第2章　图像的基本知识及运算

- 颜色模型
- 图像文件格式
- 图像像素间的关系
- 图像间的基本代数运算
- 图像间的基本几何运算

1. 颜色模型

为了科学地定量描述和使用颜色，人们提出了各种颜色模型。

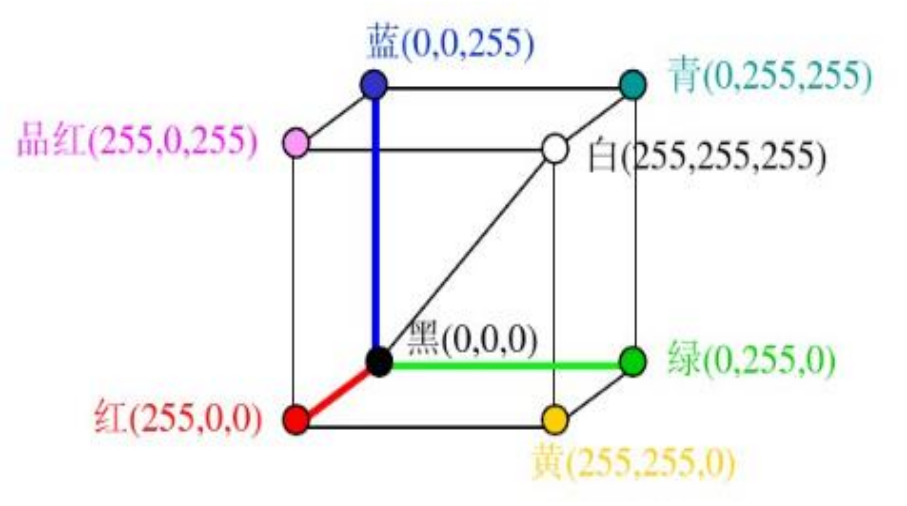
颜色模型是颜色在三维空间中的排列方式。

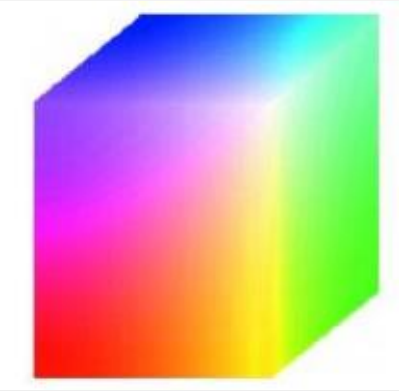
- RGB
- CMY
- YUV
- HSI

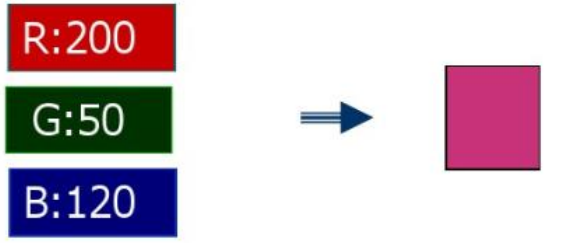
RGB彩色模型

- RGB颜色模型是图像成像、显示、打印等设备的基础，它是我们使用最多、最熟悉的颜色模型。它采用三维直角坐标系，红、绿、蓝为原色，各个原色混合在一起可以产生复合色。

面向硬件设备常用的模型



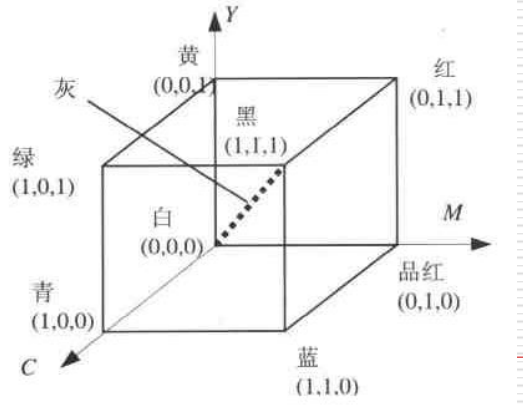




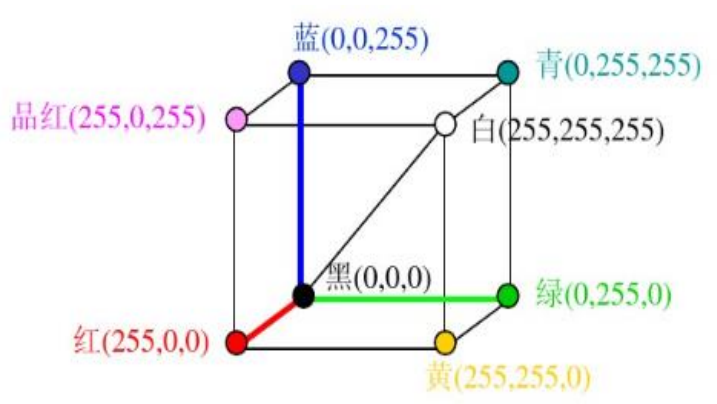

Original
Red
Green
Blue

CMY模型

- CMY-用在印刷行业 ， CMY分别是青色（Cyan）、品红(Magenta)、黄色(Yellow)三种油墨色
- CMY常用于从白光中滤去某种颜色，又被称为减性原色系统。
- CMY颜色模型与RGB颜色模型几乎完全相同。差别仅仅在于前者的原点为白，而后者的原点为黑。前者是定义在白色中减去某种颜色来定义一种颜色，而后者是通过从黑色中加入颜色来定义一种颜色。



CMY模型



RGB模型

CMY模型与RGB模型之间关系

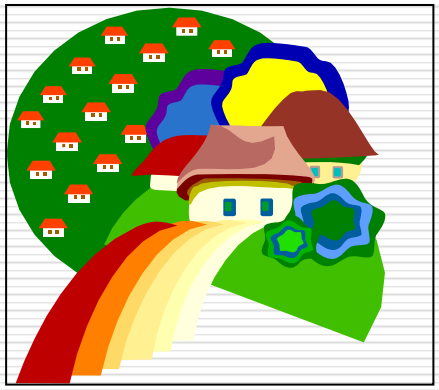
$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

RGB和CMY值都归一化到[0, 1]

YUV彩色空间

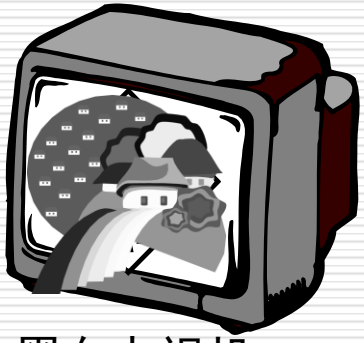
- 一种彩色传输模型，主要用于彩色电视信号传输标准
- Y—黑白亮度分量，U, V—彩色信息用以显示彩色图像

彩色电视信号



Y,U,V

Y



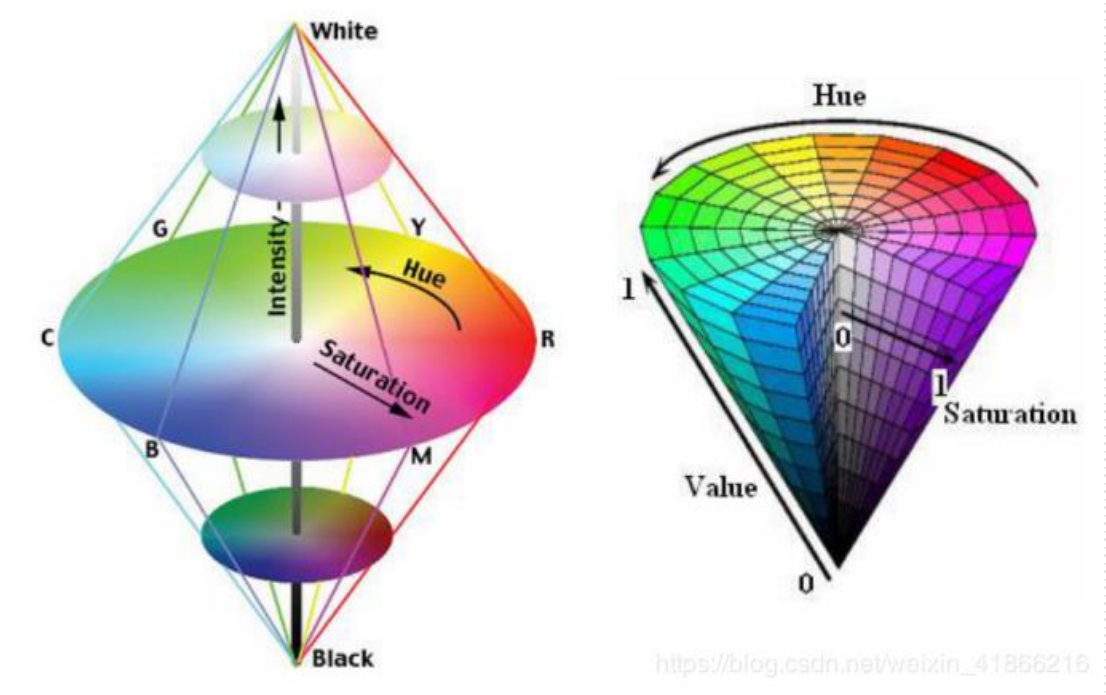
黑白电视机

与RGB之间的转换关系：

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.147 & -0.287 & 0.436 \\ 0.615 & -0.515 & -0.10 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

HSI彩色空间

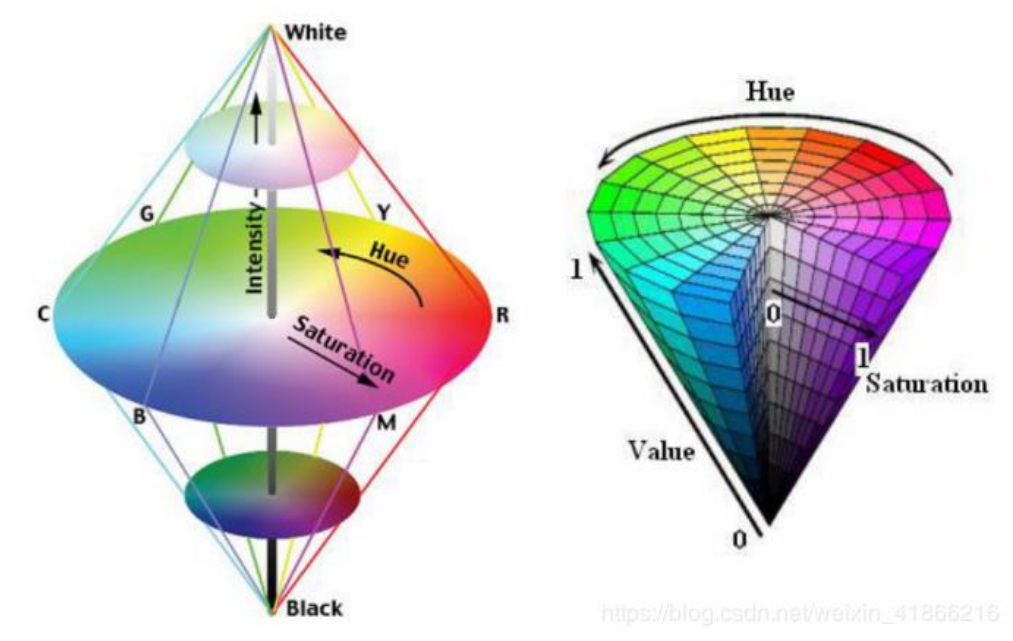
用色调(Hue)**、饱和度**(Saturation)**和强度**(Intensity)**来表示某个颜色。**



