

2023春

IB00129

数字图像处理



深圳技术大学  
SHENZHEN TECHNOLOGY UNIVERSITY



大数据与互联网学院  
COLLEGE OF BIG DATA AND INTERNET

# Chap.3 数字图像的基本运算

曹劲舟 博士  
助理教授

深圳技术大学  
大数据与互联网学院

2023年2月

深圳技术大学  
大数据与互联网学院  
曹劲舟 版权所有



# 主要内容

---

- 1、图像的点运算（灰度变换）：灰度反转，对数变换；
- 2、灰度直方图；
- 3、图像的代数运算；
- 4、图像的几何运算；

深圳技术大学  
大数据与互联网学院  
曹劲舟 版权所有

- 从一般意义上来说，各种图像处理方法都是一种图像运算方法。
- 从严格意义上来说，图像运算仅指对图像中的**所有像素实施了相同处理的那些运算**，比如对图像的点运算、直方图运算、代数运算、几何运算、灰度插值运算等。

# 灰度变换

深圳技术大学  
大数据与互联网学院  
曹劲舟 版权所有

# 灰度反转

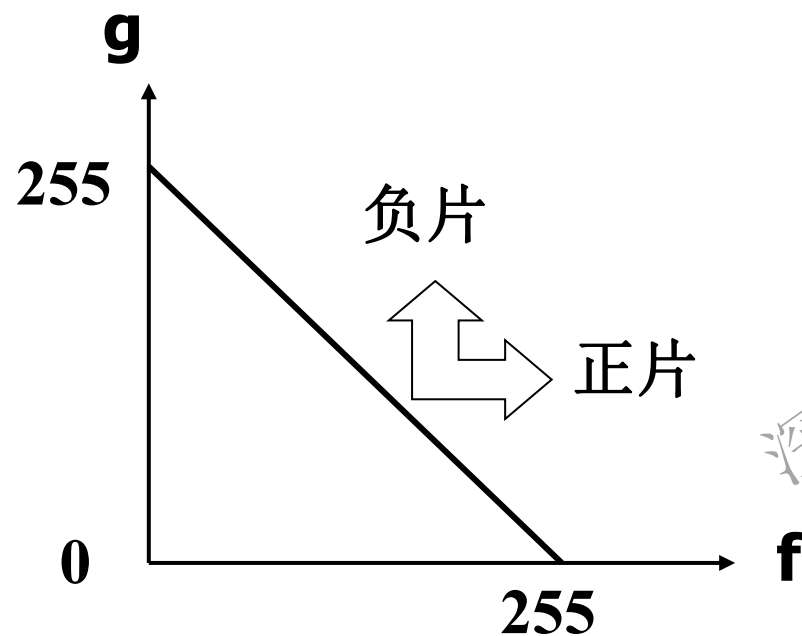
- 黑白图像的反转就是使灰度值为1的像素值变成0，使灰度值为0的像素值变成1。
- 对于256灰度级图像来说，图像的灰度反转值就是用255分别减去原图像的各个像素的灰度值。

设图像的灰度级为L，则图像的灰度反转可表示为：

$$g(x, y) = L - 1 - f(x, y)$$

# 灰度反转

■256灰度级图像的灰度反转如图所示。



# 灰度反转

## ■灰度反转matlab程序

%灰度反转matlab程序

```
clc; clear all; close all;
```

```
img0=imread('lena.jpg');
```

```
result_img=256-1-img0; % 图像阵列中的所有像素的灰度值反转
```

```
subplot(1,2,1); imshow(img0); title('原图像'); %显示原图像
```

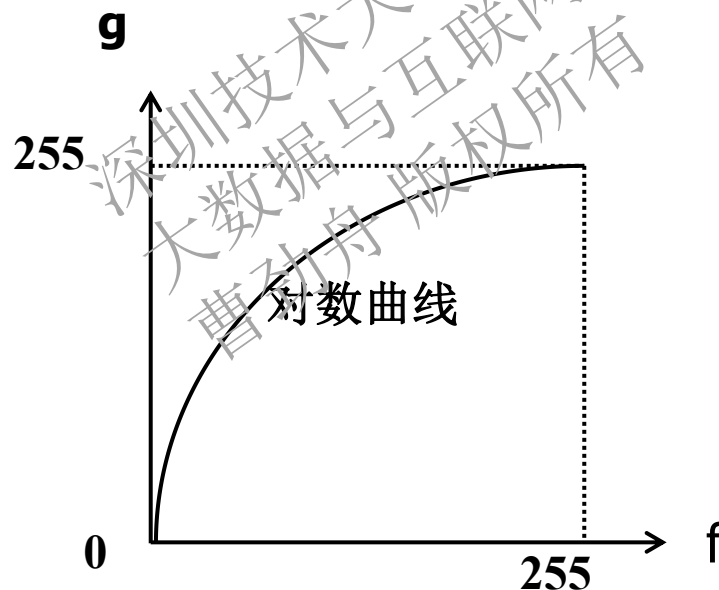
```
subplot(1,2,2); imshow(result_img); title('灰度反转图像(负片)');
```

# 对数变换

■对原图像 $f(x,y)$ 进行对数变换的解析式可表示为：

$$g(x,y) = c \cdot \log(1 + f(x,y))$$

对数变换曲线如下图所示：





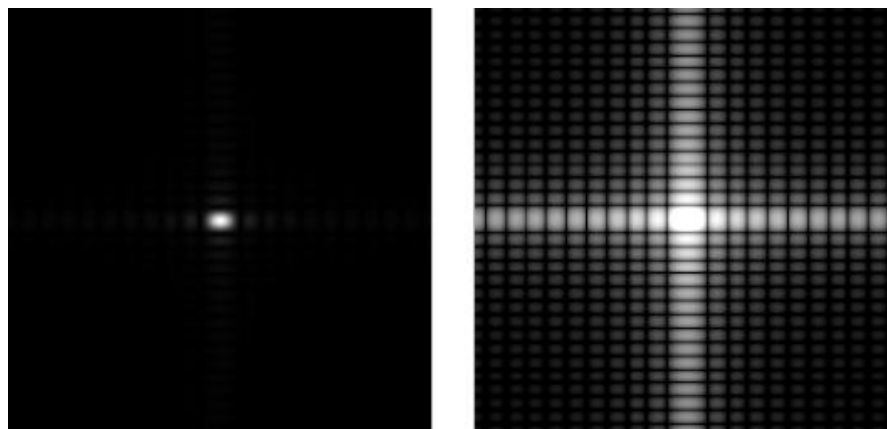
# 对数变换

## ■对数变换的作用：

- ( 1 ) 通过对图像的灰度值的**动态范围进行压缩**，调高输入图像的低灰度值。
- ( 2 ) 人的视觉感觉与进入人眼的光的强度成**对数关系**，常先给图像进行对数变换后再显示输出。

# 对数变换

**比如：**傅里叶频谱要显示的值的范围往往比较大，而傅里叶频谱要显示的重点是突出最亮的像素（对应于低频成分），而这在一个8比特的显示系统中会把频谱图像中的低灰度值部分（对应于高频成分）损失掉。这样，就可利用对数变换调高频谱图像的低灰度值而对高灰度值又尽可能地影响最小。



(a) 图像的傅里叶频谱      (b) 对(a)进行 $c=1$ 的对数变换的结果

# 直方图

深圳技术大学  
大数据与互联网学院  
曹劲舟 版权所有

# 灰度图像直方图

## ■灰度直方图的概念

- 直方图是图像的**灰度—像素个数**统计图，即对于每个灰度值，统计在图像中具有该灰度值的像素个数出现的频数，并绘制成图形，称为**灰度直方图**（简称**直方图**）。
- 直方图的横坐标表示图像中像素的**灰度级**，取值范围为0 ~ 255；纵坐标表示图像中各个**灰度级像素的数量**。

