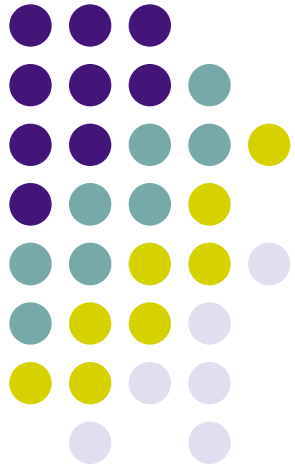
http://lamda.nju.edu.cn/liyf/dip19/Ch04.pdf

数字图像处理

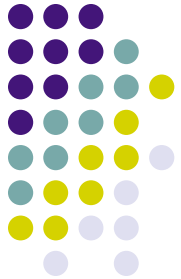


第四讲

空间域图像增强（Part III）

几何变换

几何变换



- 引言
- 空间变换
- 灰度级插值

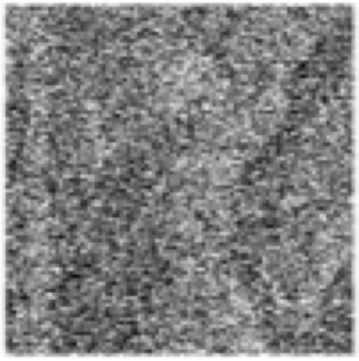
引言

- 点运算对单幅图像做处理，不改变像素的空间位置

$$s = r + a$$

引言

- 算术/逻辑运算对多幅图像做处理，也不改变像素的空间位置



降噪

引言

- 几何变换改变像素的空间位置使得图像得到增强






引言

- 几何变换对单幅图像做处理，改变像素的空间位置；

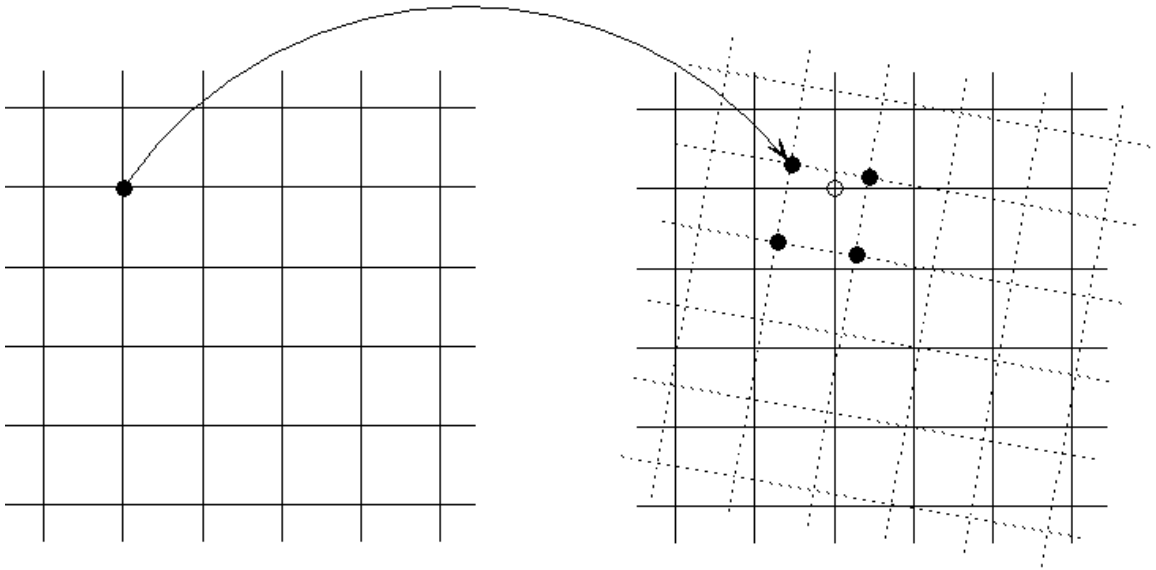
- 几何变换包含两个独立的算法：空间变换算法和灰度级插值算法
  - 空间变换：描述每个像素空间位置的变换
  - 灰度级插值：确定变换后图像像素的灰度级

空间变换

图像的每个坐标点(x,y)变换到新坐标点(u,v)

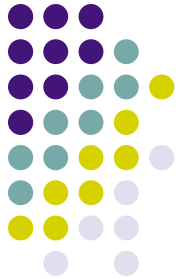
$$u = f_1(x, y)$$

$$v = f_2(x, y)$$



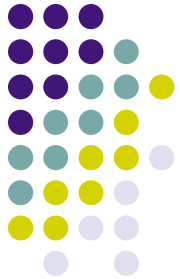
Note：图像坐标是离散的，网格的。变换后的坐标点可能不落在网格点上。

空间变换



- 空间变换需要满足一个条件
  - 保持图像中曲线型特征的连续性和各物体的连通性
  - 简而言之的话——相邻的输入产生相邻的输出
- 任意的空间变换会弄乱图像内容，或者内容支离破碎
- 一种常用的空间变换：仿射变换 （Affine Transformation）

空间变换



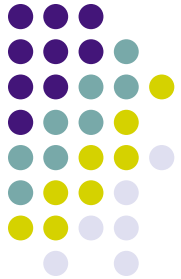
- 仿射变换（Affine Transformation）
  包括了旋转、伸缩、平移、倾斜等变换

$$u = c_{11}x + c_{12}y + c_{13}$$

$$v = c_{21}x + c_{22}y + c_{23}$$

  - C13和C23刻画了平移量
  - C11和C21刻画了伸缩比例
  - 整体组合刻画了旋转角度、倾斜程度

空间变换



- 仿射变换（Affine Transformation）包括了旋转、伸缩、平移、倾斜等变换

$$u = c_{11}x + c_{12}y + c_{13}$$

$$v = c_{21}x + c_{22}y + c_{23}$$

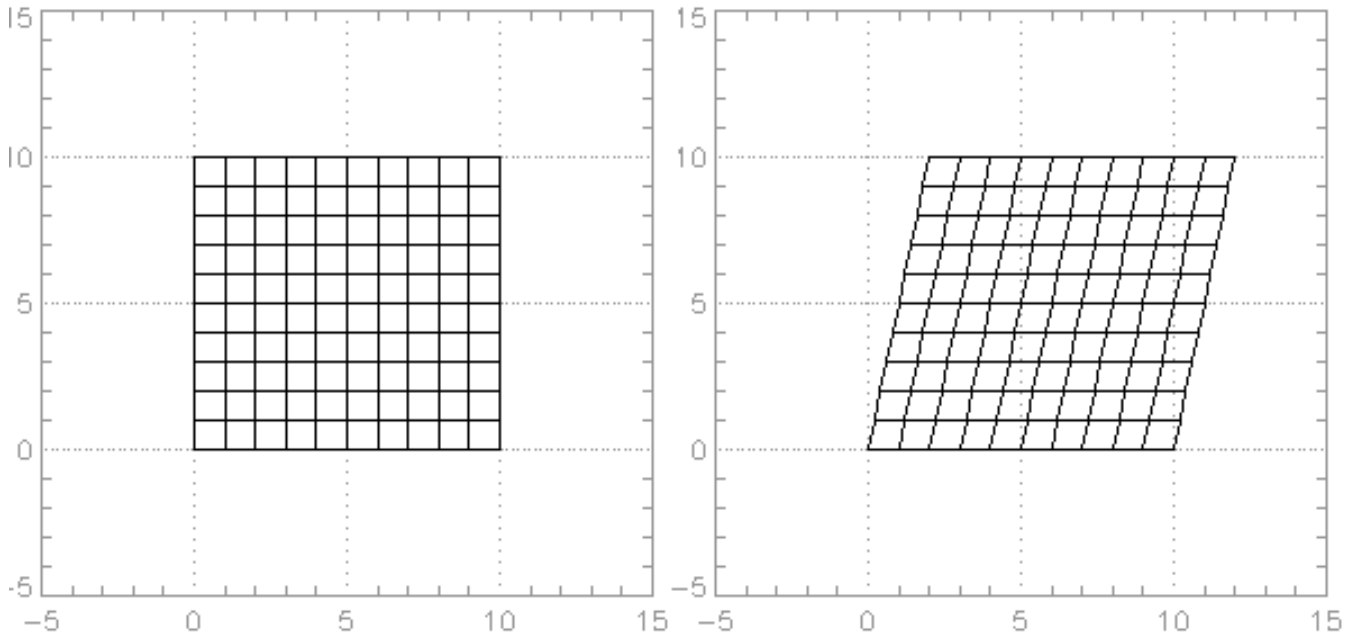
- 仿射变换具有以下良好的性质
  - 保持共线性（co-linearity）
    - 共线的点变换后依然共线
  - 保持距离比例（ratios of distance）
    - 线的中心变换后依然是线的中心