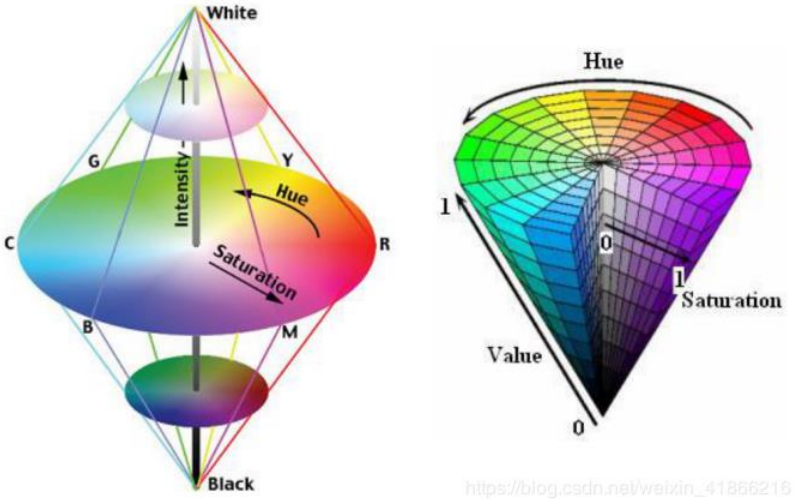
HSI 模型　　HSV 模型

色调H

- 描述纯色的属性（如红色、黄色等），一圈360^0表示不同的颜色。

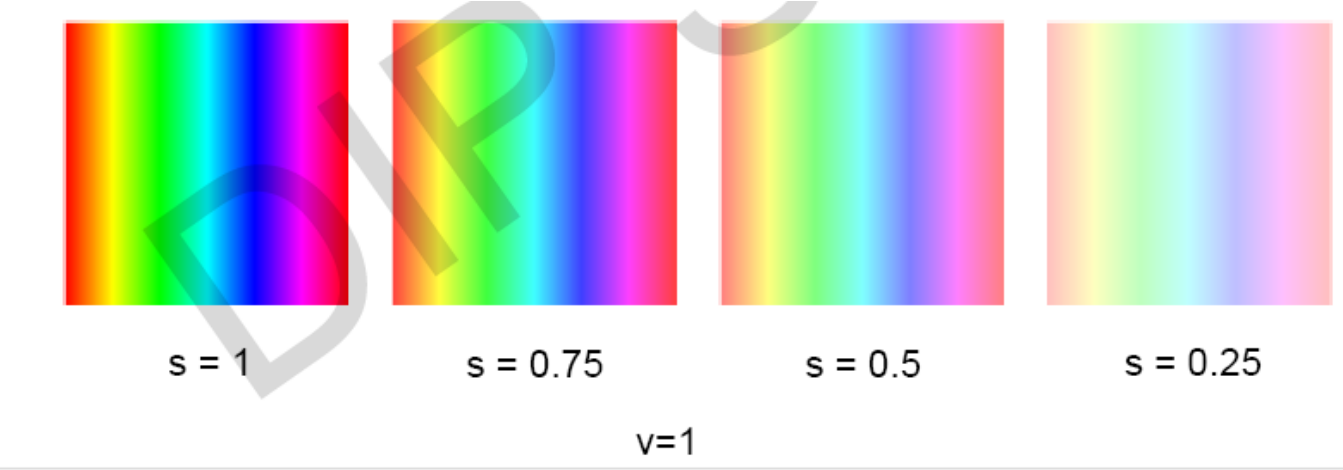


HSI 模型　　HSV 模型



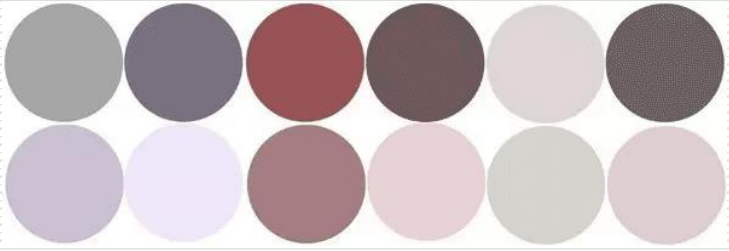
饱和度 Saturation

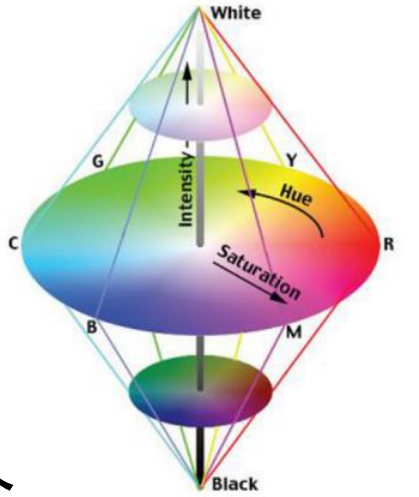
- 表示颜色的深浅程度，从内到外，颜色越来越鲜艳。
- 饱和度下降，感觉褪色。
- 色调和饱和度，通称为色度。

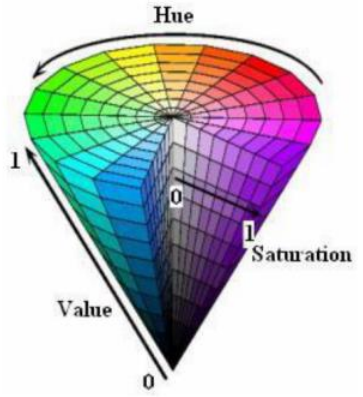


不同饱和度的绿色外套：

饱和度低的莫兰迪色系：

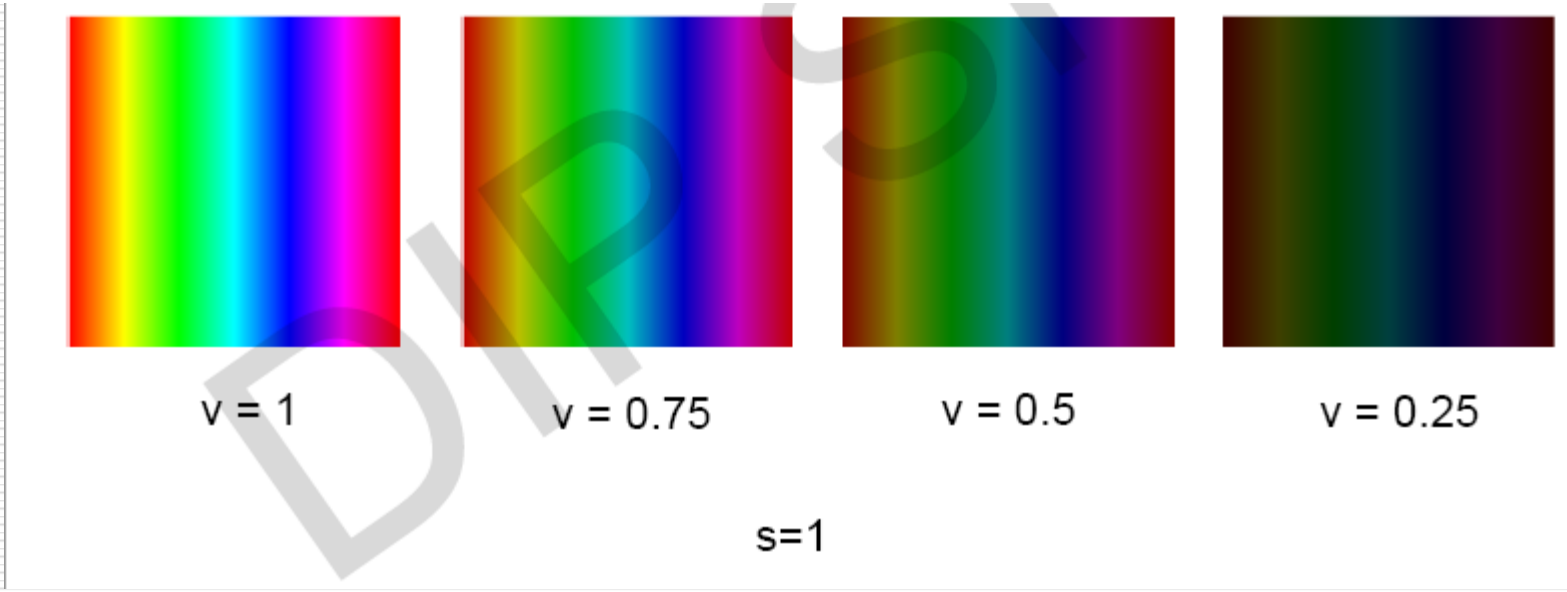






亮度 Intensity

- 体现了无色的光强度概念，是一个主观的描述。
- 从下到上，亮度增强，越来越亮。



图像的HSI颜色表示

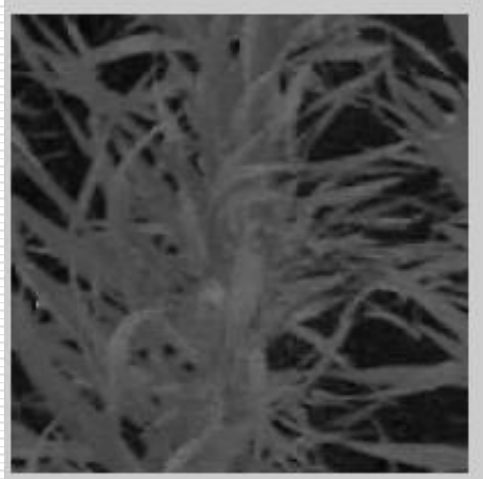
光照明暗给物体颜色带来的直接影响就是亮度分量I，HSI模型将亮度I与反映色彩本质特性的两个参数—色调H和饱和度S——分开 。

优点

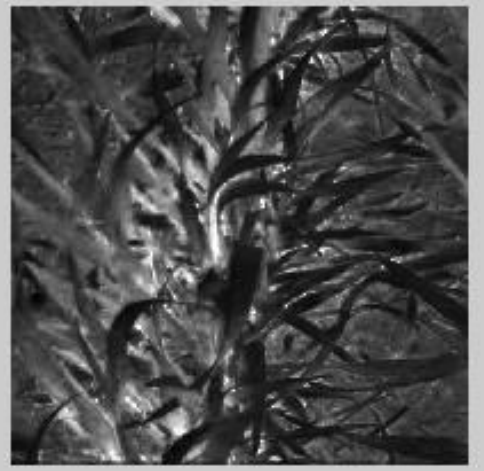
- ✓ 避免颜色受到光照明暗(I)等条件的干扰
- ✓ 仅仅分析反映色彩本质的色调和饱和度
- ✓ 广泛用于计算机视觉、图像检索和视频检索

原图像

色度图H



饱和度图S



强度图I

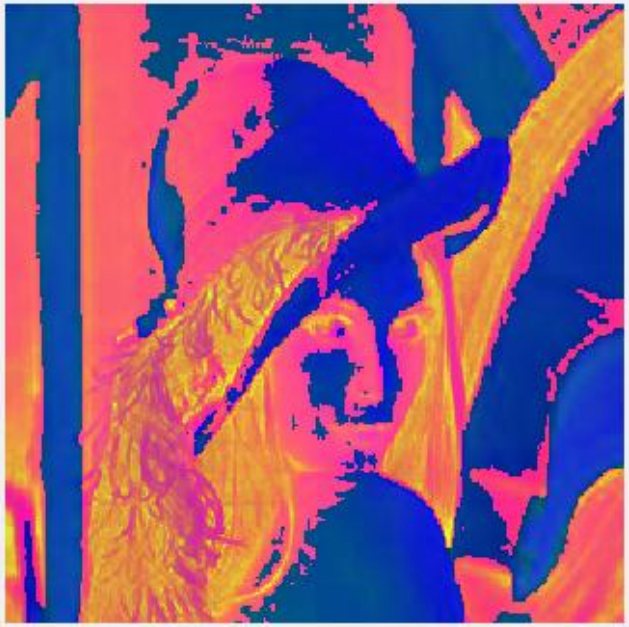
RGB 转换至 HSI 的几种常见方法

算法	色调	饱和度	亮度
算法 1 几何推导法	$H=\begin{cases}\theta, & G\geq B\\ 2\pi-\theta, & G<B\end{cases}$ $where\ \theta=\cos^{-1}\left(\frac{(R-G)+(R-B)}{2\sqrt{(R-G)^2+(R-B)(G-B)}}\right)$	$S=1-\frac{3\min(R,G,B)}{R+G+B}$	$I=\frac{R+G+B}{3}$
算法 2 坐标变换法	$H=\begin{cases}\theta, & G\geq B\\ \theta+\pi, & G<B\end{cases}$ $where\ \theta=\frac{\pi}{2}-\tan^{-1}\left(\frac{2R-G-B}{\sqrt{3}(G-B)}\right)$	$S=\frac{2}{\sqrt{6}}\times\sqrt{(R-G)^2+(R-B)(G-B)}$	$I=\frac{R+G+B}{\sqrt{3}}$
算法 3 分段定义法	$H=\begin{cases}\frac{\pi}{3}\times\frac{G-B}{Max-Min}, & if\ \ Max=R\\ \frac{\pi}{3}\times\frac{B-R}{Max-Min}+\frac{2\pi}{3}, & if\ \ Max=G\\ \frac{\pi}{3}\times\frac{R-G}{Max-Min}+\frac{4\pi}{3}, & if\ \ Max=B\end{cases}$ $H=H+2\pi,\quad if\ \ H<0$	$S=\begin{cases}\frac{Max-Min}{Max+Min}, & 0<I\leq\frac{1}{2}\\ \frac{Max-Min}{2-(Max+Min)}, & I>\frac{1}{2}\end{cases}$	$I=\frac{1}{2}(Max+Min)$
算法 4 Bajon 近似 算法	$H=\begin{cases}\frac{G-B}{3(R+G-2B)}, & if\ \ Min=B\\ \frac{B-R}{3(G+B-2R)}+\frac{1}{3}, & if\ \ Min=R\\ \frac{R-G}{3(R+B-2G)}+\frac{2}{3}, & if\ \ Min=G\end{cases}$ $H=H*2\pi$	同算法 1	同算法 1
算法 5 标准模型法	同算法 3	$S=Max-Min$	同算法 3