# 灰度图像直方图

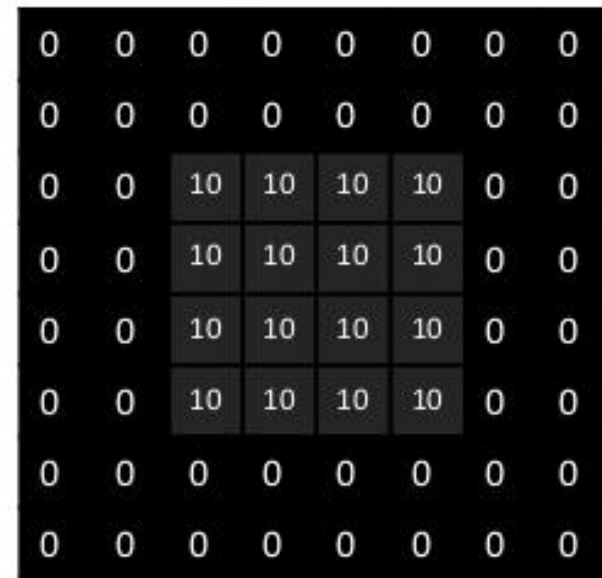
## ■归一化直方图（类似概率密度函数pdf）

$$p(k) = \frac{n_k}{n}$$

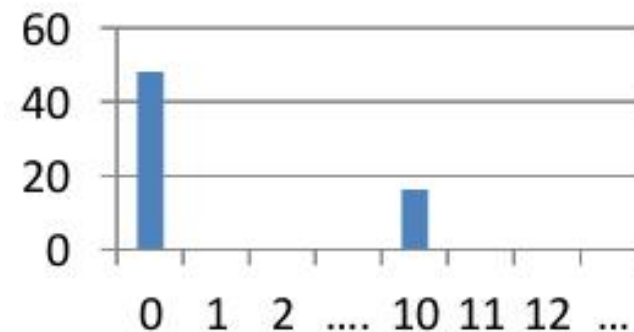
$n_k$ —灰度级为 $k$ 像素点的个数

$n$ —总像素点个数

$p(k)$ :具有该灰度级的像素出现的频数



Pixel Count



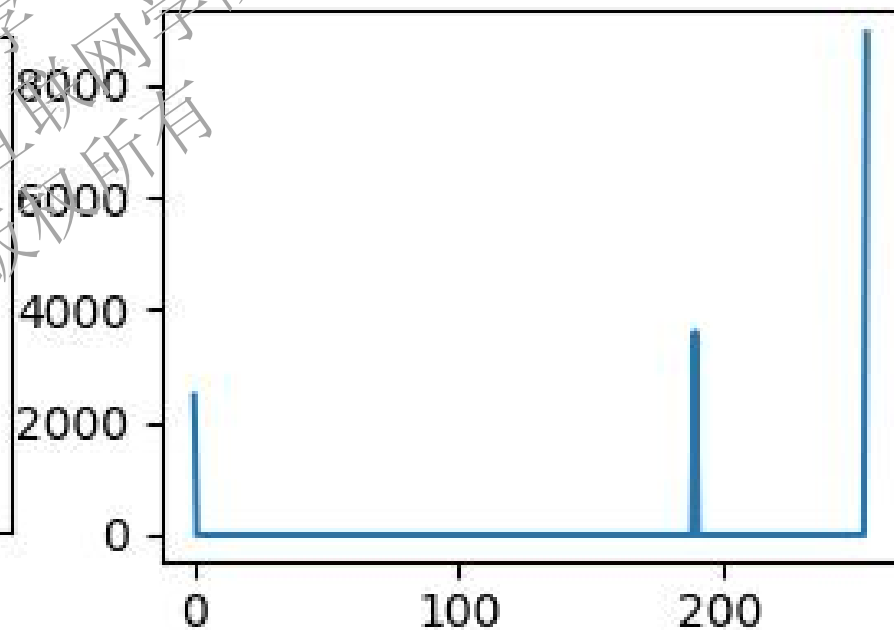
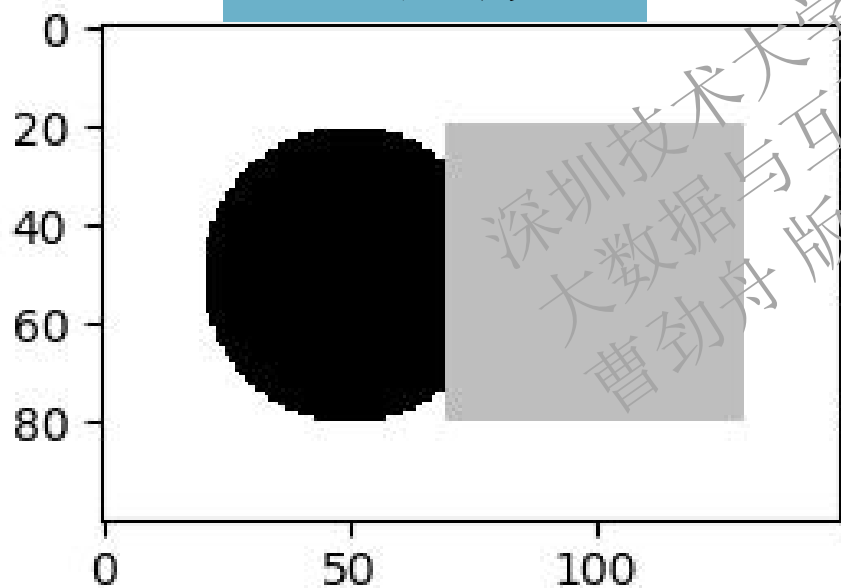
# 直方图举例

