
华北电力大学

NORTH CHINA ELECTRIC POWER UNIVERSITY

# 2 数字图像基本运算

# 主要内容

- 点运算
  - ✓ 线性变换
  - ✓ 对数变换
  - ✓ 伽马变换
  - ✓ 灰度阈值变换
- 代数运算
  - ✓ 图像加法
  - ✓ 图像减法
  - ✓ 图像乘法
  - ✓ 图像除法

# 主要内容

- 几何运算
  - ✓ 齐次坐标
  - ✓ 图像平移
  - ✓ 图像镜像
  - ✓ 图像旋转
  - ✓ 图像缩放
- 插值运算
  - ✓ 最近邻插值
  - ✓ 线性（双线性）插值

# 2.3 几何运算

- 图像几何运算（Geometric Operation）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。

$$g(x,y) = f(\mathrm{a}(x,y), \mathrm{b}(x,y))$$

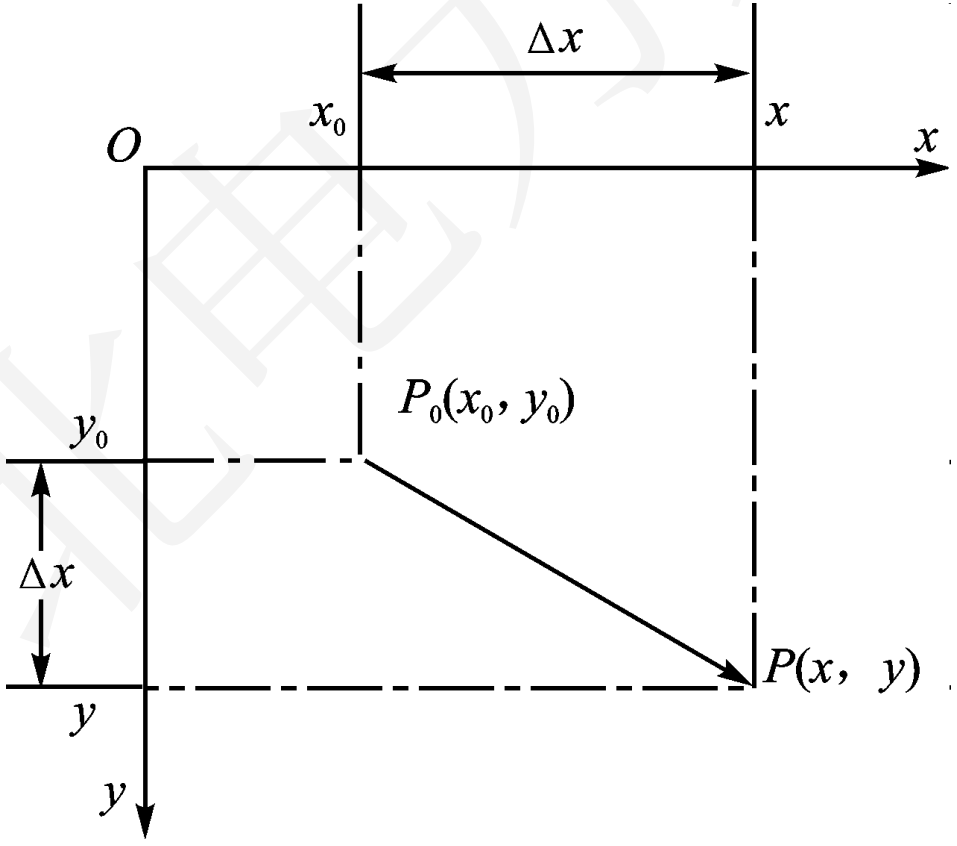
✓ a(x,y)和b(x,y)表示空间变换。

# 2.3.1 几何变换基础

- 图像的几何变换是通过改变图像中物体（像素）之间的空间关系的过程。
- 一个图像的几何变换一般包括两个过程：
  - ✓需要一个算法来定义空间变换本身，描述每个像素如何从其初始位置“移动”到终止位置。
  - ✓需要一个用于灰度插值的算法，在一般情况下，输入图像的位置坐标(x，y)为整数，而输出图像的位置坐标为非整数，反过来也是如此。

# 2.3.1.1 齐次坐标

设点$P_0(x_0, y_0)$进行平移后，移到P(x,y)，其中x方向的平移量为Δx，y方向的平移量为Δy。



# 2.3.1.1 齐次坐标

那么，点P(x,y)的坐标为

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

这个变换用矩阵的形式可以表示为

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

对上式进一步变换，可得：

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

# 2.3.1.1 齐次坐标

增加附加坐标后，可用三维坐标(x,y,1)表示二维空间点(x,y)，则实现平移变换结果如下：

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

对上式的各个矩阵进行定义：

$$T = \begin{bmatrix} 1 & 0 & \Delta x \\ 0 & 1 & \Delta y \\ 0 & 0 & 1 \end{bmatrix}$$ 为变换矩阵