注：Max=max(R, G, B), Min=min(R, G, B)。公式中的 RGB 在计算前均归一化至[0，1]区间。

RGB与HSI 颜色空间的图像比较：（利用算法1）

RGB空间原图像



变换后的HSI 空间图像

RGB与HSV 颜色空间的图像比较：

RGB空间原图像

变换后的HSV 空间图像

2 图像文件格式

- 熟悉常用的图像文件名
- 了解它们有什么特性和适用的场合

文件格式	说　明
BMP	是Microsoft公司在windows环境中普遍采用的一种与设备无关的图像格式。
JPEG	由CCITT和ISO联合组成的一个图像专家组制定的静态图像压缩国际标准，也是应用最广泛的图像格式之一，所有格式中压缩率最高的格式。
GIF	不仅可以是一张图像，也可以是动画，网上很多小动画都是GIF格式。
PNG	与JPG格式类似，支持图像透明，支持Alpha通道调节图像的透明度。
TIFF	是由Aldus和Microsoft公司为扫描仪和桌上出版系统开发的一种通用的图像文件格式，图像格式复杂。

数字图像有多种存储格式，每种格式一般由不同的开发商支持。因此，要进行图像处理，必须了解图像文件的格式， 即图像文件的数据构成。

图像的一般结构：包含有文件头、文件体和文件尾等三部分

文件头：文件类型、文件制作者、制作时间、版本号、 文件大小 软件ID、软件版本号、图像分辨率、图像尺寸、图像深度、彩色类型、编码方式、压缩算法

文件体：图像数据、彩色变换表

文件尾：用户名、注释、开发日期、工作时间

Bmp图像文件格式

BMP(Bitmap),意为“位图”。BMP格式的图像文件是微软公司特为Windows环境应用图像而设计的，在Windows环境下运行的所有图像处理软件都支持这种格式。

BMP格式的主要特点有：

- BMP位图文件默认的文件扩展名是“.bmp”。
- 一般情况下，BMP格式的图像是非压缩格式。
- 可以多种彩色模式保存图像．如16色、256色、24bit真彩色，甚至32bit真彩色。

Bmp文件格式

BMP文件存储结构的格式可以在Windows中的WINGDI.h文件中找到定义。

BMP文件总体上由4部分组成：位图文件头、位图信息头、调色板和图像数据

位图文件头（bitmap-file header）
位图信息头（bitmap-information header）
调色板（color table）
位图数据（bitmap-data）

表 2－1　BMP 图像文件结构

文件部分	属　性	说　明
BITMAPFILEHEADER（位图文件头）	BfType	文件类型，必须是 0x424D，即字符串"BM"
	bfSize	指定文件大小，包括这 14 个字节
	bfReserved1	保留字，不用考虑
	bfReserved2	保留字，不用考虑
	bfOffBits	从文件头到实际位图数据的偏移字节数
BITMAPINFOHEADER（位图信息头）	bfSize	该结构的长度，为 40
	biWidth	图像的宽度，单位是像素
	biHeight	图像的高度，单位是像素
	biplanes	位平面数，必须是 1，不用考虑
	biBitCount	指定颜色位数，1 为二值，4 为 16 色，8 为 256 色，16、24、32 为真彩色
	biCompression	指定是否压缩，有效的值为 BI_RGB，BI_RLE8，BI_RLE4，BI_BITFIELDS
	biSizeImage	实际的位图数据占用的字节数
	biXPelsPerMeter	目标设备水平分辨率，单位是每米的像素数
	biYPelsPerMeter	目标设备垂直分辨率，单位是每米的像素数
	biClrUsed	实际使用的颜色数，若该值为 0，则使用颜色数为 2 的 biBitCount 次方种
	biClrImportant	图像中重要的颜色数，若该值为 0，则所有的颜色都是重要的
Palette（调色板）	rgbBlue	该颜色的蓝色分量
	rgbGreen	该颜色的绿色分量
	rgbRed	该颜色的红色分量
	rgbReserved	保留值
ImageData（位图数据）	像素按行优先顺序排序，每一行的字节数必须是 4 的整倍数	

1. 位图文件头

第1部分为位图文件头BITMAPFILEHEADER，是一个结构体类型，该结构的长度是固定的，为14个字节。其定义如下：

```
typedef struct tagBITMAPFILEHEADER {
    WORD bfType;      //文件类型，必须是0x42D，即字符串“BM”
    DWORD bfSize;     // 位图文件大小
    UINT bfReserved1;     //保留字，暂不用
    UINT bfReserved2;     //保留字，暂不用
    DWORD bfOffBits;    //从文件头到位图数据的偏移字节数
} BITMAPFILEHEADER;
```

2. 位图信息头

位图信息头BITMAPINFOHEADER,也是一个结构体类型。该结构的长度也是固定的，为40个字节。

```
typedef struct tagBITMAPINFOHEADER{
    DWORD  biSize;            //该结构的长度，40个字节
    LONG   biWidth;           //图像的宽度，单位是像素
    LONG   biHeight;          //图像的高度，单位是像素
    WORD   biPlanes;          //必须是1
    WORD   biBitCount         //颜色位数，如1，4，8，24
    DWORD  biCompression;     //压缩类型，如BI_RGB,BI_RLE4
    DWORD  biSizeImage;       //实际位图数据占用的字节数
    LONG   biXPelsPerMeter;   //水平分辨率
    LONG   biYPelsPerMeter;   //垂直分辨率
    DWORD  biClrUsed;         //实际使用的颜色数
    DWORD  biClrImportant;    //重要的颜色数
} BITMAPINFOHEADER;
```

3. 调色板

调色板实际上是一个RGBQUAD结构的数组。RGBQUAD结构是一个结构体类型，占4个字节。其定义如下：

```
typedef struct tagRGBQUAD{
    BYTE    rgbBlue;          //该颜色的蓝色分量
    BYTE    rgbGreen;         //该颜色的绿色分量
    BYTE    rgbRed;           //该颜色的红色分量
    BYTE    rgbReserved;      //保留值，不考虑
} RGBQUAD;
```

注：有些位图不需要调色板，如对于biBitCount=24的真彩色图，由于每像素3个字节中分别代表了R、G、B三分量的值，因此不需要调色板。它们的BITMAPINFOHEADER后面直接是位图数据。

4. 位图数据

（1）用到调色板的位图，图像数据是该像素颜色在调色板中的索引值。

（2）真彩色图像，图像数据就是实际的R、G、B值（三个分量的存储顺序是B、G、R）。

注意：

BMP文件的数据存放是从下到上，从左到右的。


位图数据
B G R B G R B G R B G R ……
最后一行
第一列像素
的颜色值.
最后一行
第二列像素
的颜色值.

3 图像像素间的关系

图像是由其基本单元----像素组成的，像素在图像空间是按某种规律排列的，互相之间有一定的联系。

主要介绍数字图像中基本而又重要的像素间的关系 。

像素间的一些基本关系：

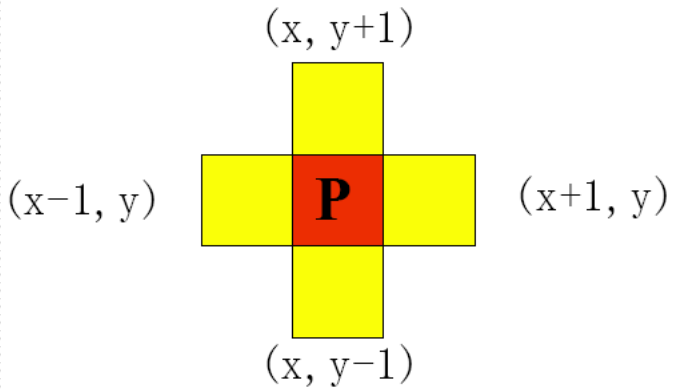
- **相邻像素：**
  - **4**邻域
  - **D**邻域
  - **8**邻域
- **连通性**
  - **4**连通
  - **8**连通
  - **m**连通
- **距离**

3.1 像素的邻域

对一个像素来说，与它关系最密切的常是它的近邻像素，它们组成该像素的邻域。根据对一个坐标为(x, y)的像素p的近邻像素的不同定义，可以得到由不同近邻像素所组成的不同的相应邻域。

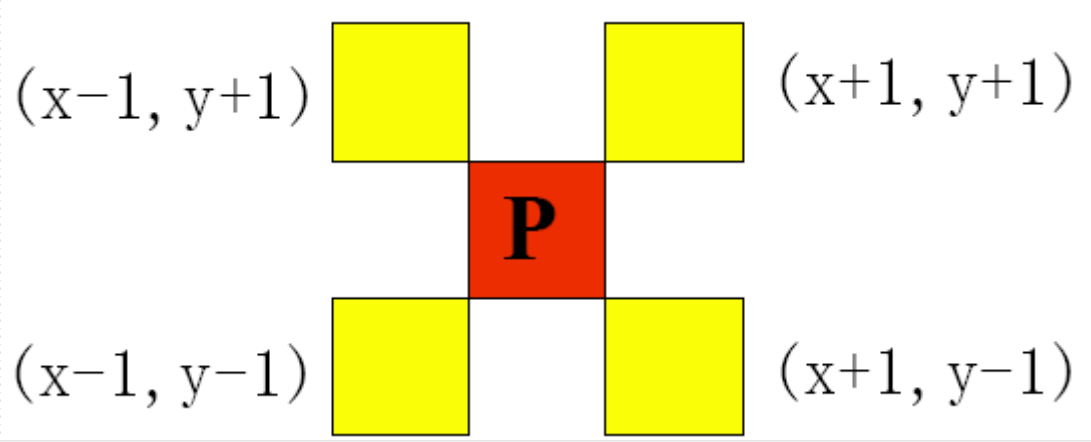
（1）4-邻域$N_4(p)$

对坐标为(x, y)的像素p，它有4个水平和垂直的近邻像素：(x+1, y)，(x-1, y)，(x, y+1)，(x, y-1)。这些像素组成p的4-邻域，记为$N_4(p)$。



（2）D-邻域$N_D(p)$

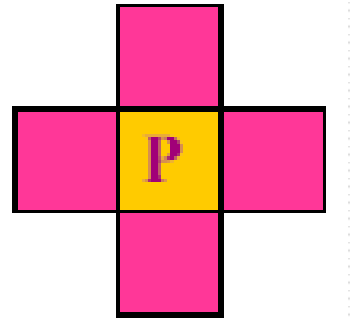
像素点p(x, y)的4个对角近邻像素(x+1, y+1)，(x+1, y-1)，(x-1, y+1)，(x-1, y-1)，这些像素组成p的D-邻域，记为$N_D(p)$



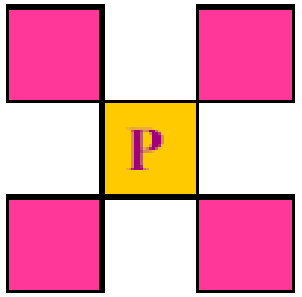
（3）8-邻域$N_8(p)$

像素p的D-邻域再加上p的4-邻域像素则合称为p的8—邻域，记为$N_8(p)$.

