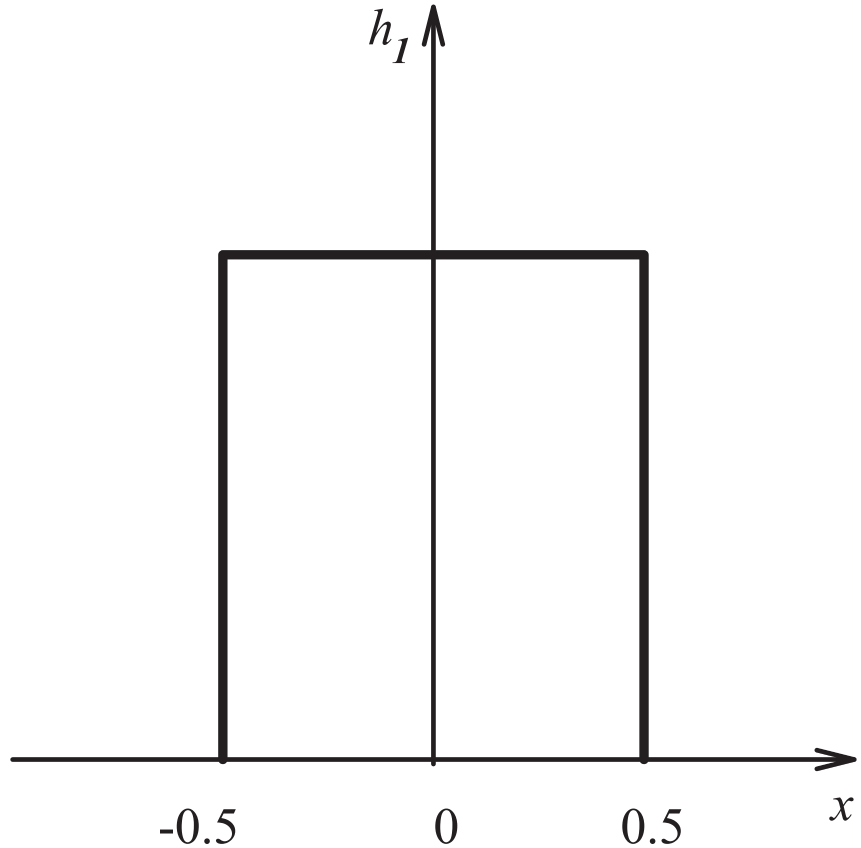
We have sampled (possibly outside the regular grid) function $s(x, y)$ instead of the wanted $f(x, y)$.

We want to find a good approximation by convolution

$$\hat{f}(x, y) = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} s(k, l) h(x - k, y - l)$$

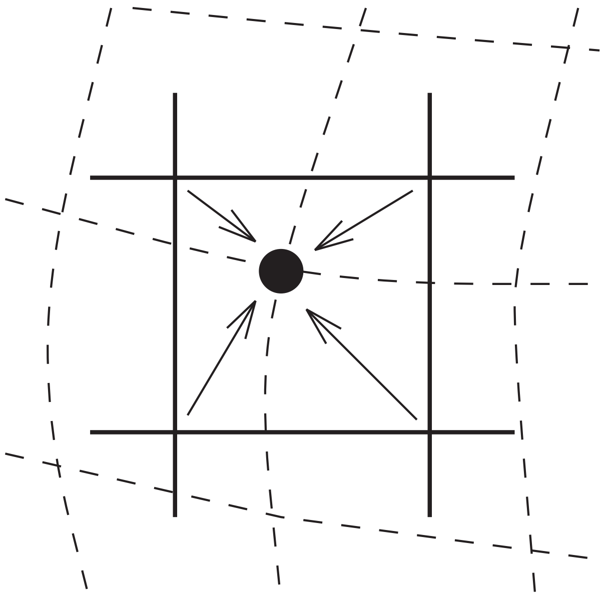
Interpolation — nearest neighbour

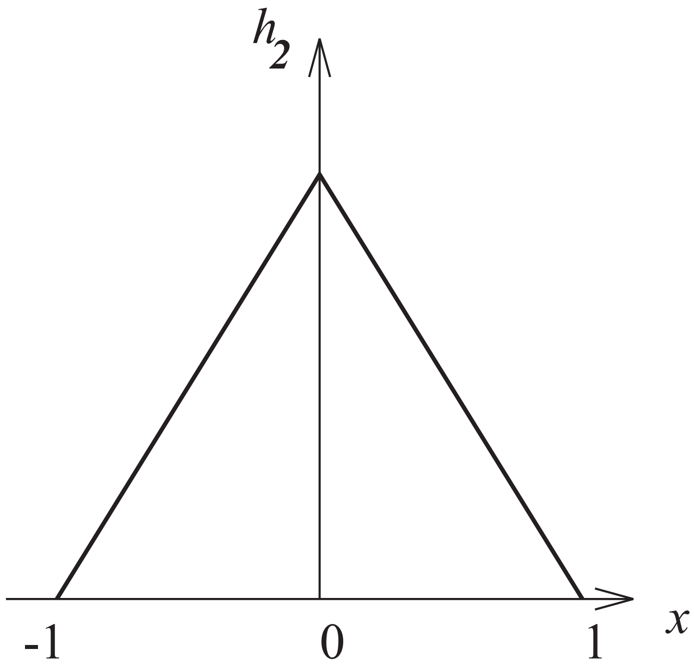




Interpolation — bilinear

$$
\begin{aligned}
\hat{f}(x,y) = \quad & (1-a)(1-b)\,s(l,k) + a(1-b)\,s(l+1,k) \\
& + b(1-a)\,s(l,k+1) + ab\,s(l+1,k+1)\,, \\
\text{where} \quad & l = round\,(x)\,, \quad a = x - l\,, \\
& k = round\,(y)\,, \quad b = y - k\,.
\end{aligned}
$$

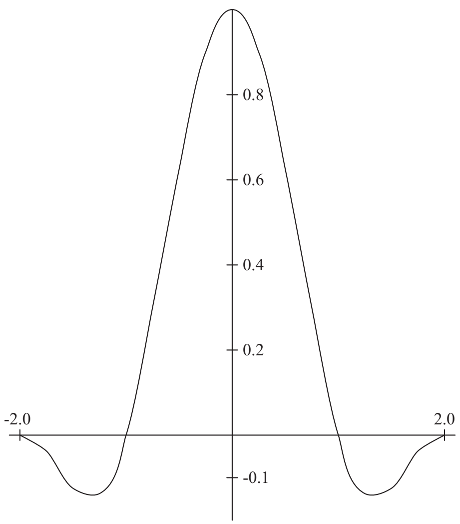




Interpolation — bicubic

Just 1D, for clarity

$$\hat{f} = \begin{cases} 1 - 2|x|^2 + |x|^3 & \text{pro } 0 \le |x| < 1\,, \\ 4 - 8|x| + 5|x|^2 - |x|^3 & \text{pro } 1 \le |x| < 2\,, \\ 0 & \text{elsewhere.} \end{cases}$$

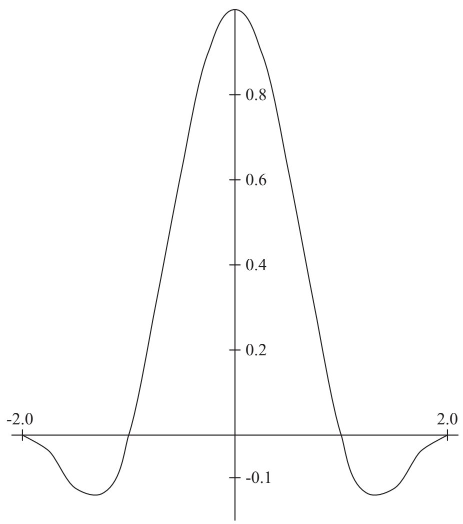


Does not remind you the shape something?

Interpolation — bicubic

Just 1D, for clarity

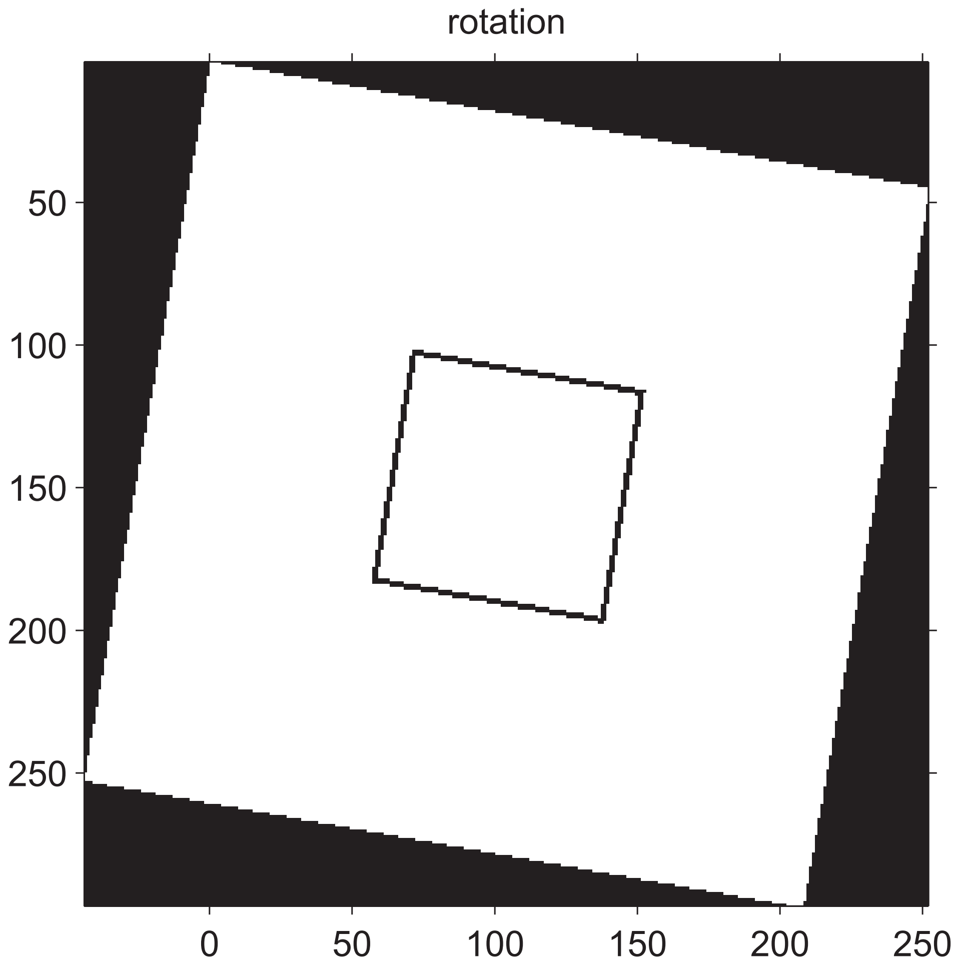
$$\hat{f} = \begin{cases} 1 - 2|x|^2 + |x|^3 & \text{pro } 0 \le |x| < 1\,, \\ 4 - 8|x| + 5|x|^2 - |x|^3 & \text{pro } 1 \le |x| < 2\,, \\ 0 & \text{elsewhere.} \end{cases}$$



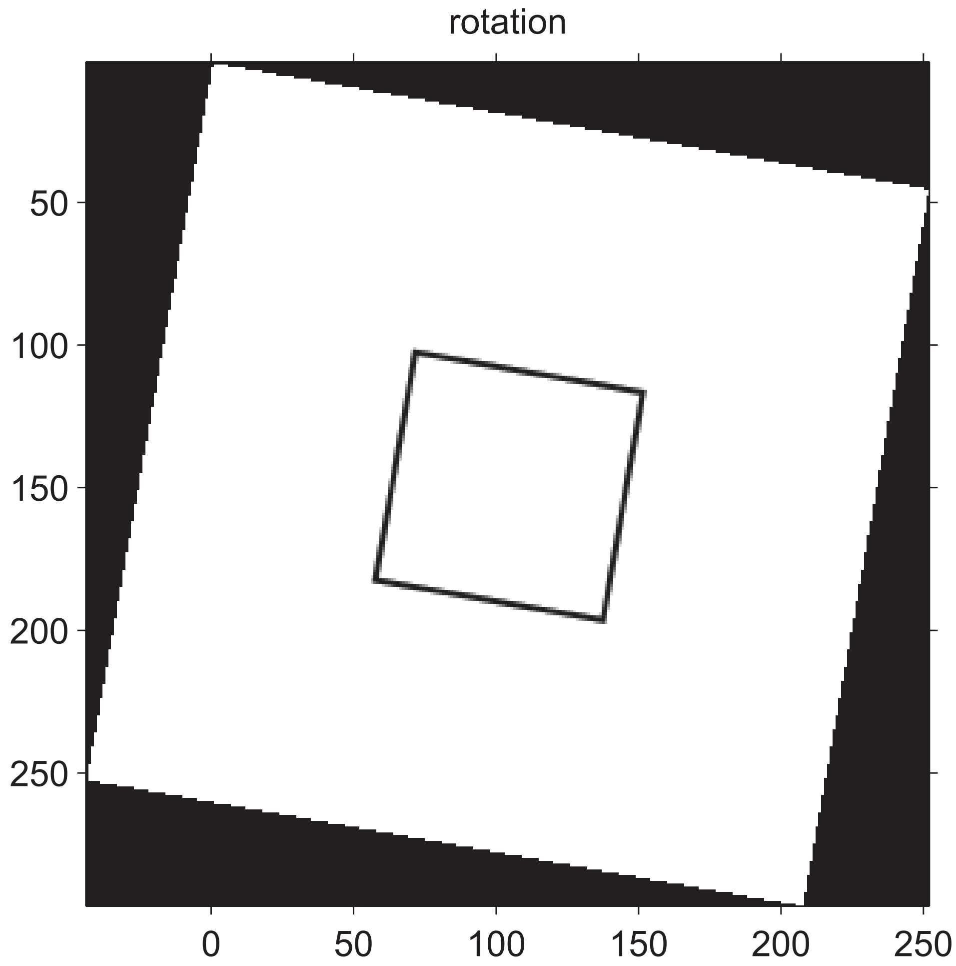
Does not remind you the shape something?