# 2.3.1.1 齐次坐标

$$P=\begin{bmatrix}x\\y\\1\end{bmatrix}$$ 为变换后的坐标矩阵

$$P_0=\begin{bmatrix}x_0\\y_0\\1\end{bmatrix}$$ 为变换前的坐标矩阵

则有 $P=T\cdot P_0$

这种用n＋1维向量表示n维向量的方法称为齐次坐标表示法。

# 2.3.1.2 齐次坐标的一般形式及意义

齐次坐标的一般表示形式为：

$$P=\begin{bmatrix} Hx \\ Hy \\ H \end{bmatrix}$$

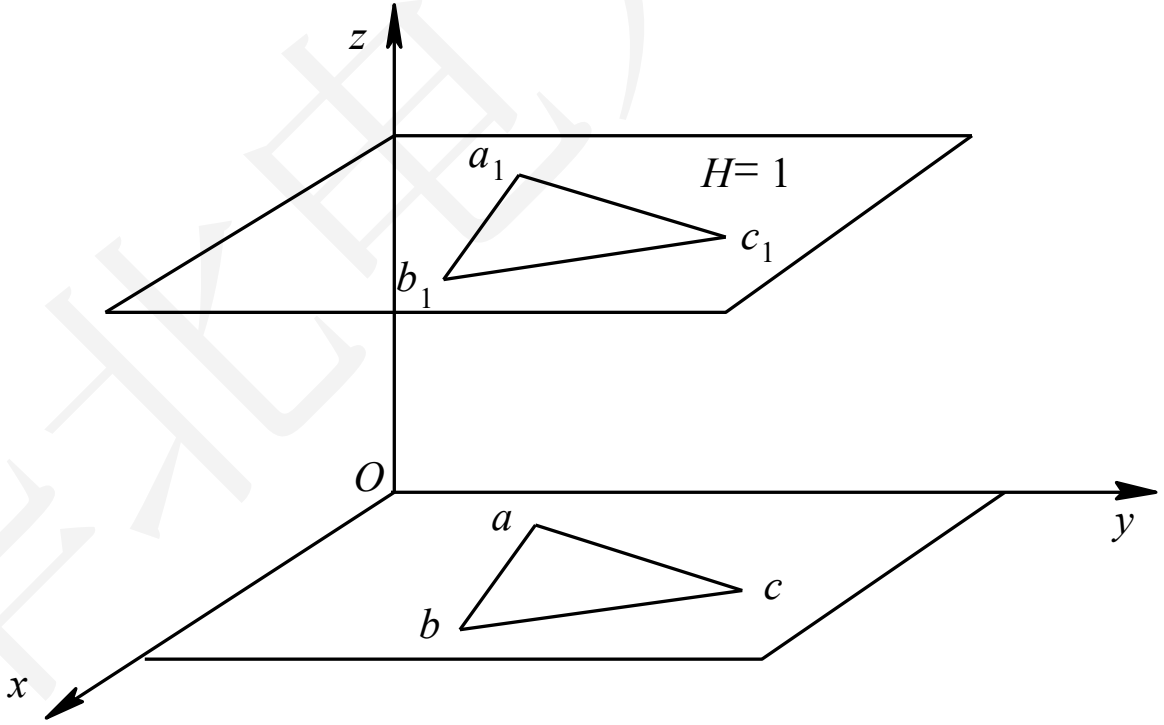
式中，H为非零的实数。当H=1，$P=\begin{bmatrix} x & y & 1 \end{bmatrix}^T$ 称规范化齐次坐标。

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

$$x=\frac{Hx}{H} \qquad y=\frac{Hy}{H}$$

# 2.3.1.2 齐次坐标的一般形式及意义

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。



# 2.3.1.3 二维图像的几何变换矩阵

对于一副二维图像，对P，T，P0进行重新定义。

$$P=\begin{bmatrix} x_1 & x_2 & \cdots & x_n \\ y_1 & y_2 & \cdots & y_n \\ 1 & 1 & \cdots & 1 \end{bmatrix} \qquad T=\begin{bmatrix} a & b & p \\ c & d & q \\ l & m & s \end{bmatrix}$$

$$\begin{bmatrix} x_1 & x_2 & \cdots & x_n \\ y_1 & y_2 & \cdots & y_n \\ 1 & 1 & \cdots & 1 \end{bmatrix} = T. \begin{bmatrix} x_{01} & x_{02} & \cdots & x_{0n} \\ y_{01} & y_{02} & \cdots & y_{0n} \\ 1 & 1 & \cdots & 1 \end{bmatrix}$$

# 2.3.1.3 二维图像的几何变换矩阵

3×3阶矩阵T可以分成四个子矩阵。 $T=\begin{bmatrix} a & b & p \\ c & d & q \\ l & m & s \end{bmatrix}$

- ✓ $\begin{bmatrix} a & b \\ c & d \end{bmatrix}_{2\times2}$ 矩阵可使图像实现恒等、 比例、 镜像、错切和旋转变换。
- ✓ $[p\ q]^T$这一列矩阵可以使图像实现平移变换。
- ✓ [l m]这一行矩阵可以使图像实现透视变换，但当l=0，m=0时它无透视作用。
- ✓ [s]这一元素可以使图像实现全比例变换。

# 2.3.2 图像平移

✓ 平移变换是将一幅图像上的所有点都按照给定的偏移量在水平方向沿x轴、垂直方向沿y轴移动。

✓ 利用齐次坐标，平移变换的矩阵表示为

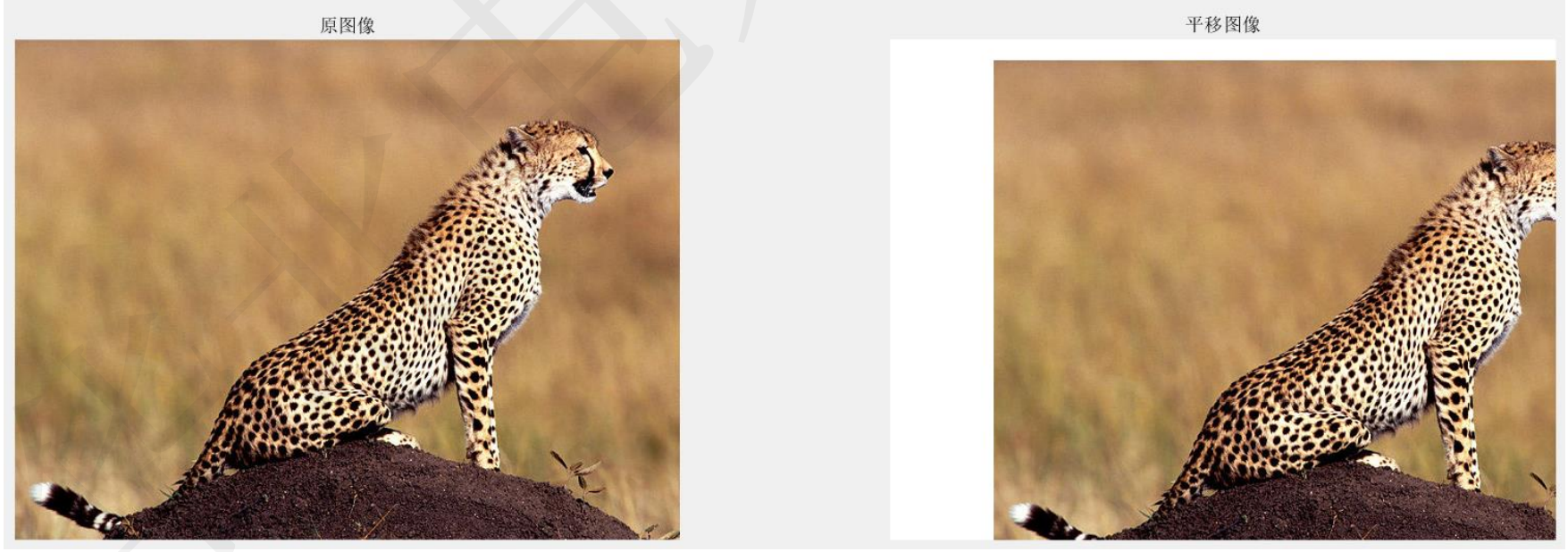
$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & \Delta x \\ 0 & 1 & \Delta y \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x_0 \\ y_0 \\ 1 \end{bmatrix}$$

逆变换为:

$$\begin{bmatrix} x_0 \\ y_0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\Delta x \\ 0 & 1 & -\Delta y \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad \begin{cases} x_0 = x - \Delta x \\ y_0 = y - \Delta y \end{cases}$$