■ Black = 0

■ Gray = 190

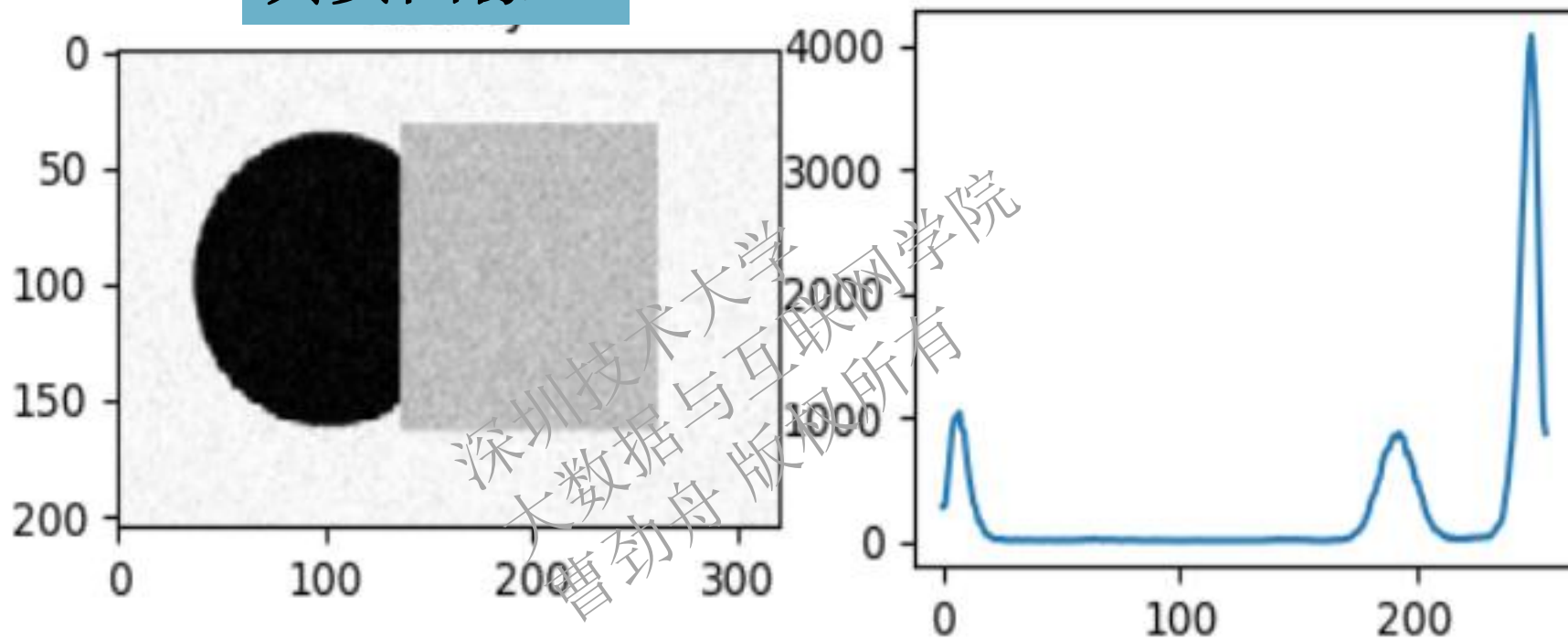
■ White = 254

理想图像



# 直方图举例

真实图像



# 灰度图像直方图

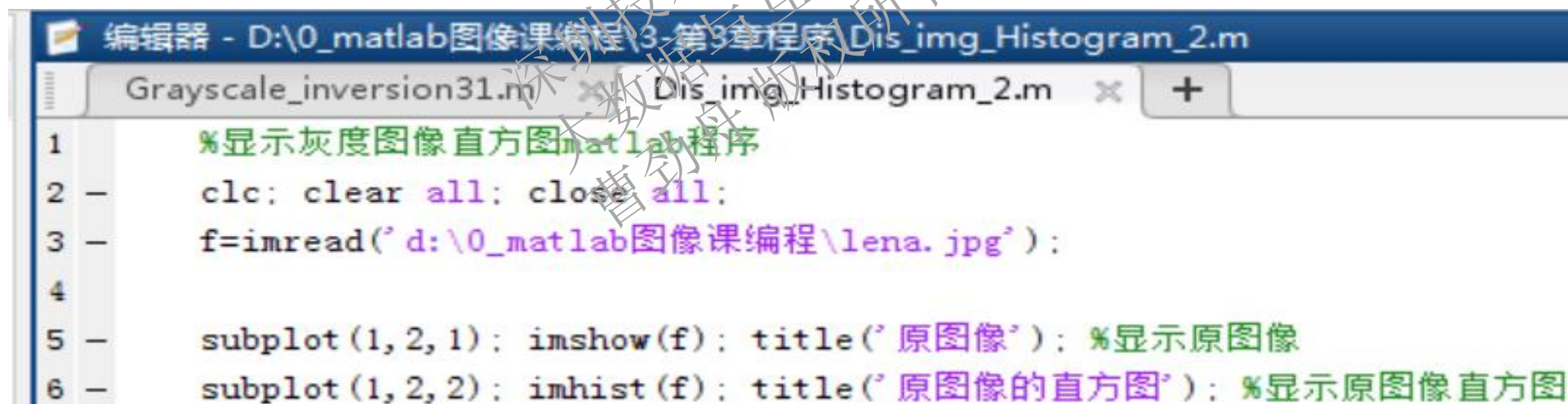
**%显示灰度图像直方图matlab程序**

**clc; clear all; close all;**

**f=imread('d:\0\_matlab图像课编程\lena.jpg');**

**subplot(1,2,1); imshow(f); title('原图像'); %显示原图像**

**subplot(1,2,2); imhist(f); title('原图像的直方图');**



The screenshot shows a MATLAB editor window with the following code:

```
1 %显示灰度图像直方图matlab程序
2 clc; clear all; close all;
3 f=imread('d:\0_matlab图像课编程\lena.jpg');
4
5 subplot(1,2,1); imshow(f); title('原图像'); %显示原图像
6 subplot(1,2,2); imhist(f); title('原图像的直方图'); %显示原图像直方图
```



# 灰度直方图的特征总结

(1) 直方图仅能描述图像中每个灰度值具有的**像素个数**，不能表示图像中每个像素的**位置(空间)信息**；

(2) 任一特定的图像都有**唯一**的直方图，不同的图像可以具有相同的直方图；

(3) 对于空间分辨率为 $M \times N$ ，且灰度级范围为 $[0, L-1]$ 的图像，有：

$$\sum_{j=0}^{L-1} h(j) = M \times N$$



Figure 1

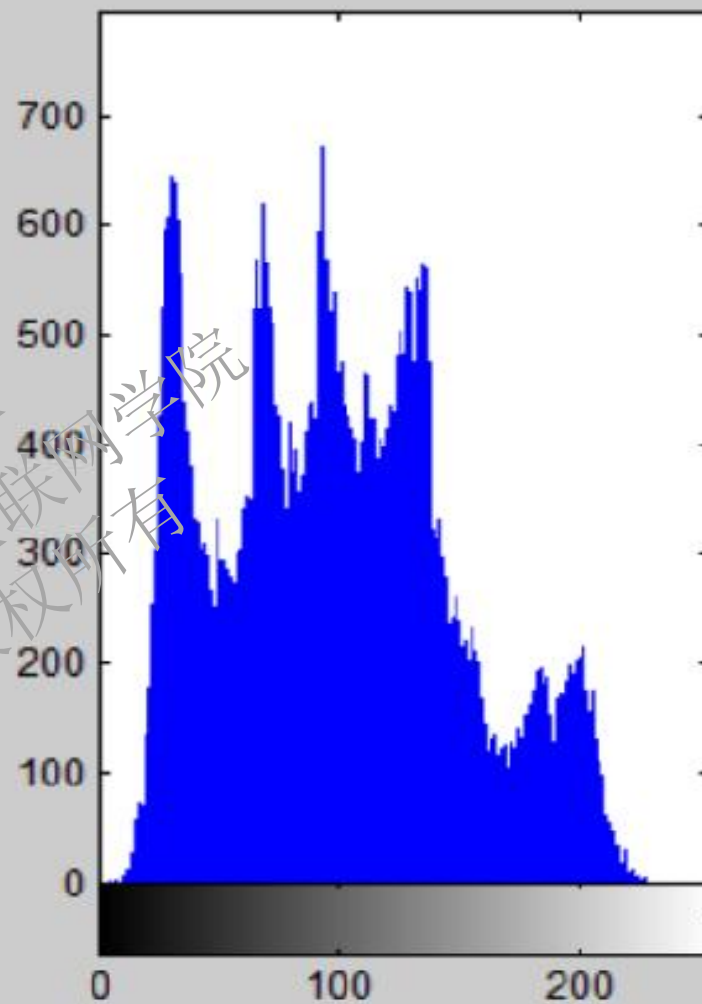
文件(F) 编辑(E) 查看(V) 插入(I) 工具(T) 桌面(D) 窗口(W) 帮助(H)



原图像

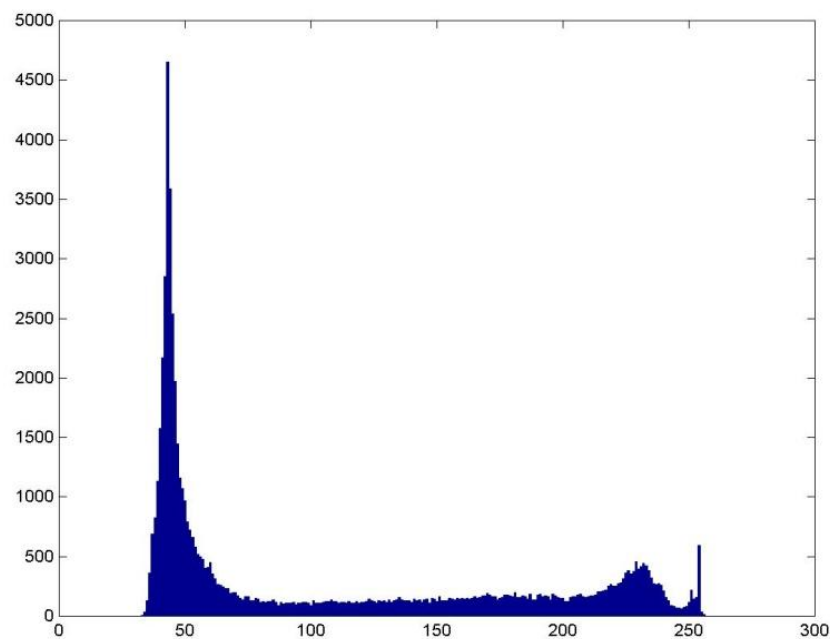


原图像的直方图



# 灰度图像直方图与灰度图像的对比度

## ■灰度直方图表明图像中每一个灰度级有多少个像素



由matlab直方图计算程序计算所得

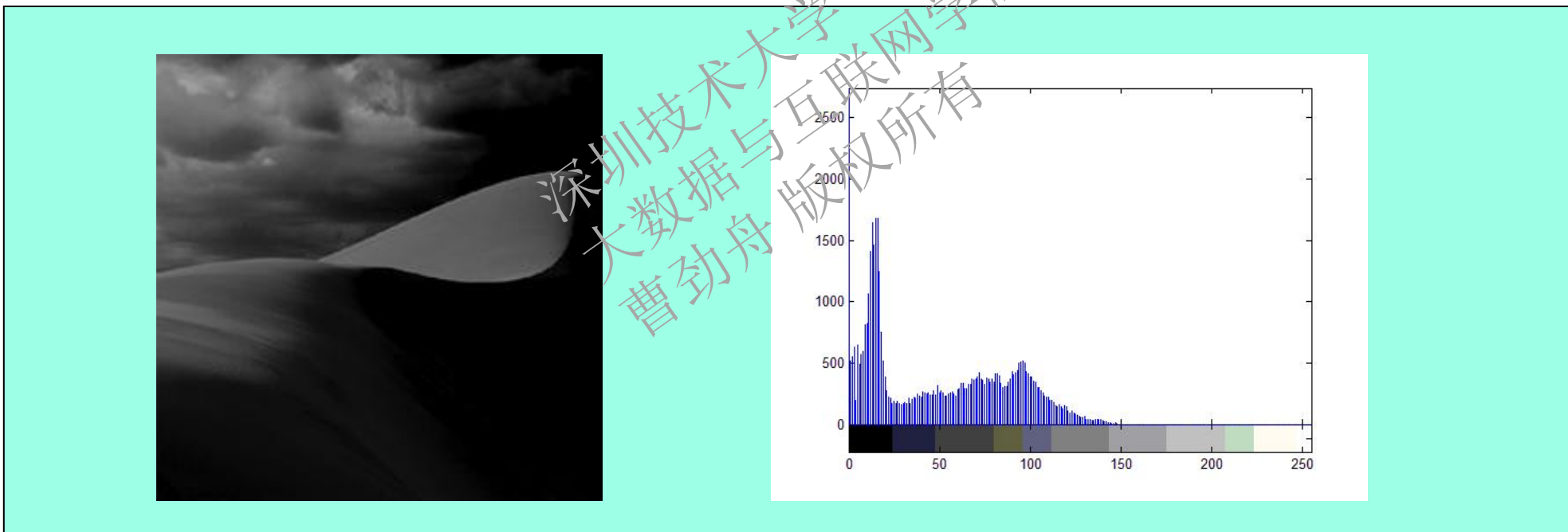
# 灰度图像的对比度

- 灰度图像的对比度是对一幅图像中最亮的白和最暗的黑之间亮度层级的测量，一般用图像画面中最大的灰度值（最白亮度）与最小的灰度值（最黑亮度）的比值来表示。
- 因此，白色越亮、黑色越暗，对比度就越高（大）；对比度越大，图像越清晰醒目；对比度越小，图像画面会显得灰蒙蒙而不清晰。