空间变换

- 倾斜变换（Sheer Transformation）

$$u = x + 0.2y$$

$$v = y$$

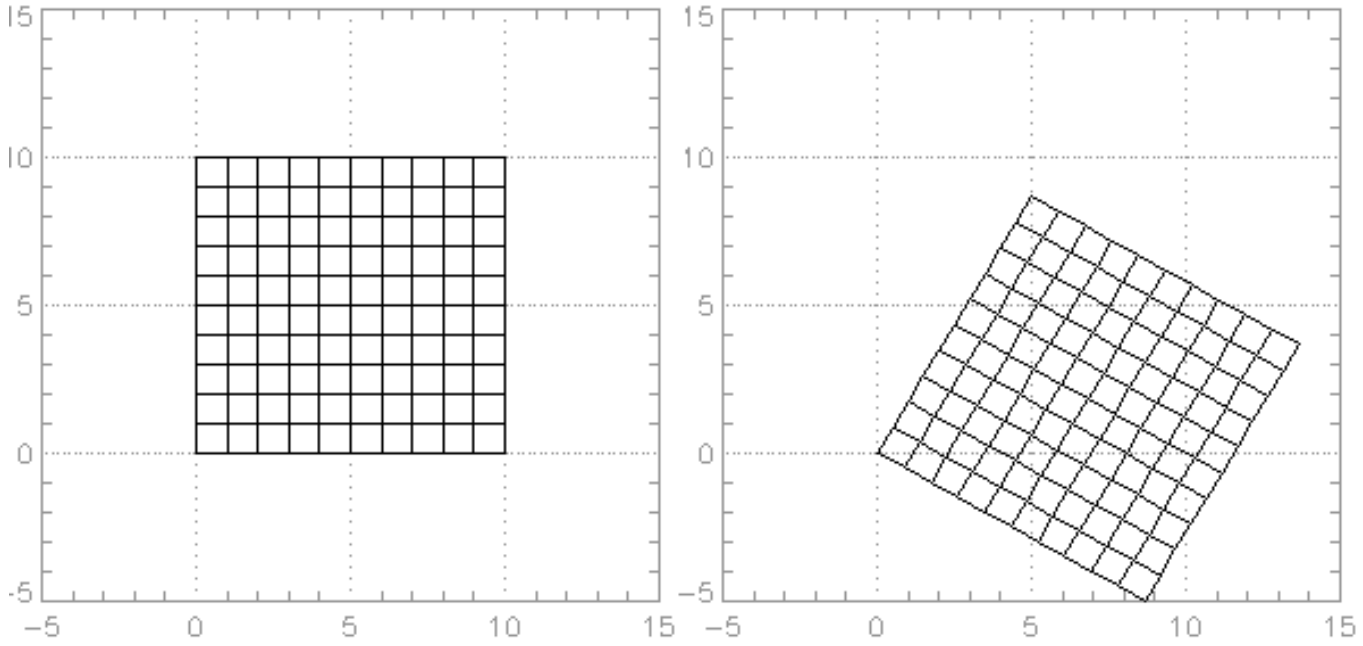


空间变换

- 旋转变换（Rotation Transformation）

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

$$v = -x\sin\theta + y\cos\theta$$

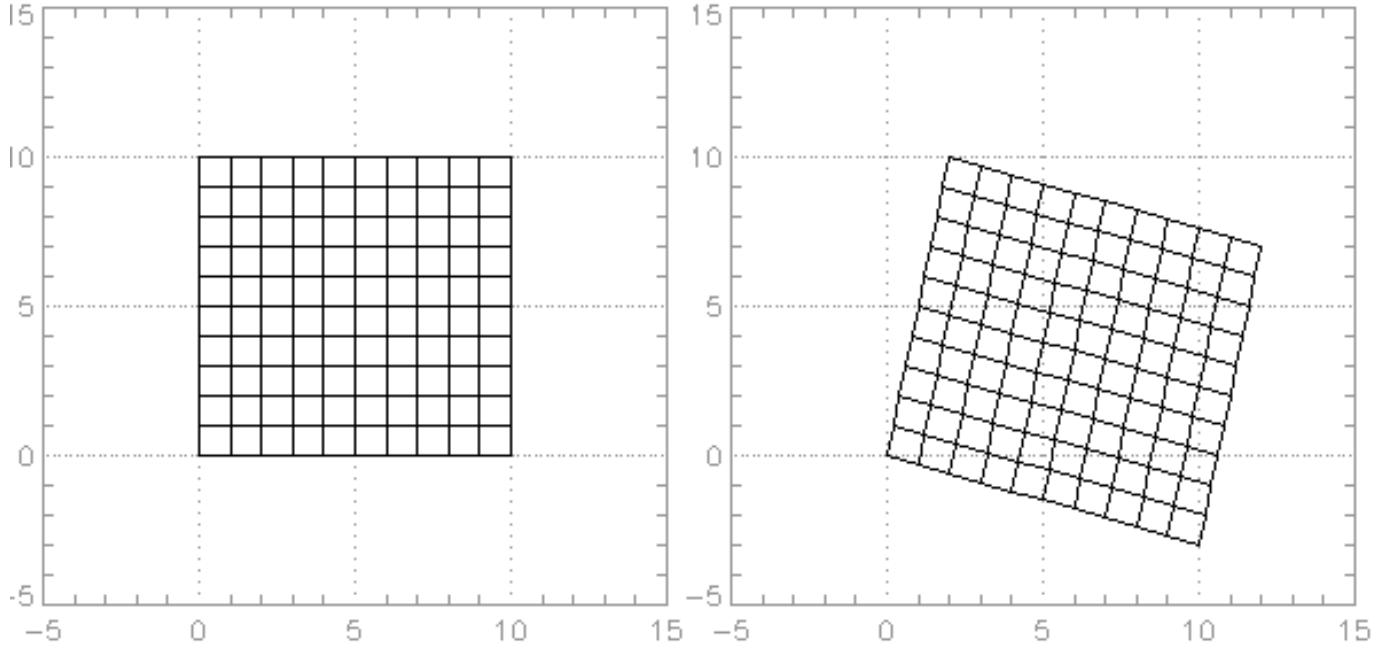


空间变换

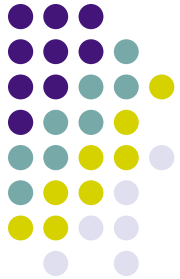
- 倾斜加旋转变换

$$u = x + 0.2y$$

$$v = -0.3x + y$$



空间变换



- 复杂的仿射变换可通过一系列仿射变换操作完成
  - **因为仿射变换的组合还是仿射变换**

- 齐次坐标系（homogeneous coordinates）：
  为了刻画仿射变换的组合，可以通过矩阵操作来简单、统一表达

空间变换

- 齐次坐标系（homogeneous coordinates）

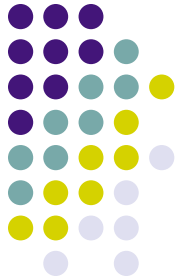
$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

  - 上式可等价的写成矩阵形式，**T是满秩矩阵**

$$\mathbf{q} = \mathbf{T}\mathbf{p}$$

变换矩阵

空间变换



- 基本变换矩阵（重要）

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & x_0 \\ 0 & 1 & y_0 \\ 0 & 0 & 1 \end{bmatrix}$$

(X,Y)轴平移(x0,y0)个单位

$$\mathbf{T} = \begin{bmatrix} s_1 & 0 & 0 \\ 0 & s_2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

(X,Y)轴伸缩(s1,s2)倍

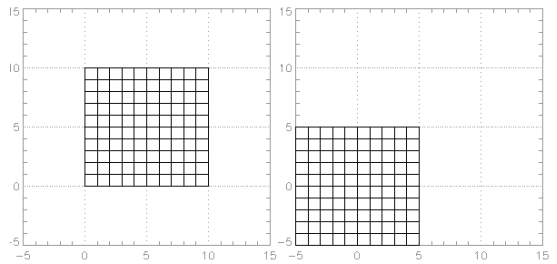
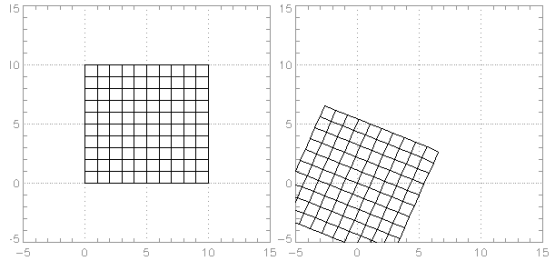
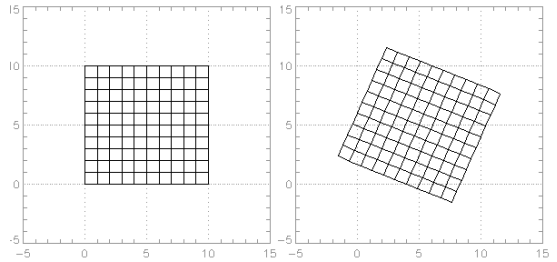
$$\mathbf{T} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

旋转θ 度

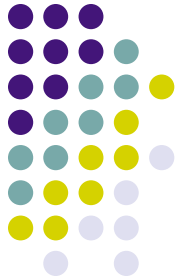
通过以上基本变换矩阵可以组合出一些复杂的变换操作

空间变换

- 组合的变换操作

Operation	Expression	Result
Translate to Origin	$\mathbf{T}_1 = \begin{bmatrix} 1.00 & 0.00 & -5.00 \\ 0.00 & 1.00 & -5.00 \\ 0.00 & 0.00 & 1.00 \end{bmatrix}$	
Rotate by 23 degrees	$\mathbf{T}_2 = \begin{bmatrix} 0.92 & 0.39 & 0.00 \\ -0.39 & 0.92 & 0.00 \\ 0.00 & 0.00 & 1.00 \end{bmatrix}$	
Translate to original location	$\mathbf{T}_3 = \begin{bmatrix} 1.00 & 0.00 & 5.00 \\ 0.00 & 1.00 & 5.00 \\ 0.00 & 0.00 & 1.00 \end{bmatrix}$	

空间变换



- 计算相应的变换矩阵
  - 上例中空间变换经过三个步骤，先经过平移变换T1，然后旋转变换T2，然后是平移变换T3
  - 那么总的变换矩阵为

$$\mathbf{T} = \mathbf{T}_3\mathbf{T}_2\mathbf{T}_1$$

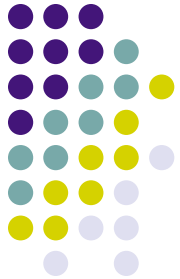
  - 计算一下得到

$$\mathbf{T} = \mathbf{T}_3\mathbf{T}_2\mathbf{T}_1 = \begin{bmatrix} 0.92 & 0.39 & -1.56 \\ -0.39 & 0.92 & 2.35 \\ 0.00 & 0.00 & 1.00 \end{bmatrix}$$

任意组合的仿射变换还是仿射变换

任意组合的仿射变换都可以通过这种方式得到变换矩阵

空间变换



- 逆仿射变换
  - 如果变换后，想把图像复原，怎么操作?
  - 仿射变换是可逆的
  - 逆变换矩阵为

$$\mathbf{T}^{-1} = \mathbf{T}_1^{-1}\mathbf{T}_2^{-1}\mathbf{T}_3^{-1}$$

- 条件：T1、T2、T3是可逆矩阵。
  - 基本变换矩阵都是可逆矩阵

空间变换

- 找到合适的空间变换

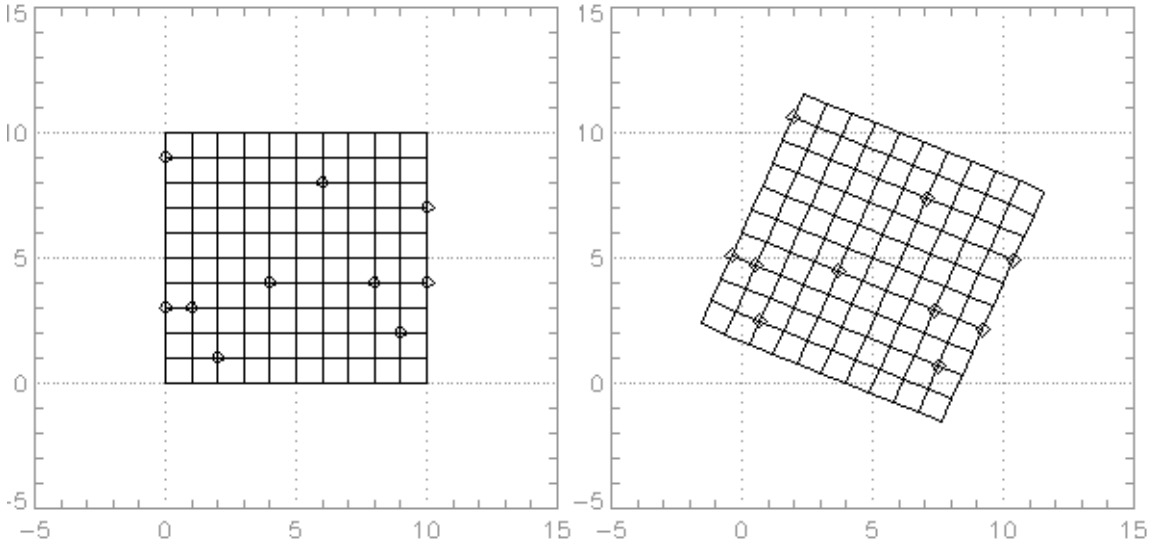
给定图像A和图像B，如何将A、B对齐?