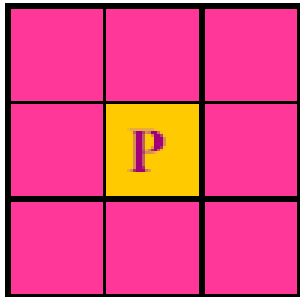
$$N_8(p) = N_4(p) + N_D(p)$$



$N_4(p)$



$N_D(p)$



$N_8(p)$

3.2 连通性

- 像素间的连通性是描述图像中目标边界和区域元素时的一个重要概念。

- 连通：空间上邻接且像素灰度值相似
- 两个像素连通的两个必要条件：
  - 两个像素的位置是否相邻
  - 两个像素的灰度值是否满足某个特定的相似性准则。

    灰度值相同或同在一个灰度值集合中

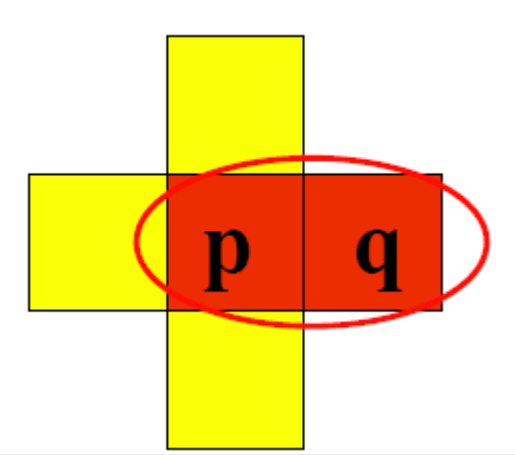
用V定义连通的灰度值集合。如在一幅二值图中，为考虑两个灰度值为1的像素之间的连通性，则V={1}。而在一幅有32个灰度级的灰度图中，假设考虑具有灰度值在8到16之间像素的连通性，则V={8，9，...，15，16}。

0	1	1
0	1	0
0	0	1

V={1}

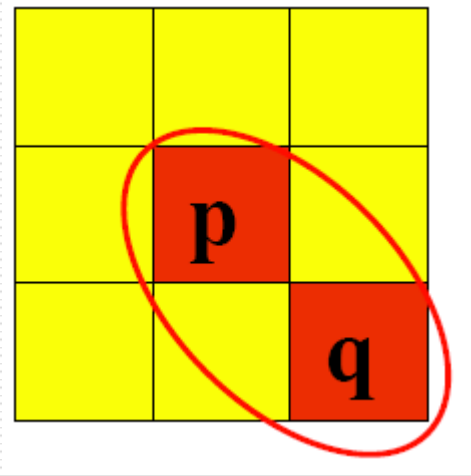
像素的连通性

（1）4-连通：2个像素p和q在V中取值且q在集合$N_4(p)$中，则这两个像素为4-连通；



4-连通

（2）8-连通：2个像素*p*和*q*在V中取值且*q*在 $N_8(p)$ 中，则它们为8-连通；



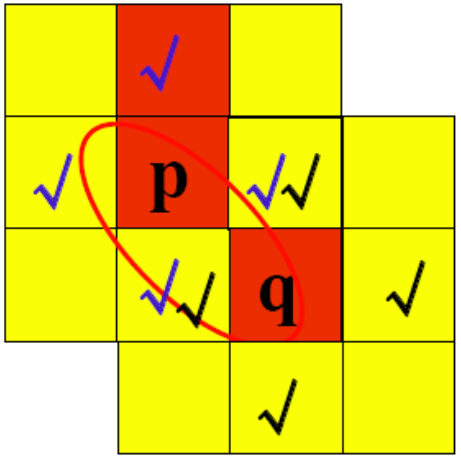
（3）m-连通

- 对于具有值V的像素p和q，如果：

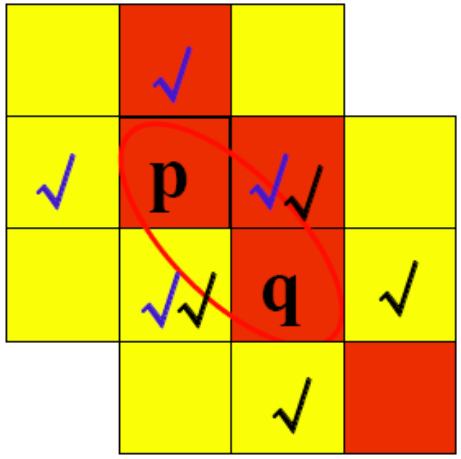
  I. q在集合$N_4(p)$中，或

  II. q在集合$N_D(p)$中，并且$N_4(p)$与$N_4(q)$的交集为空（没有值V的像素）

  则称这两个像素是m连通的，即4连通和D连通的混合连通。



是m连通



不是m连通

3.3 像素间的距离测量

- 像素之间距离的定义
- 欧氏距离定义
- D_4距离（城市距离）定义
- D_8距离（棋盘距离）定义

像素在空间的接近程度可以用像素之间的距离来测量。为测量距离，需要定义距离量度函数。

给定3个像素p，q，r，坐标分别为(x, y), (s, t), (u, v)，如果下列条件满足的话，称函数D是距离量度函数：

（1）D(p,q)>=0（D(p,q)=0当且仅当p=q），

两点之间的距离 大于等于0；

（2）D(p,q)=D(q,p)，　距离与方向无关；

（3）D(p,r)<=D(p,q)+D(q,r)，两点之间的直线距离最短。

最常用的距离：

(1). **欧几里德距离（欧氏距离）**

点p(x, y)和q(s, t)之间的Euclidean距离定义为

$$D_e(p,q)=\sqrt{(x-s)^2+(y-t)^2}$$

与(x, y)的欧氏距离小于等于某个值d的像素都包含在以(x, y)为中心，以d为半径的圆中。

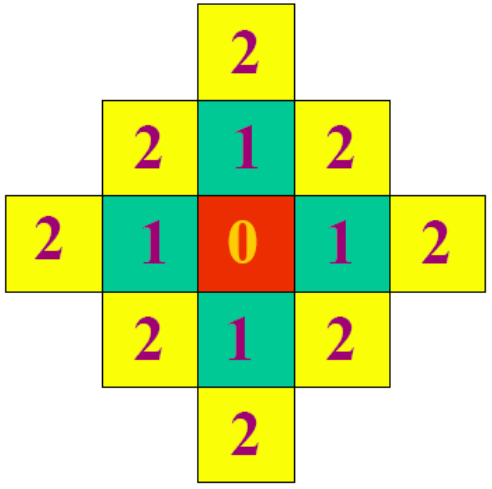
(2). 城市距离D_4

点***p***和***q***之间的D_4 距离，即城市**(city-block)**距离,定义为:

$$D_4(p,q)=|x-s|+|y-t|$$

与(x, y)的D_4距离小于等于某个值d的像素组成以(x, y)为中心的菱形。

- 例如，与点(x, y)（中心点）D_4距离小于等于2的像素，形成右边固定距离的轮廓
- 具有D_4 = 1的像素是(x, y)的4邻域



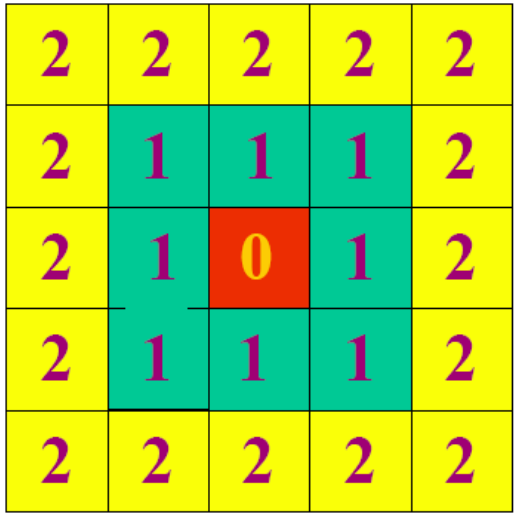
(3). 棋盘距离D_8

点*p*和*q*之间的D_8距离，即棋盘(chessboard)距离，定义为：

$$D_8(p,q)=\max\left(|x-s|,|y-t|\right)$$

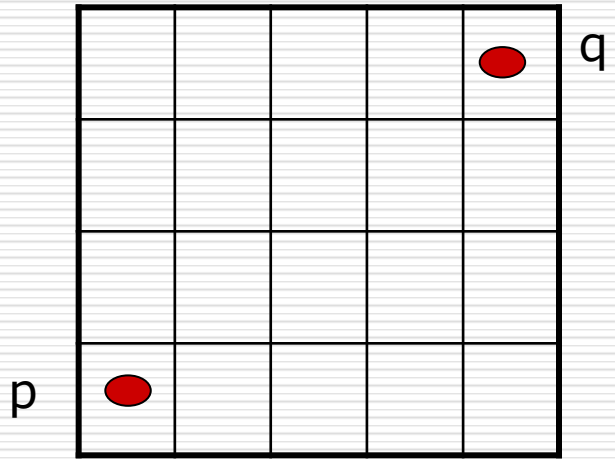
与(x, y)的D_8距离小于等于某个值d的像素组成以(x, y)为中心的正方形。

- 例如，与点(x, y)（中心点）D_8距离小于等于2的像素，形成右边固定距离的轮廓
- 具有D_8 = 1的像素是(x, y)的8邻域



例：距离计算示例

求像素p和q之间的距离。



$D_E=5$

$D_4=7$

$D_8=4$

- 欧氏距离给出的结果准确，但由于计算时需要进行平方和开方运算，计算量大。
- 城市距离和棋盘距离均为非欧氏距离，计算量小，但有一定的误差。

4　数字图像处理中的基本运算

图像的基本运算可分为图像的基本代数运算和基本几何运算两种。

4.1基本代数运算

- 代数运算是指对图像中参与运算的像素几何位置不变化，图像之间灰度级的四则运算。也就是对两幅输入图像进行点对点的加、减、乘和除计算而得到输出图像的运算 。

如果将待运算的两幅相同尺寸的图像f(x,y)和g(x,y)以矩阵的形式表达，即

$$f(x,y)=\begin{bmatrix} f_{11} f_{12} \ldots f_{1N} \\ f_{21} f_{22} \ldots f_{2N} \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ f_{M1} f_{M2} \ldots f_{MN} \end{bmatrix}$$

$$g(x,y)=\begin{bmatrix} g_{11} & g_{12} & \cdots & g_{1N} \\ g_{21} & g_{22} & \cdots & g_{2N} \\ \cdots & \cdots & \cdots & \\ g_{M1} & g_{M2} & \cdots & g_{MN} \end{bmatrix}$$

则两幅图像间的代数运算可以表示为如下:

$$s(x,y)=f(x,y)+g(x,y)$$
$$s(x,y)=f(x,y)- g(x,y)$$
$$s(x,y)=f(x,y)\times g(x,y)$$
$$s(x,y)=f(x,y) / g(x,y)$$

其中，f(x, y)和g(x, y)是输入图像，而s(x, y)是输出图像。

4.2 几种代数运算的应用

□ 加运算应用—求平均降噪

在采集实际图像的时候，由于各种不同的原因，常有一些干扰或噪声混入到最后采集的图像中。

图像相加的一个重要应用是对同一场景的多帧图像求平均值，因此被经常用来有效地降低加性随机噪声的影响。

假设我们有M帧待分析的图像序列，则第k帧图像表示为

噪声图像

$$f_k(x,y) = s(x,y) + n_k(x,y)$$

理想图像

假设噪声图像具有零均值的统计特性以及各帧独立，即

$$E\{n_k(x,y)\} = 0$$

$$E\{n_j(x,y) + n_k(x,y)\} = E\{n_j(x,y)\} + E\{n_k(x,y)\}$$

$$E\{n_j(x,y)n_k(x,y)\} = E\{n_j(x,y)\}E\{n_k(x,y)\}$$

则可以通过将一系列采集的图像$\{f_i(x,y)\}$相加来消除噪声。

对于图像的任意点，定义功率信噪比为

$$\eta_1(x,y)=\frac{s^2(x,y)}{E\{n^2(x,y)\}}$$

对M帧图像进行平均，则有

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

则输出图像的信噪比为

$$\eta_2(x,y)=\frac{s^2(x,y)}{E\{[(\frac{1}{M})\sum_{i=1}^{M}n_i(x,y)]^2\}}\Rightarrow\eta_2(x,y)=M\eta_1(x,y)$$