# 2.3.2 图像平移

✓平移后图像上的每一点都可以在原图像中找到对应的点。对于不在原图像中的点，可以直接将它的像素值统一设置为0或者255（对于灰度图就是黑色或白色）。



# 2.3.2 图像平移

```
I1 = imread('leopard.jpg');
I1 = im2double(I1);%转换为double

%创建图形窗口
figure(1);
subplot(1,2,1);imshow(I1);title('原图像');

I2 = imtranslate(I1, [100, 20],'FillValues',255);
%平移
subplot(1,2,2);imshow(I2);title('平移图像');
```

# 2.3.3 图像镜像

- 图像的镜像变换不改变图像的形状。图像的镜像(Mirror)变换分为三种：水平镜像，垂直镜像和对角镜像。

# 2.3.3 图像镜像

✓ 水平镜像

图像的水平镜像操作是将图像左半部分和右半部分以图像垂直中轴线为中心进行镜像对换。设图像的大小为M×N，水平镜象的计算可按式计算

$$x = M - x_0 + 1$$

$$y = y_0$$

# 2.3.3 图像镜像

✓ 垂直镜像

图像的垂直镜像操作是将图像上半部分和下半部分以图像水平中轴线为中心进行镜像对换。设图像的大小为M×N，垂直镜象的计算可按式计算

$$x = x_0$$

$$y = N - y_0 + 1$$

# 2.3.3 图像镜像

✓ 垂直镜像

图像的对角镜像操作是将图像以图像水平中轴线和垂直中轴线的交点为中心进行镜像对换。相当于将图像先后进行水平镜像和垂直镜像。设图像的大小为M×N，对角镜象的计算可按式计算

$$x = M - x_0 + 1$$

$$y = N - y_0 + 1$$

# 2.3.3 图像镜像

原图像

水平图像

垂直图像

对角图像

# 2.3.3 图像镜像

```
I1 = imread('leopard.jpg');
I1 = im2double(I1);%转换为double
%创建图形窗口
figure(1);
subplot(2,2,1);imshow(I1);title('原图像');
%水平镜像
I2 = fliplr(I1);
%垂直镜像
I3 = flipud(I1);
%对角镜像
I4 = flipud(I2);
subplot(2,2,2);imshow(I2);title('水平图像');
subplot(2,2,3);imshow(I3);title('垂直图像');
subplot(2,2,4);imshow(I4);title('对角图像');
```

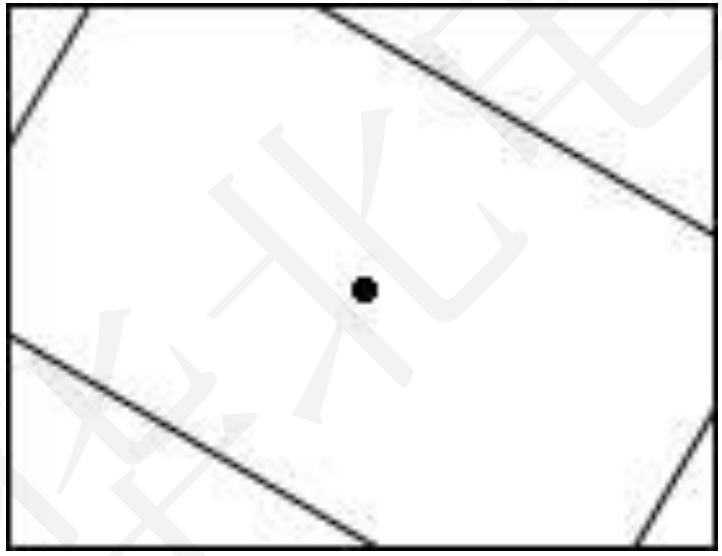
# 2.3.4 图像旋转

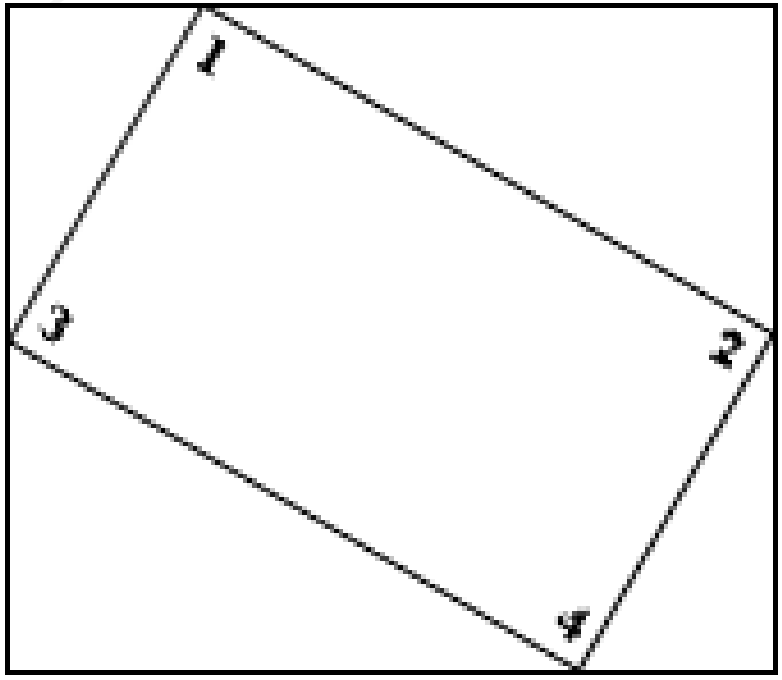
- 图像旋转(rotation)有一个绕着位置转的问题，通常做法是以图像的中心为圆心旋转，将图像上的所有像素都旋转一个相同的角度。