Image A

Image B

空间变换

- 点匹配法（point matching method）



在图像A和图像B中随机的标上n对匹配点

2 空间变换

- 点匹配法（point matching method）

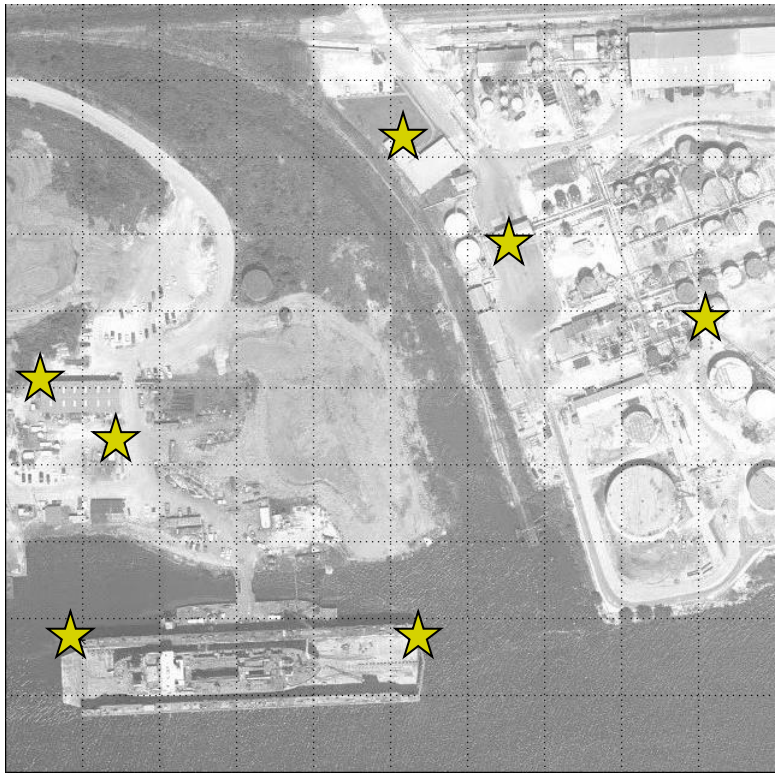


Image A

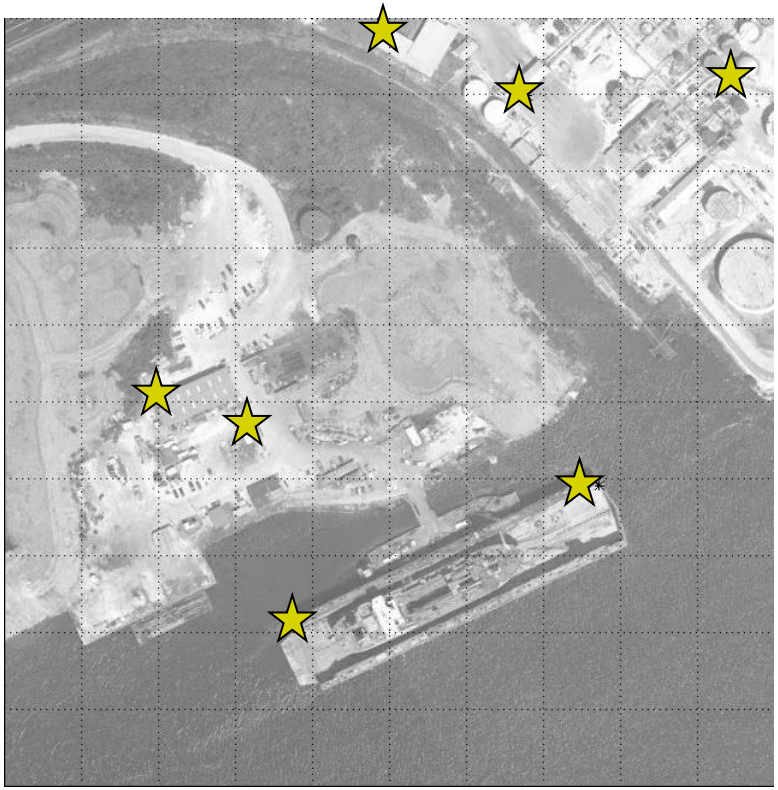
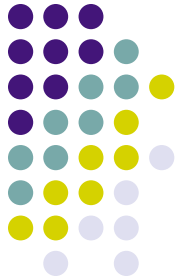


Image B

空间变换



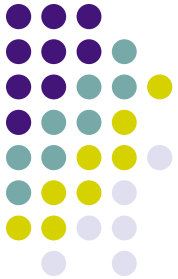
- 点匹配法（point matching method）
  - 假设图像A的n个点为$\{\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_{n-1}\}$

$$\mathbf{P} = \begin{bmatrix} x_0 & x_1 & \ldots & x_{n-1} \\ y_0 & y_1 & \ldots & y_{n-1} \\ 1 & 1 & \ldots & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{p}_0 & \mathbf{p}_1 & \ldots & \mathbf{p}_{n-1} \end{bmatrix}$$

  - 相应地，假设图像B中的n个匹配点为$\{\mathbf{q}_0, \mathbf{q}_1, \ldots, \mathbf{q}_{n-1}\}$

$$\mathbf{Q} = \begin{bmatrix} u_0 & u_1 & \ldots & u_{n-1} \\ v_0 & v_1 & \ldots & v_{n-1} \\ 1 & 1 & \ldots & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{q}_0 & \mathbf{q}_1 & \ldots & \mathbf{q}_{n-1} \end{bmatrix}$$

空间变换



- 点匹配法（point matching method）
  - 将这n个点匹配上的变换矩阵为H，则满足

$$\mathbf{Q} = \mathbf{HP}$$

  - 通过代数计算，H的最小二乘解为：

$$\mathbf{H} = \mathbf{QP}^T(\mathbf{PP}^T)^{-1} = \mathbf{QP}^\dagger$$

    - 这里 $\mathbf{P}^\dagger = \mathbf{P}^T(\mathbf{PP}^T)^{-1}$ 指矩阵P的伪逆

空间变换

- 点匹配法（point matching method）
  - 上例中，我们得到了变换矩阵为

$$\mathbf{H} = \mathbf{Q}\mathbf{P}^{\dagger} = \begin{bmatrix} 0.92 & -0.39 & 224.17 \\ 0.39 & 0.92 & 10.93 \\ 0.00 & -0.00 & 1.00 \end{bmatrix}$$

  - 考察匹配点情况：

Table of Matching Points

X_a	Y_a	X_b	Y_b	X'_a	Y'_a
30.5	325.3	125.8	322.5	126.0	322.8
86.8	271.3	199.3	295.3	198.7	294.9
330.3	534.0	320.0	632.0	320.5	632.2
62.0	110.3	238.0	137.0	238.4	136.8
342.0	115.0	494.0	250.0	493.9	250.4
412.0	437.0	434.3	574.8	433.3	574.7
584.5	384.8	611.8	594.0	612.2	593.8