$\mathrm{sinc}(x)$! The ideal reconstructor. 用到无穷多个采样点，无失真恢复

Interpolation — nearest vs. bilinear

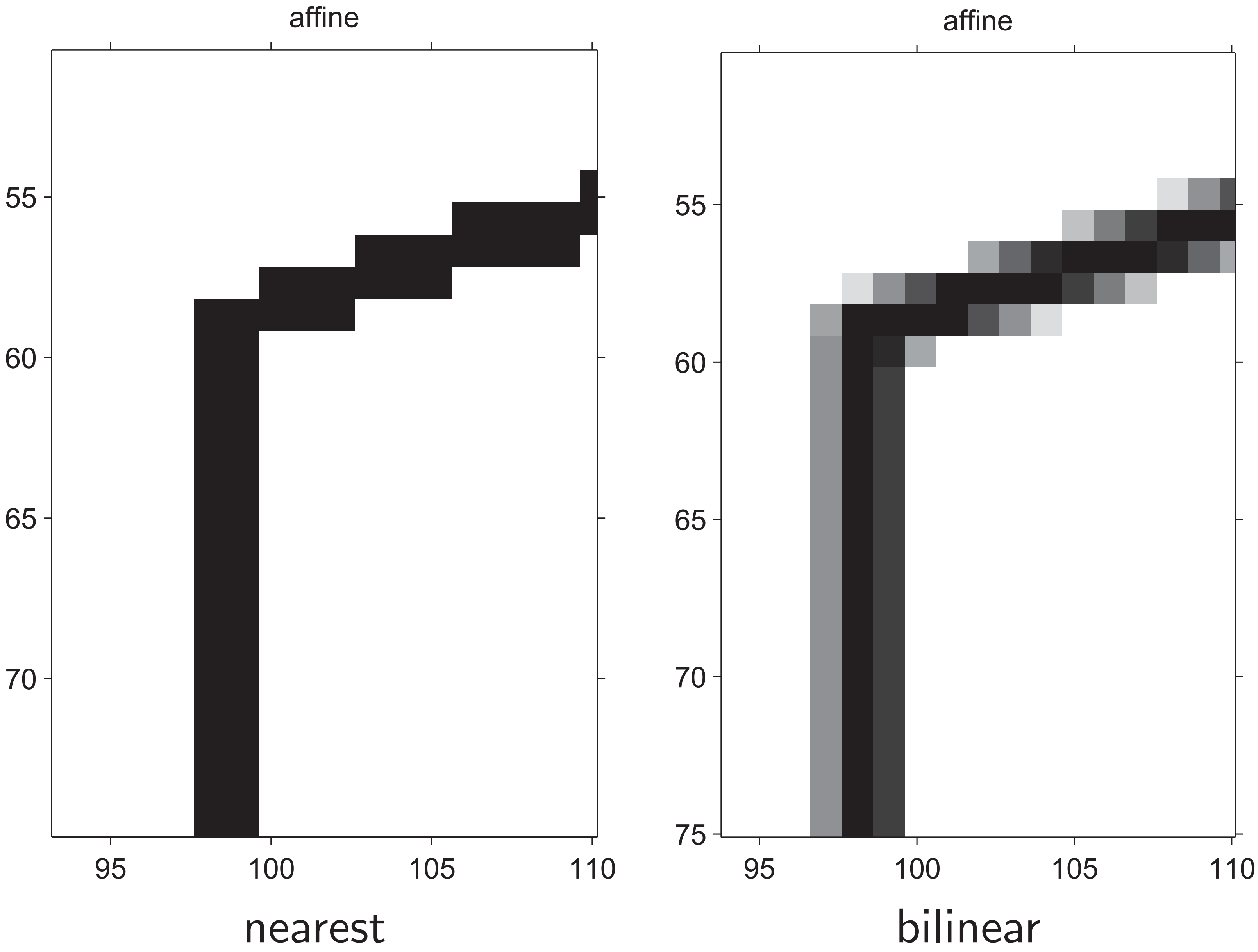


nearest



bilinear

Interpolation — nearest vs. bilinear — close up



Geometrical Transformation — Summary

- Closed form solution, all pixels undergo the same transformation
  - rotation, scaling, translation
  - affine, perspective
- General, deformation meshes. Transformation depends on position (warping, morphing)

图像的空域变换——几何变换

- 几何变换的基本概念
  - 对原始图像，按照需要改变其大小、形状和位置的变化
  - 变换的类型：二维平面图像的几何变换、三维图像的几何变换、由三维向二维平面的投影变换等
- 二维图像几何变换的定义

对于原始图像**f(x,y)**，坐标变换函数

$$x' = a(x,y); \qquad y' = b(x,y)$$

唯一确定了几何变换：

关键点：灰度值不变

$$g(x',y') = f(x, y)$$ **g(x,y)**是目标图像

- 二维图像几何变换的基本方式
  - 多项式变换、透视变换

图像的空域变换——几何变换

- 多项式变换
  - 基本公式

注：(x_i, y_i)是(x,y)的邻域像素坐标

$$\begin{cases} x' = \sum_{i=0}^{M} \sum_{j=0}^{N} a_{ij} x_i y_j \\ y' = \sum_{i=0}^{M} \sum_{j=0}^{N} b_{ij} x_i y_j \end{cases}$$

  - 线性变换——多项式变换中的一阶变换 **可以通过矩阵变换实现**

$$x' = ax + by + e, \qquad y' = cx + dy + f$$

  - 二维图像的基本变换

    由线性变换确定的图像的平移、缩放、旋转、镜像与错切

图像的空域变换——几何变换

- 二维数字图像基本几何变换的矩阵计算
  - 原始图像与目标图像之间的坐标变换函数为线性函数
    - ✓ 可以通过与之对应的线性矩阵变换来实现
  - 齐次坐标表示法——用**n + 1**维向量表示**n**维向量

设有变换矩阵***T***，则二维图像的基本几何变换矩阵为：

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = T \times \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \qquad T = \begin{bmatrix} a & b & e \\ c & d & f \\ 0 & 0 & 1 \end{bmatrix}$$

上述变换方式又称之为仿射变换

图像的空域变换—几何变换

- 数字图像基本几何变换的矩阵计算（续）
  - 二维图像的基本几何变换具有特征:
    - 变换前图形上的每一点，在变换后的图形上都有一确定的对应点，如原来直线上的中点变换为新直线的中点
    - 平行直线变换后仍保持平行，相交直线变换后仍相交
    - 变换前直线上的线段比等于变换后对应的线段比

图像的空域变换——几何变换

- 数字图像基本几何变换的矩阵计算（续）
  - 变换矩阵***T***可以分解为二个子矩阵:

$$T = \begin{bmatrix} a & b & e \\ c & d & f \\ 0 & 0 & 1 \end{bmatrix}$$

  - 子矩阵**1**:

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}_{2\times 2}$$

  - 可实现图像的恒等、比例、镜像、旋转和错切变换
  - 子矩阵**2**： $[e\ \ f]^T$

可实现图像的平移变换（**e=0**，**f=0**时无平移作用）

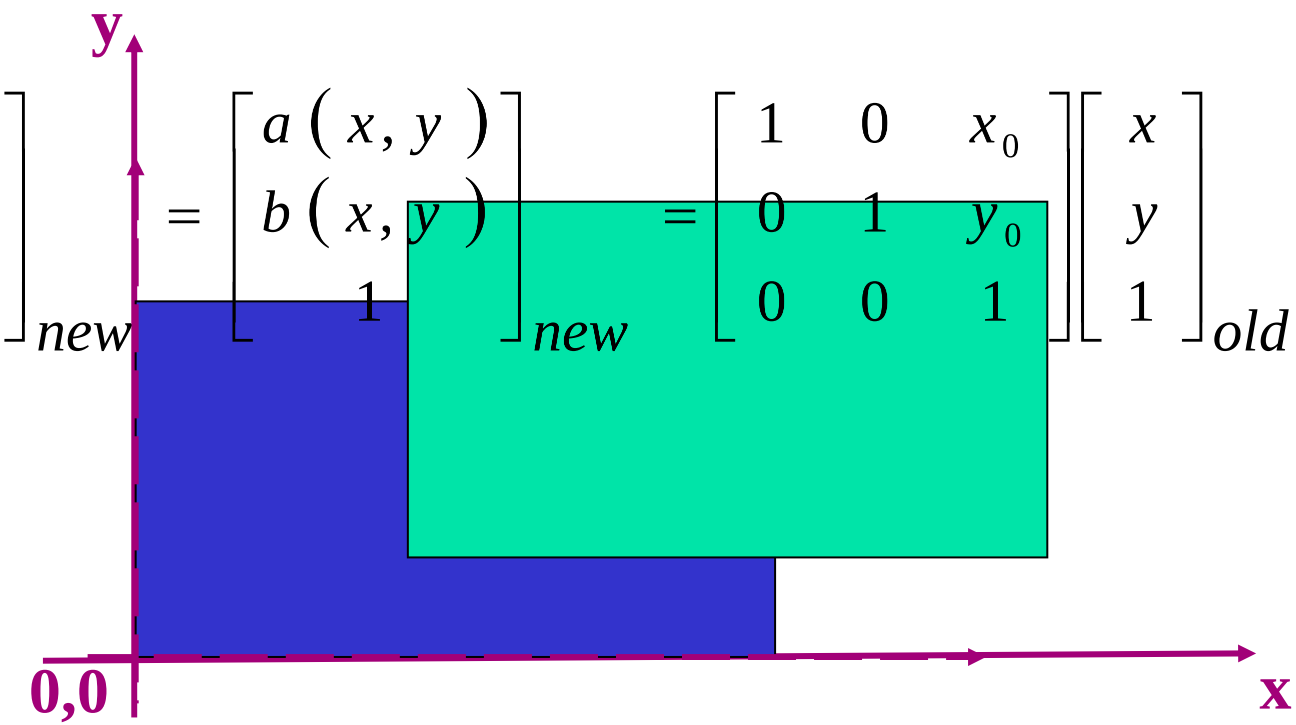
图像的空域变换——几何变换

- 平移变换（只改变图像位置，不改变图像的大小和形状）

  设: $a(x,y) = x + x_0$;　　$b(x,y) = y + y_0$;

  可有：$g(x', y')=g(x+x0, y+y0)=f(x，y)$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}_{new} = \begin{bmatrix} a(x,y) \\ b(x,y) \\ 1 \end{bmatrix}_{new} = \begin{bmatrix} 1 & 0 & x_0 \\ 0 & 1 & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}_{old}$$

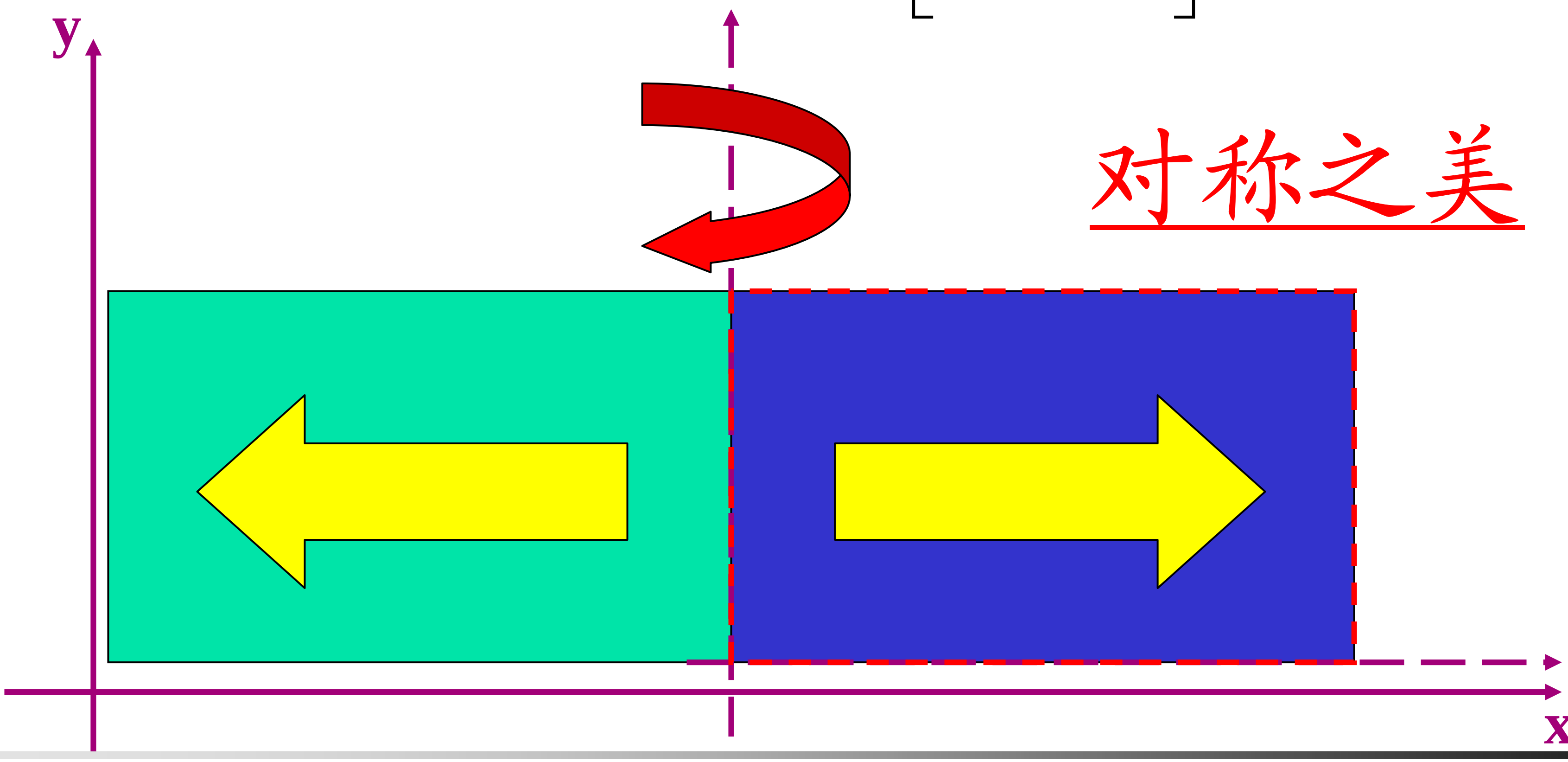


图像的空域变换——几何变换

- 水平镜像

a(x,y) = -x; b(x,y) = y;

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



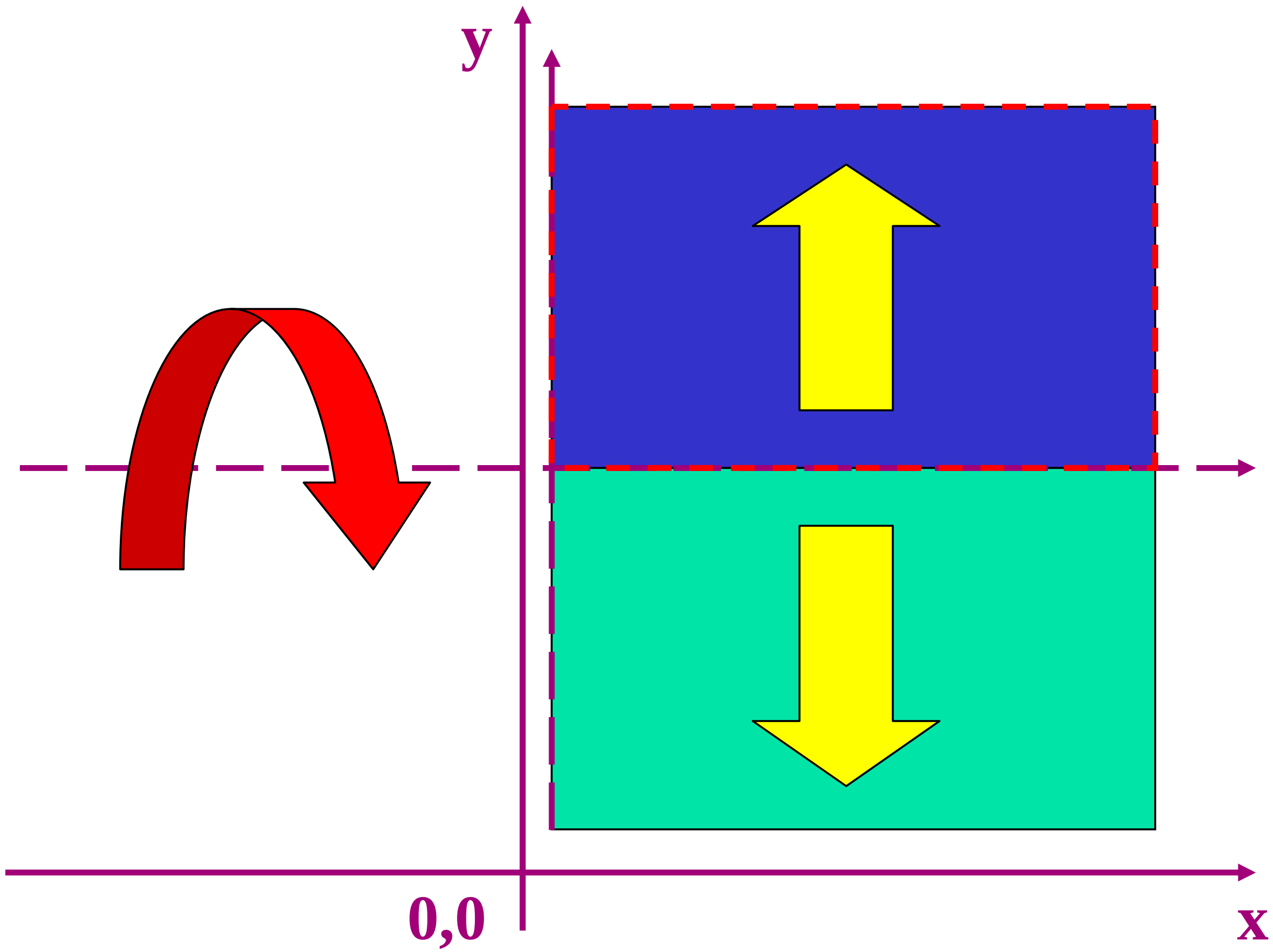
图像的空域变换——几何变换

- 垂直镜像

a(x,y) = x;

b(x,y) = -y

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



图像的空域变换——几何变换

- 缩放变换:

x方向缩放**c**倍，**y**方向缩放**d**倍

a(x,y) = x × c;　　b(x,y) = y × d;

