

# 图像增强 (1)

叶山 中国地质大学（北京）

yes@cugb.edu.cn

# 色彩空间

# 色彩空间



世界上一共有多少种颜色？

# 色彩空间

色彩空间（color space）是对不同颜色的系统组织方式（通过坐标系定义不同的颜色）。常见的色彩空间类型：

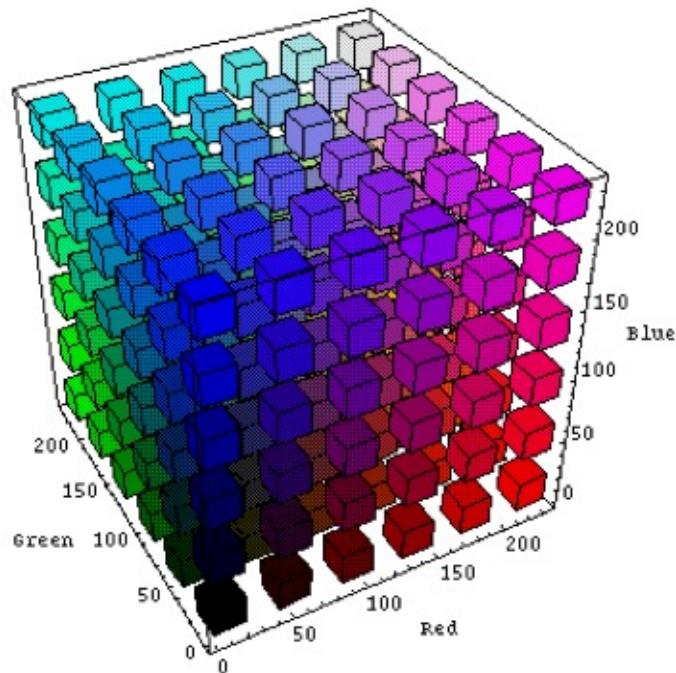
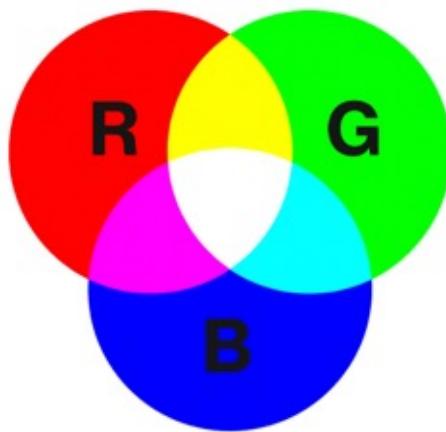
- RGB：多用于屏幕显示
- CMYK：多用于彩色印刷