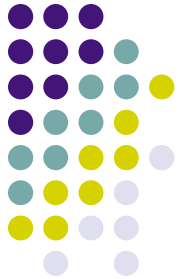
空间变换

- 点匹配法（point matching method）
  - 效果图

Mapped A

Original B

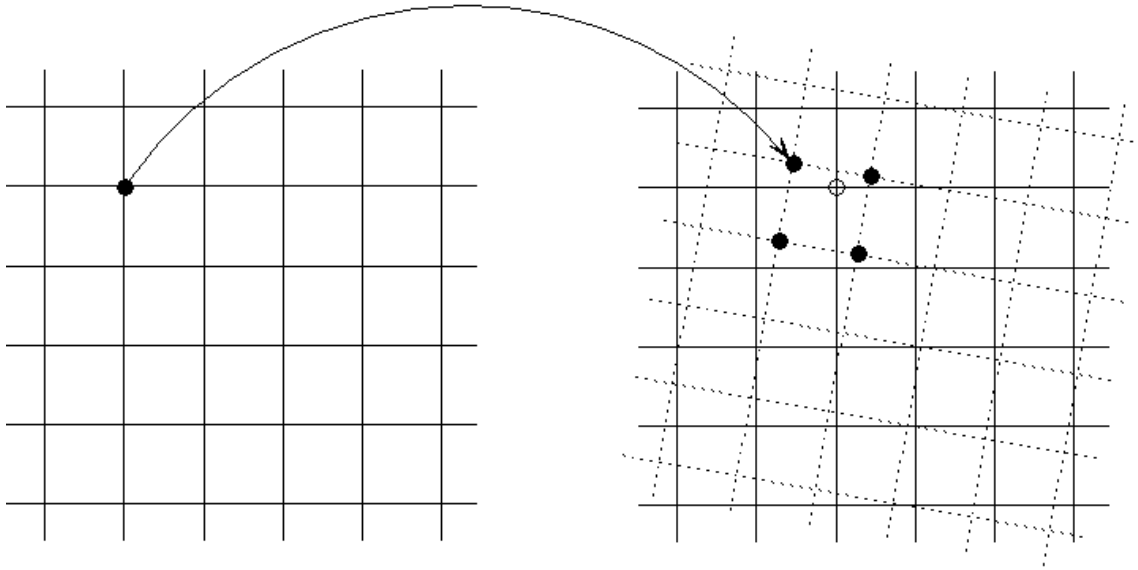
几何变换



- 引言
- 空间变换
- 灰度级插值

灰度级插值

- 灰度级插值是必要的
  - 变换后，原图像的网格点未必落入网格点



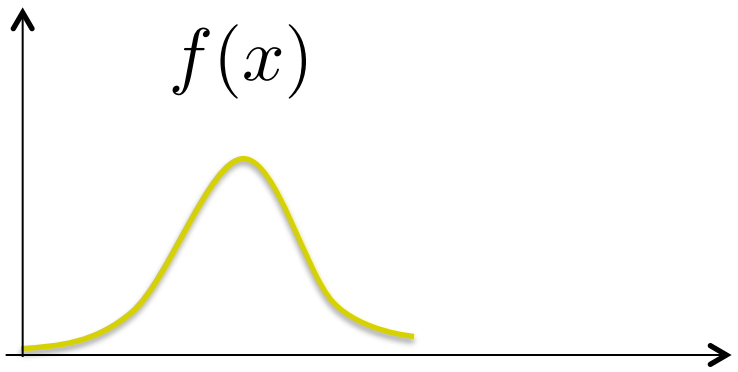
  - 为了得到变换图像网格点的灰度级，我们需要进行灰度级插值

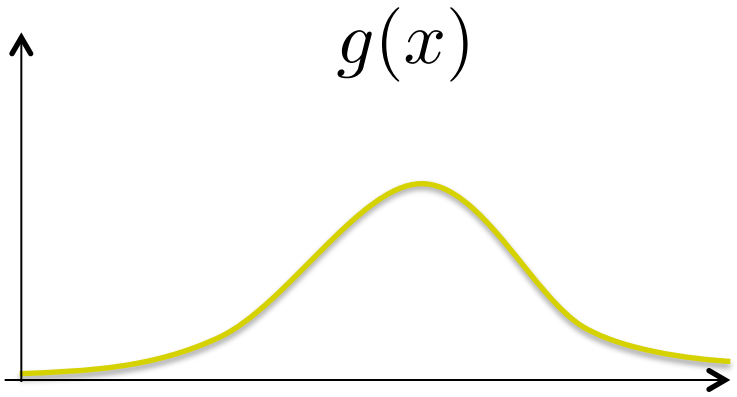
引言

函数图像的放大缩小

放大一倍：

$$g(x) = f(\frac{1}{2}x) \qquad g(2x) = f(x)$$

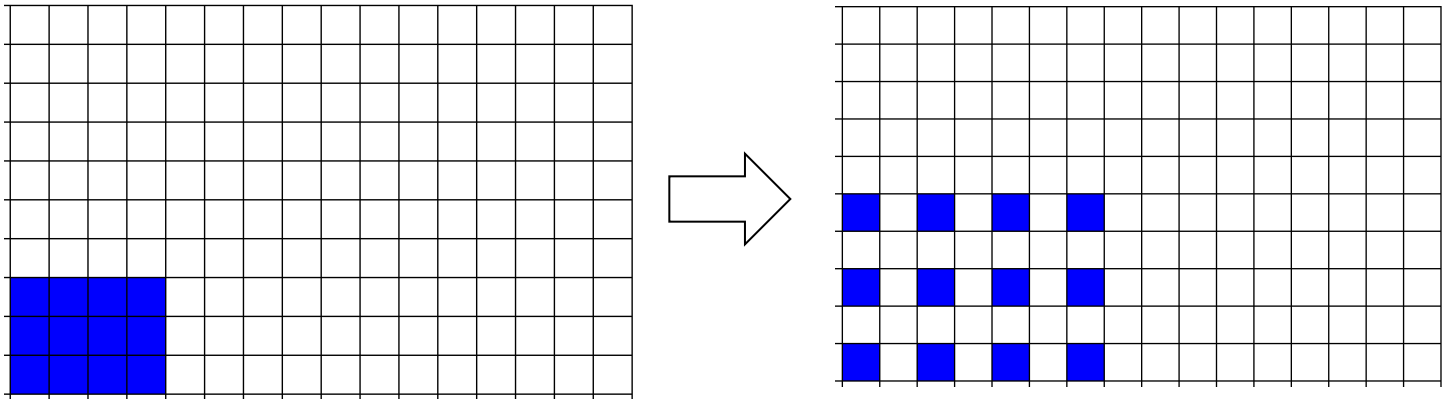




引言

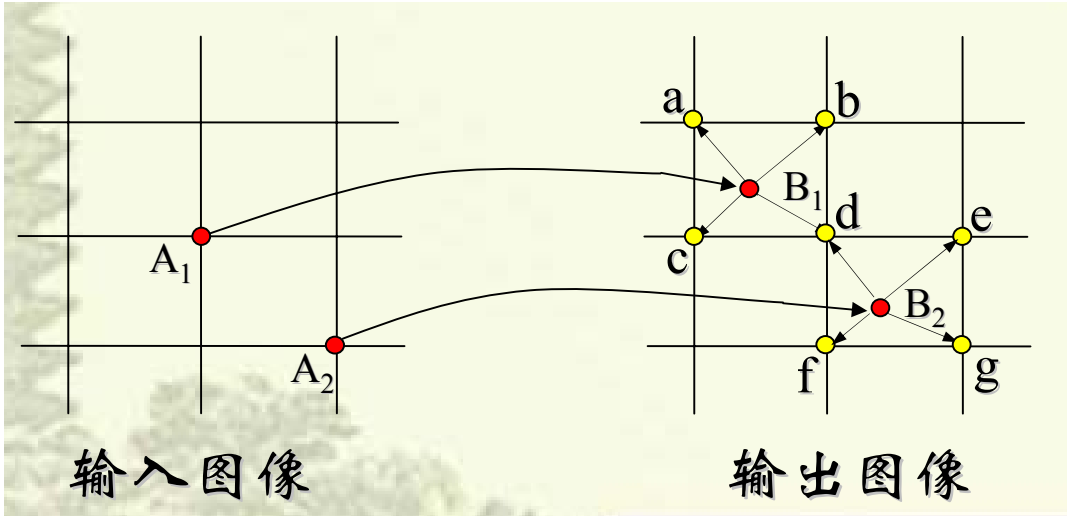
图像的放大缩小

$$g[2x, 2y] = f[x, y]$$



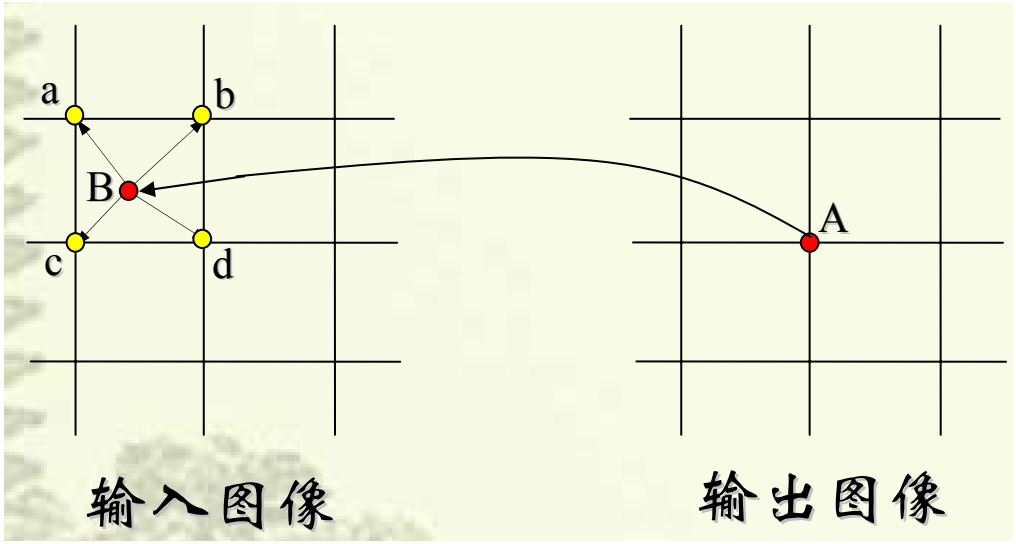
灰度级插值

- 向前映射法
  - 通过输入图像像素位置，计算输出图像对应的像素位置
  - 将该位置像素的灰度值分配给其相邻四个网格位置



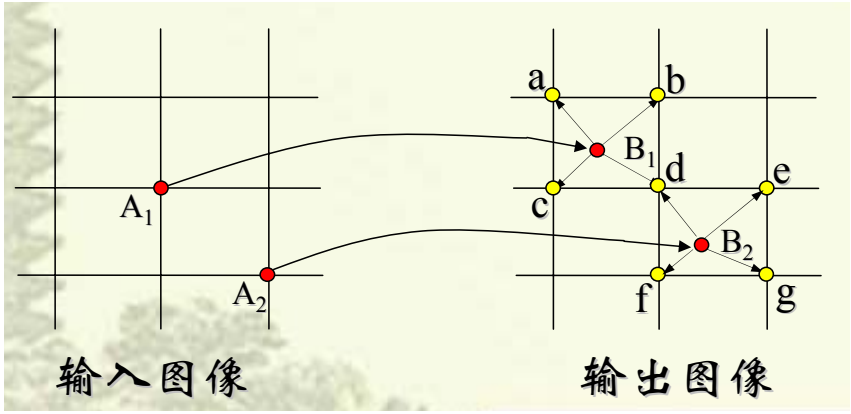
灰度级插值

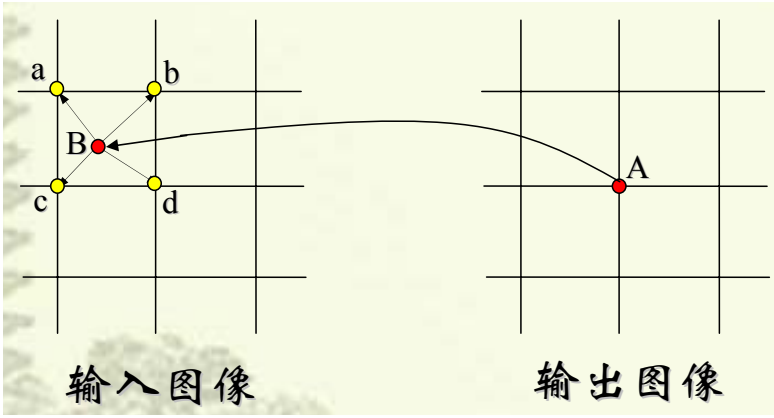
- 向后映射法
  - 通过输出图像像素位置, 计算输入图像中涉及到所有对应的像素位置;
  - 根据输入图像相邻四个像素的灰度值计算该位置像素的灰度值.



比较

- 向前映射、向后映射哪个更实用?