# 灰度直方图的概念及分布特征

■根据图像直方图分布特点，分析对应图像的特征：

- 1、直方图总体上**偏左**，说明图像的灰度值普遍**偏小**，也即图像**偏暗**。
- 2、直方图分布在**局部区域**，说明图像的对比度**较低**。

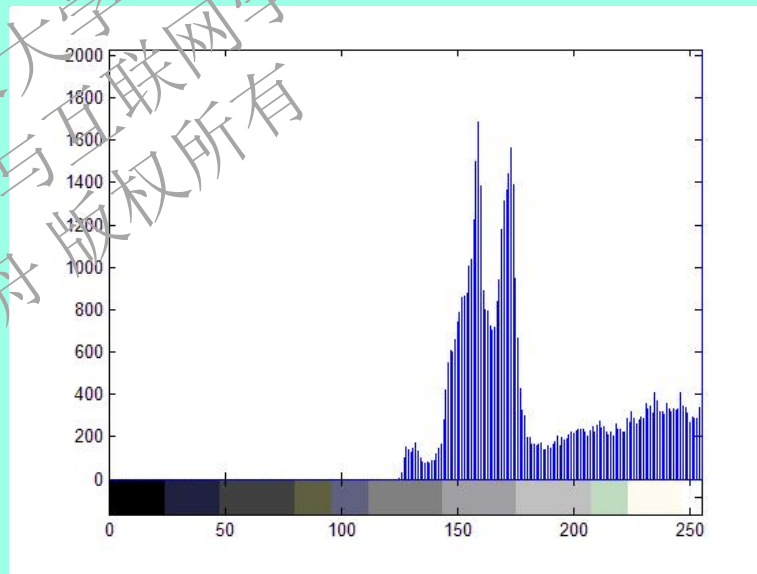


# 灰度直方图的概念及分布特征

■根据图像直方图分布特点，分析对应图像的特征：

■1、直方图总体上**偏右**，说明图像的灰度值普遍**偏大**，也即图像**偏亮**。

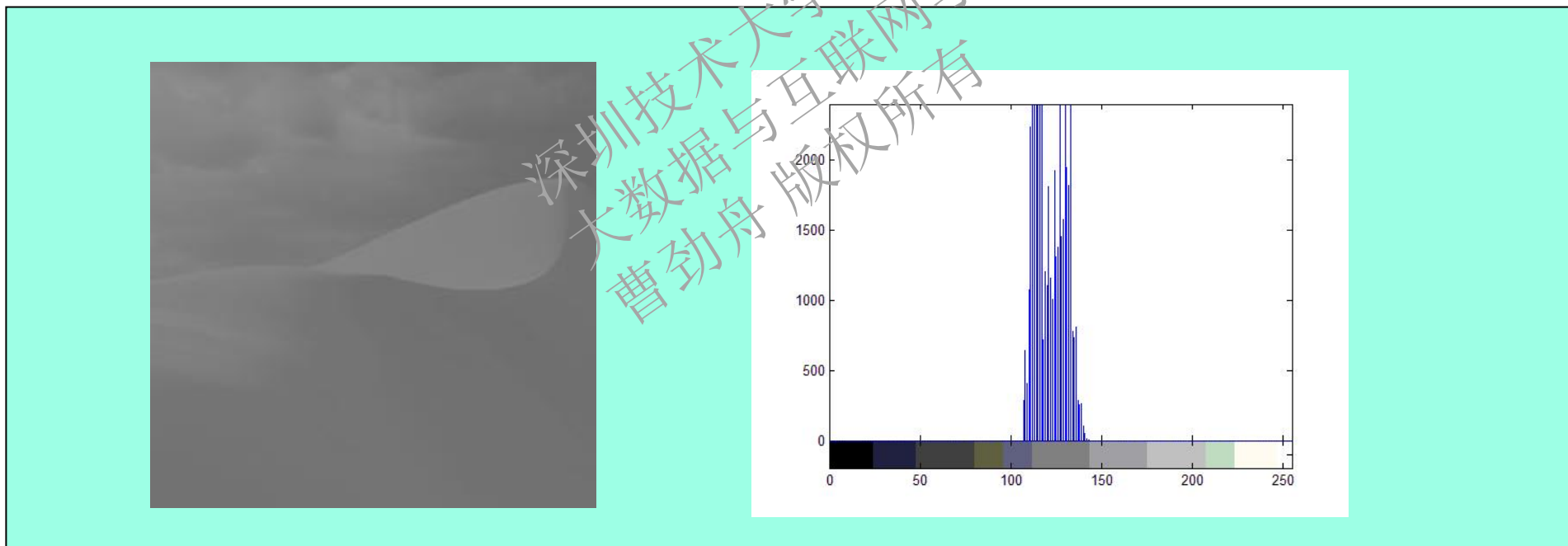
2、直方图分布在**局部区域**，说明图像的对比度**较低**。



# 灰度直方图的概念及分布特征

■根据图像直方图分布特点，分析对应图像的特征：

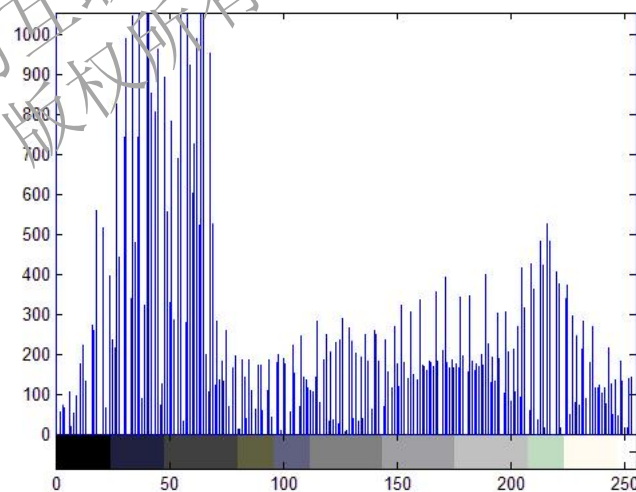
- 1、直方图总体上**居中**，说明图像的灰度值**不太大也不太小**，也即图像亮度**适中**。
- 2、直方图分布在**局部区域**，说明图像的对比度**较低**。



# 灰度直方图的概念及分布特征

## ■根据图像直方图分布特点，分析对应图像的特征：

- 1、直方图总体上**均匀分布**于整个灰度区间，且**有峰值**，说明图像亮度值**黑白分明**，也即图像**清晰醒目**。
- 2、直方图**分布均匀**，说明图像的**对比度高**。





# 灰度图像的对比度总结

---

■直观的来理解直方图：

■横轴上各（亮度值）点对应的柱状高度就是分布在该亮度的像素个数。

■当柱状接近分布在整个横轴上，且至少有一个峰值时，图像的对比度较好。

深圳技术大学  
大数据与互联网学院  
曹劲舟 版权所有

# 图像的代数运算

深圳技术大学  
大数据与互联网学院  
曹劲舟 版权所有

# 图像的代数运算——相加运算

- 图像相加是通过对两幅大小相同的图像对应位置像素的相加运算，以产生一幅新的含有两幅图像信息的图像的方法。图像相加也称为图像合成。

设  $f_1(x, y)$  和  $f_2(x, y)$  分别表示大小为  $M \times N$  的两幅输入图像，图像  $f_1(x, y)$  和  $f_2(x, y)$  图像相加后得到的结果输出图像为  $g(x, y)$ ，且： $x \in [0, M-1]$ ， $y \in [0, N-1]$ ，则两幅图像的相加运算可表示为：

$$g(x, y) = f_1(x, y) + f_2(x, y)$$

# 图像的代数运算——相加运算

## 1. 两幅图像内容的叠加/合成

### ■图像相加运算的实现方式:

#### (1) 灰度值折半相加

$$g(x, y) = \text{IntegerRound}\left(\frac{1}{2}f_1(x, y) + \frac{1}{2}f_2(x, y)\right)$$