# 色彩空间：RGB

RGB：加法混色的空间

利用红、绿、蓝三个基本色的线性组合来表示其他颜色，即任何颜色都是由这三个基本色的分量决定的。

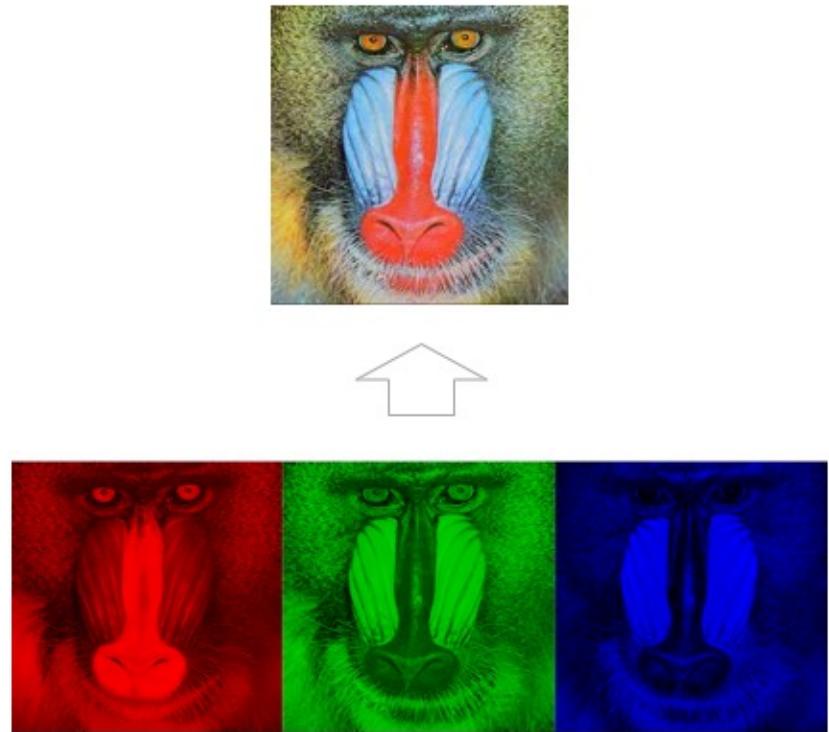


RGB的图像分3个色彩通道

24bit模式下，每个基本色的强度范围是0到255  
共有1680万种颜色

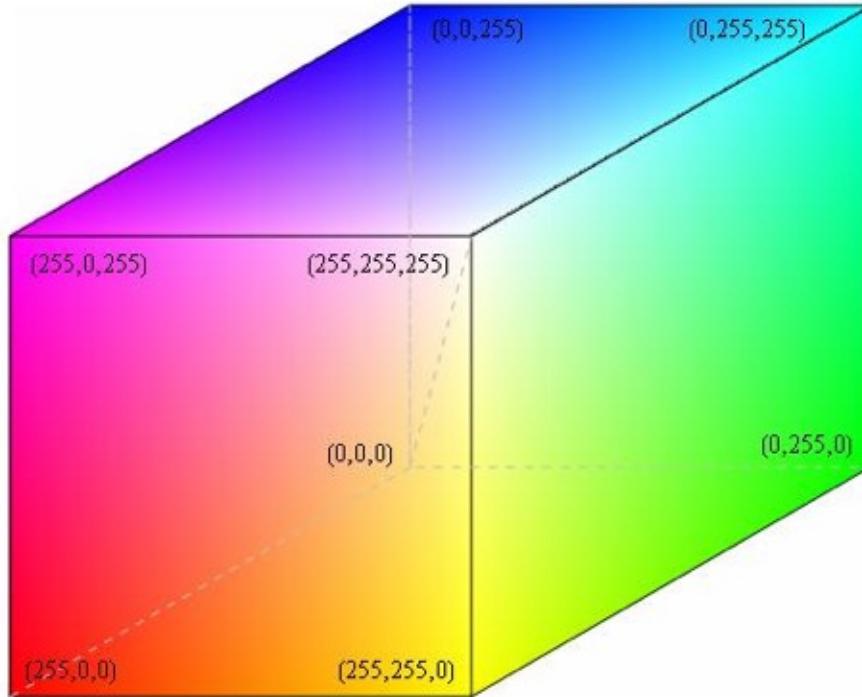
# 色彩空间：RGB

| row   |      |      |      |
|-------|------|------|------|
| 0 1 2 |      |      |      |
| 0     | .392 | .482 | .576 |
| 1     | .478 | .63  | .169 |
| 2     | .580 | .79  | .263 |
| 0     | .169 | .263 | .376 |
| 1     | .263 | .44  | .306 |
| 2     | .373 | .60  | .376 |
| 0     | .376 | .478 | .451 |
| 1     | .443 | .569 | .561 |
| 2     | .443 | .569 | .674 |



RGB图像有3个色彩通道 (channels)  
因此这类图像的数组为3维向量  
即宽 (像素数量) 、 高 (像素数量) 、 深 (3个色彩通道)