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}_{new} = \begin{bmatrix} a(x,y) \\ b(x,y) \\ 1 \end{bmatrix}_{new} = \begin{bmatrix} \mathrm{c} & 0 & 0 \\ 0 & \mathrm{d} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}_{old}$$

c, d 相等，<u>按比例缩放</u>

c, d 不相等，不按比例缩放—<u>几何畸变</u>

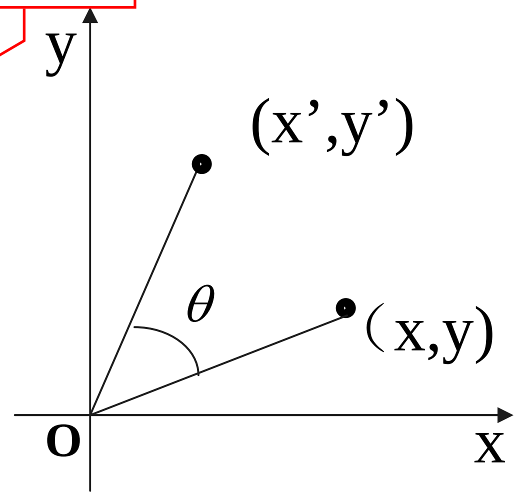
图像的空域变换——几何变换

- 旋转变换:绕原点旋转θ度

按原点逆时针旋转一个角度

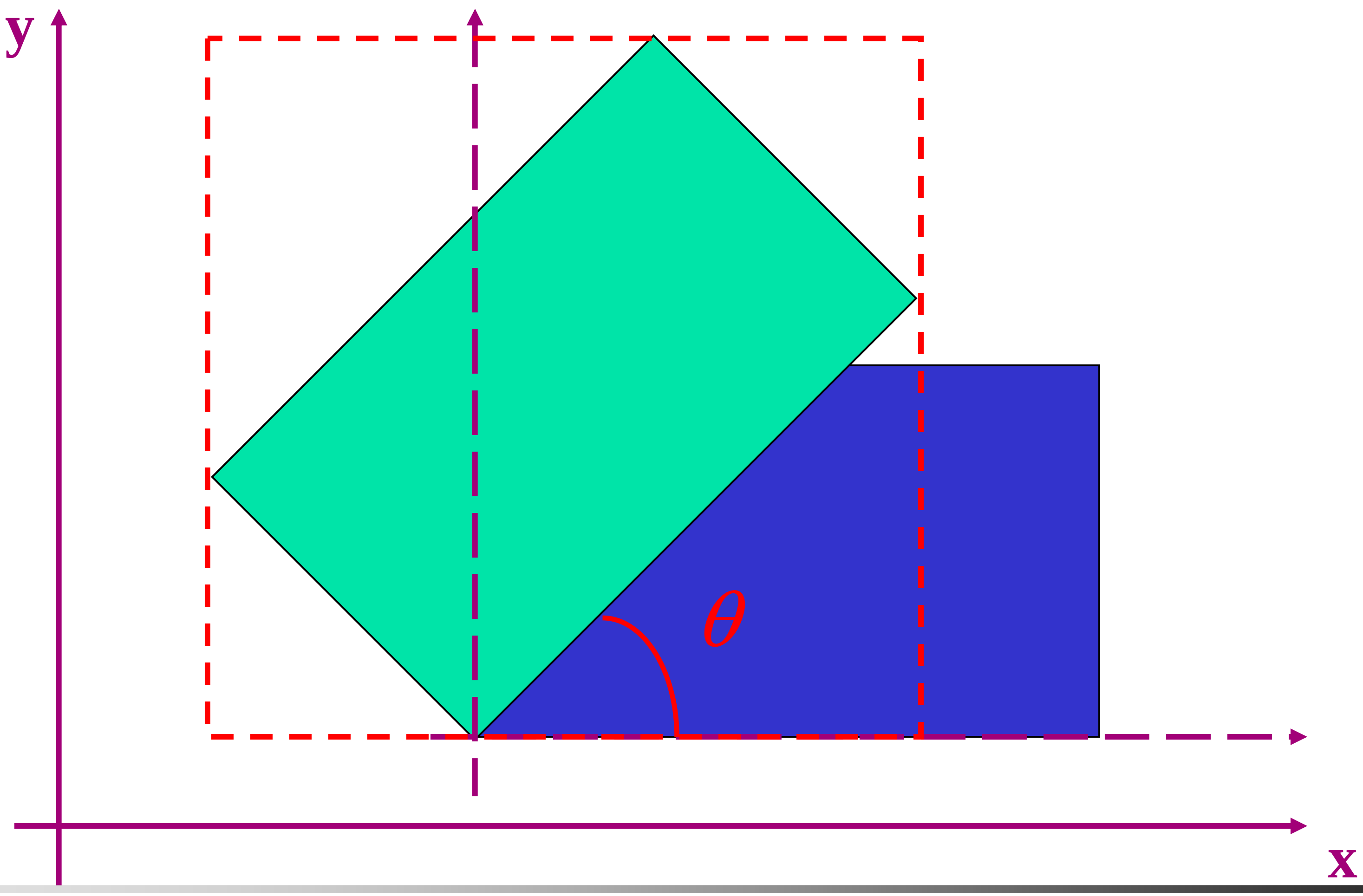
设:

$$\begin{cases} x' = x\cos\theta - y\sin\theta \\ y' = x\sin\theta + y\cos\theta \end{cases}$$



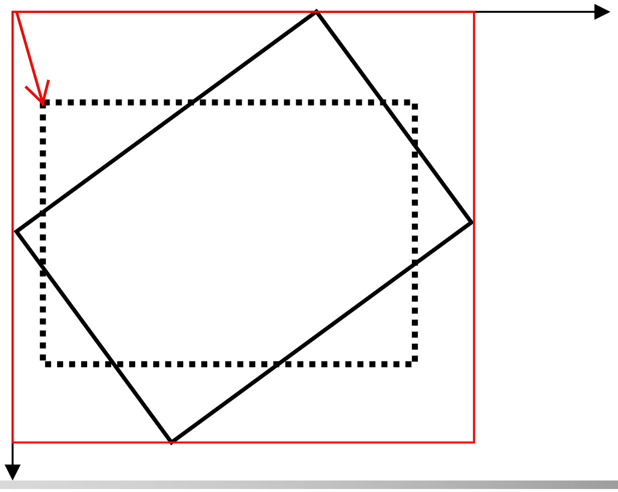
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}_{new} = \begin{bmatrix} a(x,y) \\ b(x,y) \\ 1 \end{bmatrix}_{new} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}_{old}$$

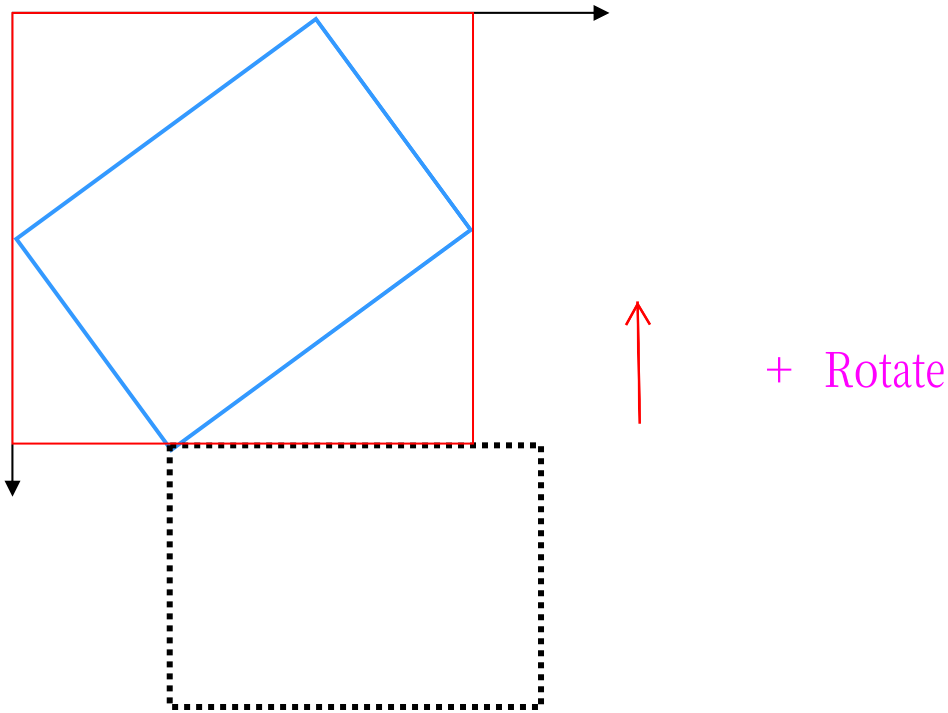
- 旋转变换:绕原点旋转θ度



- 旋转变换的注意点

1）图像旋转之前，为了避免信息的丢失，一定有平移坐标，具体的做法有如图所示的两种方法。

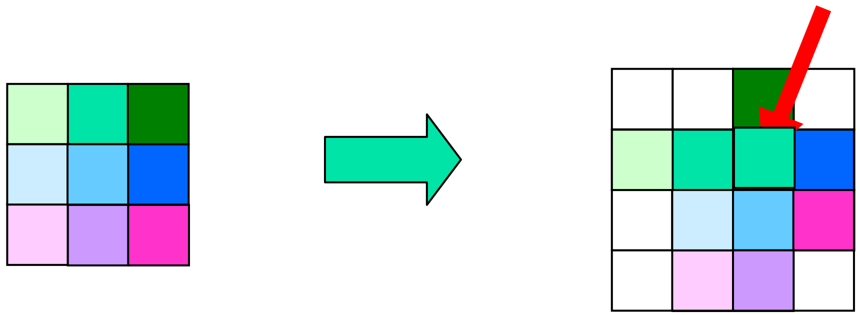




■ 旋转变换的注意点

图像的旋转注意点：

2）图像旋转之后，会出现许多的空洞点，对这些空洞点必须进行填充处理，否则画面效果不好。称这种操作为插值处理。



经过插值处理之后，图像效果就变得自然。

图像的空域变换——几何变换

- 图像的错切变换

图像的错切变换实际上是景物在平面上的非垂直投影效果。

$$\begin{cases} x' = x + d_x y \\ y' = y \end{cases} (x\text{方向的错切})$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}_{new} = \begin{bmatrix} 1 & d_x & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}_{old}$$

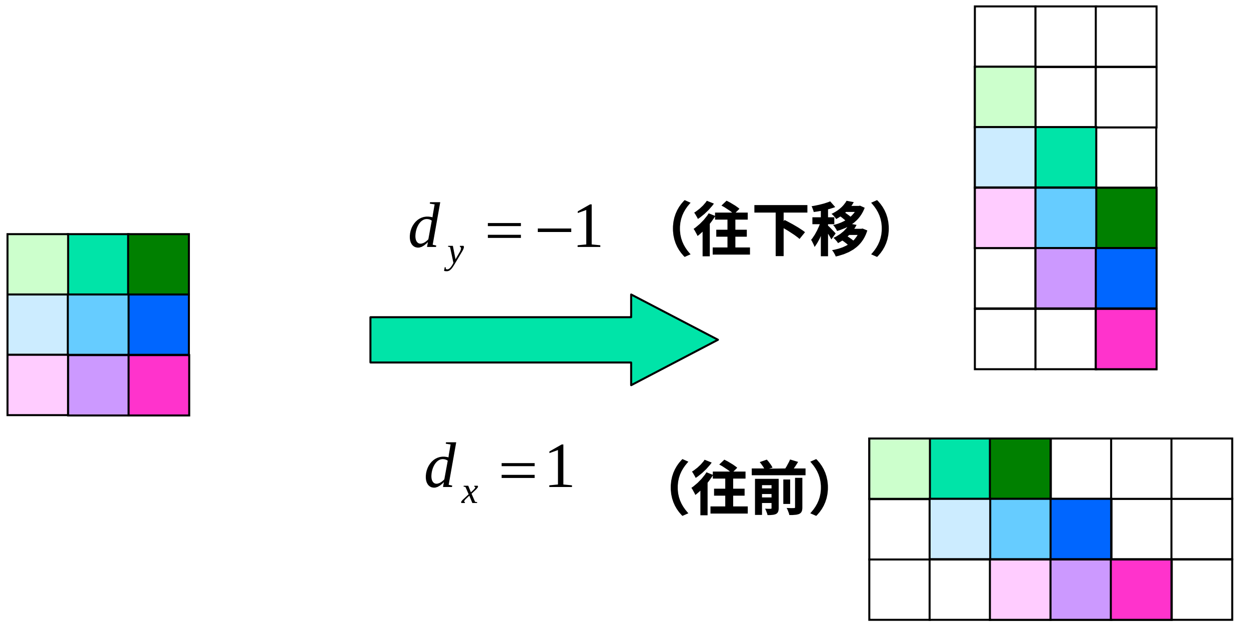
$$\begin{cases} x' = x \\ y' = y + d_y x \end{cases} (y\text{方向的错切})$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}_{new} = \begin{bmatrix} 1 & 0 & 0 \\ d_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}_{old}$$

图像的空域变换——几何变换

例：图像的错切变换

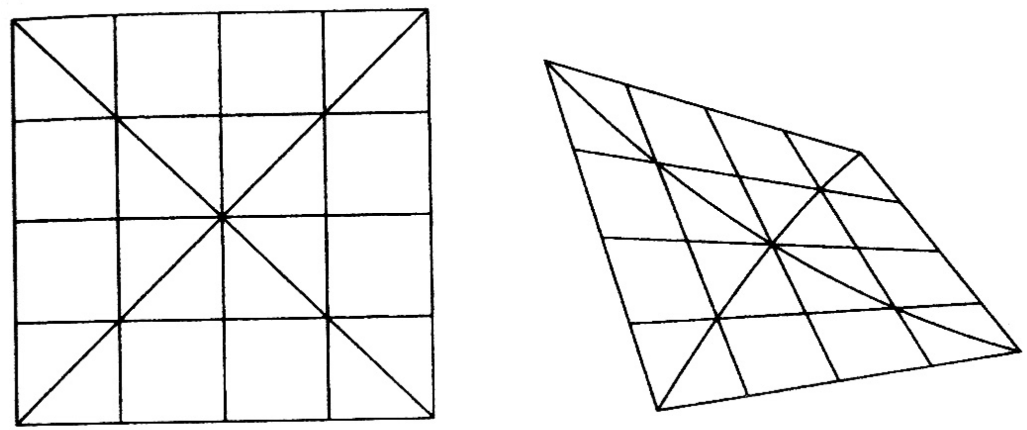
✓可以看到，错切之后原图像的像素排列方向改变。与旋转不同的是，x方向与y方向独立变化。