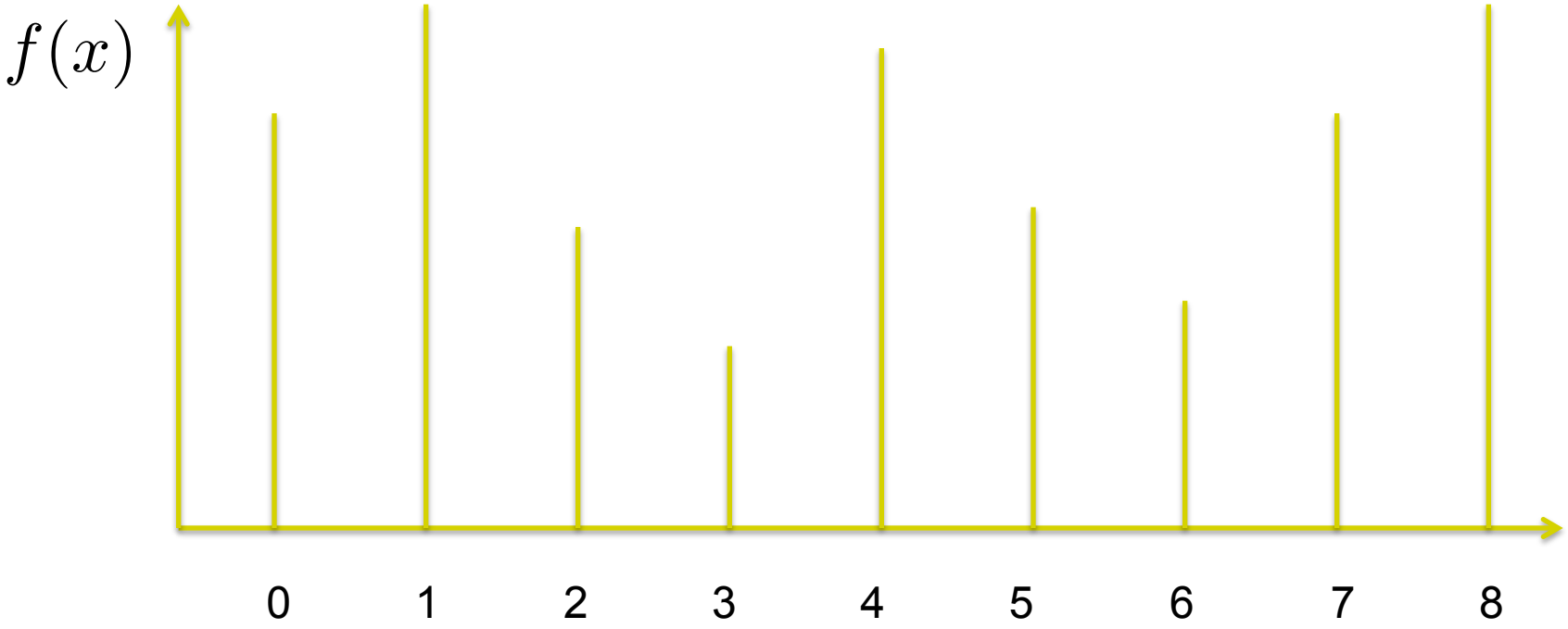
灰度级插值

- 两种映射方法的对比
  - 对于向前映射：每个输出图像的灰度要经过多次运算；
  - 对于向后映射：每个输出图像的灰度只要经过一次运算。

实际应用中，更经常采用向后映射法。

插值

$$f(0) = 10$$
$$f(1) = 12$$
$$f(0.5) = ?$$

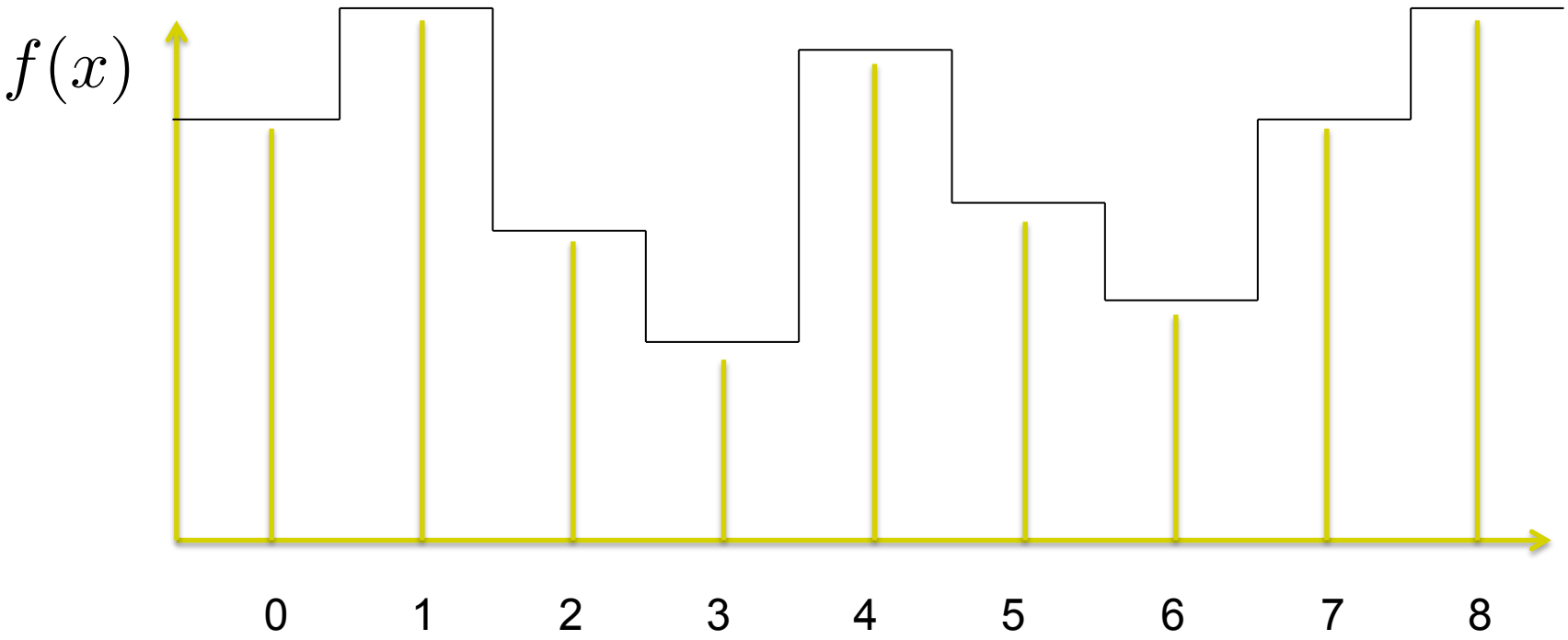


最近邻插值

$$f(0) = 10$$

$$f(1) = 12$$

$$f(0.4) = f(0), f(0.6) = f(1), f(0.5) = \ldots$$

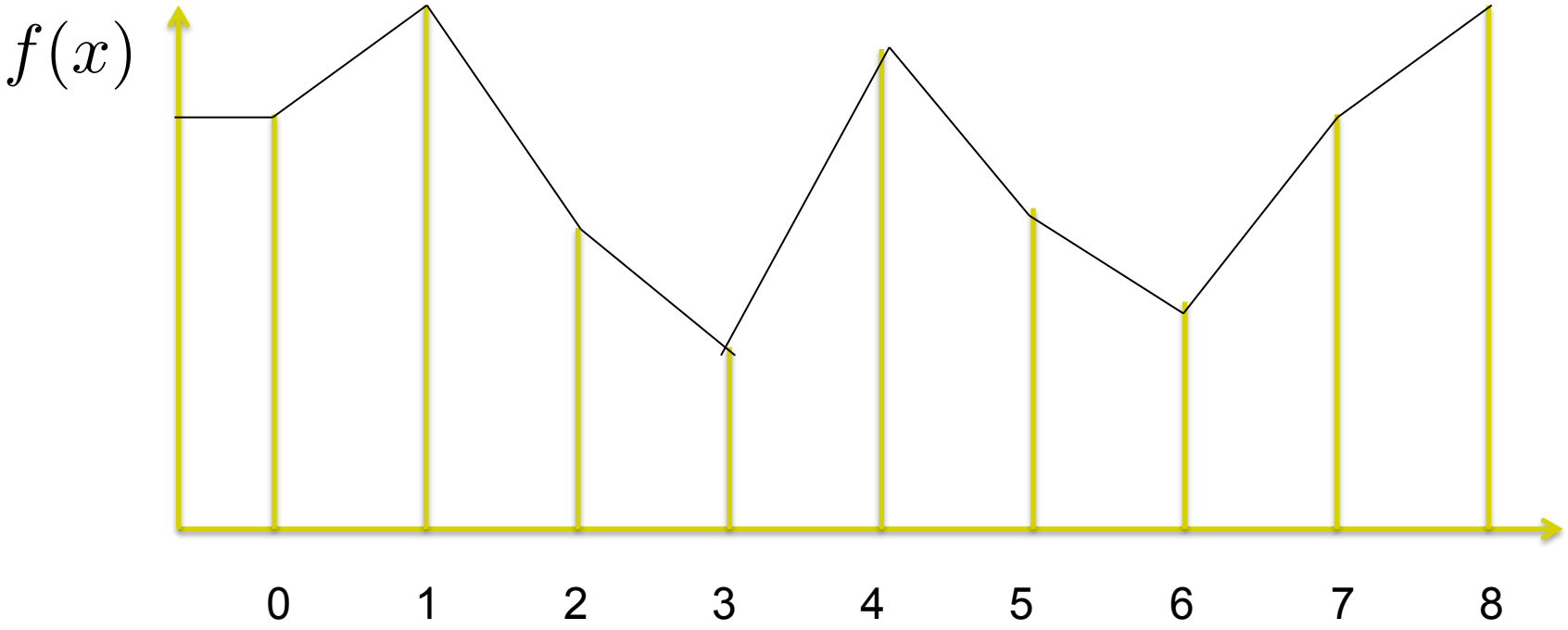


线性插值

$$f(0) = 10, f(1) = 12$$

解出[0,1]： $f(x) = 10 + x * 2$

$$f(0.4) = 10.8$$

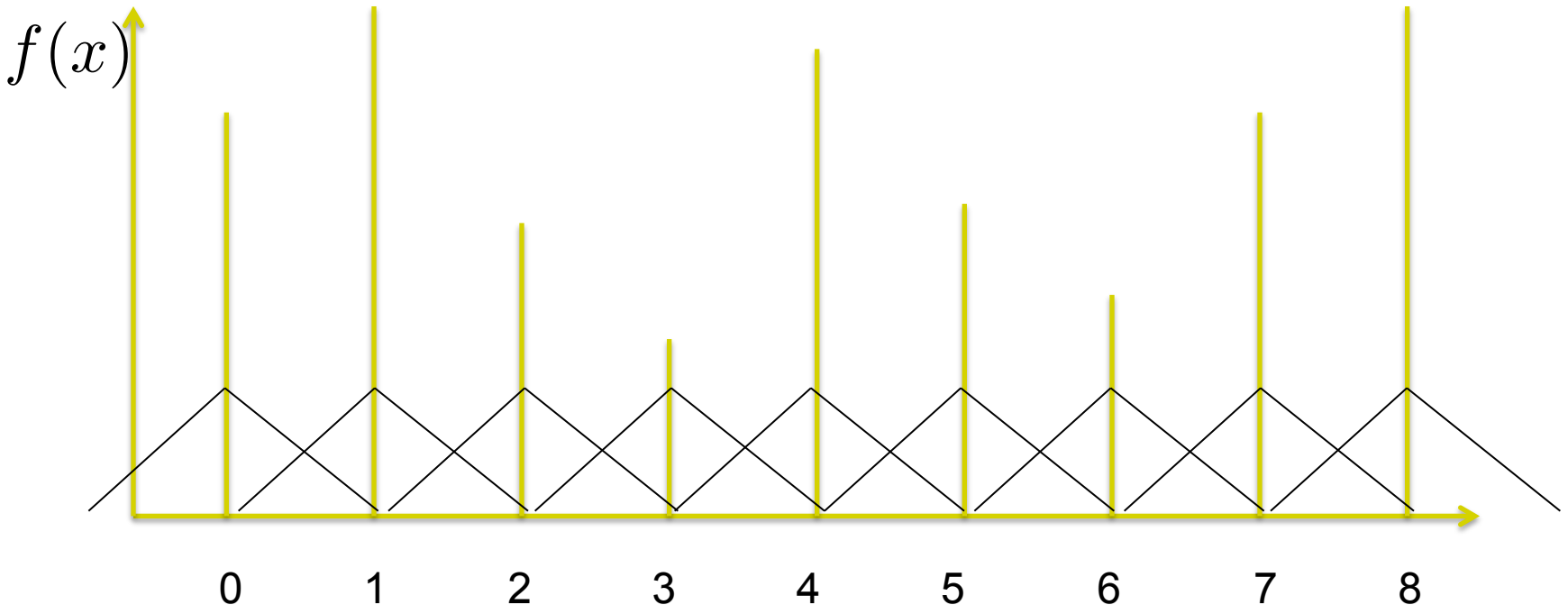


线性插值

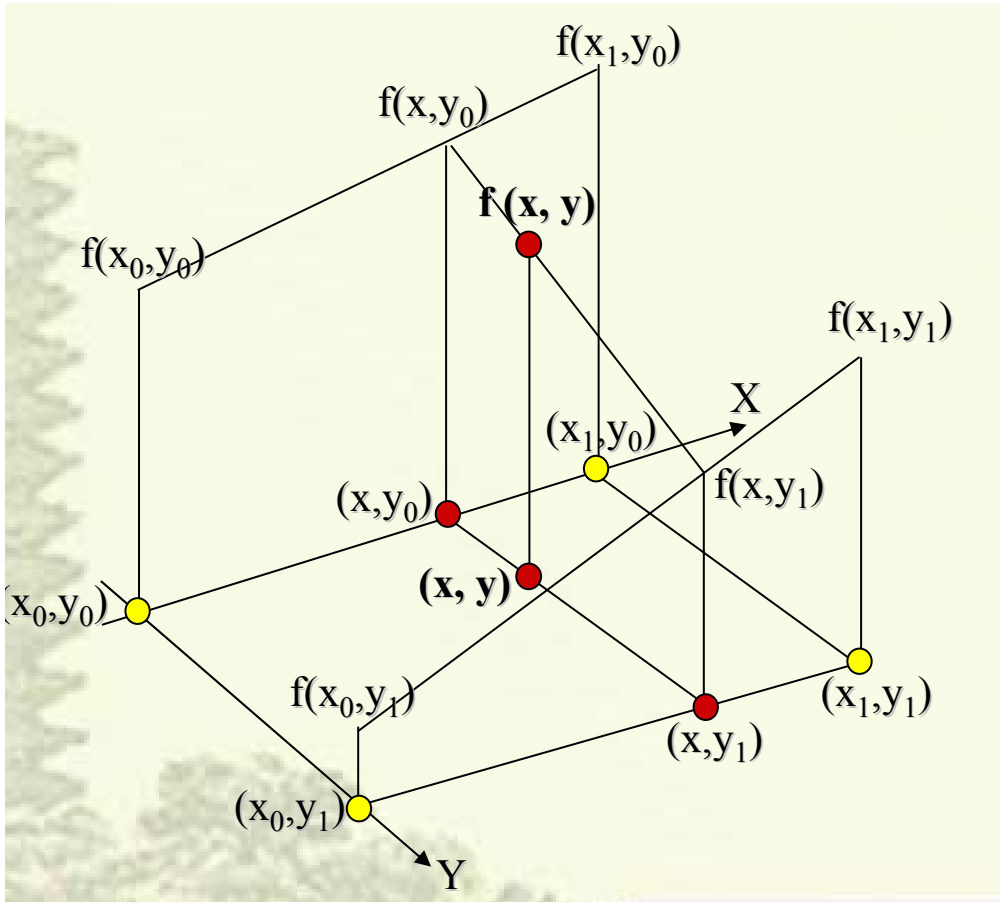
线性插值 $f(0) = 10, f(1) = 12$

另一种写法：$f(x) = (1 - x) * f(0) + (x - 0) * f(1)$

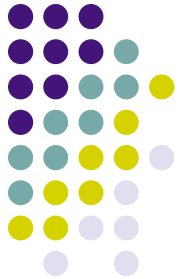
$$f(0.4) = 0.6 * 10 + 0.4 * 12 = 6 + 4.8 = 10.8$$



双线性插值


f(x1,y0)
f(x,y0)
f(x, y)
f(x0,y0)
f(x1,y1)
X
(x1,y0)
(x,y0)
f(x,y1)
(x, y)
(x0,y0)
f(x0,y1)
(x1,y1)
(x,y1)
(x0,y1)
Y

双线性插值



$setp1$：定义双线性方程$f(x,y)=ax+by+cxy+d$

$step2$：代入已知四点的值

$$\begin{cases} f(0,0)=d \\ f(1,0)=a+d \\ f(0,1)=b+d \\ f(1,1)=a+b+c+d \end{cases}$$