  ✓图像的旋转变换是图像的位置变换，但旋转后，图像的大小一般会改变。

  ✓和图像平移一样，在图像旋转变换中既可以把转出显示区域的图像截去，也可以扩大图像范围以显示所有的图像。

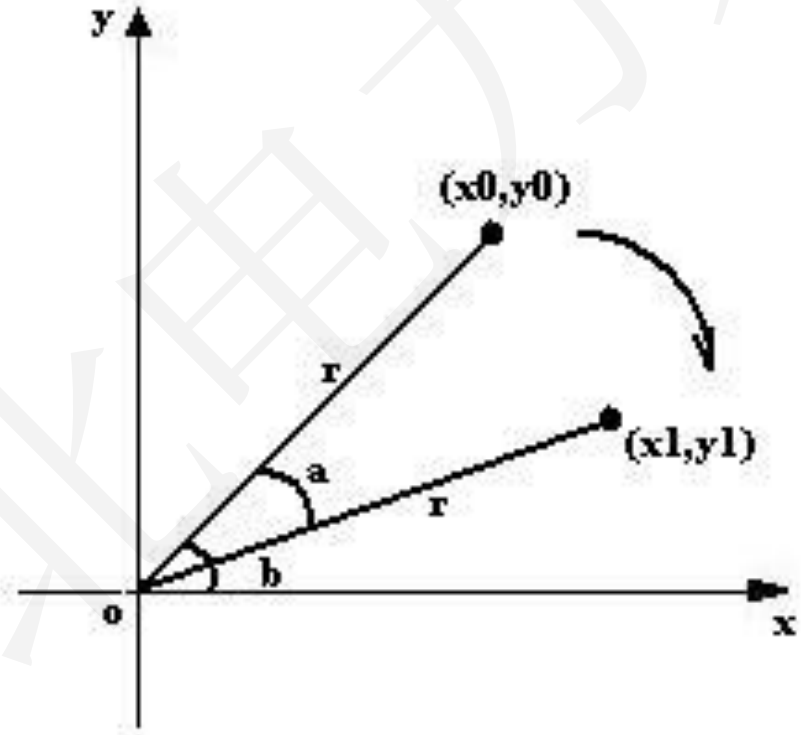
# 2.3.4 图像旋转

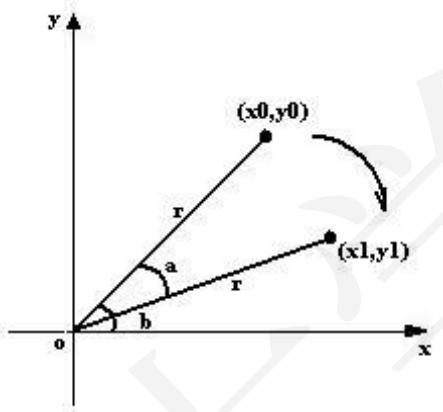


# 2.3.4 图像旋转

设旋转前$x_0$，$y_0$的坐标分别为$x_0=r\cos b$；$y_0=r\sin b$



# 2.3.4 图像旋转



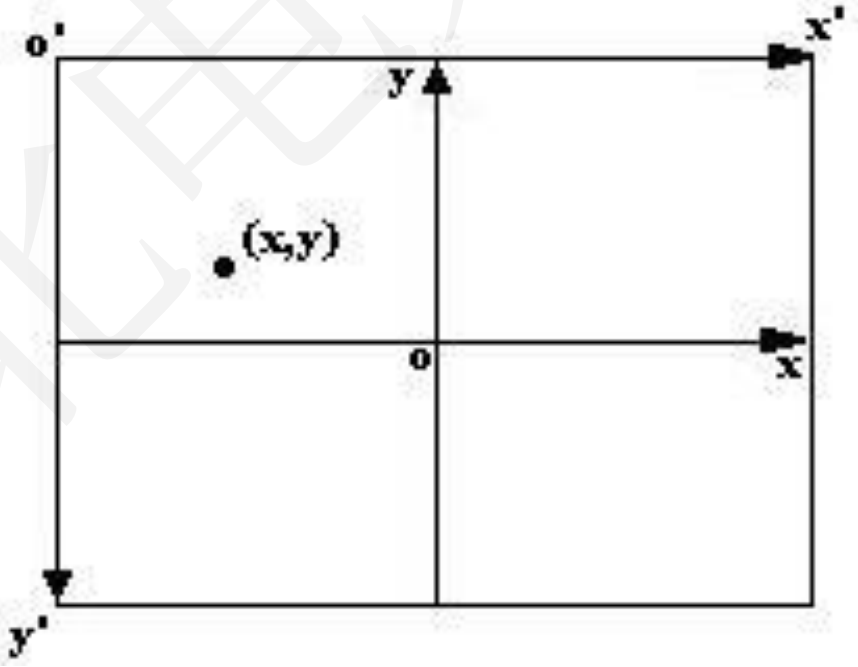

当旋转a角度后：旋转后的坐标$x_1$，$y_1$的坐标分别为

$$\begin{cases} x_1 = r\cos(b-a) = r\cos b\cos a + r\sin b\sin a = x_0\cos a + y_0\sin a \\ y_1 = r\sin(b-a) = r\sin b\cos a - r\cos b\sin a = -x_0\sin a + y_0\cos a \end{cases}$$

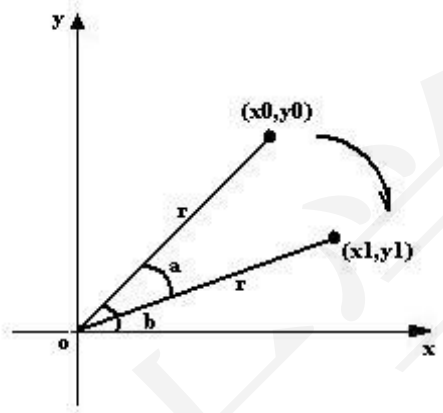
$$\begin{bmatrix} x_1 & y_1 & 1 \end{bmatrix} = \begin{bmatrix} x_0 & y_0 & 1 \end{bmatrix} \begin{bmatrix} \cos a & -\sin a & 0 \\ \sin a & \cos a & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

# 2.3.4 图像旋转

上式中，坐标系xy是以图像的中心为原点，向右为x轴正方向，向上为y轴正方向。它与以图像左上角为原点O’，向右为x’轴正方向，向下为y’轴正方向的坐标系x’y’之间的转换关系如图所示。



# 2.3.4 图像旋转



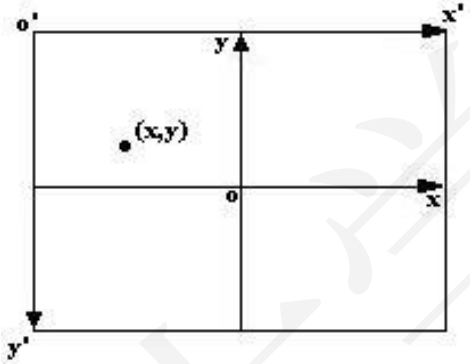

设图像的宽为w，高为h，得到：

$$[x \quad y \quad 1]=[x' \quad y' \quad 1]\times\begin{bmatrix}1 & 0 & 0\\ 0 & -1 & 0\\ -0.5w & 0.5h & 1\end{bmatrix}$$

逆变换为：

$$[x' \quad y' \quad 1]=[x \quad y \quad 1]\times\begin{bmatrix}1 & 0 & 0\\ 0 & -1 & 0\\ 0.5w & 0.5h & 1\end{bmatrix}$$