# 色彩空间：RGB



在**24bit**模式下，每个基本色的取值范围是**0**到**255**之间，因此RGB共能定义大约**1680**万种颜色。

# 色彩空间：色彩三要素

色彩三要素空间（RGB的变种）

- HSL: 色相、饱和度、亮度模式
- HSV: 色相、饱和度、明度模式

二者区别

色相 Hue

- 由光波长决定的颜色种类

饱和度 Saturation

- 颜色的纯度（饱和度越高，灰色成分越少）

HSL

- 假设物体自己会发光
- L调到最大，会让物体呈白色

HSV

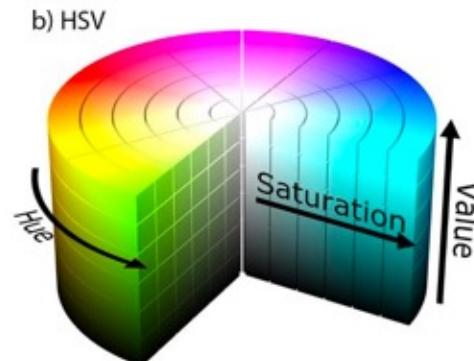
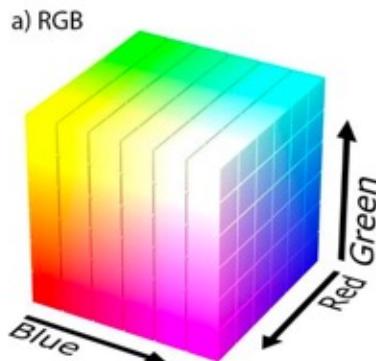
- 模拟物体被白色光所照亮
- V调到最大，是强烈白光照在物体底色上的效果

亮度 Lightness

- 物体本身的亮度

明度 Value

- 物体被照亮的程度



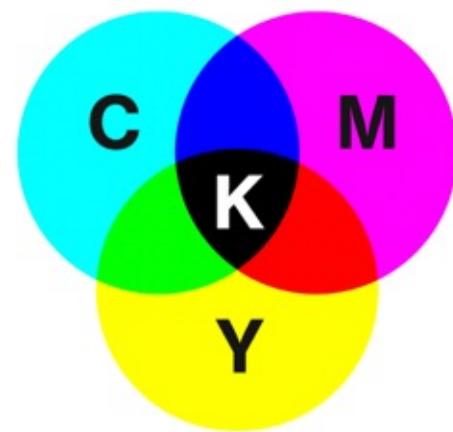
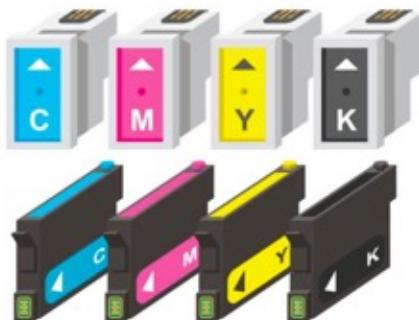
首先选定一个基本色，然后调整饱和度以及明度或亮度。