图像的空域变换—几何变换

- 伪仿射变换——双线性几何变换

$$x' = ax + by + \boxed{gxy} + e$$
$$y' = cx + dy + \boxed{hxy} + f$$

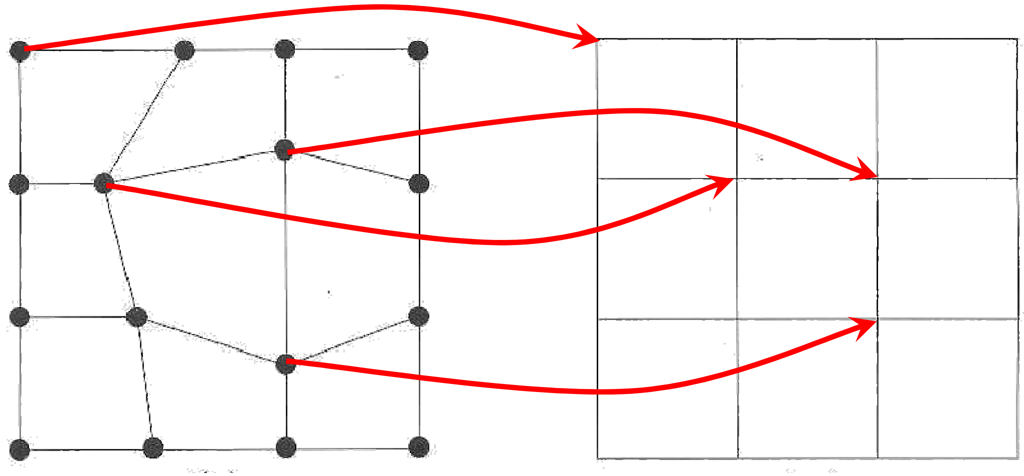


(a) 变换前　　(b) 变换后

- 特点:
  - ✓ 与**xy**平面上坐标轴平行的直线，变换为**x'y'**平面上的直线
  - ✓ 与**xy**平面上坐标轴不平行的直线，变换为**x'y'**平面上的曲线

图像的空域变换——几何变换

- 任意变形变换——非线性几何变换
  - 在二维平面上，<u>实现图像几何形状的任意变换</u>
  - 在二维平面上，<u>校正图像的几何失真</u>
  - 特征：一般的，原始图像与目标图像之间，存在一一对应的特征点（**tiepoints**，**GCPs**）



原始图像　　　　目标图像

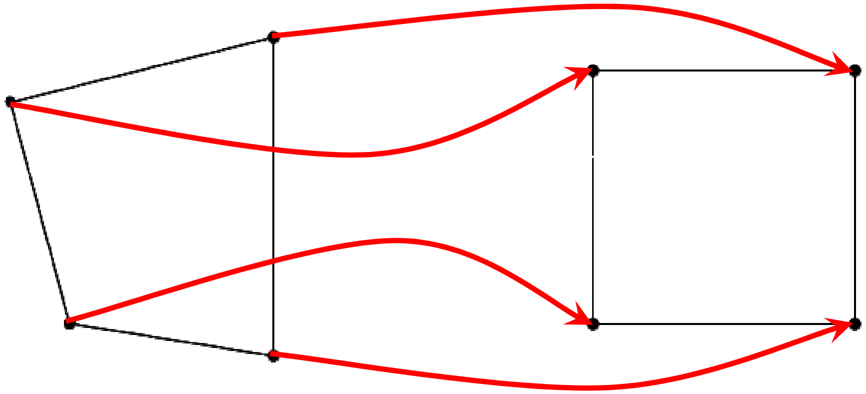
图像的空域变换—几何变换

- 任意变形变换——非线性几何变换（续）
  - 模型：一般的，原始图像与目标图像之间的坐标变换函数为非线性函数，需用高阶多项式进行近似描述
    - ✓ 例：三阶多项式变换

$$x = a_0 + a_1X + a_2Y + a_3X^2 + a_4XY + a_5Y^2 + a_6X^3 + a_7X^2Y + a_8XY^2 + a_9Y^3$$

$$y = b_0 + b_1X + b_2Y + b_3X^2 + b_4XY + b_5Y^2 + b_6X^3 + b_7X^2Y + b_8XY^2 + b_9Y^3$$

  - 通过原始图像与目标图像之间多个对应特征点（**GCP**点），可以确定上述多项式中的未知参数



利用多个对应特征点对的的坐标，确定多项式参数

图像的空域变换——几何变换

- 任意变形变换——非线性几何变换（续）
  - 多项式阶数与GCP数量的关系：

确定多项式变换参数的必要条件

$$\mathrm{GCP}s \geq \frac{(t+1)(t+2)}{2}$$

t: 多项式阶数

  - 通过多项式变换进行任意变形变换后的误差，通常用均方误差表示：

参考坐标点和变换坐标点间的RMS

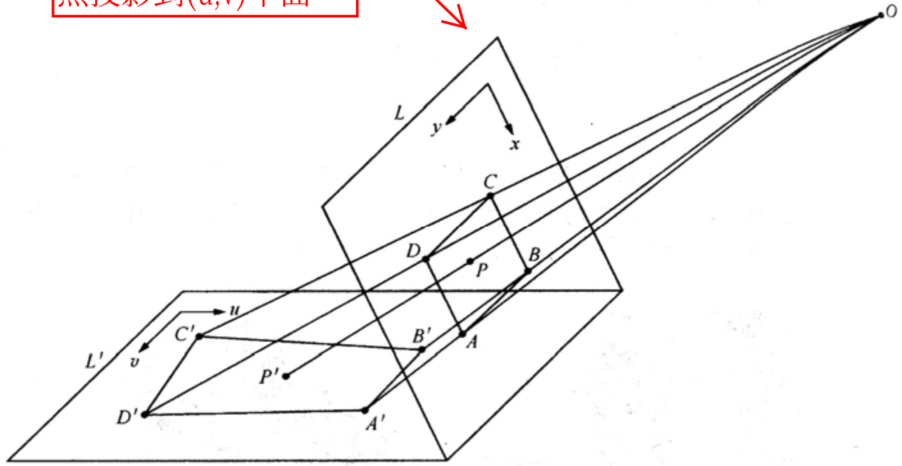
$$RMS = \sqrt{\frac{\sum_{j=1}^{n}(x_{jr} - x_{ji})^2 + \sum_{j=1}^{n}(y_{jr} - y_{ji})^2}{n}}$$

  - 变换示例

图像的空域变换——几何变换

- 二维图像的透视变换
  - 将一个平面上的点**P(x,y)**，以投影中心**O**为基准，投影成另一个平面上的点**P’(x,y)**
  - 可<u>看作为三维物体向二维图像透视投影的特殊形式</u>

以o为投影中心基准，将(x,y)平面上的点投影到(u,v)平面



<u>二维图像透视变换示例</u>

图像的空域变换——几何变换

- 二维图像的透视变换（续）
  - 二维图像透视变换函数及其齐次坐标表示为:

$$x' = \frac{ax+by+e}{mx+ly+1}; \quad y' = \frac{cx+dy+f}{mx+ly+1}$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = T \times \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad T = \left[\begin{array}{cc:c} a & b & e \\ c & d & f \\ \hdashline m & l & 1 \end{array}\right]$$

注：mx+ly+1=1

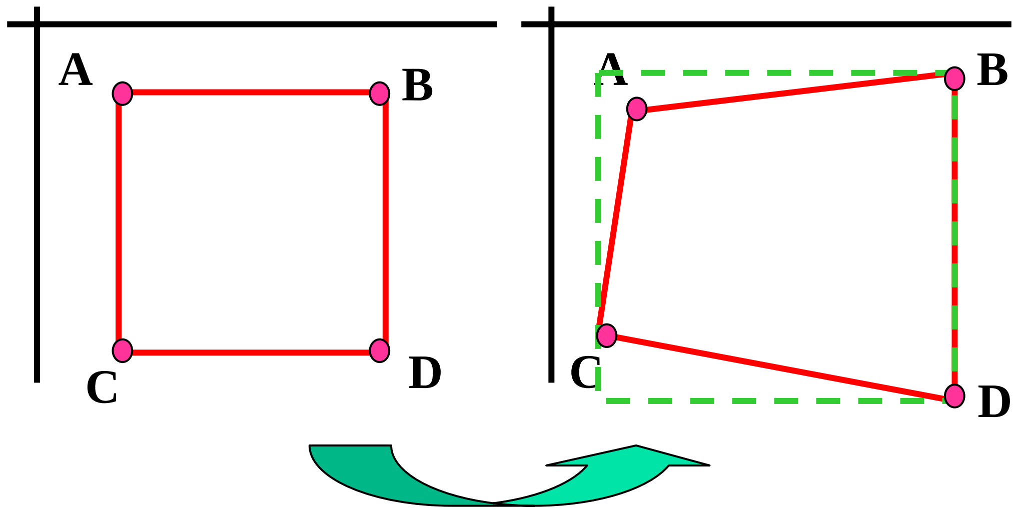
  - 与前面关于齐次变换矩阵的描述类似，这里引入第三个子矩阵[*m l*]，实现图像的透视变换

图像的空域变换——几何变换

- 二维图像的透视变换（续）
  - 变换式中共有**8**个独立的参数，可采用图像点对的方式，进行二维平面图像的透视投影计算

$$x' = \frac{ax+by+e}{mx+ly+1};\quad y' = \frac{cx+dy+f}{mx+ly+1}$$

  - **2**个独立的方程
  - **8**个独立的参数
  - **4**个**GCP**，**4x2**≥**8**



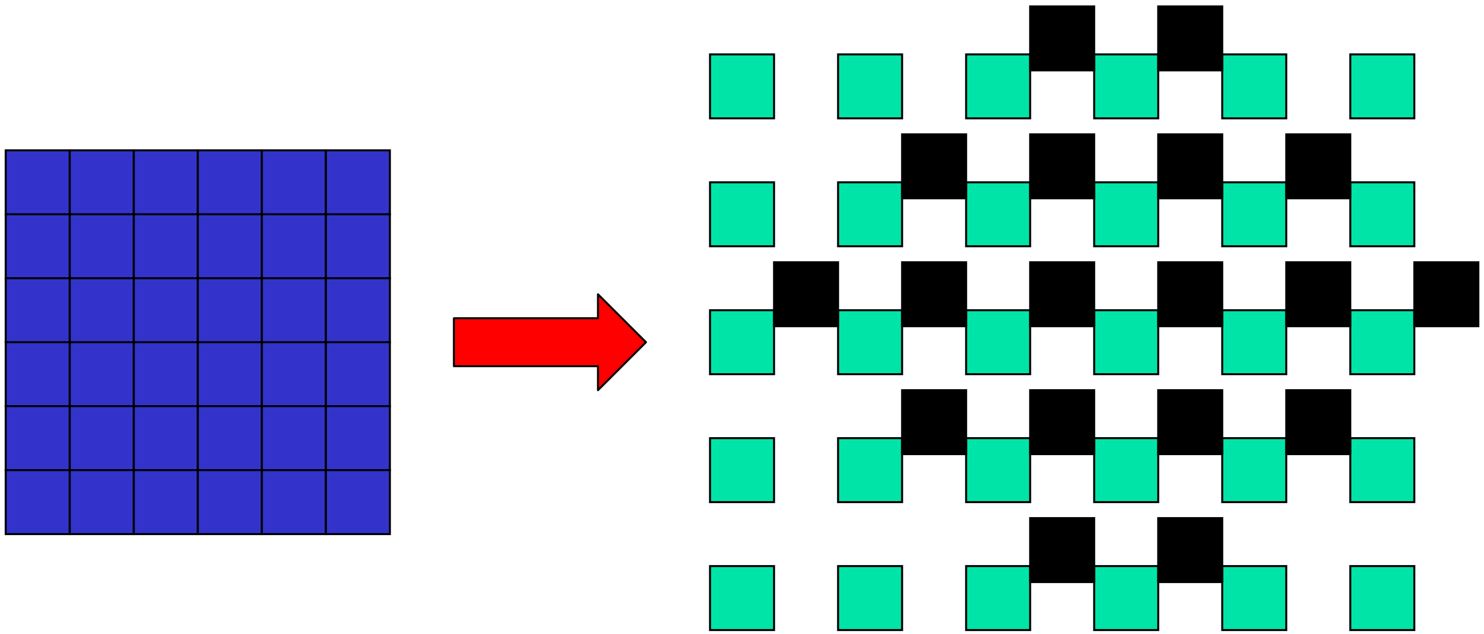
图像的空域变换——几何变换

- 基本几何变换的特征
  - 坐标空间的变化
    - ✓范围发生变化
    - ✓大小发生变化
  - 像素值的变化
    - ✓像素值不发生变化——位置改变

图像的空域变换——几何变换

- 旋转、缩放、变形变换中的漏点、不规则点问题

新坐标系显示格点
上的像素值待确定



图像的空域变换——几何变换

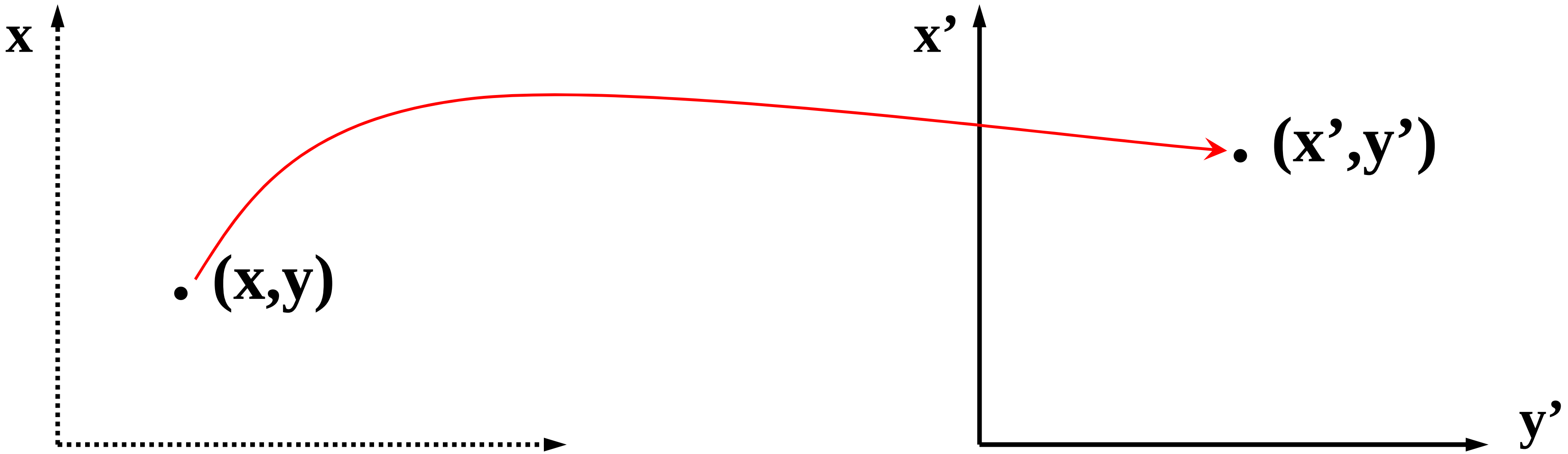
- 离散几何变换的计算问题
  - 空间坐标
    - ✓向前映射法
    - ✓向后映射法
  - 像素值计算——灰度插值（重采样）
    - ✓最近邻插值法
    - ✓双线性插值（一阶插值）
    - ✓高阶插值