#### (2) 按不同比例灰度值的相加

$$g(x, y) = \alpha f_1(x, y) + \beta f_2(x, y)$$

其中:  $\alpha + \beta = 1$

# 图像的代数运算——相加运算



+





# 图像的代数运算——相加运算



+



=



# 图像的代数运算——相加运算

## 2、相加多幅图像的叠加去加性噪声

■通过对**同一场景的多幅图像**的灰度值求平均值（多幅灰度图像的叠加运算），可以**去除加性（Additive）随机噪声**。

$$g(x, y) = \frac{1}{N} \sum_{i=1}^N f_i(x, y) \quad (3.9)$$

# 图像的代数运算——相加运算



(a) 有加性噪声图像    (a) 2幅去噪效果    (a) 4幅去噪效果    (a) 16幅去噪效果。

图 3.5 利用图像叠加运算去加性噪声效果示例。



# 图像的代数运算——相减运算

设  $f_1(x, y)$  和  $f_2(x, y)$  分别表示大小为  $M \times N$  的两幅输入图像，从图像  $f_1(x, y)$  中的各位置的像素中减去的相应位置的像素值后，得到的结果输出图像为  $g(x, y)$ ，且  $x \in [0, M-1]$ ,  $y \in [0, N-1]$ ，则两幅图像的相减运算可表示为：

$$g(x, y) = f_1(x, y) - f_2(x, y)$$

# 图像的代数运算——相减运算



—



深圳技术大学  
大数据与互联网学院  
曹劲舟 版权所有

# 图像的代数运算——相减运算

- 当两幅256灰度级图像对应坐标位置像素值相减的结果大于或等于零时，则取其为结果图像中的像素的灰度值；
- 当相减结果小于零时，一般都是**取零**为结果值。
- 对于某些特殊的应用目的，也可以取其**绝对值**为结果值。
- 图像相减运算的典型应用是**图像的变化检测**。

# 图像的几何运算

深圳技术大学  
大数据与互联网学院  
曹劲舟 版权所有

# 图像的几何运算

■图像的几何运算又称为图像的几何变换，包括：

- 1、图像的平移变换；
- 2、图像的旋转变换；
- 3、图像镜像；
- 4、图像转置；
- 5、图像的缩放。

深圳技术大学  
大数据与互联网学院  
曹劲舟 版权所有

# 图像的平移变换

- 图像平移变换是指将一幅图像或一幅图像中的子图像块（以下简称为**图像块**）中的所有像素点，都按指定的X方向偏移量  $\Delta X$  和Y方向偏移量  $\Delta Y$  进行移动。

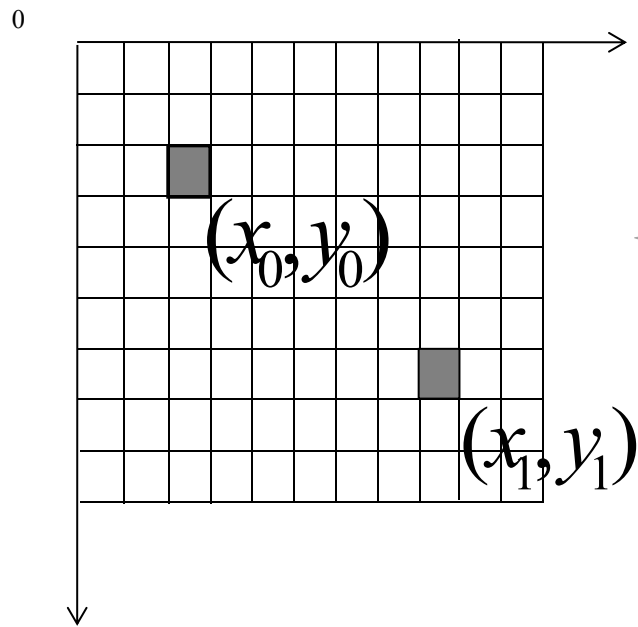
深圳技术大学  
大数据与互联网学院  
曹劲舟 版权所有

# 图像的平移变换

■ 设初始坐标为  $(x_0, y_0)$  的像素平移  $(\Delta x, \Delta y)$  后的坐标为  $(x_1, y_1)$ ，则

有

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



图像平移变换式的矩阵形式为

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

# 图像的平移变换

## ■实现方式:

### (1) 图像块平移

(2) 整幅图像平移: 图像平移保持原幅面大小; 被移出的部分被截掉。



原图像



平移图像子块后的图像



平移后的图像



# 图像的旋转变换

■ 图像旋转变换是指，以图像的中心为原点，将图像中的所有像素（也即整幅图像）旋转一个相同的角度。

■ 实现方式：

- （1）旋转后的图像幅面被放大（按外接矩形尺寸）；
- （2）保持图像旋转前后的幅面大小，把旋转后图像被转出原幅面大小的那部分截断。

# 图像的旋转变换

## 1、相加旋转后图像幅面放大的图像旋转变换

### ■图像旋转变换公式的推导分为两步：

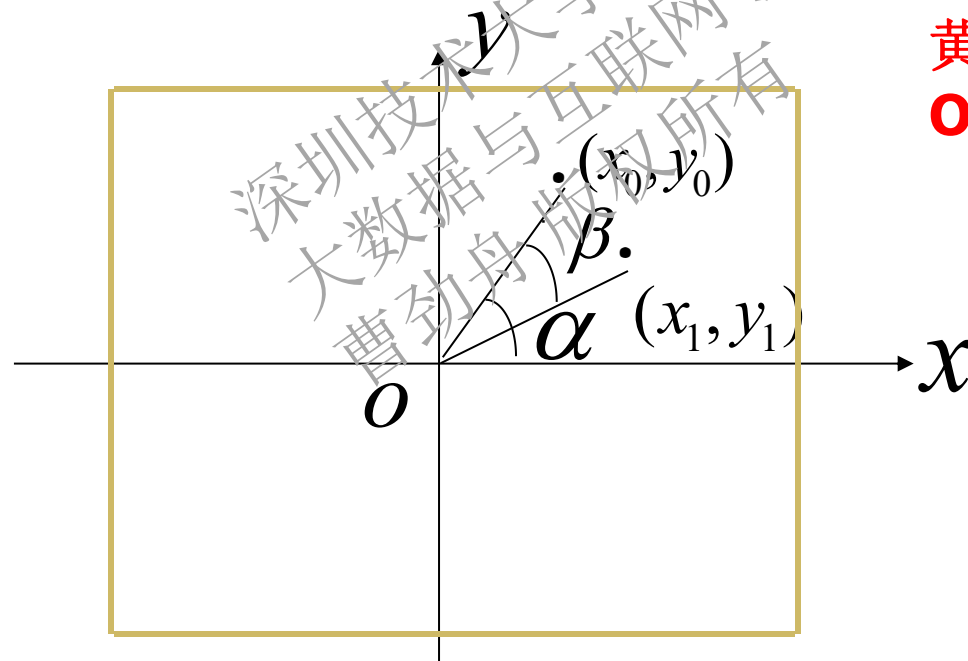
- 第一步：先把待旋转图像的中心看作为 $xoy$ 平面数学坐标系原点，并基于该坐标系推导出图像上像素点旋转变换公式。
- 第二步：把第一步推导的变换公式映射到图像的显示坐标，最终得出在显示坐标下的图像旋转变换公式。

# 图像的旋转变换

## ■1、相加旋转后图像幅面放大的图像旋转变换

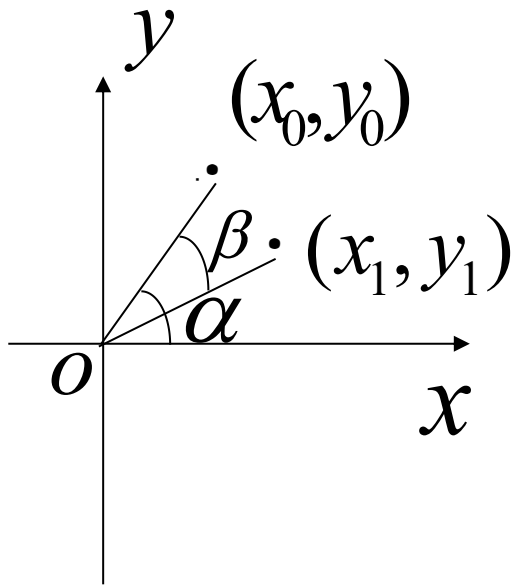
### 1) 基于xoy平面坐标系的点旋转变换

设位于  $(x_0, y_0)$  处的点到坐标原点的直线  $r$  与轴  $x$  的夹角为  $\alpha$ ，直线  $r$  顺时针旋转  $\beta$  角度后使位于  $(x_0, y_0)$  处的点被旋转至  $(x_1, y_1)$  处。



黄色框相当于原图像，  
O点位于原图像中心。

# 图像的旋转变换



旋转前:

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

旋转后:

$$\begin{cases} x_1 = r \cos(\alpha - \beta) = r \cos \alpha \cos \beta + r \sin \alpha \sin \beta \\ y_1 = r \sin(\alpha - \beta) = r \sin \alpha \cos \beta - r \cos \alpha \sin \beta \end{cases}$$

# 图像的旋转变换

旋转前式代入旋转后式(点旋转变换表示形式):