# 2.3.4 图像旋转




图像的旋转变换比较复杂，它需要3步变换来完成：

（1）将坐标系O’变成O；（平移）

（2）将点顺时针旋转α角；（旋转）

（3）将坐标系O变回O’。（平移）

因此，图像的旋转变换矩阵是上面三个矩阵的级联：

$$[x_1 \quad y_1 \quad 1]=[x_0 \quad y_0 \quad 1]\begin{bmatrix}1 & 0 & 0\\ 0 & -1 & 0\\ -0.5w_{old} & 0.5h_{old} & 1\end{bmatrix}\times\begin{bmatrix}\cos\alpha & -\sin\alpha & 0\\ \sin\alpha & \cos\alpha & 0\\ 0 & 0 & 1\end{bmatrix}\times\begin{bmatrix}1 & 0 & 0\\ 0 & -1 & 0\\ 0.5w_{new} & 0.5h_{new} & 1\end{bmatrix}$$

$$=[x_0 \quad y_0 \quad 1]\times\begin{bmatrix}\cos a & \sin a & 0\\ -\sin a & \cos a & 0\\ -0.5w_{old}\cos a+ & -0.5w_{old}\sin a- & \\ 0.5h_{old}\sin a+0.5w_{new} & 0.5h_{old}\cos a+0.5h_{new} & 1\end{bmatrix}$$

# 2.3.4 图像旋转

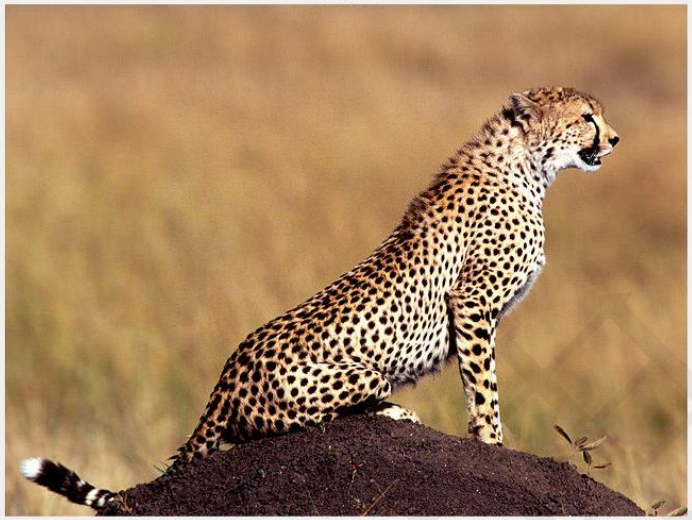
逆变换为：

$$\begin{bmatrix} x_0 & y_0 & 1 \end{bmatrix} = \begin{bmatrix} x_1 & y_1 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ -0.5w_{new} & 0.5h_{new} & 1 \end{bmatrix} \times \begin{bmatrix} \cos a & \sin a & 0 \\ -\sin a & \cos a & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0.5w_{old} & 0.5h_{old} & 1 \end{bmatrix}$$

$$= \begin{bmatrix} x_1 & y_1 & 1 \end{bmatrix} \times \begin{bmatrix} \cos a & -\sin a & 0 \\ \sin a & \cos a & 0 \\ -0.5w_{new}\cos a - 0.5h_{new}\sin a + 0.5w_{old} & 0.5w_{new}\sin a - 0.5h_{new}\cos a + 0.5w_{old} & 1 \end{bmatrix}$$

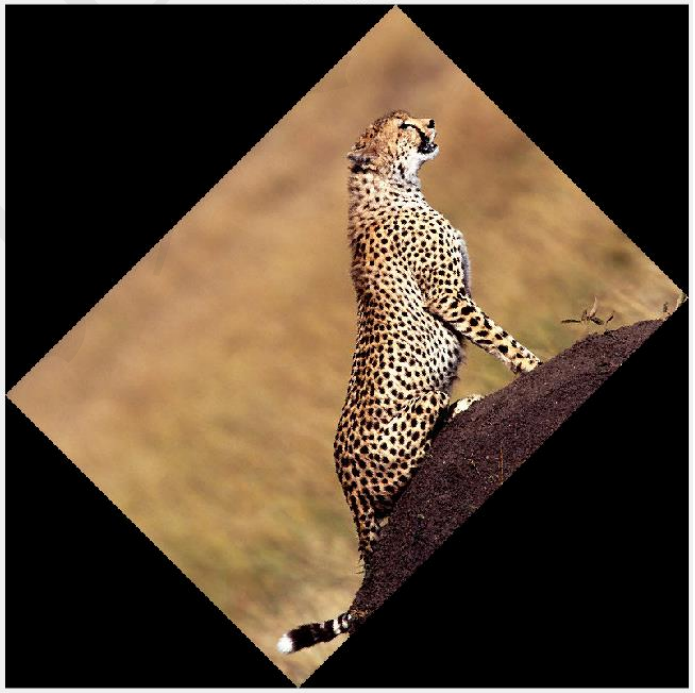
# 2.3.4 图像旋转

原图像



旋转45度剪切图像

旋转45度图像



# 2.3.4 图像旋转

```
I1 = imread('leopard.jpg');
I1 = im2double(I1);%转换为double
%创建图形窗口
figure(1);
subplot(2,2,1);imshow(I1);title('原图像');
% 旋转45度
I2 = imrotate(I1, 45);
% 旋转45度，剪切
I3 = imrotate(I1, 45, 'crop');
% 旋转-45度，剪切
I4 = imrotate(I1, -45, 'crop');
subplot(2,2,2);imshow(I2);title('旋转45度图像');
subplot(2,2,3);imshow(I3);title('旋转45度剪切图像');
subplot(2,2,4);imshow(I4);title('旋转-45度剪切图像');
```

# 2.3.5 图像缩放

- 图像比例缩放(Scaling)是指将给定的图像在x轴方向按比例缩放fx倍，在y轴方向按比例缩放fy倍，从而获得一幅新的图像。

# 2.3.5 图像缩放

- 比例缩放前后两点$P_0(x_0,y_0)$、$P(x, y)$之间的关系用矩阵形式可以表示为

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & 0 \\ 0 & f_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x_0 \\ y_0 \\ 1 \end{bmatrix}$$

$$\begin{cases} x = f_x \cdot x_0 \\ y = f_y \cdot y_0 \end{cases}$$

# 2.3.5.1 图像缩小

- 图像的缩小是将通过减少像素个数来实现的，因此，需要根据所期望缩小的尺寸数据，从原图像中选择合适的像素点，使图像缩小之后可以尽可能保持原有图像的概貌特征不丢失。

# 2.3.5.1 图像缩小

✓ 基于等间隔采样的图像缩小

通过对画面像素的均匀采样来保持所选择到的像素仍旧可以保持像素的概貌特征。该方法的具体实现步骤为：

设原图为f(i,j)，大小为：M×N，i＝1,2,…,M，j=1,2,…,N，缩小后的图像为g(i,j)，大小为k1·M×k2·N，（k1＝k2时为按比例缩小，kl≠k2时为不按比例缩小,kl <1，k2 <1），i＝1,2,…,k1·M; j =1,2,…,k2·N。则有：