对M帧图像进行平均，使输出图像中每一点的信噪比提高了M倍。


原始图像


噪声图像


2幅平均


6幅平均


12幅平均

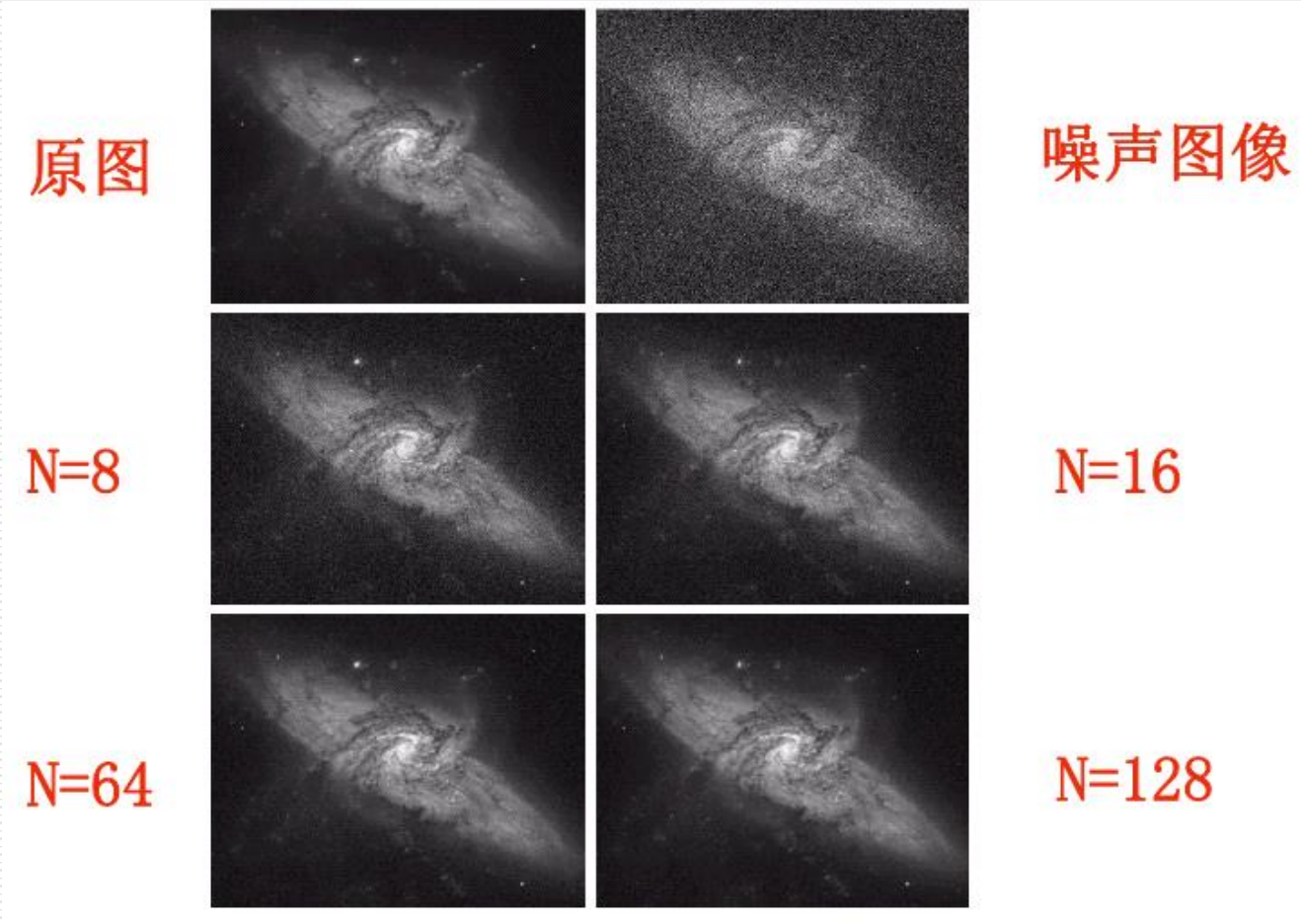

原始图像


噪声图像


2幅平均


16幅平均

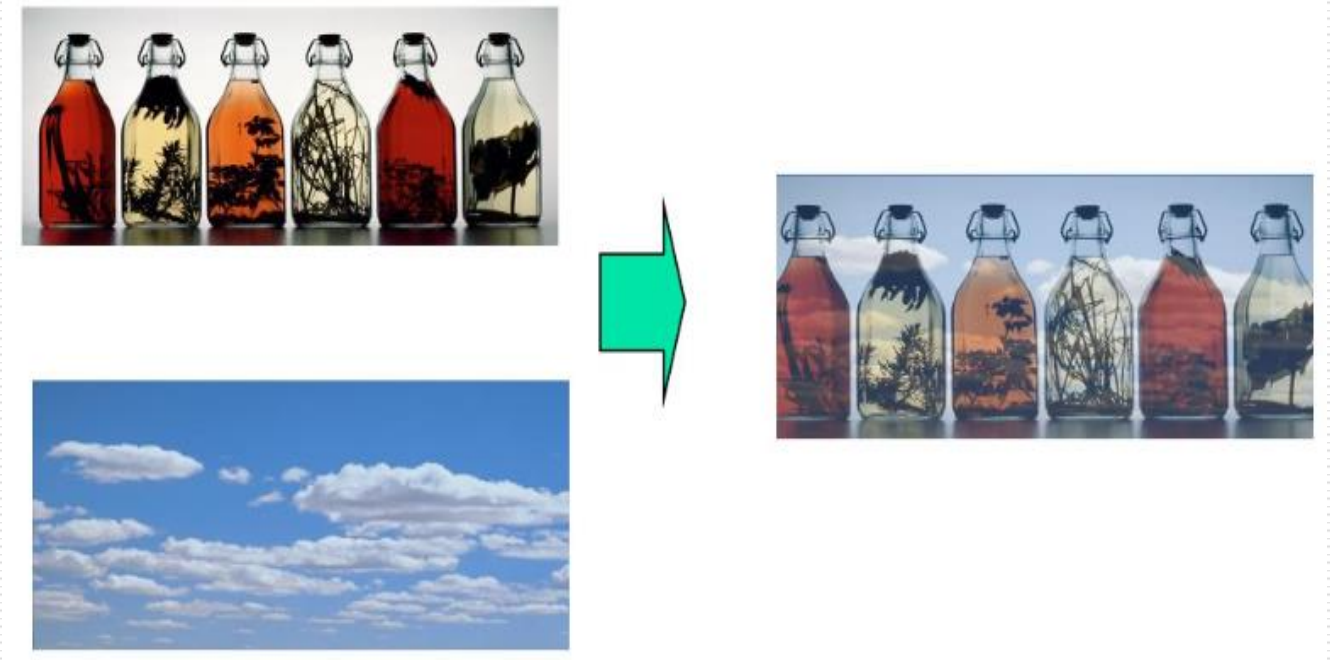
多幅图像求平均降噪



去除叠加性噪声----星系图

□ 加运算应用—生成图像叠加效果

图像相加也可用于将一幅图像的内容叠加到另一幅图像上去，生成图像叠加效果，可以得到图像合成的效果。

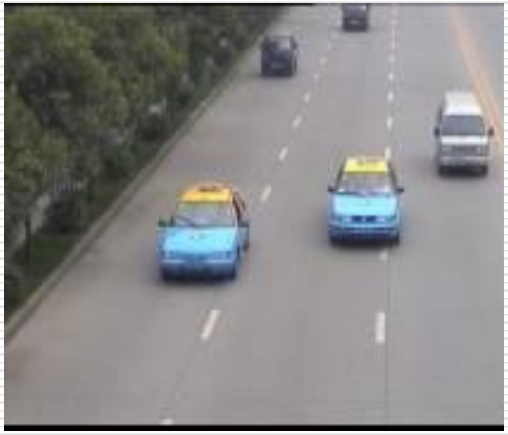


□ 减运算应用—序列图像求运动目标

设有图像f(x, y)和h(x, y)，对它们进行相减运算，可把两图的差异显示出来：

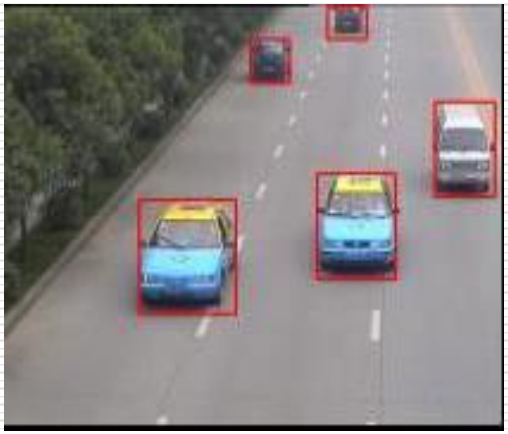
$$g(x, y)=f(x, y)-h(x, y)$$

图像相减运算可提取某运动序列图像相邻两帧或多帧之间的差异以及运动帧与背景帧之间的差。利用图像相减方法，可以对运动目标进行检测，从而实现运动目标定位、跟踪等。



a.原始图像

b.与背景图像相减后的差分图像



c.运动目标定位结果图

□ 乘法运算应用—图像的局部显示

5 基本几何运算

- 图像的几何运算是指通过图像像素位置的变换，运算后，直接确定该像素灰度的运算。与代数运算不同，几何运算可改变图像中各物体之间的空间关系。

几种基本的几何运算

- 图像平移
- 图像放大、缩小
- 图像旋转
- 图像镜像

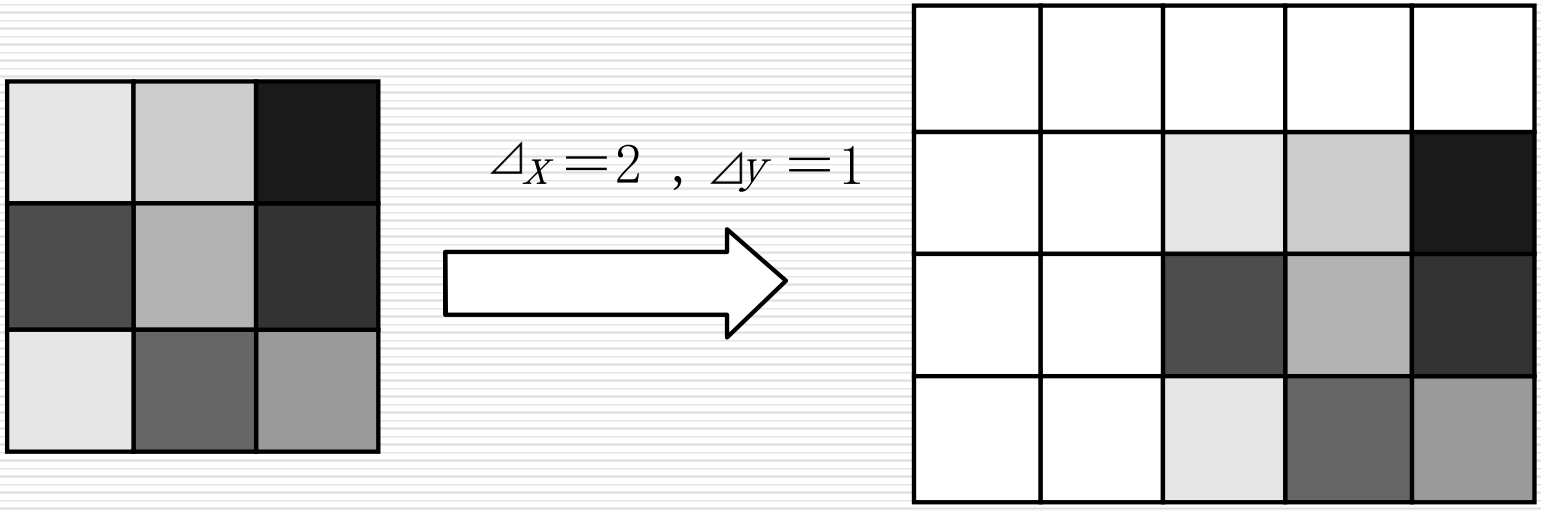
- 一个几何运算需要两个独立的算法。

a. 空间变换，用它来描述每个像素如何从其初始位置“移动”到终止位置；

b. 灰度级插值的算法

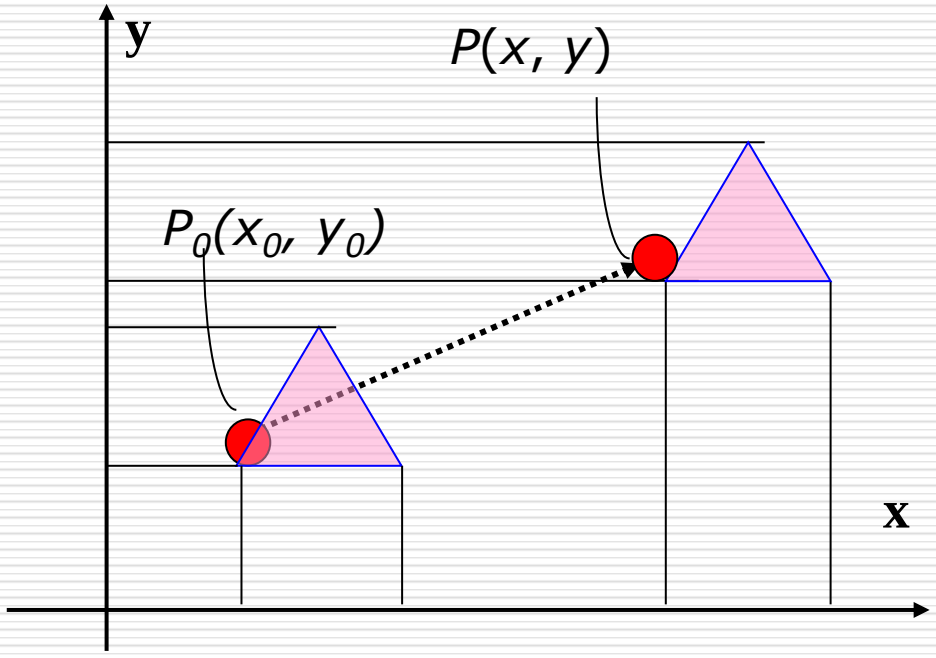
平移变换

将一幅图像中所有的点都按照指定的平移量在水平、垂直方向移动。



现设点$P_0(x_0, y_0)$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。那么，点$P(x, y)$的坐标为