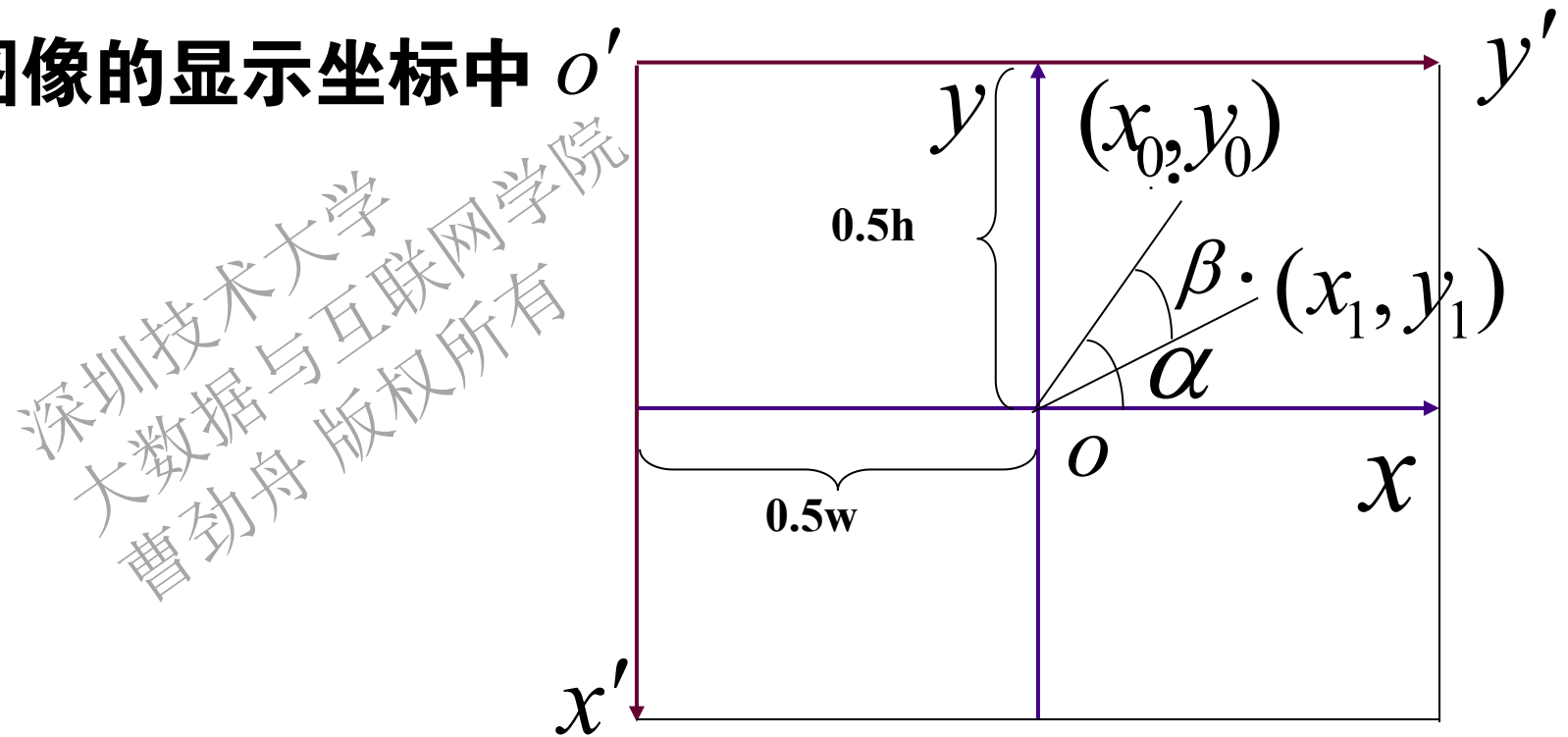
$$\begin{cases} x_1 = x_0 \cos \beta + y_0 \sin \beta \\ y_1 = -x_0 \sin \beta + y_0 \cos \beta \end{cases}$$

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

# 图像的旋转变换

## 2) 图像像素点的旋转变换

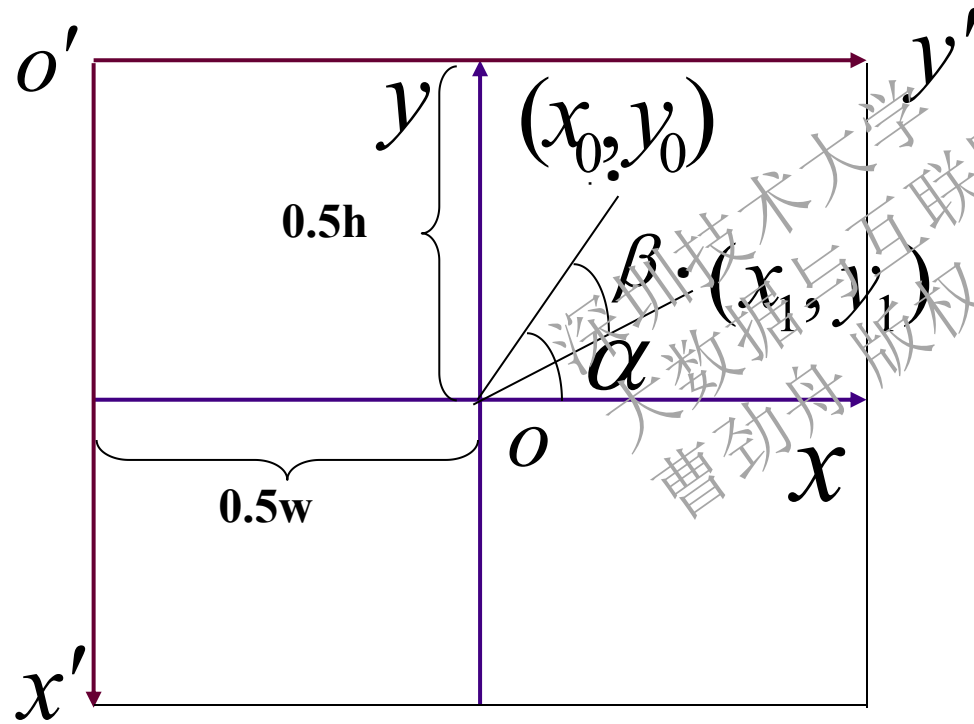
- 由于图像像素点的旋转变换是基于图像的显示坐标的，所以还需将变换结果映射到图像的显示坐标中



图像的像素点与图像的显示坐标和  $xoy$  平面坐标的关系示例

# 图像的

假设图像的宽度width用 $w$ 表示，图像的高度high用表示 $h$ ，由图可知：在原点 $o'$ 位于左上角、 $x'$ 轴方向朝下、 $y'$ 轴方向朝右的图像显示坐标中，如果用 $(x'_0, y'_0)$ 表示 $(x_0, y_0)$ 在图像显示坐标 $x'o'y'$ 中的位置，用 $(x'_1, y'_1)$ 表示 $(x_1, y_1)$ 在图像显示坐标 $x'o'y'$ 中的位置，则有：



$$\begin{cases} x'_0 = 0.5h - y_0 \\ y'_0 = 0.5w + x_0 \end{cases}$$

$$\begin{cases} x'_1 = 0.5h - y_1 \\ y'_1 = 0.5w + x_1 \end{cases}$$

## 2) 图像像素点的旋转变换

上两式的矩阵形式分别为：

$$\begin{bmatrix} x'_0 \\ y'_0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0.5h \\ 1 & 0 & 0.5w \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x'_1 \\ y'_1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0.5h \\ 1 & 0 & 0.5w \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$$



# 图像的旋转变换

## 2) 图像像素点的旋转变换

将式:

( 基于xoy坐标的点旋转变换矩阵表示形式 )

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

代入, 可得图像像素点的旋转变换公式为:

$$\begin{bmatrix} x'_1 \\ y'_1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0.5h \\ 1 & 0 & 0.5w \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos \beta & \sin \beta & 0 \\ -\sin \beta & \cos \beta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \\ 1 \end{bmatrix}$$

# 图像的旋转变换

## ■2、旋转前后幅面大小保持不变的图像旋转变换

- 当图像旋转前后幅面大小保持不变时，就要把旋转后超出原幅面大小的那部分图像像素截断。



旋转前的图像



旋转后的图像

## ■2、旋转前后幅面大小保持不变的图像旋转变换

- 根据上述公式对原图像进行变换时，需要判断每一个像素值的坐标值 $(x_1, y_1)$ 是否超出了原图像的尺寸范围，只有那些 $(x_1, y_1)$ 值都没有超出图像尺寸范围的像素才得以保留，其它像素由于其坐标超出了原图像尺寸而没办法显示而被丢弃。