$$step3: f(x,y)=[f(1,0)-f(0,0)]x+[f(0,1)-f(0,0)]y$$

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

双线性插值

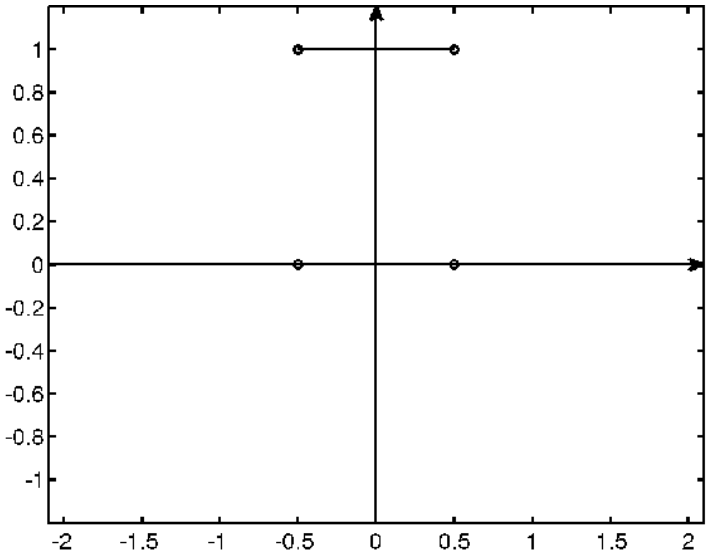
- 双线性插值的第二种算法

$$f(x,0) = f(0,0) + x\left[f(1,0) - f(0,0)\right]$$
$$f(x,1) = f(0,1) + x\left[f(1,1) - f(0,1)\right]$$
$$f(x,y) = f(x,0) + y\left[f(x,1) - f(x,0)\right]$$

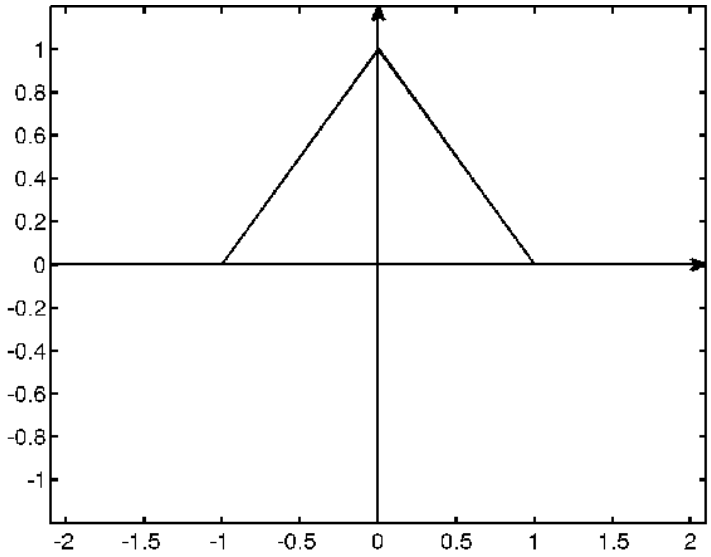
B样条插值

$$N_{i,0}(u) = \begin{cases} 1 & \text{if } u_i \le u < u_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

$$N_{i,p}(u) = \frac{u - u_i}{u_{i+p} - u_i} N_{i,p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(u)$$



p=0

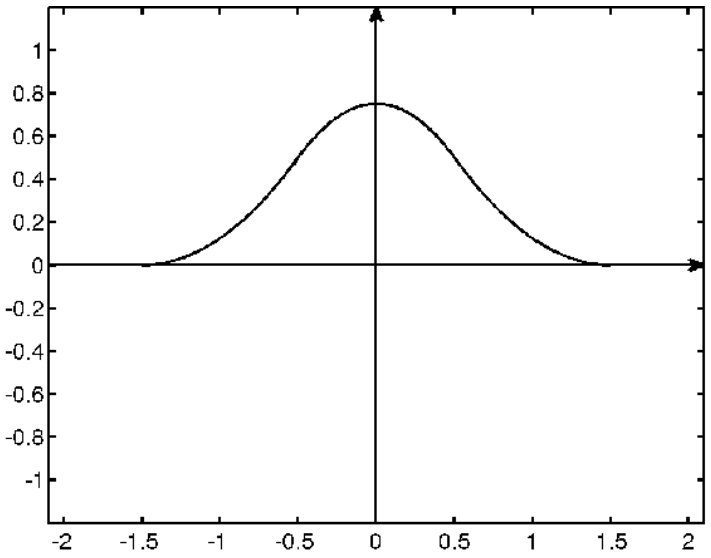


p=1

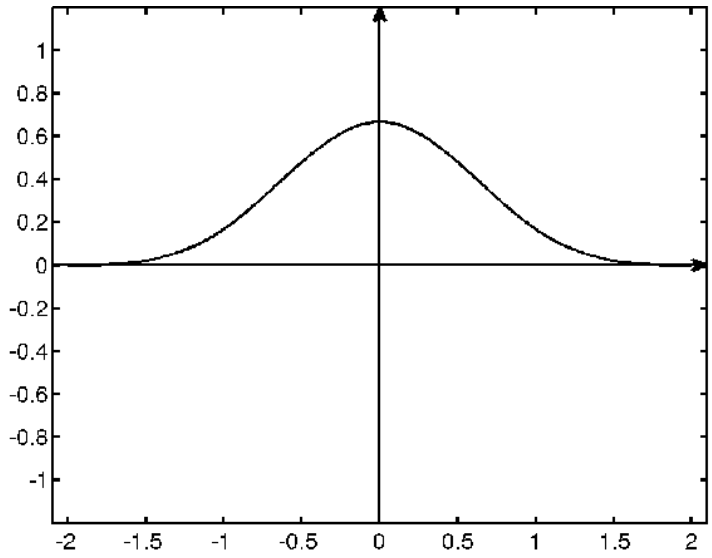
B样条插值

$$N_{i,0}(u) = \begin{cases} 1 & \text{if } u_i \le u < u_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

$$N_{i,p}(u) = \frac{u - u_i}{u_{i+p} - u_i} N_{i,p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(u)$$