重采样变换 —》对于显示格点的重采样

- 向前映射计算法

$$g(x',y') = g(a(x,y), b(x,y))=f(x, y);$$

  - 从原图像坐标计算出目标图像坐标
    - ✓镜像、平移变换使用这种计算方法



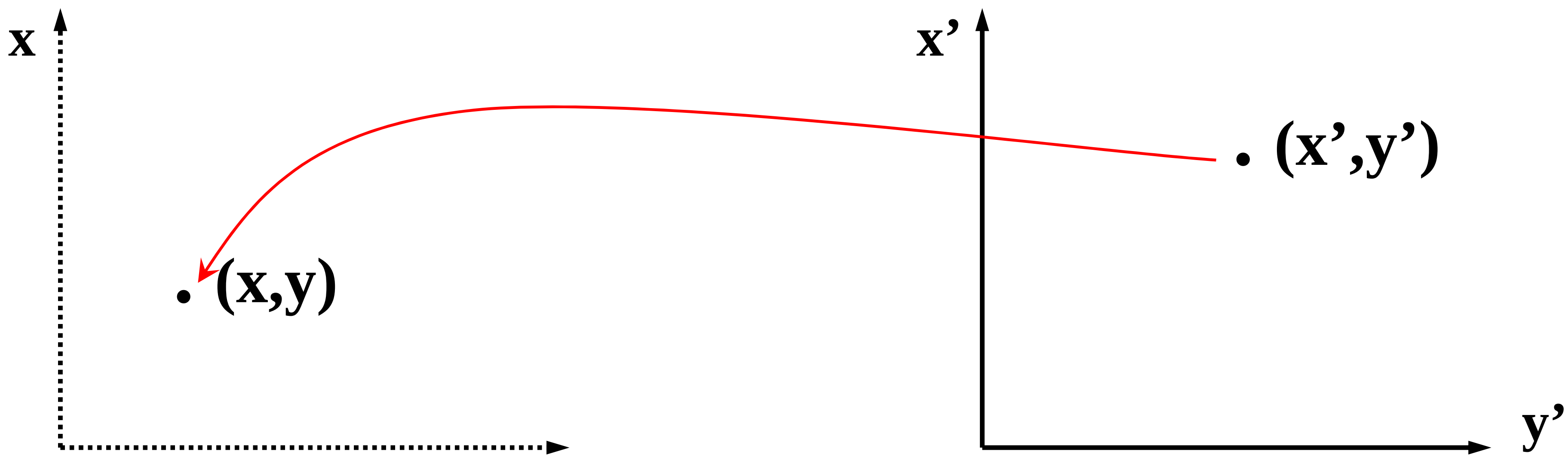
图像的空域变换——几何变换

- 向后映射计算法

  $$g(x', y')=f(a\text{^}{-1}(x',y'), b\text{^}{-1}(x',y'))\ ;$$

  X　Y

  - 从结果图像的坐标计算原图像的坐标
    - ✓旋转、缩放、变形可以使用

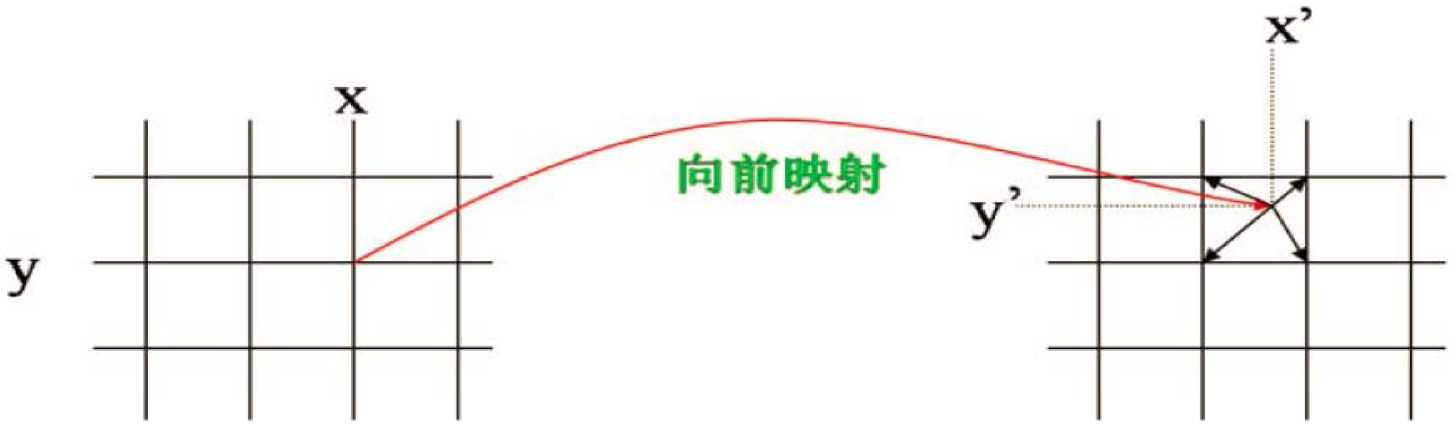


有关向前和向后映射

插值：向前映射

■显示格点(x, y)经向前映射后一般都不在显示格点上（见下图），因此变换后图像显示格点上的像素值需要据映射后的像素插值得到。

■除平移或对称映射情形，计算复杂。



有关向前和向后映射（续）

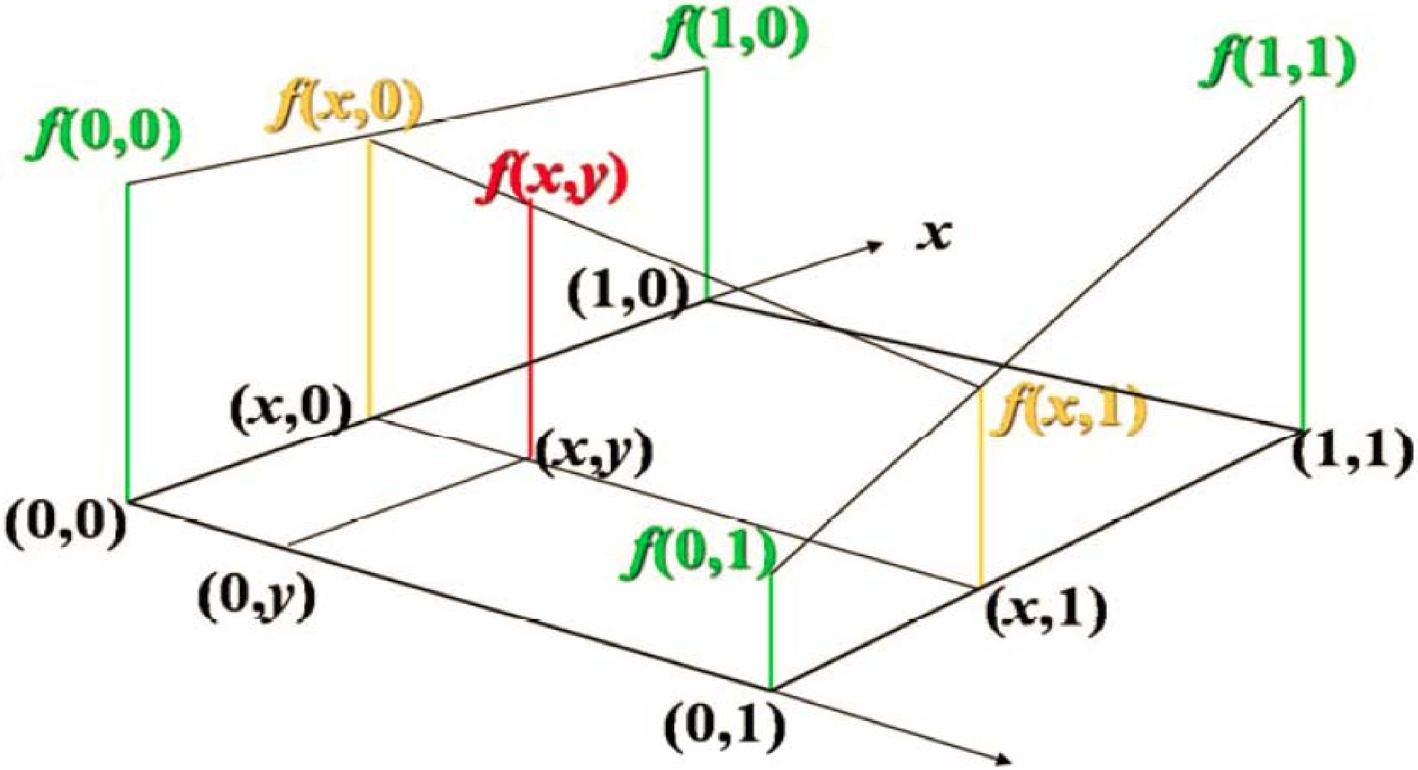
插值：向后映射

■对于一般几何变换，~~先~~向后映射比较直观和简单。给定目标图像显示格点上(x',y')的像素值，经向后映射得到在原图像上的位置$(x,y)=(a^{-1}(x',y'),b^{-1}(x',y'))$一般不在显示格点上。

■可以由其在原图像上近邻的 4 个显示格点，通过插值（例如双线性插值），得到$g(x',y')$。

有关向前和向后映射（续）

以下给出了双线性插值图示：



$$f(x,y) = (1-x)*(1-y)*f(0,0) + (1-x)*y*f(0,1) + x*(1-y)*f(1,0) + x*y*f(1,1)$$

图像的空域变换——几何变换

- 复合变换——多种变换的计算
  - 注意计算的顺序
  - 将多级变换合并为一级变换
- 例：围绕任意坐标点的旋转（**x0, y0**）

(1) 将（**x0, y0**）点平移至坐标原点（**0, 0**）

(2) 旋转

(3) 平移回（**x0, y0**）点

图像的空域变换——几何变换

- 复合变换——多种变换的计算

(1) $$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}_{new} = \begin{bmatrix} a(x,y) \\ b(x,y) \\ 1 \end{bmatrix}_{new} = \begin{bmatrix} 1 & 0 & x_0 \\ 0 & 1 & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}_{transl}$$

(2) $$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}_{transl} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}_{trans\,\&\,rot}$$

图像的空域变换——几何变换

$$(3)\quad \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}_{trans\,\&\,rot} = \begin{bmatrix} 1 & 0 & -x_0 \\ 0 & 1 & -y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}_{old}$$

通过矩阵的线性运算形成一次变换式

第一次：平移到原点

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}_{new} = \begin{bmatrix} 1 & 0 & x_0 \\ 0 & 1 & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -x_0 \\ 0 & 1 & -y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}_{tl\&r\&rtl}$$