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



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

这个变换用矩阵的形式可以表示为

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

利用齐次坐标，变换前后图像上的点$P_0(x_0,\ y_0)$和$P(x,\ y)$之间的关系可以用如下的矩阵变换表示为

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

平移后的图像上的每一点都可以在原图像中找到对应的点。

原始图像

平移后的图像

放大、缩小

图像缩放是指将给定的图像在x轴方向按比例缩放f_x倍，在y轴方向上按比例缩放f_y倍，从而获得一幅新的图像。

$$x=f_x x_0$$
$$y=f_y y_0$$

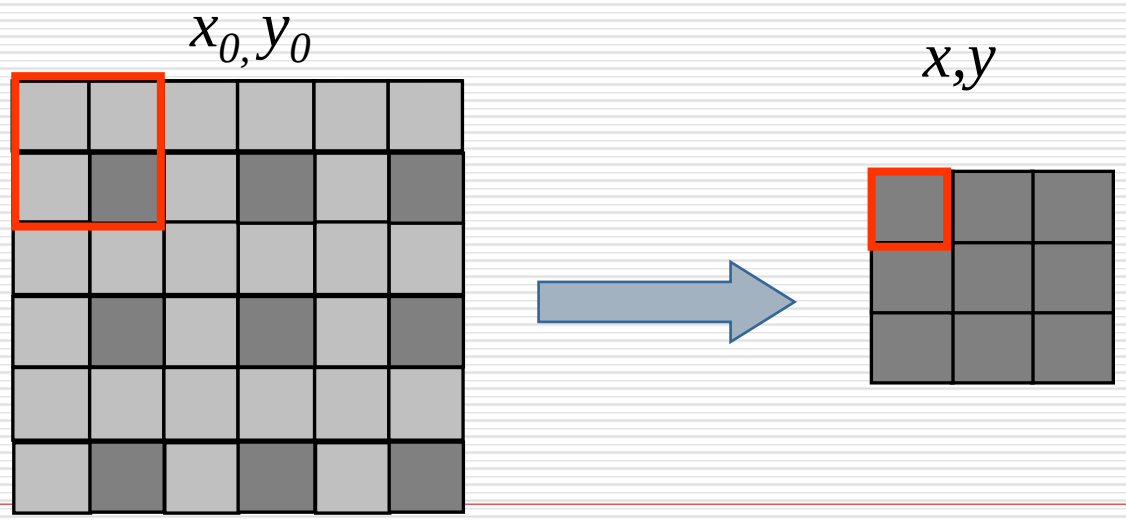
比例缩放前后两点$P_0(x_0, y_0)$、$P(x, y)$之间的关系用矩阵形式可以表示为：

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} fx & 0 & 0 \\ 0 & fy & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \\ 1 \end{bmatrix}$$

图像的比例缩小

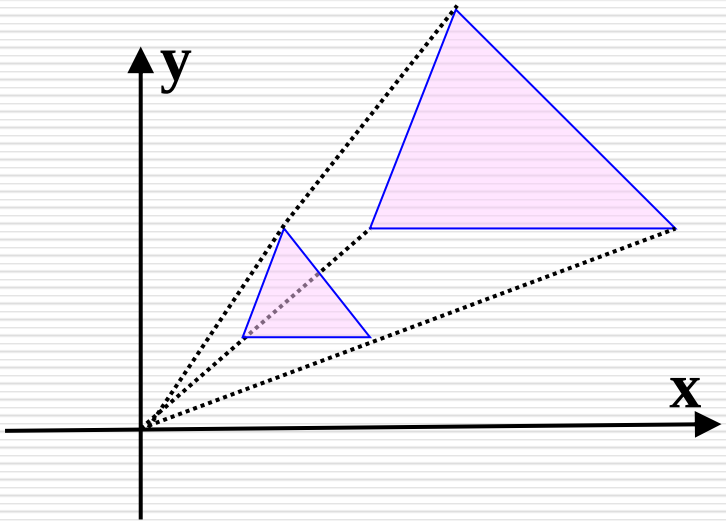
$$x=f_x x_0 \quad y=f_y y_0 \Rightarrow x=x_0/2 \quad y=y_0/2$$

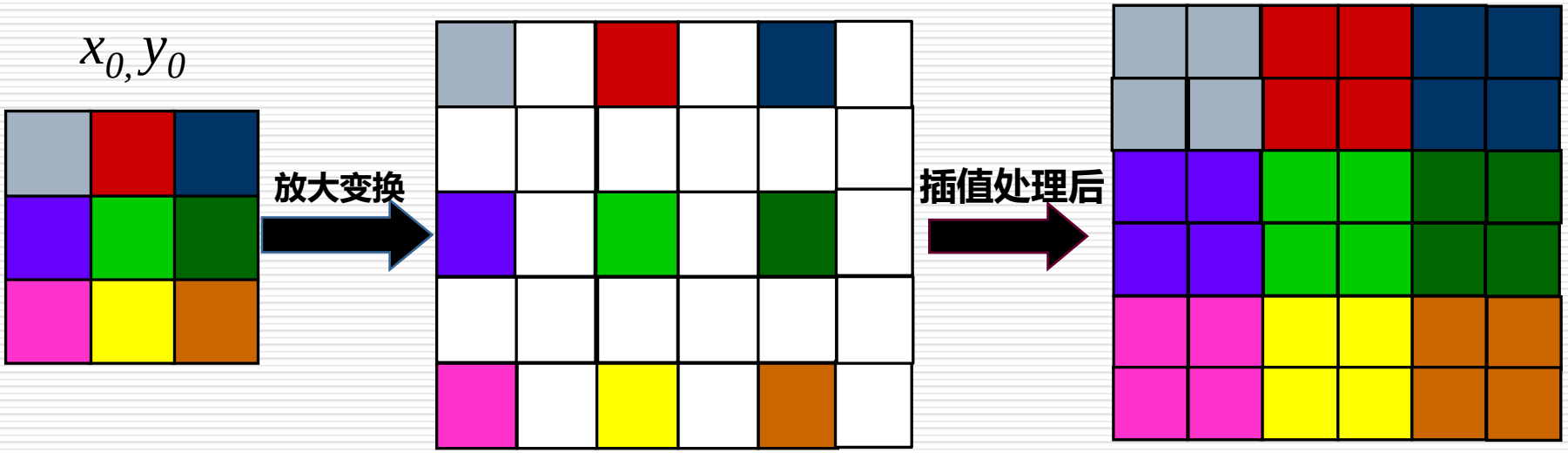
在原图像基础上，每行隔一个像素取一点，每隔一行进行操作，即取原图的偶（奇）数行和偶（奇）数列构成新的图像。如图所示：



图像的比例放大

$$x=f_x x_0 \quad y=f_y y_0 \Rightarrow x=2x_0 \quad y=2y_0$$





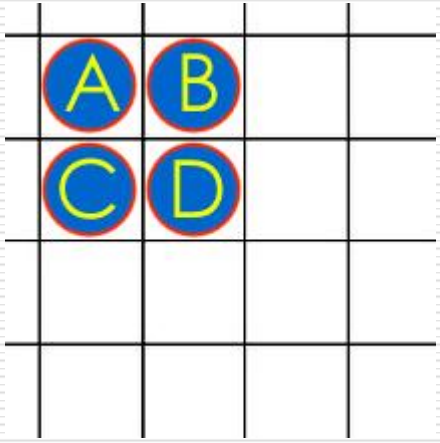
在图像放大的正变换中，出现了很多的空格。因此，需要对放大后所多出来的一些空格填入适当的像素值。一般采用**最邻近插值**和**线性插值法**。

由于放大运算不是一一映射，只是简单的重复放大，因此将产生所谓的“方块”效应，为改善这种可视效果，需要进行插值运算 。

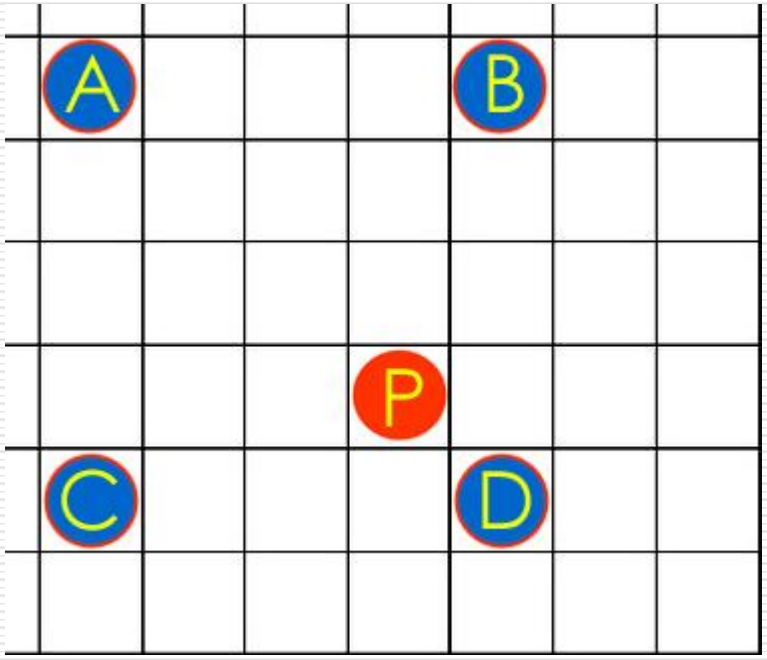
最邻近插值

最近邻法是将与$P_0(u_0, v_0)$点最近的整数坐标$P(u, v)$点的灰度值取为$P_0(u_0, y_0)$点的灰度值。

在$P_0(u_0, v_0)$点各相邻像素间灰度变化较小时，这种方法是一种简单快速的方法，但当$P_0(u_0, v_0)$点相邻像素间灰度差很大时，这种灰度估值方法会产生较大的误差。