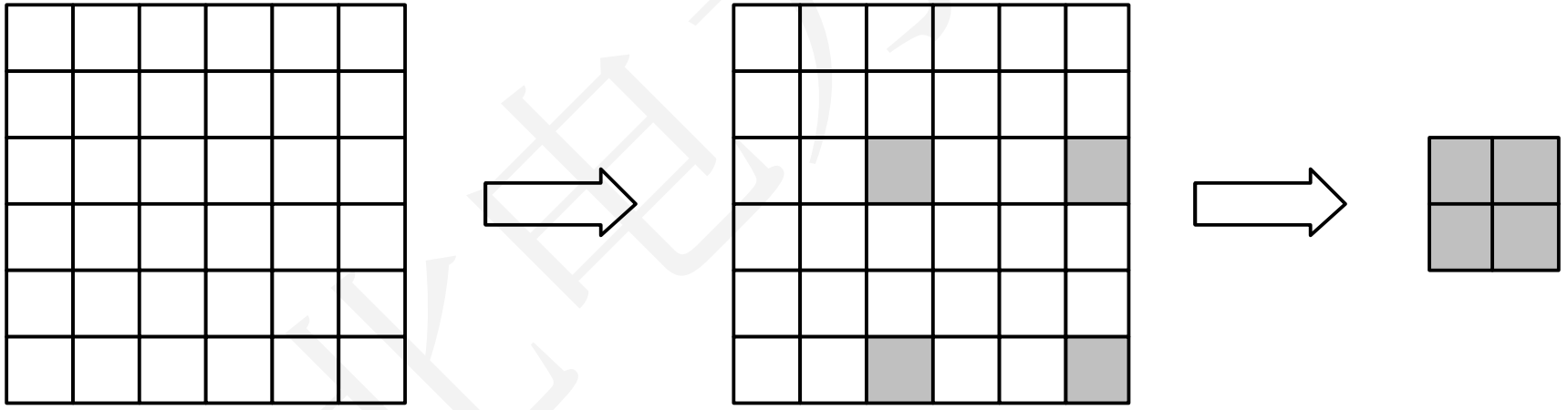
$$\Delta i = 1/k_1 \quad ,\Delta j = 1/k_2$$

$$g(i,j) = f(\Delta i \cdot i \quad ,\Delta j \cdot j)$$

# 2.3.5.1 图像缩小

✓ 基于等间隔采样的图像缩小

$$k_1 = k_2 = 1/3 \quad \Delta i = 1/k_1 \quad ,\Delta j = 1/k_2$$



# 2.3.5.1 图像缩小

图像矩阵的大小为4×6，将其进行缩小，缩小的倍数为k1 = 0.7， k2 = 0.6，则缩小图像的大小为3×4。

$$F = \begin{bmatrix} f_{11} & f_{12} & f_{13} & f_{14} & f_{15} & f_{16} \\ f_{21} & f_{22} & f_{23} & f_{24} & f_{25} & f_{26} \\ f_{31} & f_{32} & f_{33} & f_{34} & f_{35} & f_{36} \\ f_{41} & f_{42} & f_{43} & f_{44} & f_{45} & f_{46} \end{bmatrix}$$

# 2.3.5.1 图像缩小

$$\Delta i = 1/k_1 = 1.4 \quad ,\Delta j = 1/k_2 = 1.7$$

得到缩小后的图像矩阵为

$$G = \begin{bmatrix} f_{12} & f_{13} & f_{15} & f_{16} \\ f_{32} & f_{33} & f_{35} & f_{36} \\ f_{42} & f_{43} & f_{45} & f_{46} \end{bmatrix}$$

# 2.3.5.1 图像缩小

✓ 基于局部均值的图像缩小

从前面的缩小算法可以看到，算法的实现非常简单，但是采用上面的方法对没有被选取到的点的信息就无法反映在缩小后的图像中。为了解决这个问题，可以采用基于局部均值的方法来实现图像的缩小。

（1）得到采样间隔 $\Delta i = 1/k_1$ ，$\Delta j = 1/k_2$

（2）求出相邻两个采样点之间所包含的原图像子块

$$F_{(i,j)} = \begin{bmatrix} f_{\Delta i\cdot(i-1)+1,\Delta j\cdot(j-1)+1} & \cdots & f_{\Delta i\cdot(i-1)+1,\Delta j\cdot j} \\ \vdots & & \vdots \\ f_{\Delta i\cdot i,\Delta j\cdot(j-1)+1} & \cdots & f_{\Delta i\cdot i,\Delta j\cdot j} \end{bmatrix}$$

（3）利用g(i,j)=f(i,j)的均值,求出缩小的图像。

# 2.3.5.1 图像缩小

✓ 基于局部均值的图像缩小

$$F=\begin{bmatrix} f_{11} & f_{12} & f_{13} & f_{14} & f_{15} & f_{16} \\ f_{21} & f_{22} & f_{23} & f_{24} & f_{25} & f_{26} \\ f_{31} & f_{32} & f_{33} & f_{34} & f_{35} & f_{36} \\ f_{41} & f_{42} & f_{43} & f_{44} & f_{45} & f_{46} \end{bmatrix}$$

$$\Delta i = 1/k_1 = 1.4 \quad ,\Delta j = 1/k_2 = 1.7$$

$$F=\begin{bmatrix} f_{11} & f_{12} & f_{13} & f_{14} & f_{15} & f_{16} \\ f_{21} & f_{22} & f_{23} & f_{24} & f_{25} & f_{26} \\ f_{31} & f_{32} & f_{33} & f_{34} & f_{35} & f_{36} \\ f_{41} & f_{42} & f_{43} & f_{44} & f_{45} & f_{46} \end{bmatrix}$$

# 2.3.5.1 图像缩小

✓ 基于局部均值的图像缩小

$$G = \begin{bmatrix} g_{11} & g_{12} & g_{13} & g_{14} \\ g_{21} & g_{22} & g_{23} & g_{24} \\ g_{31} & g_{32} & g_{33} & g_{34} \end{bmatrix}$$

$$g_{21} = \frac{1}{4} \times (f_{21} + f_{22} + f_{31} + f_{32})$$

# 2.3.5.1 图像缩小

✓ 基于局部均值的图像缩小

$$F=\begin{bmatrix}31 & 35 & 39 & 13 & 17 & 21\\ 32 & 36 & 10 & 14 & 18 & 22\\ 33 & 37 & 11 & 15 & 19 & 23\\ 34 & 38 & 12 & 16 & 20 & 24\end{bmatrix}$$

$$G=\begin{bmatrix}35 & 39 & 17 & 21\\ 37 & 11 & 19 & 23\\ 38 & 12 & 20 & 24\end{bmatrix}$$

基于等间隔采用

$$G=\begin{bmatrix}33 & 39 & 15 & 21\\ 35 & 11 & 17 & 23\\ 36 & 12 & 18 & 24\end{bmatrix}$$