# 图像的镜像变换

---

■ 图像镜像变换分为：

( 1 ) 图像水平镜像变换

( 2 ) 图像垂直镜像变换

深圳技术大学  
大数据与互联网学院  
曹劲舟 版权所有

# 图像的镜像变换

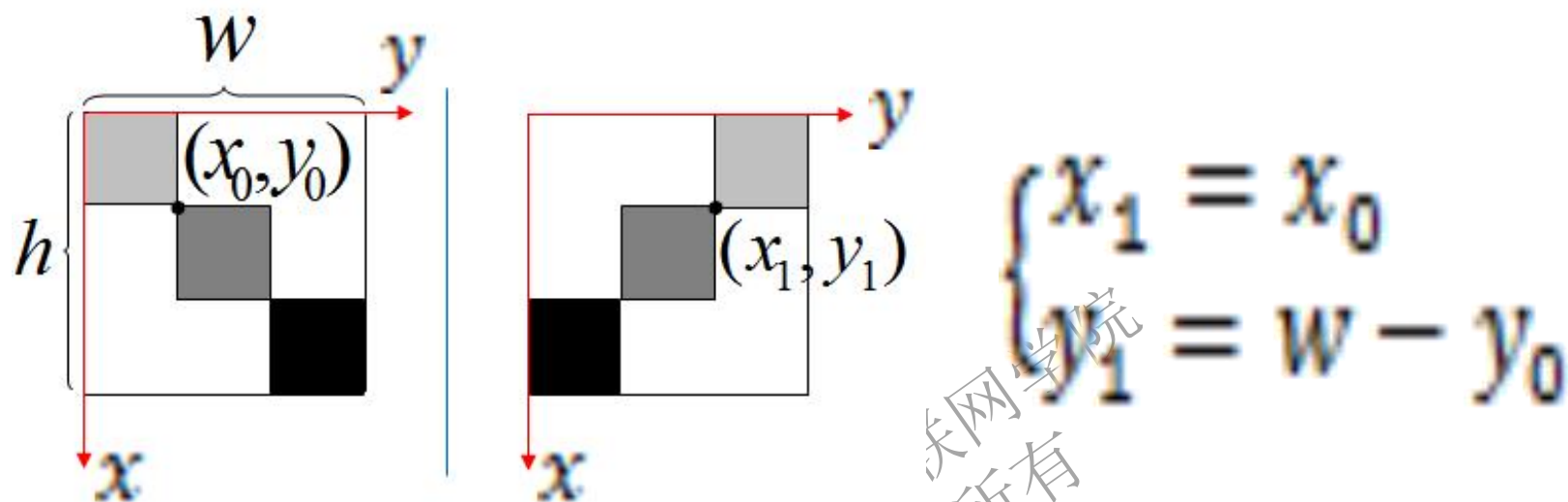
## ■ 图像水平镜像

■ 图像水平镜像是指以原图像为参照，使原图像和水平镜像结果图像与虚拟的垂直轴成对称关系。



# 图像的镜像变换

## ■ 图像水平镜像



图像水平镜像变换矩阵表示形式为：

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

# 图像的镜像变换

## ■水平镜像matlab程序

```
% 图像水平镜像matlab程序
```

```
clc; clear; close all;
```

```
img0=imread('lena.jpg');
```

```
[h,w]=size(img0); %获取矩阵的行数h和列数w
```

```
for x0=1:h %把原图像最(偏)右边的列移到新图像的最(偏)左边
```

```
    for y0=1:w
```

```
        result_img(x0,w-y0+1)=img0(x0,y0);
```

```
    end
```

```
end
```

```
subplot(1,2,1); imshow(img0); title('原图像'); %显示原图像
```

```
subplot(1,2,2); imshow(result_img); title('水平镜像结果图像');
```

# 图像的镜像变换

## ■ 图像水平镜像





# 图像的镜像变换

## ■ 图像垂直镜像

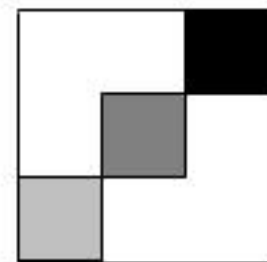
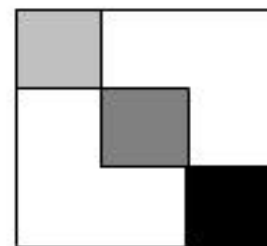
- 图像垂直镜像是指以原图像为参照，使原图像和垂直镜像结果图像与虚拟的水平轴成对称关系。

图像垂直镜像变换： $\begin{cases} x_1 = h - x_0 \\ y_1 = y_0 \end{cases}$

图像垂直镜像变换矩阵表示形式为：

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

原图像



垂直镜像结果图像

# 图像转置变换

## ■图像的转置（Image transpose）概念

- 图像转置变换是指将**图像的行坐标和列坐标互换**。
- 需要注意的是，图像转置和图像旋转是两回事，图像旋转是将图像旋转多少度。原图像不论是顺时针旋转 $90^\circ$ ，还是逆时针旋转 $90^\circ$  都不会得到图像转置后的结果。



转置前图像

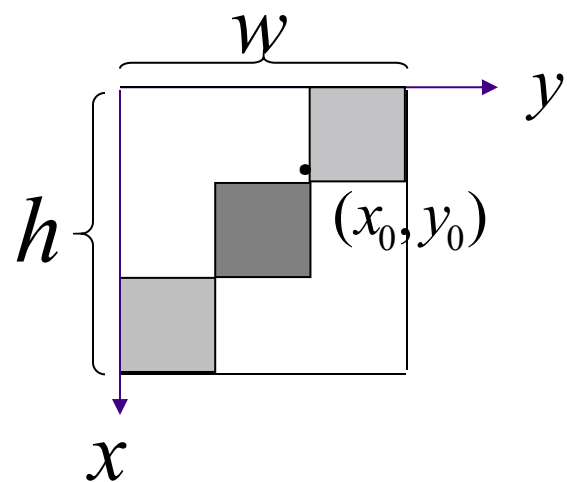


转置后图像

# 图像转置变换

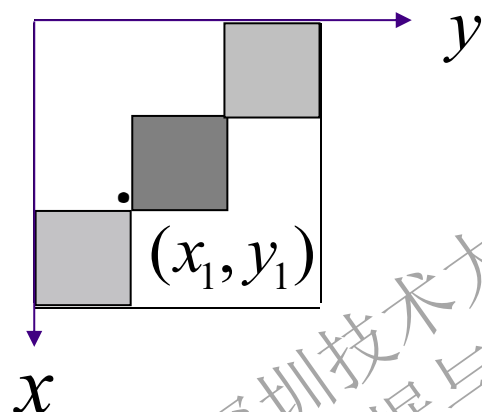
## ■ 图像转置表示及变换

原图像显示坐标



转置前图像

转置后图像显示坐标



转置后图像

$$\begin{cases} x_1 = y_0 \\ y_1 = x_0 \end{cases}$$

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

# 图像缩放

■ 图像缩放（image scaling）是指对图像进行缩小或放大，也即对数字图像的大小进行调整的过程。

$$\begin{cases} x_1 = r x_0 \\ y_1 = r y_0 \end{cases}$$

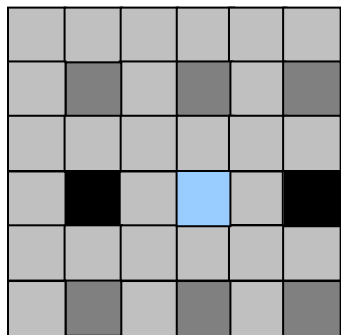
图像缩放的矩阵表示形式为：

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

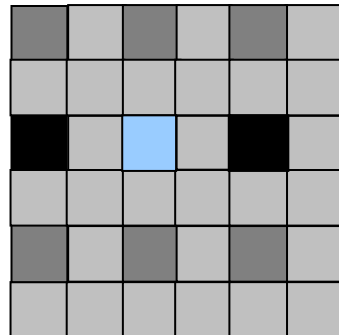
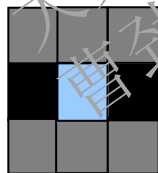
# 图像缩放

## ■ 图像缩小（下采样、降采样）

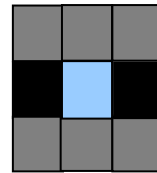
- 最简单的图像缩小方法是：将图像的行和列都缩小一半（保留原图像的**奇数行或偶数行和奇数列或偶数列**），从整体上看就是将原图像缩小到原来大小的四分之一。



仅取偶数行和偶数列像素缩小原图像



仅取奇数行和奇数列像素缩小原图像



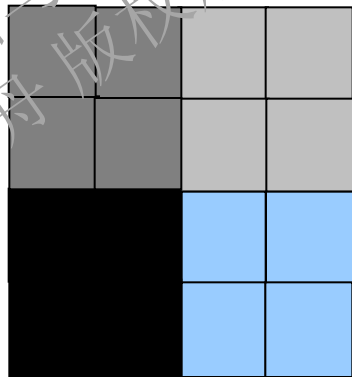
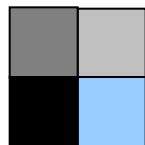
## ■放大图像

- 放大图像的目的一般是为了使放大后的图像更好地显示在更高分辨率的显示设备上。
- 放大图像的方法较多，也较复杂。典型的图像放大方法是最近邻域插值法和双线性插值放大图像方法。

## ■最近邻域插值法

### 1)按整数倍放大图像的最近邻域插值法

- 基本思想：将原图像中的每一个原像素原封不动地复制映射到放大后的新图像中的四个像素中。



## ■2)按非整数倍放大图像的最近邻域插值法

设已知有一个 $3 \times 3$ 的灰度图像如图(a)所示。按非整数倍数放大图像的最近邻域插值法将该图像放大为  $4 \times 4$ 的图像。

234 38 22

67 44 12

89 65 63

(a) 原图像

? ? ? ?