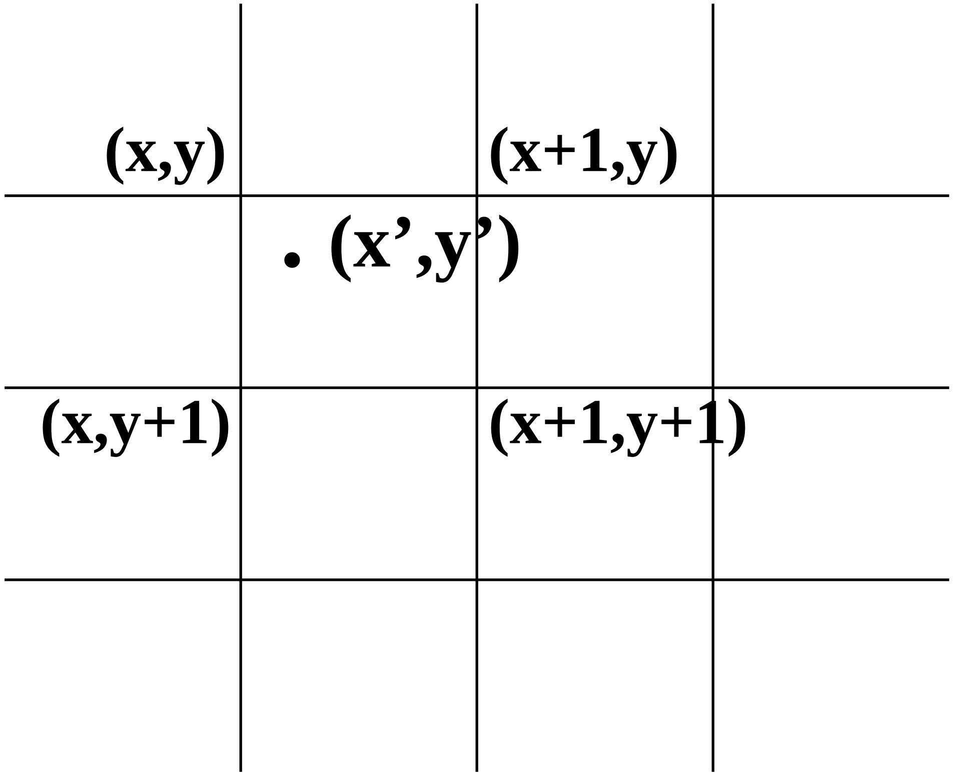
第三次：反平移

第二次：旋转

图像的空域变换——灰度插值

- 灰度插值——最近邻插值法
  - 选择最临近点像素灰度值

(x’,y’) 点 像 素 的 灰 度 值 为 原 图 中 (x,y) 点 的 像 素值



■ 灰度插值——最近邻插值法

- 特点
  - ✓简单快速
  - ✓灰度保真性好
  - ✓误差较大
  - ✓视觉特性较差

马赛克效应

图像的空域变换——灰度插值

- 灰度插值——双线性插值（一阶插值）

$f'(x',y')=$

$a\cdot f(x,y) + b\cdot f(x,y+1)$

$f''(x',y')=$

$c\cdot f(x,y) + d\cdot f(x+1,y)$

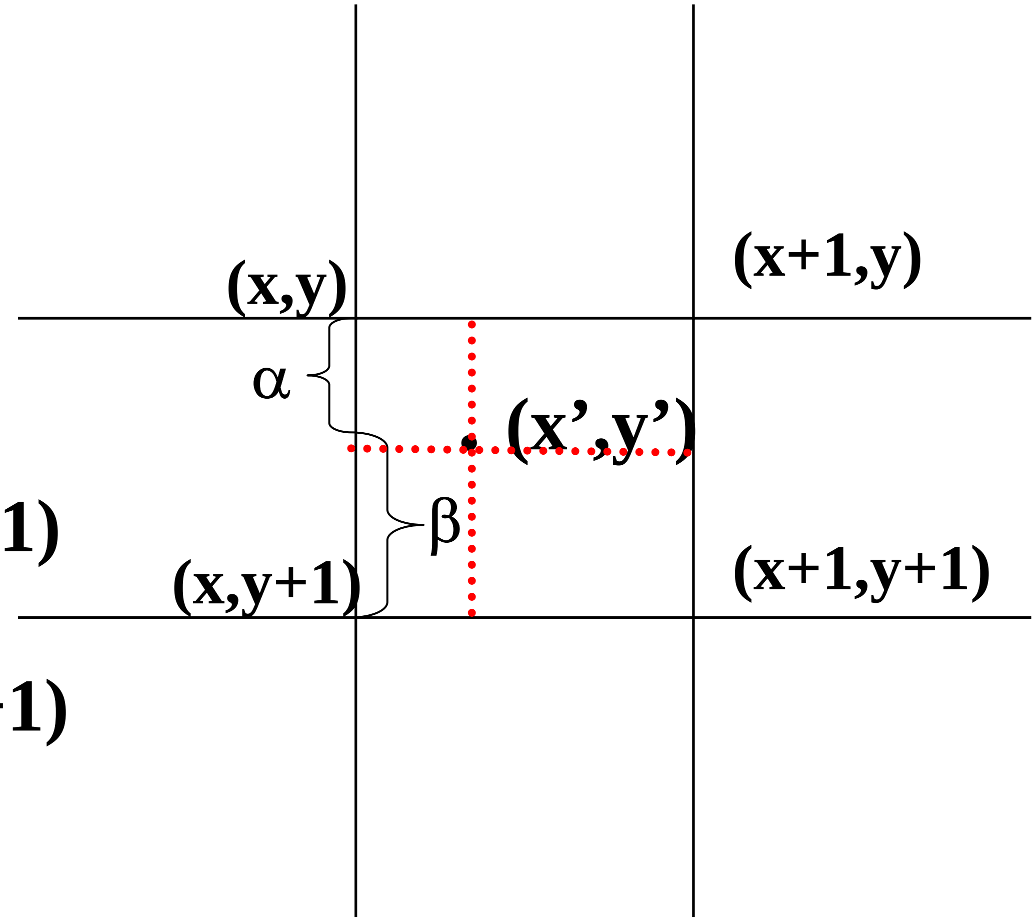
$f'''(x',y')=$

$u\cdot f(x,y+1) + v\cdot f(x+1,y+1)$

$f''''(x',y')=$

$w\cdot f(x+1,y) + z\cdot f(x+1,y+1)$

$f(x',y')=\ldots\ldots$



图像的空域变换——灰度插值

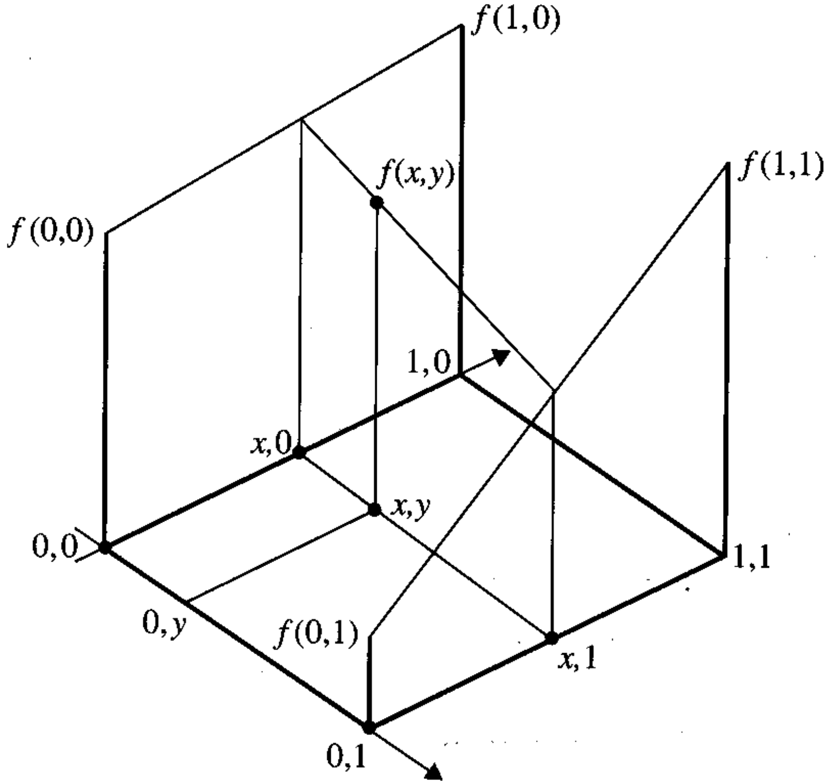
- 双线性插值——简化计算方法
  - 应用双曲抛物面方程

$$\boxed{f(x,y)=ax+by+cxy+d}$$

  - 归一化坐标值

$$0>x>1,\ 0>y>1$$

  - 可有:



$$f(x,y)=[f(1,0)-f(0,0)]x+[f(0,1)-f(0,0)]y$$
$$+[f(1,1)+f(0,0)-f(0,1)-f(1,0)]xy+f(0,0)$$

- 线性运算理论
  - 如果f是一个线性运算，则

$$f(x_1+x_2,y)=f(x_1,y)+f(x_2,y)$$

  - 但对

$$f(x,y)=ax+by+cxy+d$$

  - 则有

$$f(x_1+x_2,y)=a(x_1+x_2)+by+c(x_1+x_2)y+d$$

$$f(x_1+x_2,y)=ax_1+ax_2+by+cx_1y+cx_2y+d$$

图像的空域变换——灰度插值

- 灰度插值——双线性插值（一阶插值）
  - 双线性插值一般理论——双曲抛物面方程插值

$$f(x,y) = ax + by + cxy + d$$

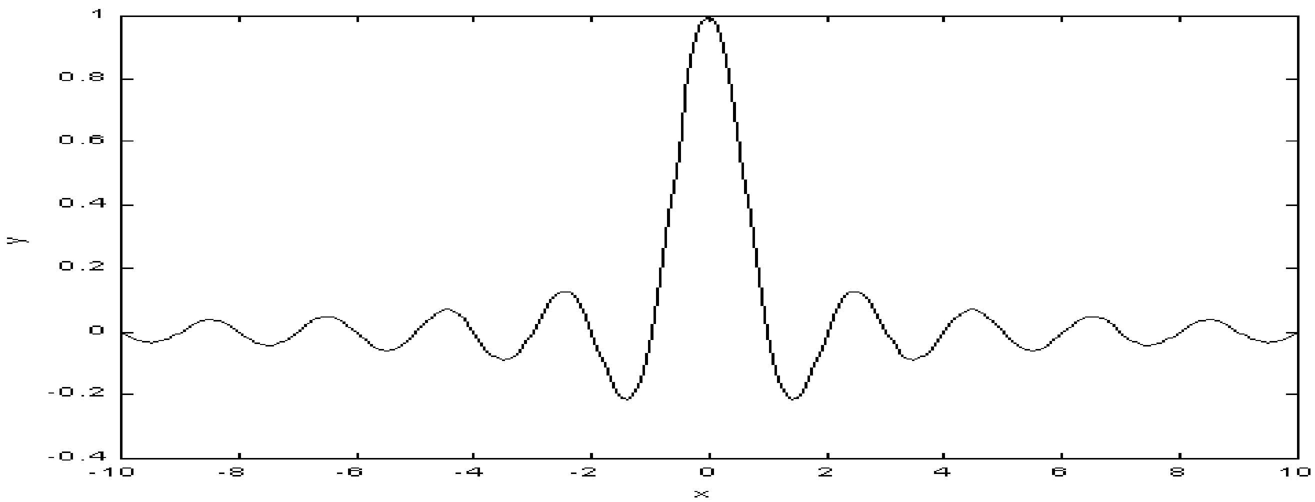
  - 需得到四个未知参数——利用四个已知点
  - 特点
    - ✓计算中较为充分地考虑相邻各点的特征，具有灰度平滑过渡特点
    - ✓一般情况下可得到满意结果
    - ✓具有低通滤波特性，使图像轮廓模糊
    - ✓平滑作用使图像细节退化，尤其在放大时
    - ✓不连续性会产生不希望的结果

图像的空域变换——灰度插值

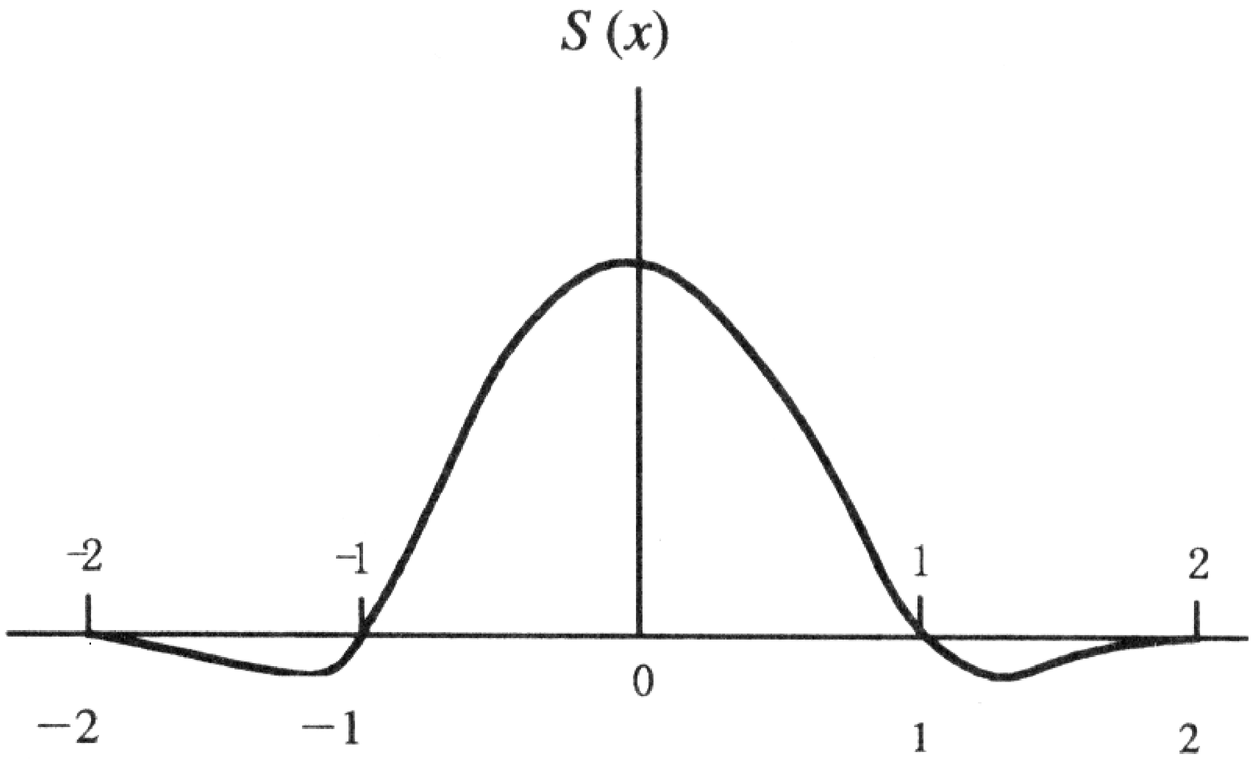
- 灰度插值——最佳插值函数
  - 在满足Nyquist条件下，从离散信号x(nTs)可恢复连续信号x(t)：

在满足乃奎斯特抽样条件下，可以无失真恢复，但需要无穷抽样点

$$\therefore \quad x(t)=\sum_{i=-\infty}^{+\infty} x(nT_s)\sin c\left(\frac{\pi}{T_S}(t-nT_s)\right)$$



- 灰度插值——高阶插值
  - 如果简化计算，仅取原点周围有限范围函数：



图像的空域变换——灰度插值

■ 灰度插值——高阶插值

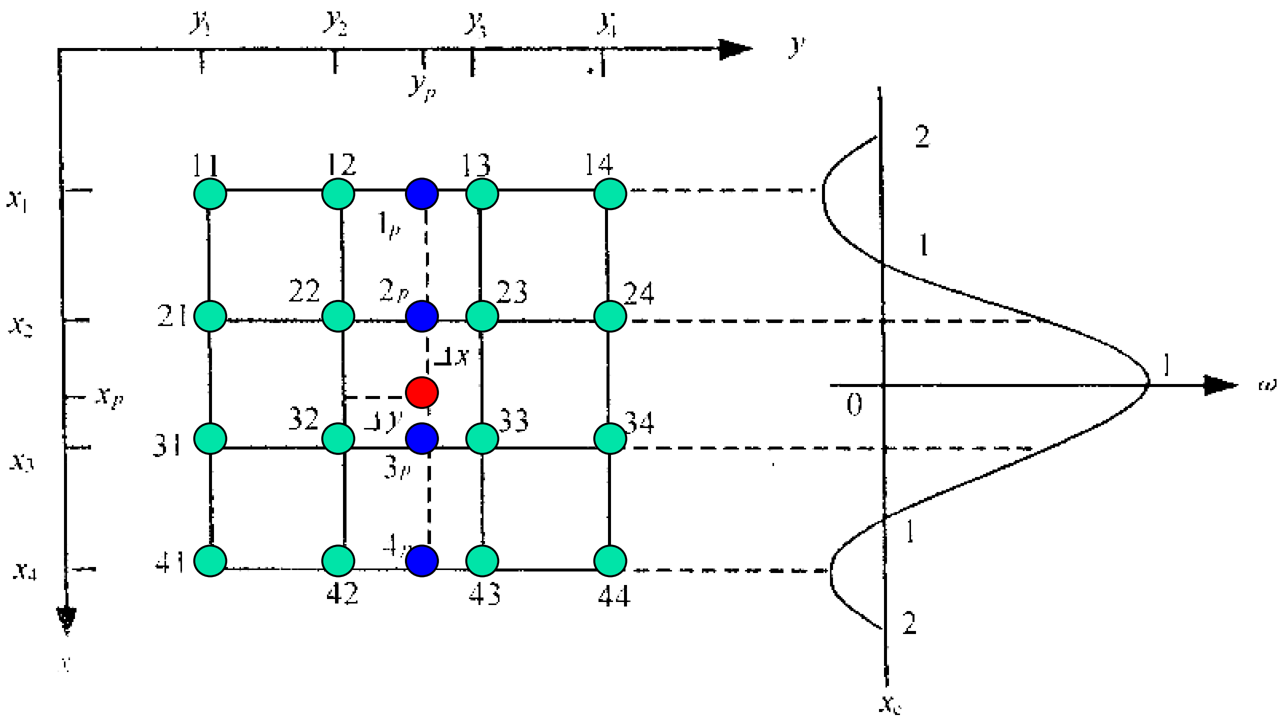
- 并利用三次多项式来近似理论上的最佳插值函数**sinc(*x*)**:

$$S(x)=\begin{cases}1-2|x|^2+|x|^3 & |x|<1\\ 4-8|x|+5|x|^2-|x|^3 & 1\le|x|\le 2\\ 0 & |x|>2\end{cases}$$

- 由此形成常用的三次卷积插值算法，又称三次内插法，两次立方法(**Cubic**)。**CC**插值法等

图像的空域变换——灰度插值

- 灰度插值——三次卷积插值算法实现
  - 利用待插值点周围的**16**个邻点像素值:



  - 首先确定辅助点位**1*p***，**2*p***，**3*p***，**4*p**各点亮度值
  - 再由确定***p***点亮度值

双立方插值（Bicubic）

插值公式:

S（x）为双立体插值核

$$f(i+x, j+y) = \mathbf{ABC}$$

其中：$0 \le |x|,|y| \le 1$，$\mathbf{A}$，$\mathbf{B}$和$\mathbf{C}$为矩阵

$$\mathbf{A} = \begin{bmatrix} S(1+x) & S(x) & S(1-x) & S(2-x) \end{bmatrix}$$

$$\mathbf{C} = \begin{bmatrix} S(1+y) & S(y) & S(1-y) & S(2-y) \end{bmatrix}^{\tau}$$

双立方插值（Bicubic）

矩阵$\mathbf{B}$则用到相邻 16 个格点的灰度值：

$$\mathbf{B}=\begin{bmatrix} f(i-1,j-2) & f(i,j-2) & f(i+1,j-2) & f(i+2,j-2) \\ f(i-1,j-1) & f(i,j-1) & f(i+1,j-1) & f(i+2,j-1) \\ f(i-1,j) & f(i,j) & f(i+1,j) & f(i+2,j) \\ f(i-1,j+1) & f(i,j+1) & f(i+1,j+1) & f(i+2,j+1) \end{bmatrix}$$