原始图像的相邻4个像素点

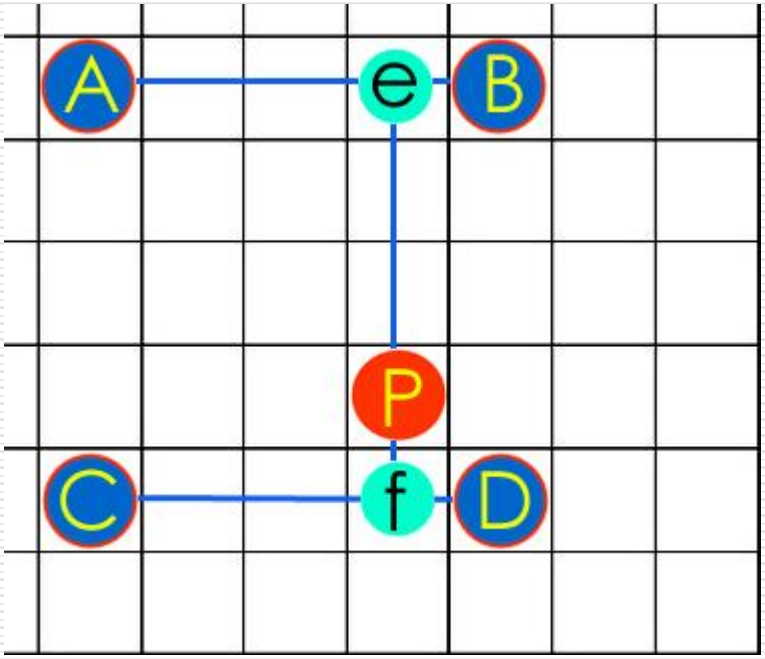


图像放大4倍后已知像素分布图

P点由于最接近D点，所以就直接取P=D。

双线性插值法

双线性插值法是对最近邻法的一种改进，即用线性内插方法，根据$P_0(u_0, v_0)$点的四个相邻点的灰度值，插值计算出$P_0(u_0, v_0)$值。



◆第一步 对上端的两个顶点AB进行线性插值　插值计算出AB两点对P点的影响得到e点的值。

◆第二步 对底端的两个顶点CD进行线性插值　插值计算出CD两点对P点的影响得到f点的值。

◆第三步 进行垂直方向的线性插值　插值计算出ef两点对P点的影响值。

习题

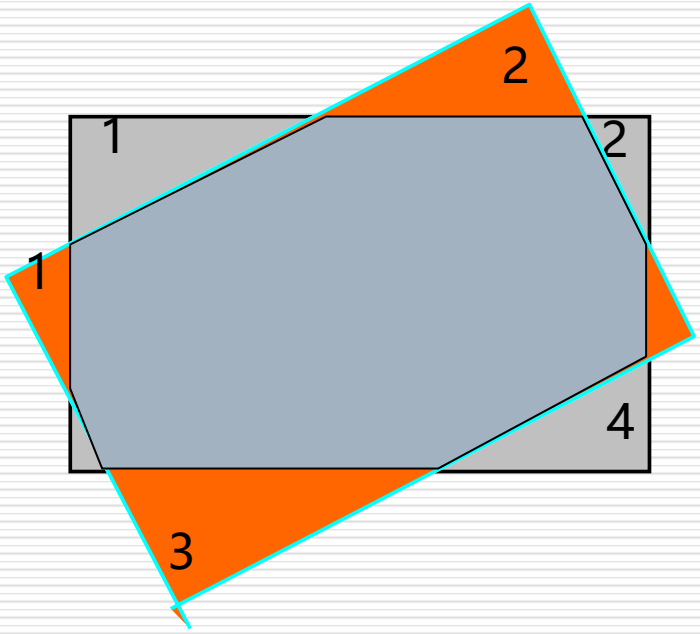
已知一幅数字图像f(x, y)， 其中f (0,0)=100，f (0,1)=150，f(1,0)=200，f (1,1)=250。

(1) 用最近邻插值求f (0.4 , 0.4)；

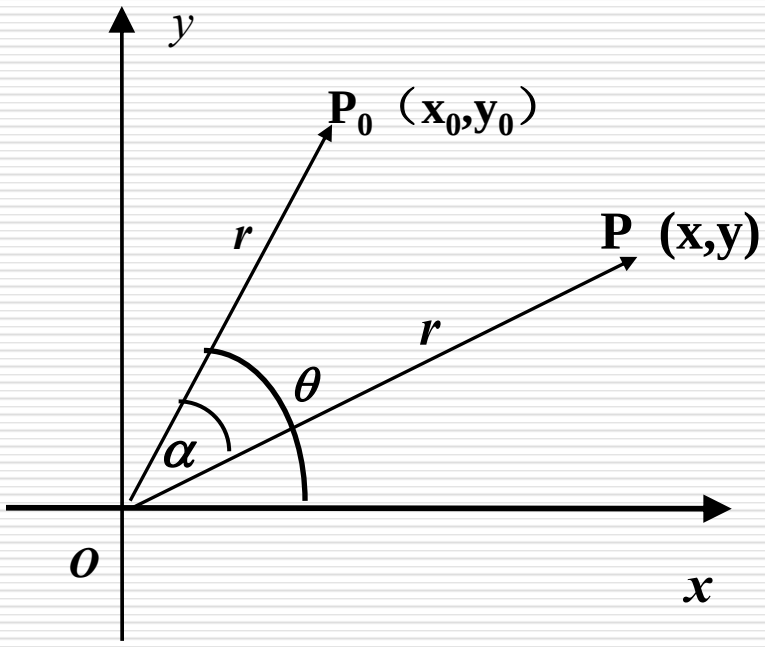
(2) 用双线性插值求f (0.4 , 0.4)。

图像旋转

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



图像的旋转变换也可以用矩阵变换表示。设点$P_0(x_0, y_0)$旋转a角后的对应点为P(x, y)。



旋转前：

$$\begin{cases} x0=r\cos\theta \\ y0=r\sin\theta \end{cases}$$

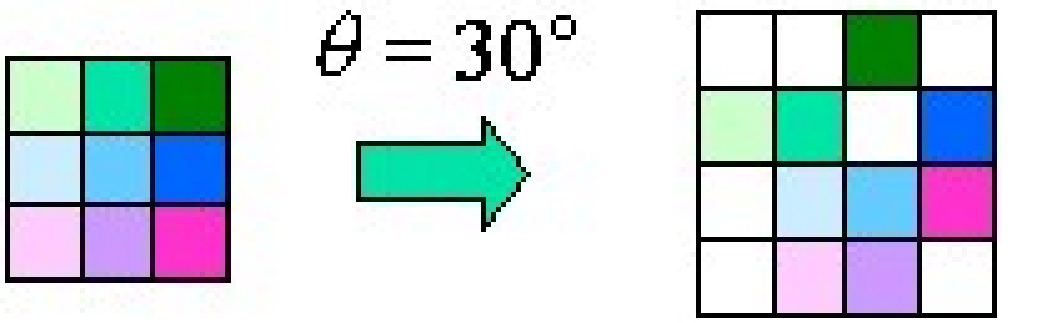
旋转后：

$$\begin{cases} x=r\cos(\theta-a)=r\cos\theta\cos a+r\sin\theta\sin a=x0\cos a+y0\sin a \\ y=r\sin(\theta-a)=r\sin\theta\cos a-r\cos\theta\sin a=-x0\sin a+y0\cos a \end{cases}$$

上式写成矩阵表达式为：

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

从上式可以看出，旋转变换也不是一一映射，输出也不总是整数，**会出现许多空洞点，需要对这些空洞点必须进行插值处理。**



图像绕任意点旋转

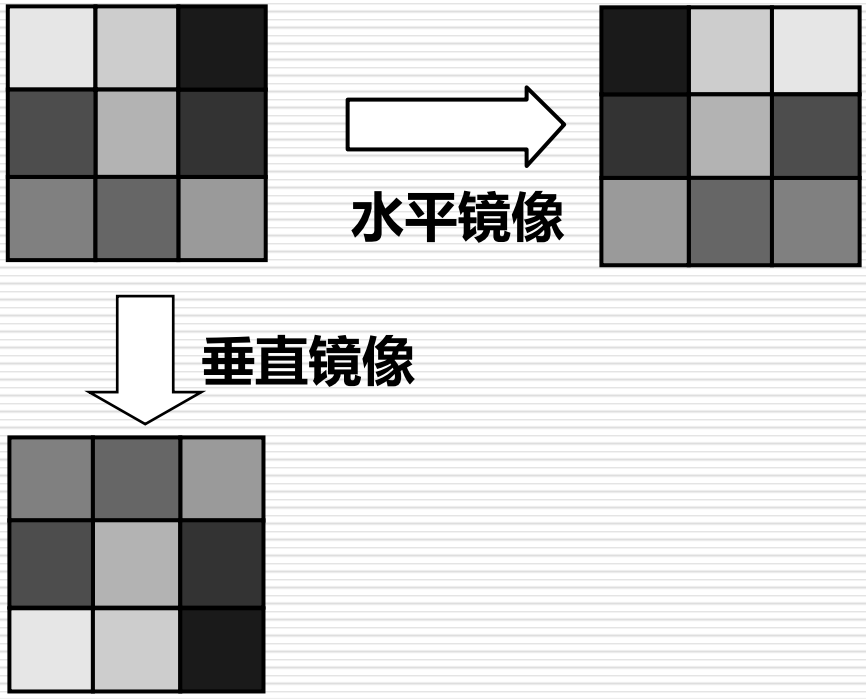
上述的旋转是绕坐标轴原点（0，0）进行的，如果是绕某任意点（a，b）旋转，则可以将平移变换和旋转变换组合起来：先将图像平移到使该点成为坐标原点，再进行旋转，最后将旋转后的图像平移回原点。

旋转前的图像

旋转15°并进行插值处理的图像

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)变换分为水平镜像和垂直镜像。

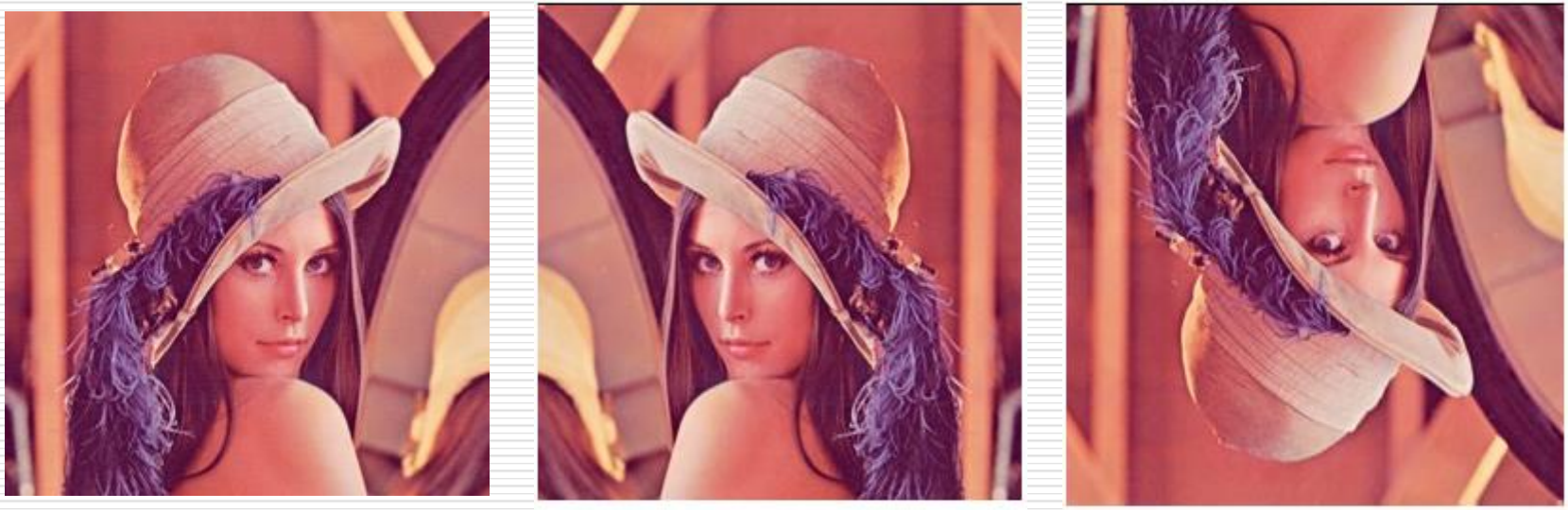


设点$P_0(x_0,\ y_0)$进行镜像后的对应点为$P(x,\ y)$，图像高度为h，宽度为w，原图像中$P_0(x_0,\ y_0)$经过水平镜像后坐标将变为（$w-x_0$，y_0），其矩阵表达式为