? ? ? ?

? ? ? ?

? ? ? ?

(b) 放大后的图像



## ■2)按非整数倍放大图像的最近邻域插值法

**设：**放大前的原图像在显示坐标的X方向和Y方向的坐标值分别用 $X_{old}$ 和 $Y_{old}$ 表示；图像高度（X方向）和宽度（Y方向）分别用 $H_{old}$ 和 $W_{old}$ 表示。

**放大后的目标图像在显示坐标的X方向和Y方向的坐标值分别用 $X_{new}$ 和 $Y_{new}$ 表示；图像高度（X方向）和宽度（Y方向）分别用 $H_{new}$ 和 $W_{new}$ 表示。**

**则最近邻域插值法的原图像和目标图像的坐标关系为**

$$\begin{cases} x_{old} = x_{new} * (h_{old} / h_{new}) \\ y_{old} = y_{new} * (w_{old} / w_{new}) \end{cases}$$

## 随堂作

解：对图（a）根据式（3.30）逐个地计算插值放大后的目标图像中从（0,1）至（3,3）的每个像素的值。

对于目标图像中坐标为（0,0）处的像素，因为有

$$x_{old} = x_{new} * (h_{old} / h_{new}) = 0 * (3/4) = 0,$$

$$y_{old} = y_{new} * (w_{old} / w_{new}) = 0 * (3/4) = 0.$$

也即，目标图像中位于（0,0）处的像素值应是原图像中位于（0,0）处的像素值，也即234。

对于目标图像中坐标为（0,1）处的像素，因为有

$$x_{old} = x_{new} * (h_{old} / h_{new}) = 0 * (3/4) = 0,$$

$$y_{old} = y_{new} * (w_{old} / w_{new}) = 1 * (3/4) = 0.75 \approx 1.$$

也即，目标图像中位于（0,1）处的像素值应是原图像中位于（0,1）处的像素值，也即38。

# 图像缩放

同理，可得目标图像中位于  $(0, 2)$  处和  $(0, 3)$  处的像素值都是22；

位于  $(1, 0)$  处、 $(1, 1)$  处、 $(1, 2)$  和  $(1, 3)$  处的像素值分别是67、44、12和12；

位于  $(2, 0)$  处、 $(2, 1)$  处、 $(2, 2)$  和  $(2, 3)$  处的像素值分别是89、65、63和63；

位于  $(3, 0)$  处、 $(3, 1)$  处、 $(3, 2)$  和  $(3, 3)$  处的像素值分别是89、65、63和63。

放大后的结果图像如图 (b) 所示。

## ■2)按非整数倍放大图像的最近邻域插值法

设已知有一个 $3 \times 3$ 的灰度图像如图(a)所示。按非整数倍数放大图像的最近邻域插值法将该图像放大为  $4 \times 4$ 的图像。

234	38	22
67	44	12
89	65	63

(a) 原图像

234	38	22	22
67	44	12	12
89	65	63	63
89	65	63	63

(b) 放大后的图像

## ■ (2) 双线性插值放大图像方法

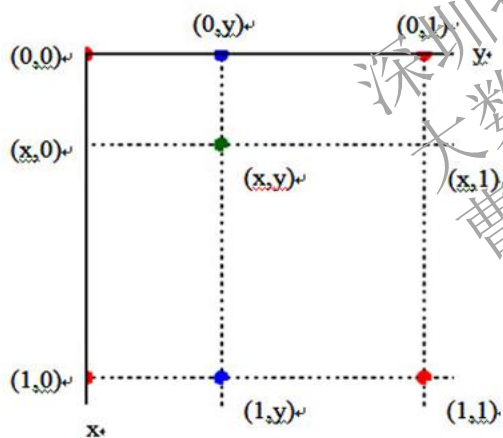
- 双线性插值法通过计算输出像素被映射到**输入图像中4个像素之间的非整数位置的那个像素点的灰度值**，并将具有该灰度值的像素点插入到该位置，来实现对输入图像的放大。

## (2) 双线性插值放大图像方法

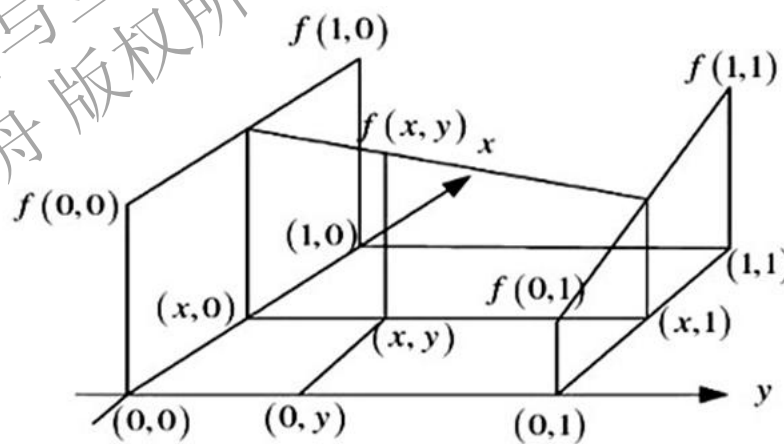
已知像素点  $(0, 0)$ 、 $(0, 1)$ 、 $(1, 0)$  和  $(1, 1)$ ，要插值的点为  $(x, y)$ 。双线性插值的基本思路是：首先在  $x$  方向上线性插值，也即在  $(0, 0)$  和  $(1, 0)$  两个点之间插入点  $(x, 0)$ ，在  $(0, 1)$  和  $(1, 1)$  两个点之间插入点  $(x, 1)$ ；然后在  $y$  方向线性插值，也即通过计算出的点  $(x, 0)$  和  $(x, 1)$ ，在  $y$  方向上插值计算出点  $(x, y)$ 。

同时，在已知  $f(0, 0)$ 、 $f(0, 1)$ 、 $f(1, 0)$  和  $f(1, 1)$  的情况下，按如下的双线性方程计算出  $f(x, y)$  值。

$$f(x, y) = ax + by + cxy + d \quad (3.31)$$



(a) 插值点关系示意图



(b) 插值运算示意图



首先，在  $x$  方向上作线性插值，对上端的两个点  $(0, 0)$  和  $(1, 0)$  进行线性插值可得：

$$f(x, 0) = f(0, 0) + x[f(1, 0) - f(0, 0)] \quad (3.32)$$

同理，对低端的两个点  $(0, 1)$  和  $(1, 1)$  进行线性插值可得：

$$f(x, 1) = f(0, 1) + x[f(1, 1) - f(0, 1)] \quad (3.33)$$

然后，在  $y$  方向上作线性插值，有：

$$f(x, y) = f(x, 0) + y[f(x, 1) - f(x, 0)] \quad (3.34)$$

将式 (3.32) 和式 (3.33) 代入式 (3.34)，有：

$$\begin{aligned} f(x, y) = & f(0, 0) + x[f(1, 0) - f(0, 0)] + \\ & + y[f(0, 1) + x[f(1, 1) - f(0, 1)] - f(0, 0) - x[f(1, 0) - f(0, 0)]] \end{aligned}$$

首先，在  $x$  方向上作线性插值，对上端的两个点  $(0, 0)$  和  $(1, 0)$  进行线性插值可得：

$$f(x, 0) = f(0, 0) + x[f(1, 0) - f(0, 0)] \quad (3.32)$$

同理，对低端的两个点  $(0, 1)$  和  $(1, 1)$  进行线性插值可得：

$$f(x, 1) = f(0, 1) + x[f(1, 1) - f(0, 1)] \quad (3.33)$$

然后，在  $y$  方向上作线性插值，有：

$$f(x, y) = f(x, 0) + y[f(x, 1) - f(x, 0)] \quad (3.34)$$

将式 (3.32) 和式 (3.33) 代入式 (3.34)，有：

$$\begin{aligned} f(x, y) = & f(0, 0) + x[f(1, 0) - f(0, 0)] + \\ & + y[f(0, 1) + x[f(1, 1) - f(0, 1)] - f(0, 0) - x[f(1, 0) - f(0, 0)]] \end{aligned}$$



整理上式，即可得双线性插值公式为：

$$f(x,y)=[f(1,0)-f(0,0)]x+[f(0,1)-f(0,0)]y+ \\ +[f(1,1)+f(0,0)-f(0,1)-f(1,0)]xy+f(0,0) \quad (3.35)$$

基于上述算法原理，进行二次线性插值的结果图像的过程如下图所示：