图像的空域变换——灰度插值

- 灰度插值——三次卷积插值算法特点
  - 为满足二维**Nyquist**条件下，最佳重构公式的近似
  - 只有图像满足特定的条件，三次卷积插值算法才可获得最佳结果
  - 可使待求点的灰度值更好地模拟实际可能值
  - 可取得更好的视觉效果
  - 三次卷积内插突出的优点是高频信息损失少，可将噪声平滑
  - **4 × 4**时，像元均值和标准差信息损失小
  - 计算量大为增加

图像的空域变换——灰度插值

- 灰度插值——图像处理中内插方法的选择
  - 内插方法的选择除了考虑图像的显示要求及计算量，还要考虑内插结果对分析的影响
    - ✓当纹理信息为主要信息时，最近邻采样将严重改变原图像的纹理信息
    - ✓但灰度信息为主要信息时，双线性内插及三次卷积内插将减少图像异质性，增加图像同质性，其中，双线性内插方法使这种变化更为明显

图像的空域变换——非几何变换

- 非几何变换的定义

  对于原图像**f(x,y)**，灰度值变换函数

  $$T(f(x,y))$$

  唯一确定了非几何变换：

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

  g(x,y)是目标图像

- 非几何变换属于像素值的变换，没有几何位置的改变——灰度变换

- 灰度变换的目的是为了改善画质，使图像的显示效果更加清晰

图像的空域变换——非几何变换

- 基本非几何变换类型

Negative: $s = L - 1 - r$

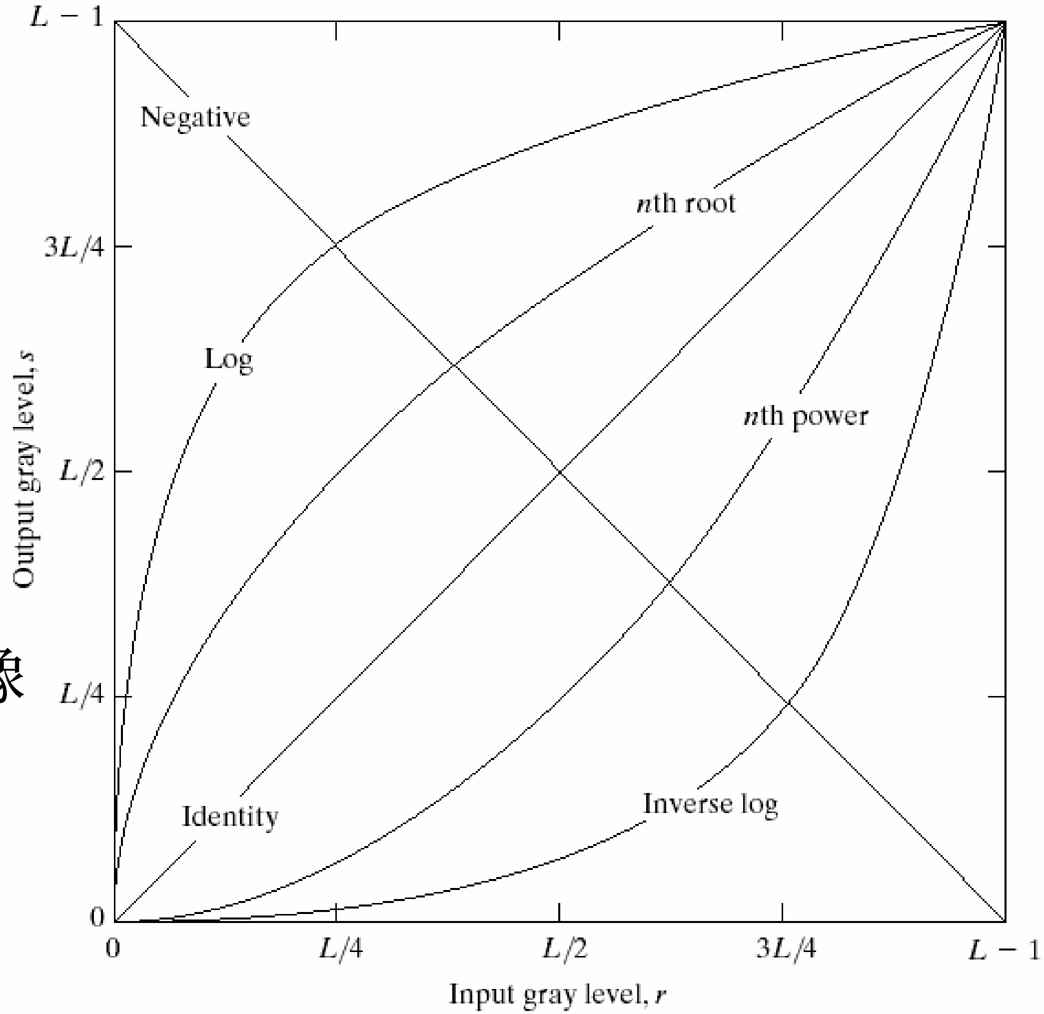
Log: $s = c\log(1 + r)$

Inverse Log: $s = e^{cr} - 1$

Power-law: $s = cr^{\gamma}$

……

r和s分别为原图像和目标图像灰度值



- 彩色图像的非几何变换定义

对于彩色原图像**f(x,y)**，颜色值变换函数

$$T_r(f(x,y));\ T_g(f(x,y));\ T_b(f(x,y));$$

唯一确定了非几何变换:

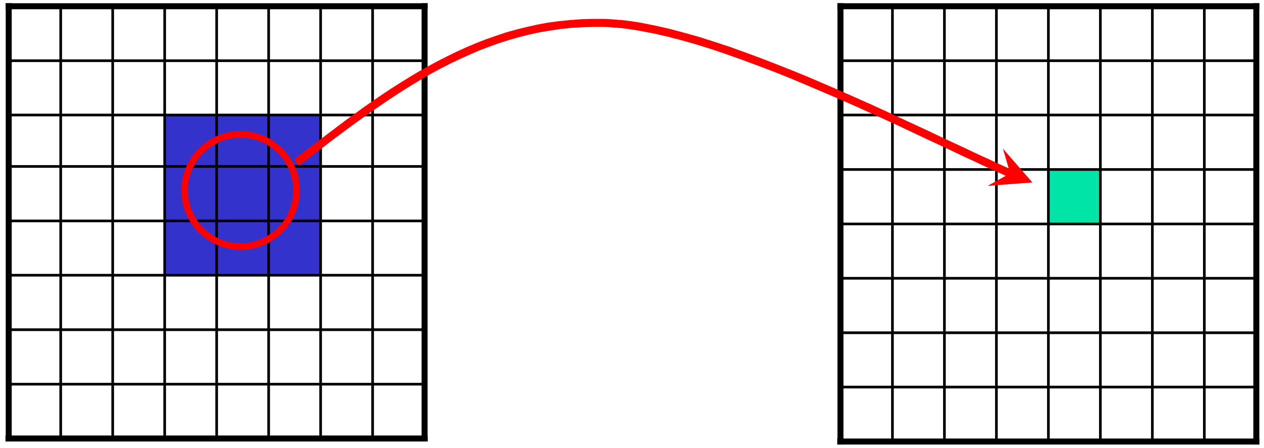
$$g_r(x,y) = T_r(f(x,y))$$

$$g_g(x,y) = T_g(f(x,y))$$

$$g_b(x,y) = T_b(f(x,y))$$

图像的空域变换——非几何变换

- 离散非几何变换的计算
  - 简单变换—像素值一一对应的映射
    - ✓如伪彩色变换
  - 复杂变换—同时考虑相邻各点的像素值



    - ✓通常通过模板运算进行

- 模板的定义
  - 所谓模板就是一个系数矩阵
  - 模板大小：经常是奇数，如:

    3x3　5x5　7x7 ……

  - 模板系数: 矩阵的元素

$$\mathbf{w_1\ w_2\ w_3}$$

$$\mathbf{w_4\ w_5\ w_6}$$

$$\mathbf{w_7\ w_8\ w_9}$$

图像的空域变换——非几何变换

模板运算的定义

对于某图像的子图像:

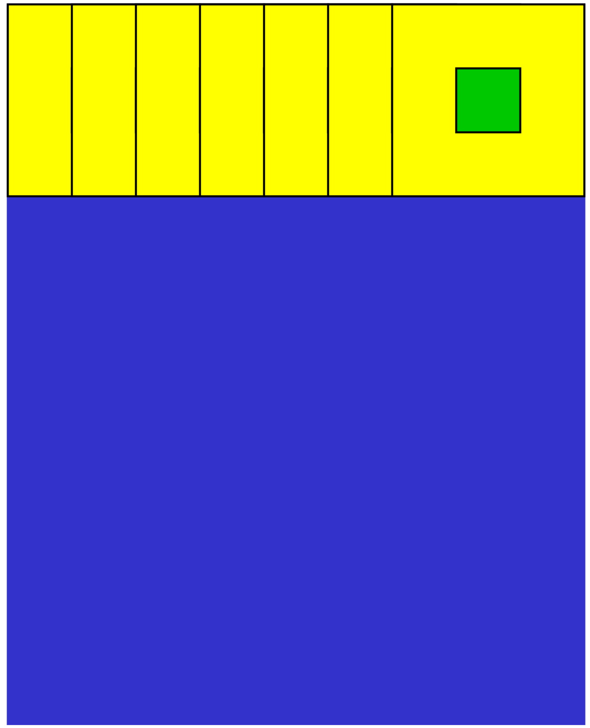
$$\begin{array}{ccc} z_1 & z_2 & z_3 \\ z_4 & z_5 & z_6 \\ z_7 & z_8 & z_9 \end{array}$$

z5的模板运算公式为:

$$R = w_1z_1 + w_2z_2 + \ldots + w_9z_9$$

$w_1, w_2, \ldots w_9$

为加权系数。



图像的空域变换——非几何变换

- 灰度变换（点运算）的定义**(1)**
  - 对于输入图像**f(x, y)**，灰度变换**T**将产生一个输出图像**g(x, y)**，**g(x, y)**的每一个像素值，均取决于**f(x, y)**中对应点的像素值

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

- 灰度变换（点运算）的定义**(2)**

  对于原图像**f(x,y)**，灰度值变换函数

  $$T(f(x,y))$$

  由于灰度值总是有限个，如：**0-255**

  非几何变换可定义为：

  $$R = T(r)$$

  其中**R,r**在**0-255**之间取值

图像的空域变换——非几何变换

- 灰度变换（点运算）的实现

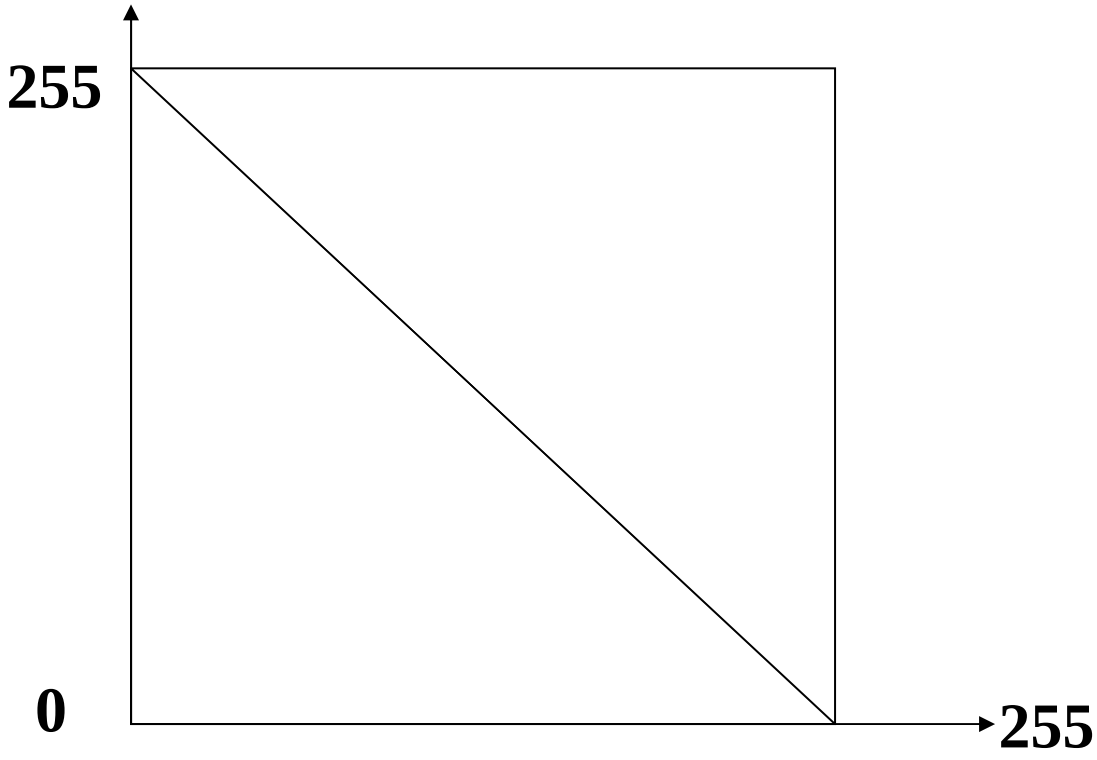
R = T(r) 定义了输入像素值与输出像素之间的映射关系，通常通过查表来实现。

因此灰度级变换也被称为**LUT**（**Look Up Table**）变换。

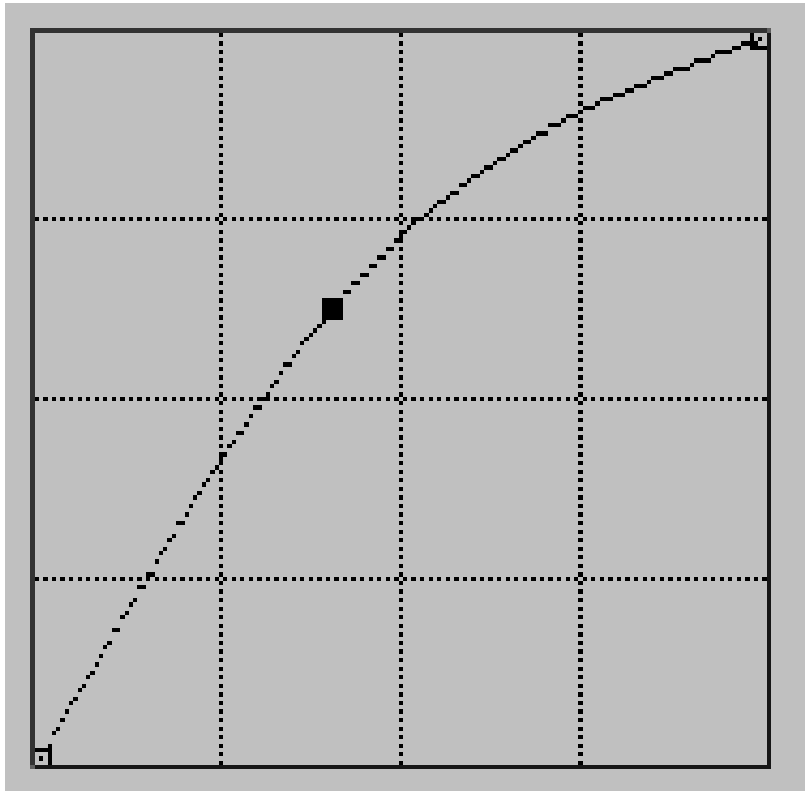
0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**……**	**250**	**251**	**252**	**253**	**254**	**255**
0	**3**	**5**	**7**	**9**	**11**	**13**	**15**	**17**	**19**	**……**	**252**	**253**	**254**	**254**	**254**	**255**