基于局部均值

# 2.3.5.1 图像缩小

原图像

缩小0.5图像

缩小为200*200图像

# 2.3.5.1 图像缩小

```
I1 = imread('leopard.jpg');
I1 = im2double(I1);%转换为double
%创建图形窗口，不能使用subplot，看不出效果
figure(1);
imshow(I1);title('原图像');
I2 = imresize(I1, 0.5);
%等比例缩小
figure(2);
imshow(I2);title('缩小0.5图像');
I3 = imresize(I1, [200,200]);
%不等比例缩小
figure(3);
imshow(I3);title('缩小为200*200图像');
```

## 2.3.5.2 图像放大

- 在图像的放大操作中，则需要对尺寸放大后所多出来的空格填入适当的像素值，这是信息的估计问题，所以相对要难一些。
- 由于图像的相邻像素之间的相关性很强，可以利用这个相关性来实现图像的放大。
- 与图像缩小相同，等比例放大不会引起图像的畸变，而不按比例放大则会产生图像的畸变，图像放大一般采用最近邻域法和线性插值法。

# 2.3.5.2 图像放大

✓ 基于最近邻的图像放大

按比例将原图像放大k倍时，如果按照最近邻域法则需要将一个像素值添在新图像的k×k的子块中。

$$g(x,y) = f(x/k, y/k)$$

# 2.3.5.2 图像放大

✓ 基于最近邻的图像放大

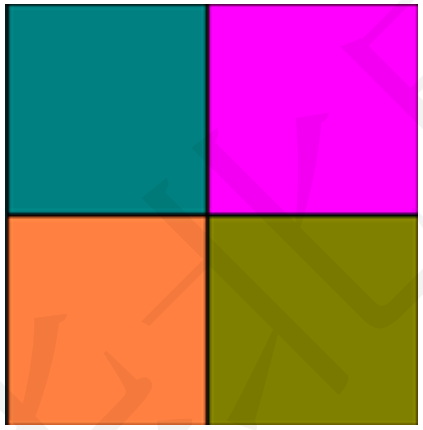
按比例将原图像放大k倍时，如果按照最近邻域法则需要将一个像素值添在新图像的k×k的子块中。

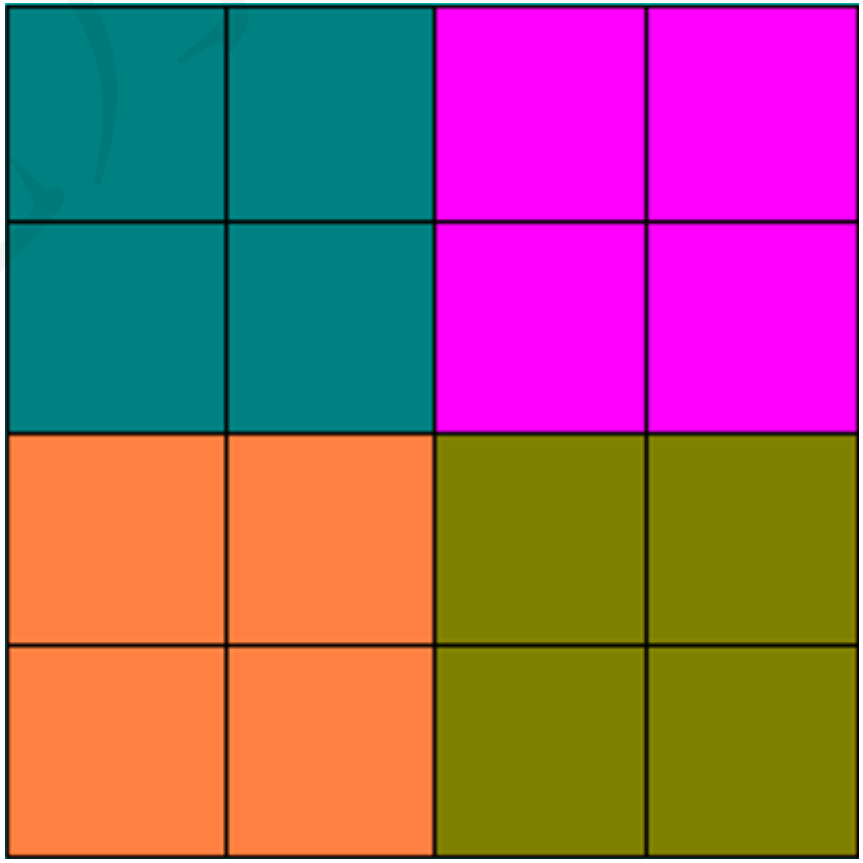
$$F=\begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \quad G=\begin{bmatrix} f_{11}\ f_{11}\ f_{11} & f_{12}\ f_{12}\ f_{12} & f_{13}\ f_{13}\ f_{13} \\ f_{11}\ f_{11}\ f_{11} & f_{12}\ f_{12}\ f_{12} & f_{13}\ f_{13}\ f_{13} \\ f_{11}\ f_{11}\ f_{11} & f_{12}\ f_{12}\ f_{12} & f_{13}\ f_{13}\ f_{13} \\ f_{21}\ f_{21}\ f_{21} & f_{22}\ f_{22}\ f_{22} & f_{23}\ f_{23}\ f_{23} \\ f_{21}\ f_{21}\ f_{21} & f_{22}\ f_{22}\ f_{22} & f_{23}\ f_{23}\ f_{23} \\ f_{21}\ f_{21}\ f_{21} & f_{22}\ f_{22}\ f_{22} & f_{23}\ f_{23}\ f_{23} \\ f_{31}\ f_{31}\ f_{31} & f_{32}\ f_{32}\ f_{32} & f_{33}\ f_{33}\ f_{33} \\ f_{31}\ f_{31}\ f_{31} & f_{32}\ f_{32}\ f_{32} & f_{33}\ f_{33}\ f_{33} \\ f_{31}\ f_{31}\ f_{31} & f_{32}\ f_{32}\ f_{32} & f_{33}\ f_{33}\ f_{33} \end{bmatrix}$$