p=2



p=3

灰度级插值

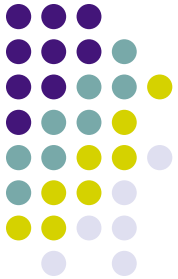
- 用最近邻插值和双线性插值的方法分别将老虎放大1.5倍。

灰度级插值

采用最近邻插值放大1.5倍

采用双线性插值放大1.5倍

完整的几何变换过程（例子）

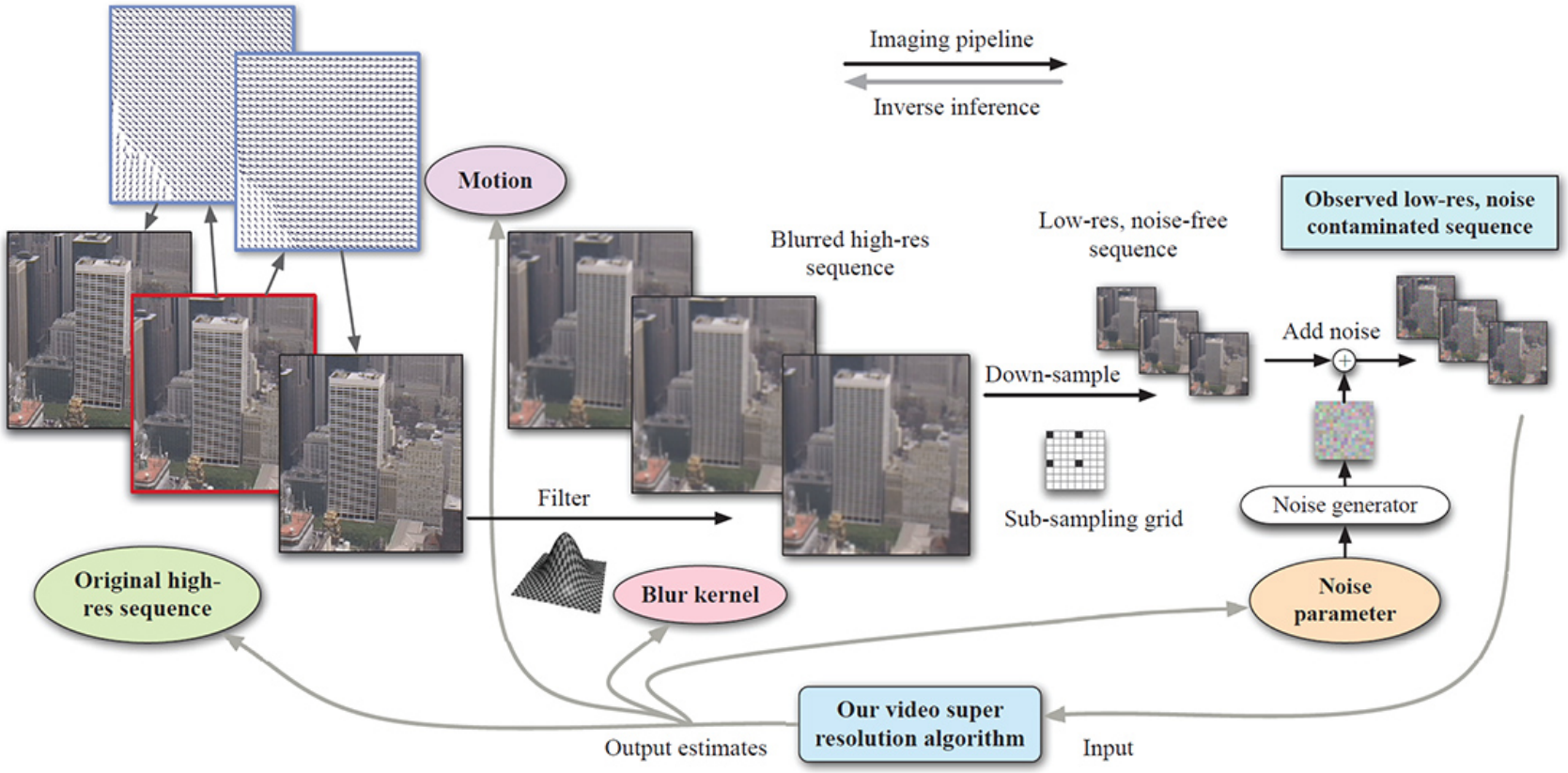


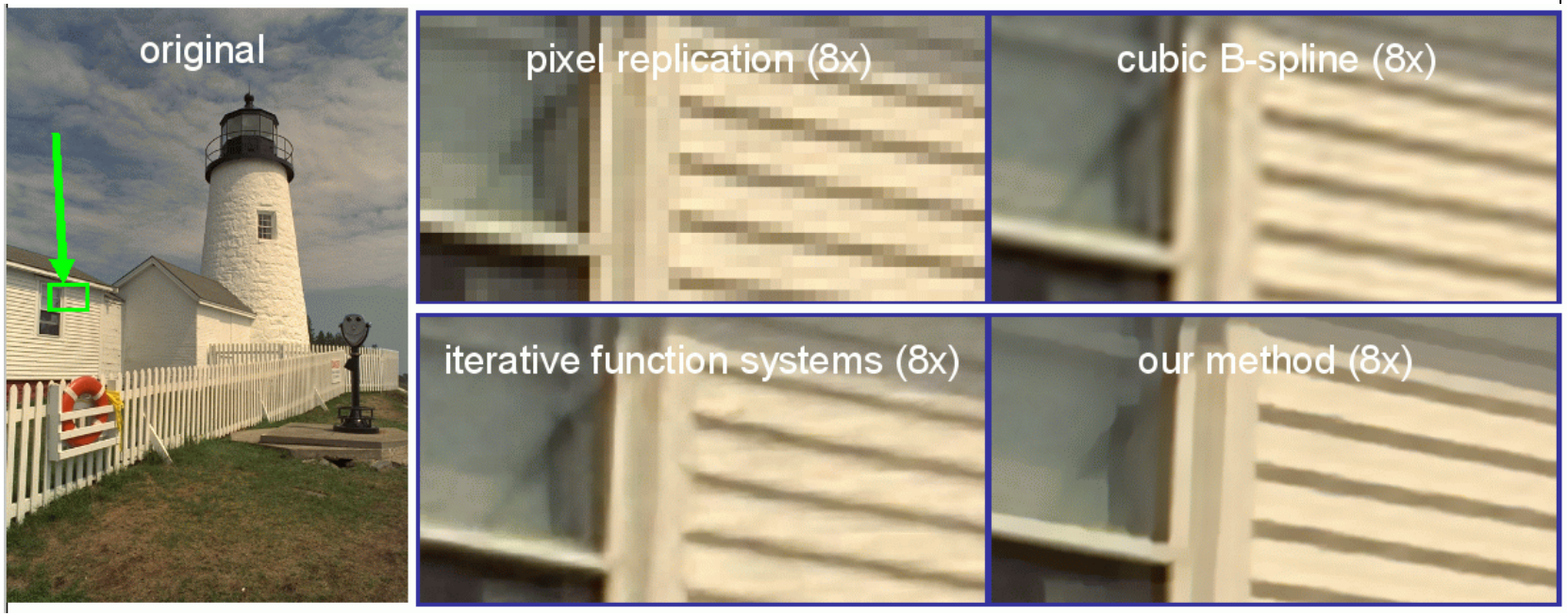
- 比例变换中对应图像的确定
  - 假设输出图像的宽度为W，高度为H；
  - 输入图像的宽度为w高度为h，要将输入图像的尺度拉伸或压缩变换至输出图像的尺度；
  - 按照线性插值的方法，将输入图像的宽度方向分为W等份，高度方向分为H等份；
  - 那么输出图像中任意一点（x，y）的灰度值就应该由输入图像中四点（a，b）、（a+1，b）、（a，b+1）和（a+1，b+1）的灰度值来确定。其中a和b的值分别为：

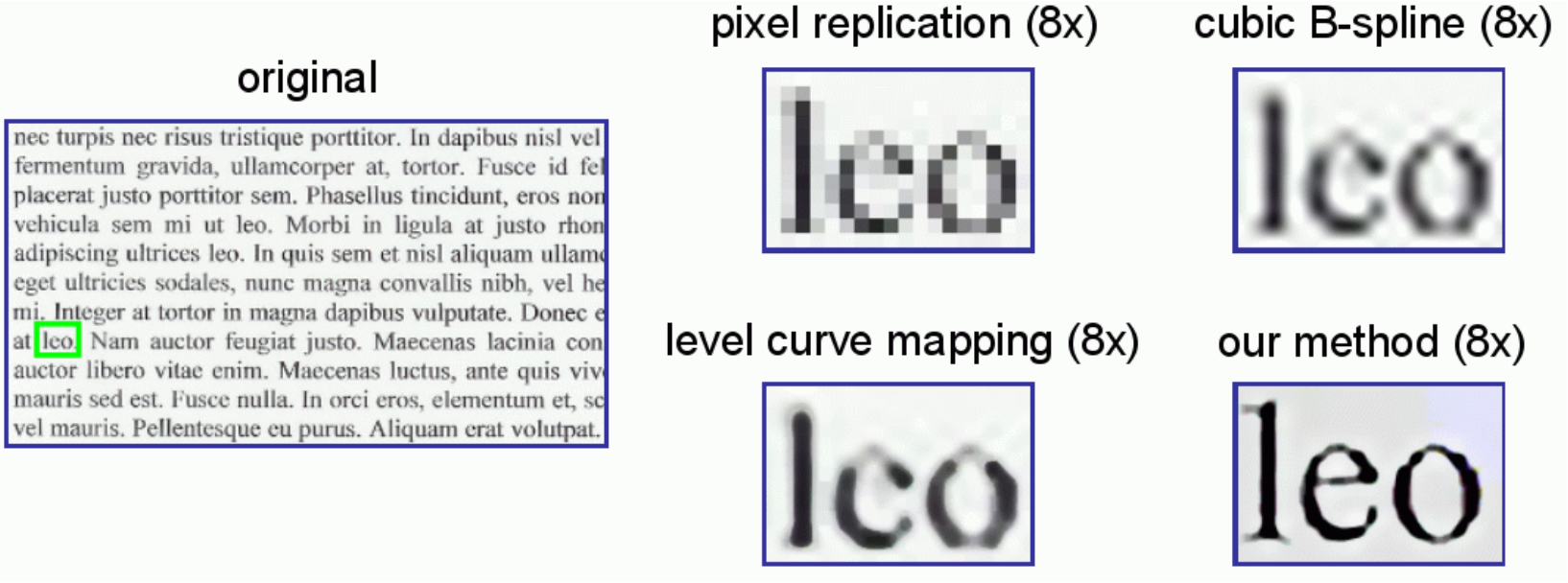
$$a = \left\lfloor x \times \frac{w}{W} \right\rfloor \qquad b = \left\lfloor y \times \frac{h}{H} \right\rfloor$$