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

同样，$P_0(x_0,\ y_0)$经过垂直镜像后坐标将变为$(x_0,\ h-y_0)$，其矩阵表表达式为

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

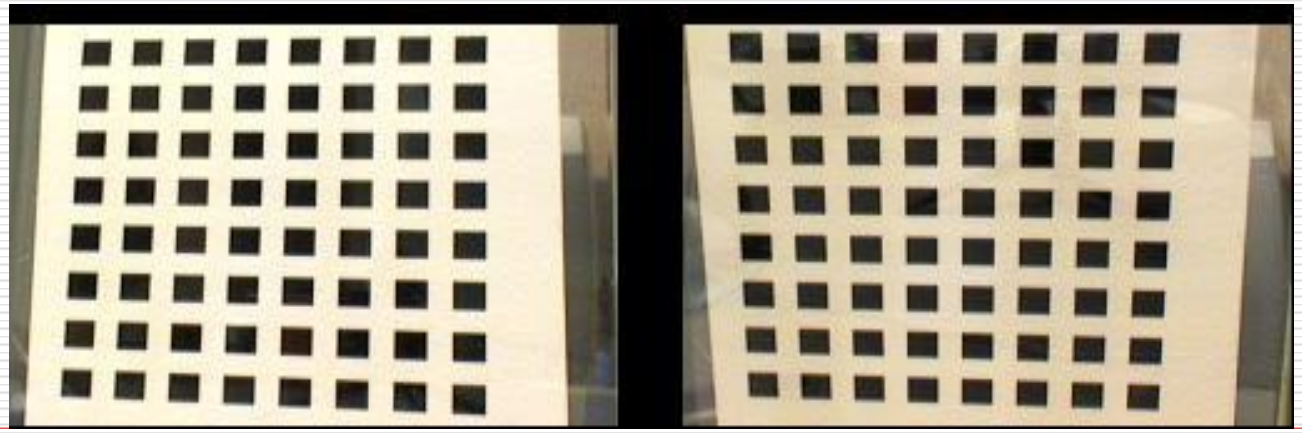


图像镜像效果图

几何运算的应用

- 消除由摄像机原因导致的数字图像几何畸变；

校正前的图片（存在几何变形）



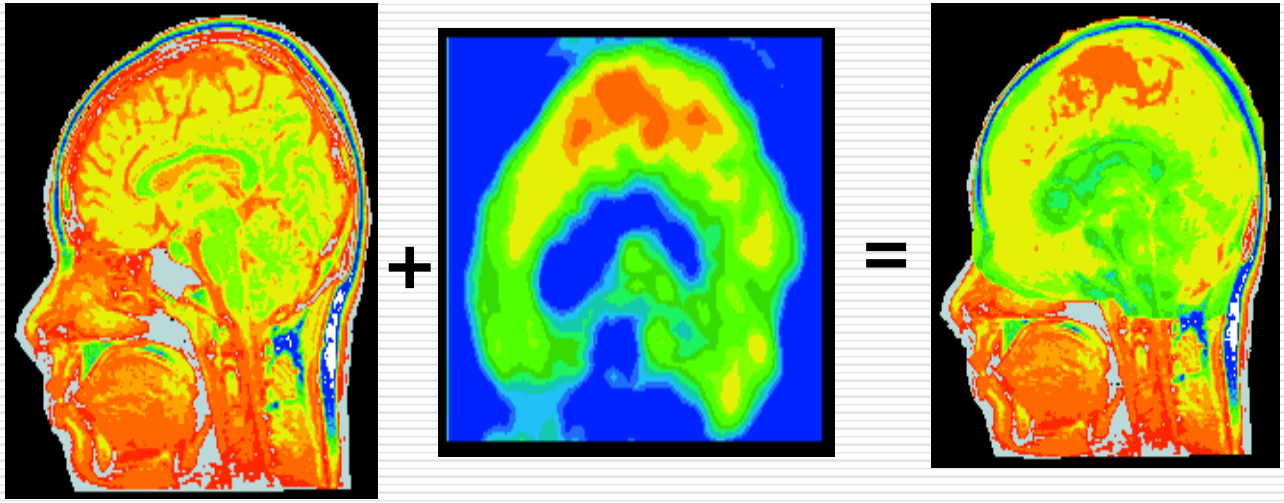
校正后的图片

- **对相似图像进行配准，以便进行图像比较。**

在遥感图像处理中，图像配准是很重要的。

通过配准来寻找两幅图像中相似的区域

医学图像融合



MRI　　SPECT　　配准融合后的结果

data augmentation

在深度学习中，有的时候训练集不够多，或者某一类数据较少，或者为了防止过拟合，让模型更加具有鲁棒性，**data augmentation**是一个不错的选择。

常见方法：

Color Jittering：对颜色的数据增强：图像亮度、饱和度、对比度变化；

Random Scale：尺度变换；

Random Crop：采用随机图像差值方式，对图像进行裁剪、缩放；包括**Scale Jittering**方法（**VGG**及**ResNet**模型使用）或者尺度和长宽比增强变换；

Horizontal/Vertical Flip：水平/垂直翻转；

Shift：平移变换；

Rotation/Reflection：旋转/仿射变换；

Noise：高斯噪声、模糊处理；

Common augmentation method

网易云课堂

Mirroring

Random Cropping

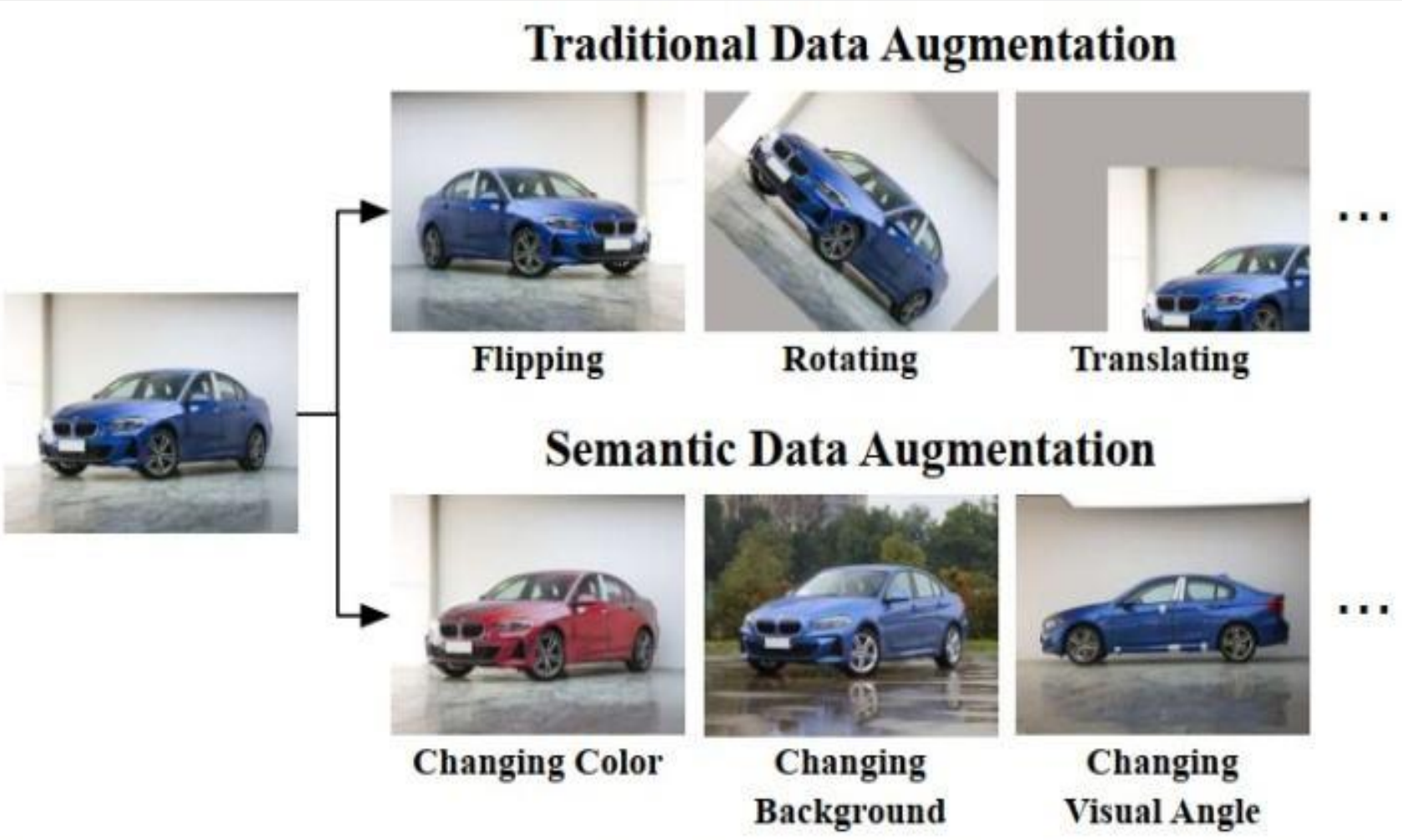
Rotation

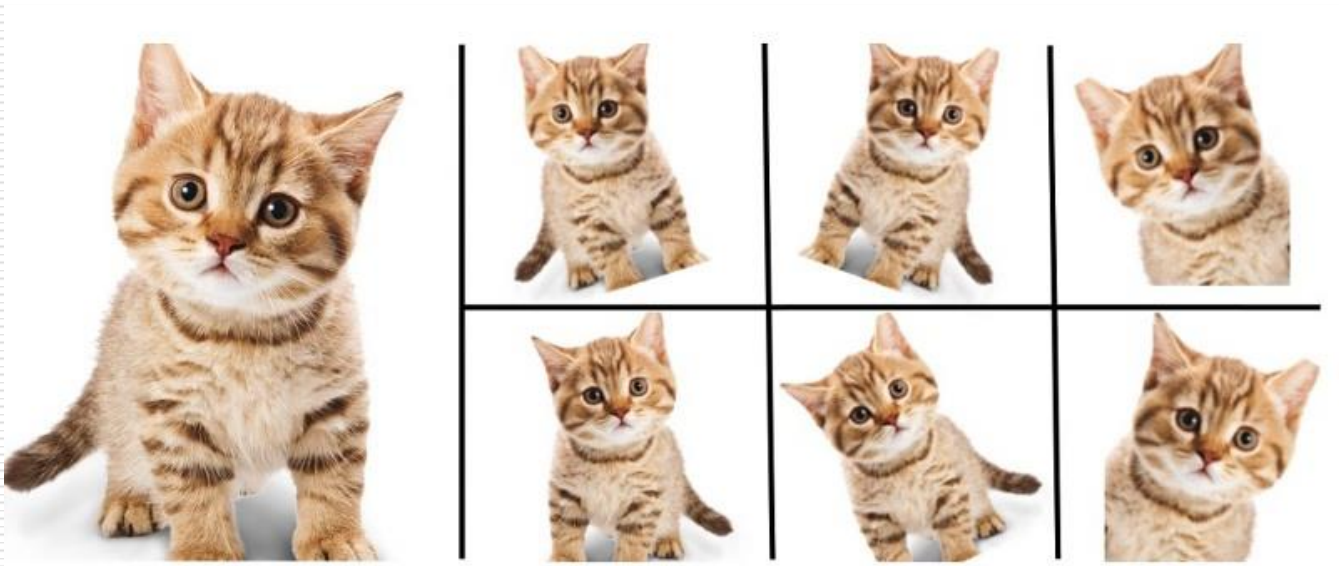
Shearing

Local warping

...

Andrew Ng


Traditional Data Augmentation
Flipping
Rotating
Translating
...
Semantic Data Augmentation
Changing Color
Changing Background
Changing Visual Angle
...


Enlarge your Dataset

补充：图像噪声

所谓噪声，就是妨碍人的感觉器官对所接收到的图像信息进行理解或分析的各种因素。

一般噪声是不可预测的随机信号，只能用概率统计的方法去认识。因此图像噪声完全可以用其概率分布函数和概率密度函数来描述。

原图像

添加噪声的图像

1. 图像噪声分类

图像噪声按其产生的原因可分为外部噪声和内部噪声。

- **外部噪声是指图像处理系统外部产生的噪声，如电气设备、天体放电现象等引起的噪声。**
- **内部噪声是指系统内部产生的，主要内部干扰如下：**

（1） 由光和电的基本性质所引起的噪声。

（2）电器的机械运动产生的噪声。如, 各种接头因抖动引起的电流变化所产生的噪声；磁头、磁带抖动引起的抖动噪声等。

（3） 元器件材料本身引起的噪声。如磁带、 磁盘表面缺陷所产生的噪声。

（4） 系统内部设备电路所引起的噪声。如 电源系统引入的交流噪声，偏转系统和箝位电路引起的噪声等

按噪声对图像的影响可分为加性噪声模型和乘性噪声模型两大类。

设f(x,y)为理想图像，n(x,y)为噪声，实际输出图像为g(x,y)。

对于加性噪声而言，其特点是n(x,y)和图像光强大小无关。即

$$g(x,y)=f(x,y)+n(x,y)$$

对于乘性噪声而言，其特点是n(x,y)和图像光强大小相关。即

$$g(x,y)=f(x,y)[1+n(x,y)]=f(x,y)+f(x,y)n(x,y)$$

按噪声频谱分类：

噪声幅度分布符合高斯分布，称其为高斯噪声

频谱均匀分布的噪声，称为白噪声。

2. 噪声的特征

对灰度图像f(x,y)来说，可看做是二维亮度分布，则噪声可看做是对亮度的干扰，用n(x,y)来表示。噪声是随机性的，因而需用随机过程来描述，即要求知道其分布函数或密度函数。但在许多情况下这些函数很难测出或描述，甚至不可能得到，所以常用统计特征来描述噪声，如均值、方差等。