# 2.3.5.2 图像放大

✓ 基于最近邻的图像放大

按比例将原图像放大k倍时，如果按照最近邻域法则需要将一个像素值添在新图像的k×k的子块中。





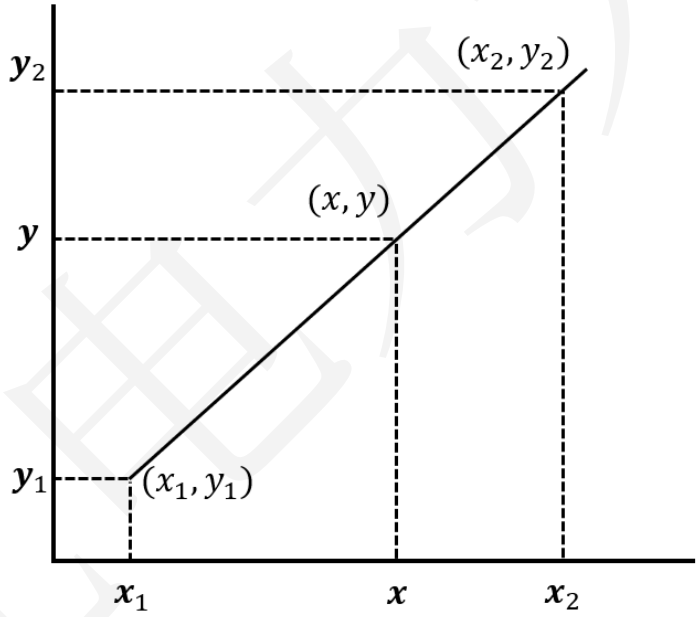
# 2.3.5.2 图像放大

✓ 基于线性（双线性）插值法的图像放大

当求出的分数地址与像素点不一致时，求出该点与周围四个像素点的距离比，根据该比率，由四个邻域的像素灰度值进行线性插值。

# 2.3.5.2 图像放大

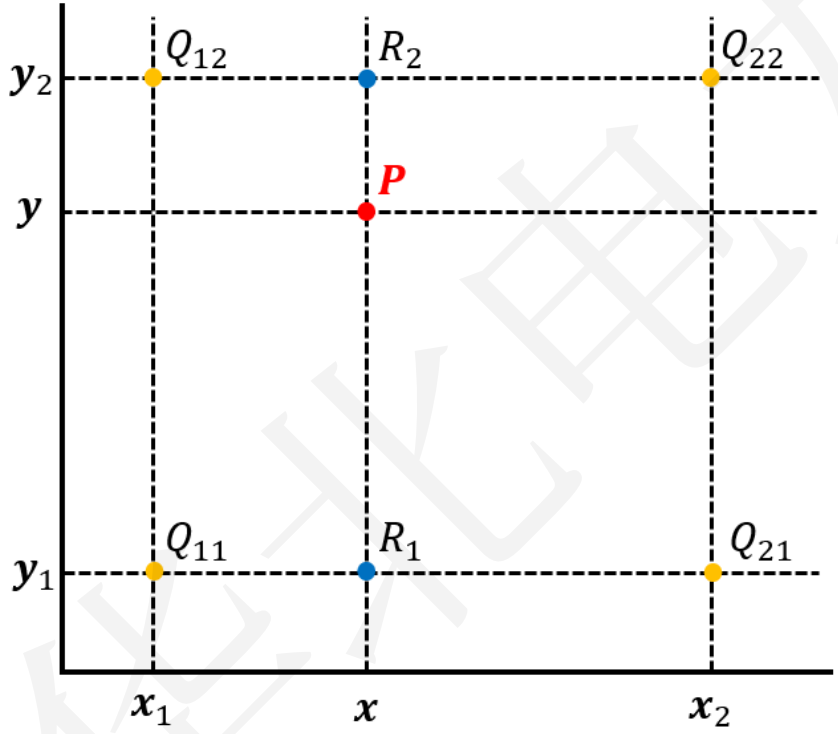
✓ 基于线性（双线性）插值法的图像放大




$$y = \frac{x_2 - x}{x_2 - x_1} y_2 + \frac{x - x_1}{x_2 - x_1} y_1$$

# 2.3.5.2 图像放大

✓ 基于线性（双线性）插值法的图像放大




$$f(R_1) = \frac{x_2 - x}{x_2 - x_1} f(Q_{11}) + \frac{x - x_1}{x_2 - x_1} f(Q_{21})$$

$$f(R_2) = \frac{x_2 - x}{x_2 - x_1} f(Q_{12}) + \frac{x - x_1}{x_2 - x_1} f(Q_{22})$$

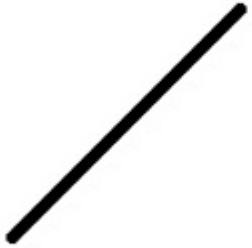
$$f(P) = \frac{y_2 - y}{y_2 - y_1} f(R_1) + \frac{y - y_1}{y_2 - y_1} f(R_2)$$

# 2.3.5.2 图像放大

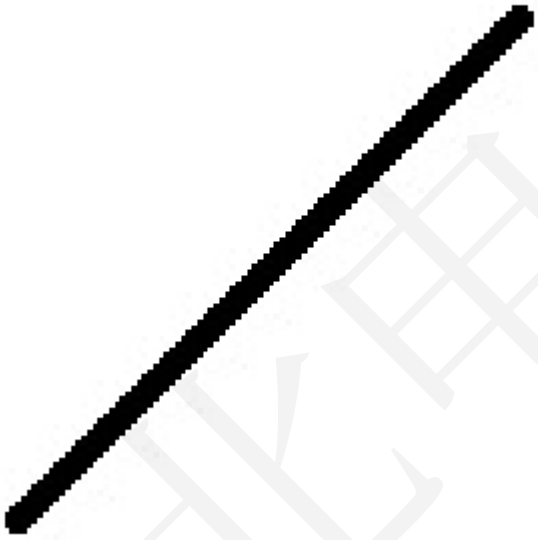
原图像

最近邻放大1.25图像

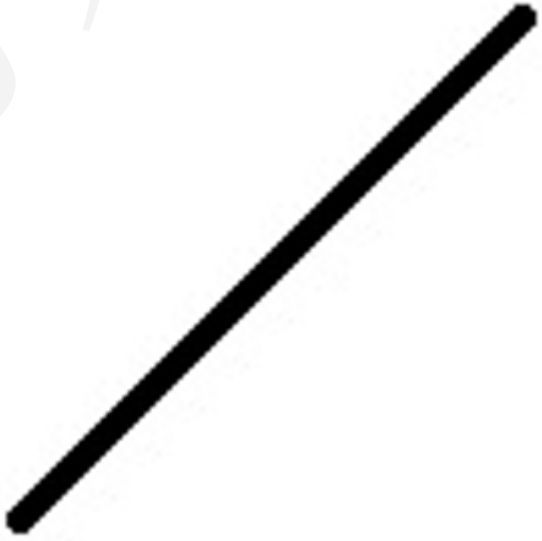
双线性放大1.25图像



# 2.3.5.2 图像放大



最近邻插值放大5倍



双线性插值放大5倍

# 2.3.5.2 图像放大

```
I1 = imread('line.jpg');
I1 = im2double(I1);%转换为double
%创建图形窗口，不能使用subplot，看不出效果
figure(1);
imshow(I1);title('原图像');

I2 = imresize(I1, 1.25, 'nearest');
%放大
figure(2);
imshow(I2);title('最近邻放大1.25图像');

I3 = imresize(I1, 1.25);
%放大
figure(3);
imshow(I3);title('双线性放大1.25图像');
```