Liu & Sun. A Bayesian Approach to Adaptive Video Super Resolution. CVPR'11

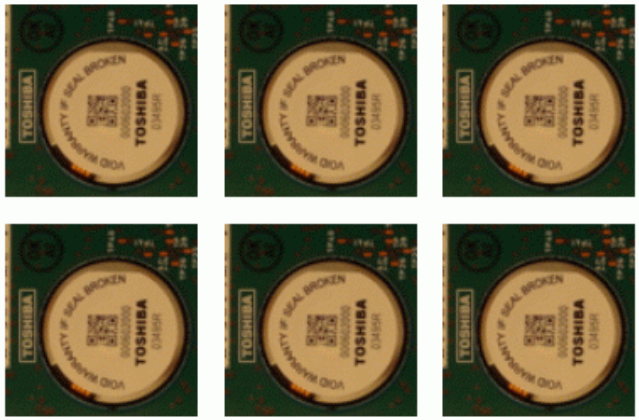
Liu & Sun. On Bayesian Adaptive Video Super Resolution. TPAMI'14



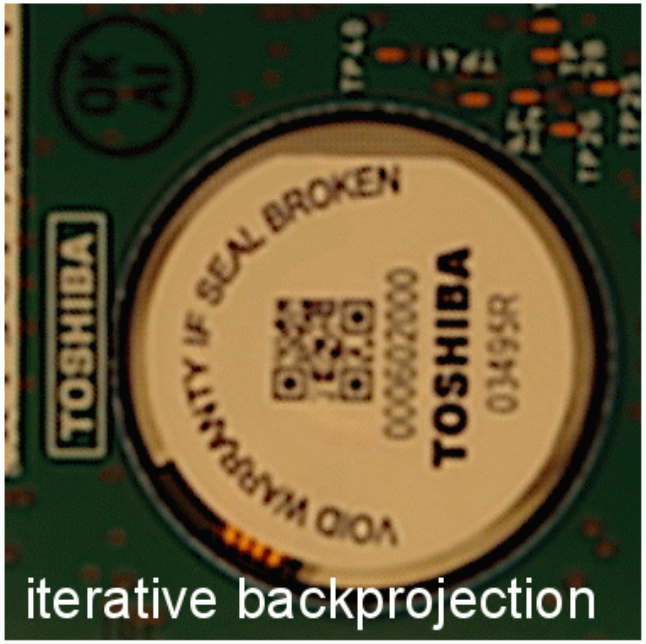

original
pixel replication (8x)
cubic B-spline (8x)
iterative function systems (8x)
our method (8x)

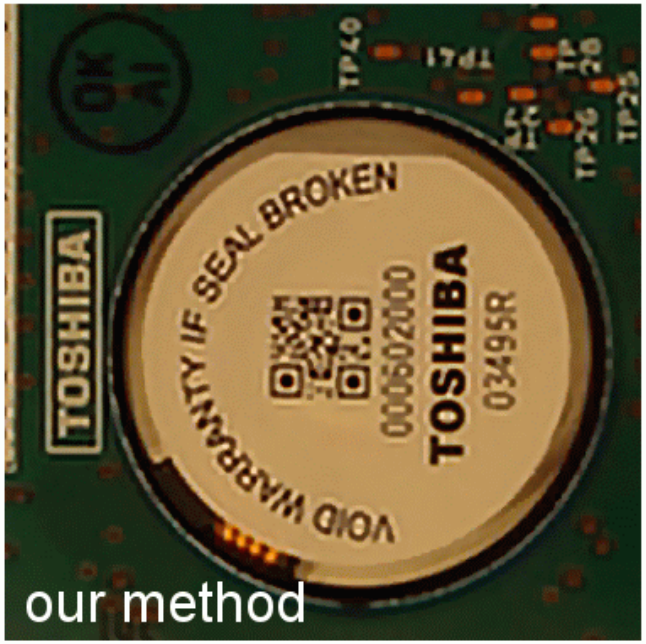

original
nec turpis nec risus tristique porttitor. In dapibus nisl vel
fermentum gravida, ullamcorper at, tortor. Fusce id fel
placerat justo porttitor sem. Phasellus tincidunt, eros non
vehicula sem mi ut leo. Morbi in ligula at justo rhon
adipiscing ultrices leo. In quis sem et nisl aliquam ullam
eget ultricies sodales, nunc magna convallis nibh, vel he
mi. Integer at tortor in magna dapibus vulputate. Donec e
at leo. Nam auctor feugiat justo. Maecenas lacinia con
auctor libero vitae enim. Maecenas luctus, ante quis viv
mauris sed est. Fusce nulla. In orci eros, elementum et, sc
vel mauris. Pellentesque eu purus. Aliquam erat volutpat.
pixel replication (8x)
cubic B-spline (8x)
level curve mapping (8x)
our method (8x)

original



…20 images (captured with Sony DSC-P120)



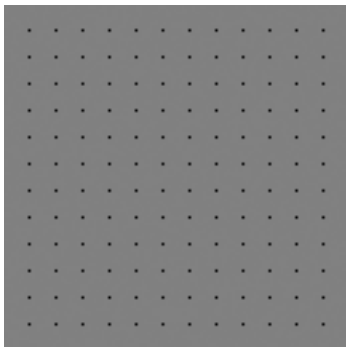



Bicubic x4
Our algorithm (x4)

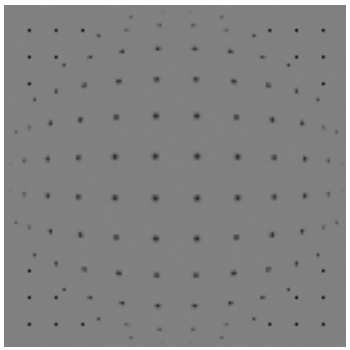
几何变换的用途

几何校正和多项式卷绕

- 几何校正（calibration）:由于镜头的几何变形，使用矩形栅格校准。
- 多项式卷绕：多项式的项数与控制点数相同，解线性方程组，得系数后矩阵求逆。



测试靶



对应的鱼眼图像

几何变换的用途

变形后的老虎

校正后的老虎

几何变换的用途

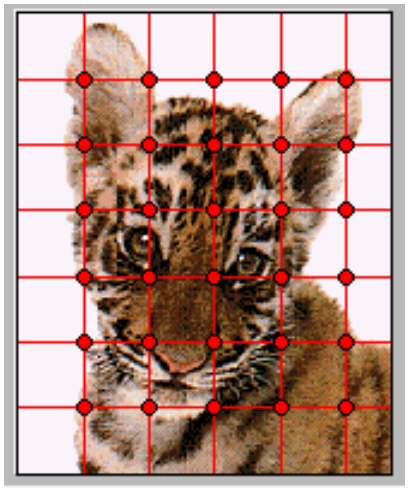
- 多项式卷绕的基本原理
- 对于一个有10个点的测试靶，可以设计一个二维三阶函数拟合。

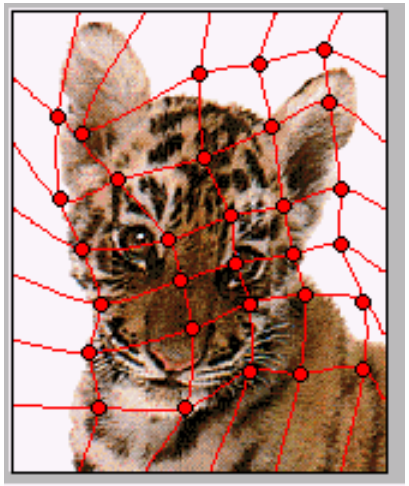
$$p(x,y) = c_0 + c_1 x + c_2 y + c_3 xy + c_4 x^2$$
$$+ c_5 y^2 + c_6 x^2 y + c_7 x y^2 + c_8 x^3 + c_9 y^3$$

$$\begin{bmatrix} p(x_1,y_1) \\ p(x_2,y_2) \\ \vdots \\ p(x_{10},y_{10}) \end{bmatrix} = \begin{bmatrix} 1 & x_1 & \cdots & y_1^3 \\ 1 & x_2 & \cdots & y_2^3 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_{10} & \cdots & y_{10}^3 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_9 \end{bmatrix}$$

几何变换的用途

- 控制栅格插值和图像卷绕
  - 控制栅格插值：将图像分成小块进行卷绕变换。
  - 常用双线性插值。





几何变换的用途

- 效果图

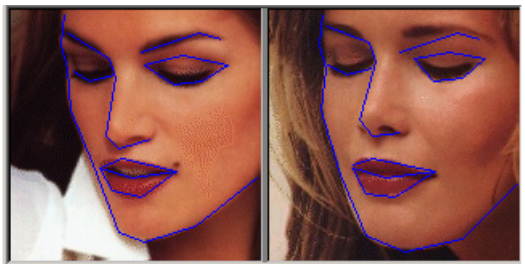
*Method*1: 双线性空间变换的表达式

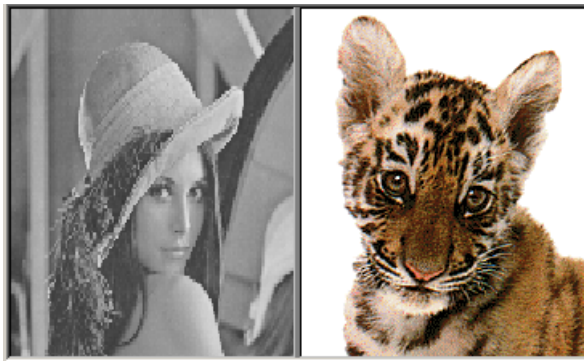
$$G(x,y) = F(ax + by + cxy + d, ex + fy + gxy + h)$$

解4个点8个方程式可得。

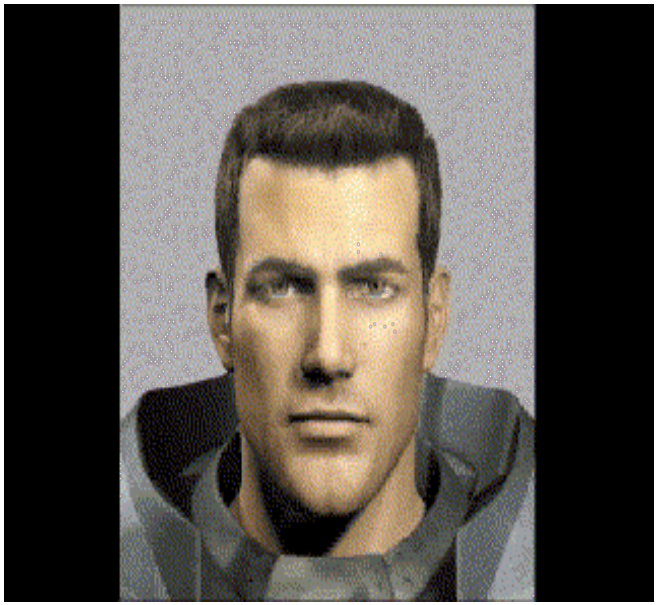
图像变形

http://
www.morpheussoftware.net/





图像变形



下一讲