图像的空域变换——非几何变换

- 灰度变换例－1
  - 图像求反

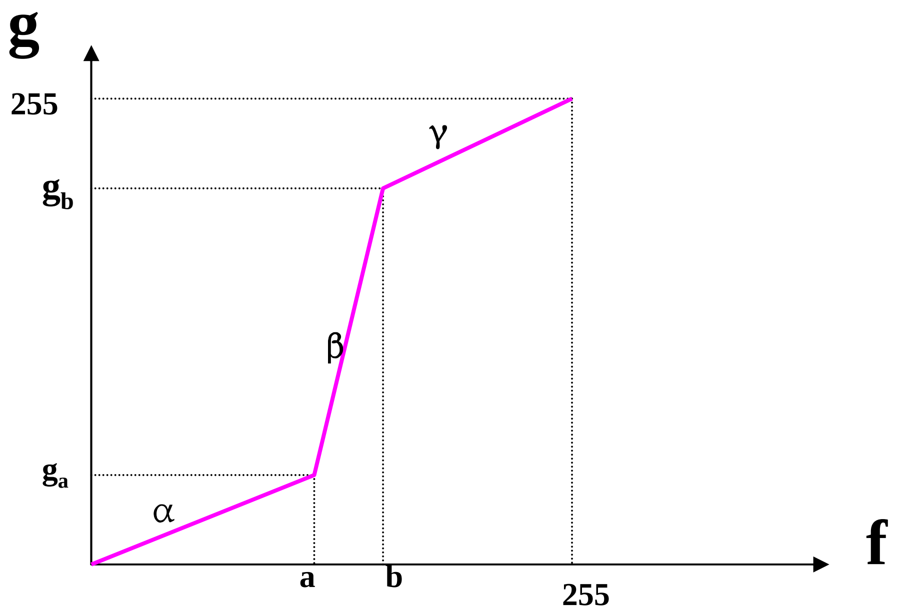


- 灰度变换例－2
  - 对比度拉伸



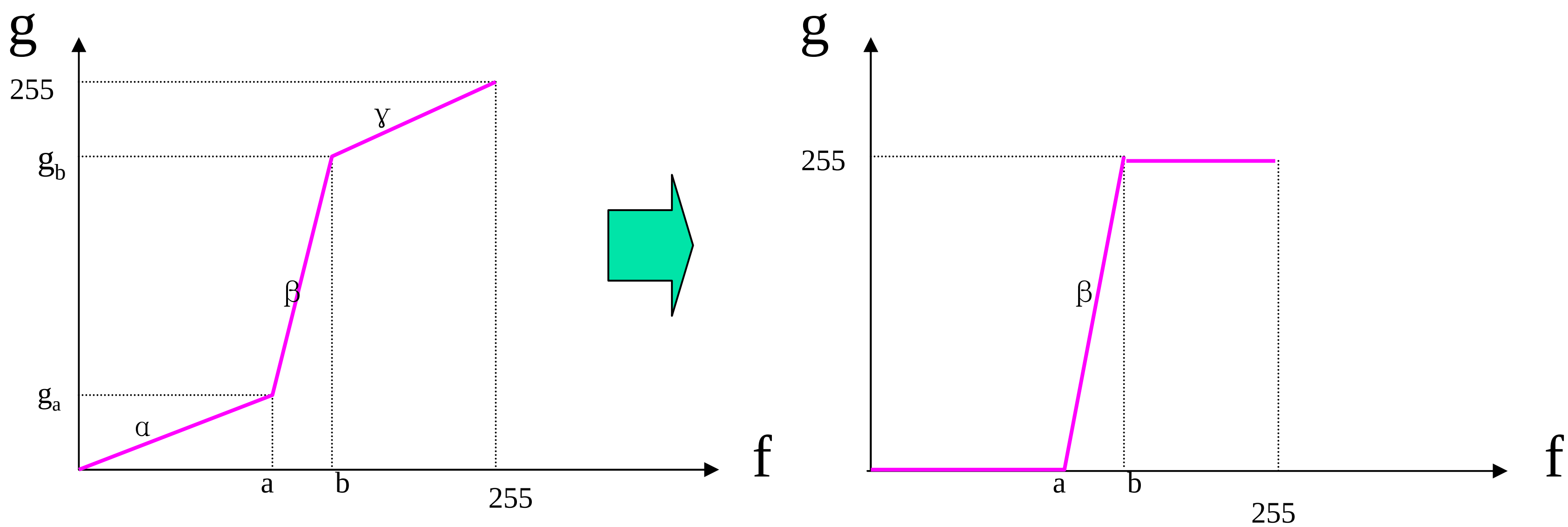
- 灰度变换例－3
  - 对比度展宽——突出图像中关心的部分
  - 方法:

$$
g=\begin{cases}\alpha f & 0\le f<a\\ \beta(f-a)+g_a & a\le f<b\\ \gamma(f-b)+g_b & b\le f<L\end{cases}
$$



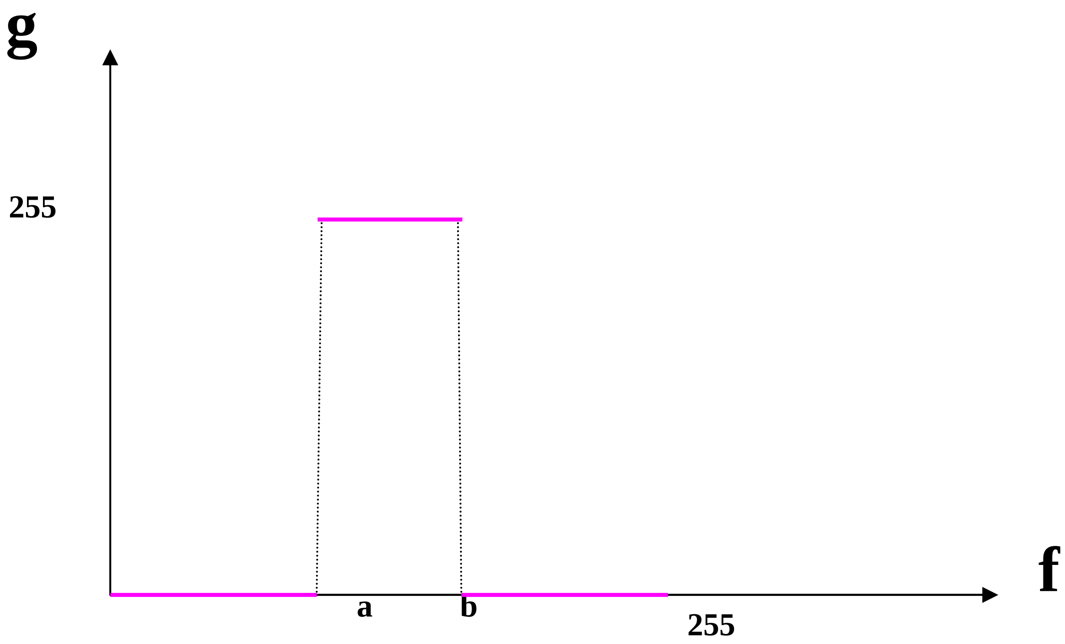
- 实例

- 灰度变换例－4
  - 灰度窗——只显示指定灰度级范围内的信息。如：$\alpha = \gamma = 0$



- 灰度变换例－5
  - 灰度级切片——只保留感兴趣的部分，其余部分置为0

只关注形状



图像的空域变换——非几何变换

- 灰度变换例－6

  - 灰度级修正

记录装置中心光轴附近光衰减较小

✓通过记录装置把一景物变成一幅图像时，景物上每一点所反射的光，并不是按同一比例转化成图像上相应点的灰度，靠近光轴的光要比远离光轴的光衰减得要少一些。

✓灰度级修正的目的是：使画面中的每个关心的细节信息通过灰度级修正之后，可以变得<u>清楚可见</u>。

例如：指数衰减律

$$I(x,y)=e(x,y)*g(x,y)$$

$$g(x,y)=e^{-1}(x,y)*I(x,y)$$

e(x,y): 衰减函数
I(x,y): 待修正图像
g(x,y): 原图像
辨识e(x,y) ->g(x,y)

图像的空域变换——非几何变换

将区间[a,b]内的灰度值拉伸到[0,255]

- 灰度变换例－7
  - ➢线性动态范围调整：

$$h^*(x,y)=\begin{cases}0 & h(x,y)<=a\\ \dfrac{255}{b-a}h(x,y)-\dfrac{255\,a}{(b-a)} & h(x,y)\in(a,b)\\ 255 & h(x,y)>=b\end{cases}$$

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

黑：0 → 2

白：9 → 7

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

y=1.8*x-3.6

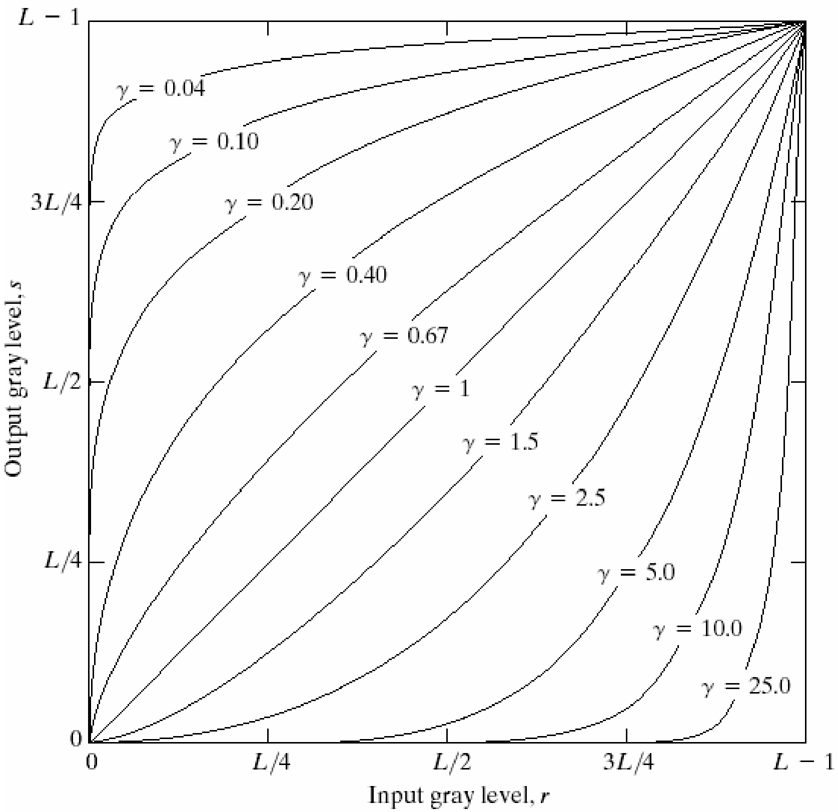
0	2	9	9	9
0	0	2	9	2
2	7	0	7	4
7	9	0	0	5
0	9	0	7	0

作用：进行亮暗限幅

图像的空域变换——非几何变换

- 灰度变换例－8
  - 图像输入输出的Gamma失真
  - Gamma校正

$$s = cr^{\gamma}$$



- ✓ 校正图像获取设备对于图像像素亮度相应的非线性
- ✓ 校正图像显示设备中输入信号与亮度显示之间的非线性关系
- ✓ 校正图像中不同像素值显示时的亮度感觉
  - 人的感知觉，心理量和物理量一般呈对数关系
  - 越暗处感觉越细，对同等光强的变化，暗处比亮处敏感

伽马校正说明（1）

伽马失真:

实际应用中，许多显示设备，包括显示器和打印机等对图像像素的灰度值的响应为指数规律（Power Law）。例如 CRT 显示器的响应指数为 2.5，图像输出:

$$s = r^{1/2.5} = r^{0.4}$$

<u>产生的图像比期望要暗</u>

伽马校正说明（2）

伽马校正:

假定待校正像素值为r，校正后的像素值为s，像素值范围$[0,255]$，则有校正公式:

$$s = 255 \times \left[\frac{r}{255} \right]^{\gamma}$$

归一化到 0-1 之间正数

$\gamma < 1$，提高灰度级，使图像变亮

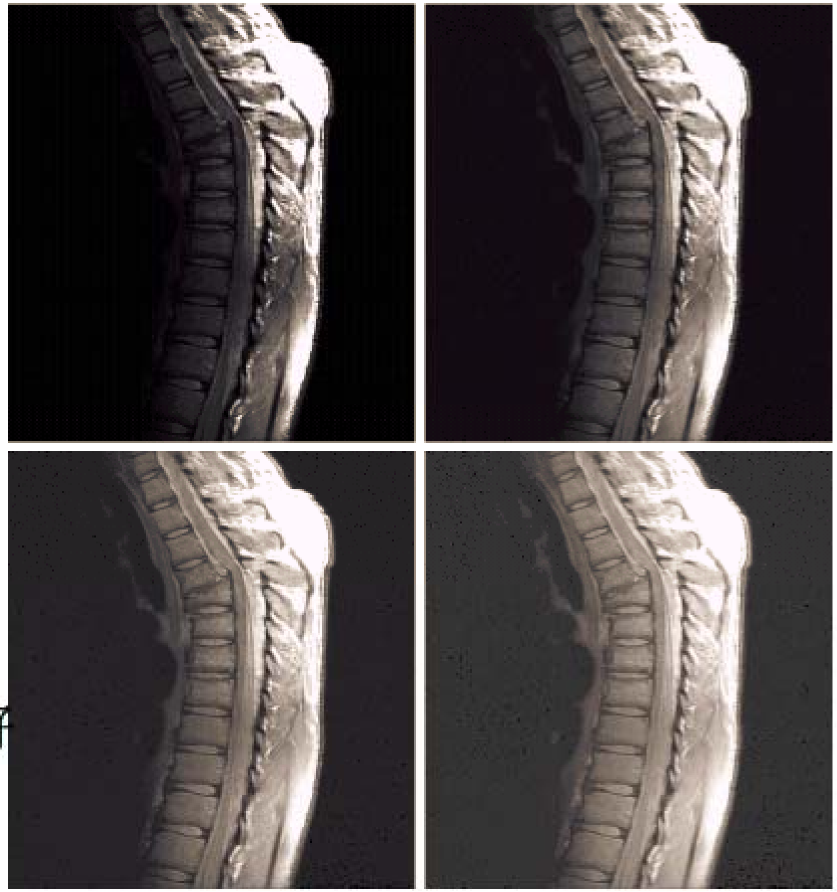
$\gamma > 1$，降低灰度级，使图像变暗

伽马校正说明（3）

- 例：人体胸上部脊椎骨折的核磁共振图像
- $\gamma < 1$ 提高灰度级，使图像变亮。c=1, $\gamma = 0.6, 0.4, 0.3$

$\gamma = 0.4$

增强效果最好



a b
c d

FIGURE 3.8 (a) Magnetic resonance (MR) image of a fractured human spine. (b)–(d) Results of applying the transformation in Eq. (3.2-3) with $c = 1$ and $\gamma = 0.6, 0.4,$ and 0.3, respectively. (Original image for this example courtesy of Dr. David R. Pickens, Department of Radiology and Radiological Sciences, Vanderbilt University Medical Center.)

伽马校正说明（4）

- 例：航空地面图像
- $\gamma > 1$降低灰度级，使图像变暗c=1，$\gamma = 3,4,5$

a b
c d

FIGURE 3.9
(a) Aerial image. (b)–(d) Results of applying the transformation in Eq. (3.2-3) with $c = 1$ and $\gamma = 3.0, 4.0$, and 5.0, respectively. (Original image for this example courtesy of NASA.)

$\gamma = 3$ √

$\gamma = 4$ √

$\gamma = 5$