颜色种类：约1680万种

# 色彩空间：CMYK

CMYK：减法混色颜色空间。

通过青色、品红、黄色三原色以及黑色通道的叠加而构成，通常用于印刷领域，也叫印刷四分色模式，可以定义大约103万种颜色。



**CMYK**

# 色彩空间：CIE-XYZ

CIE-XYZ：最精确的色彩定义

物体所呈现的颜色由物体的材料属性、光源中各种波长分布、人的心理认知有关。

色觉存在个体差异，所以颜色既是一种生理和物理现象，也是一种心理现象。

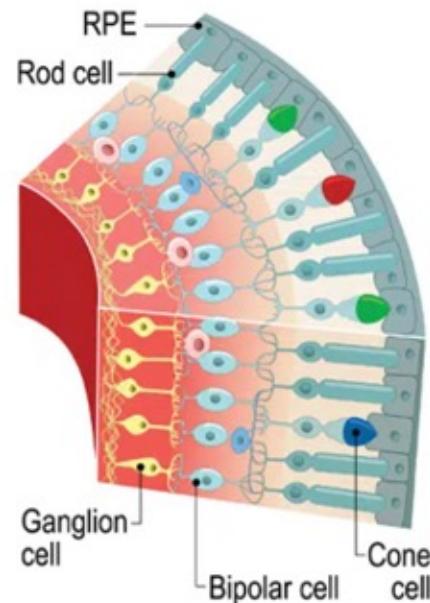
# 色彩空间：CIE-XYZ

## 杆状细胞 (rod cell)

- 也叫视杆细胞。
- 具有很强的暗视觉 (scotopic vision) , 能在低照明的情况下感知到物体的大小和形状。
- 几乎无法感知颜色信息，只有灰度视觉。
- 所能感应的可见光波长范围为400~700nm。

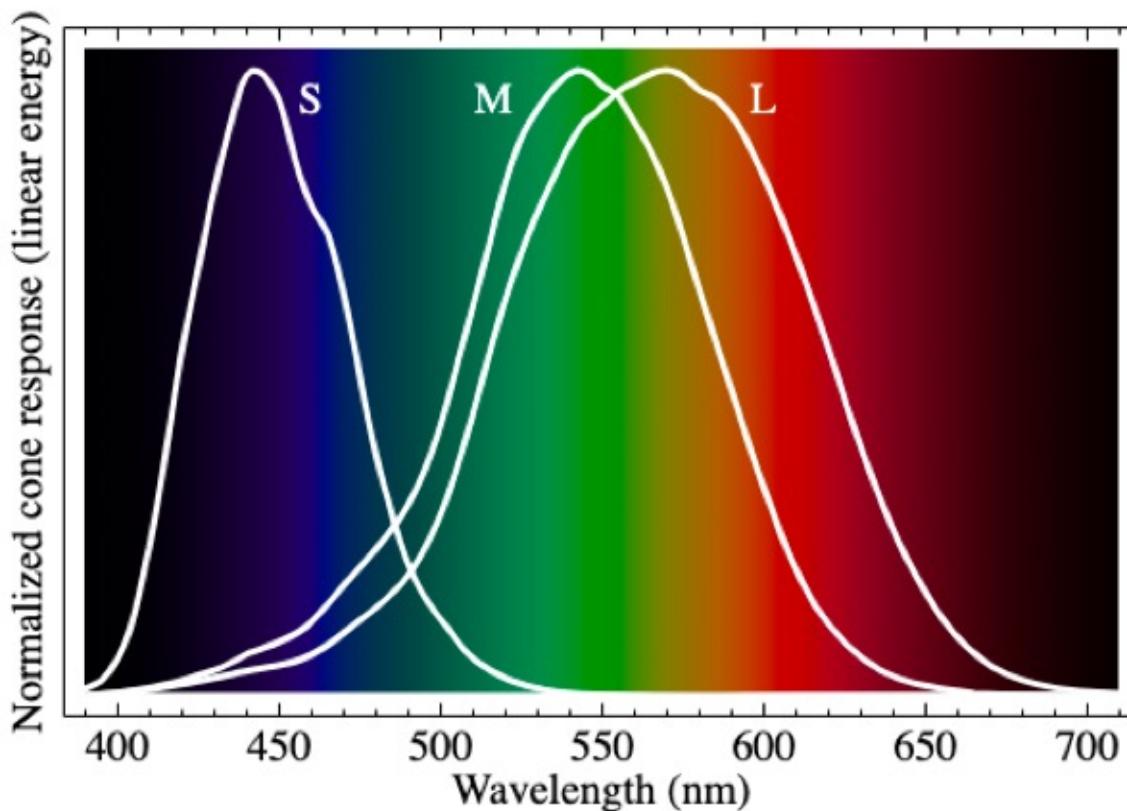
## 锥状细胞 (cone cell)

- 也叫视锥细胞。
- 只对明亮的光线产生刺激反应，主要负责明视觉 (photopic vision) 。
- 数量较少，但每个细胞单独和一条视觉神经相连，因此具有更加清晰的视觉能力。



视网膜示意图

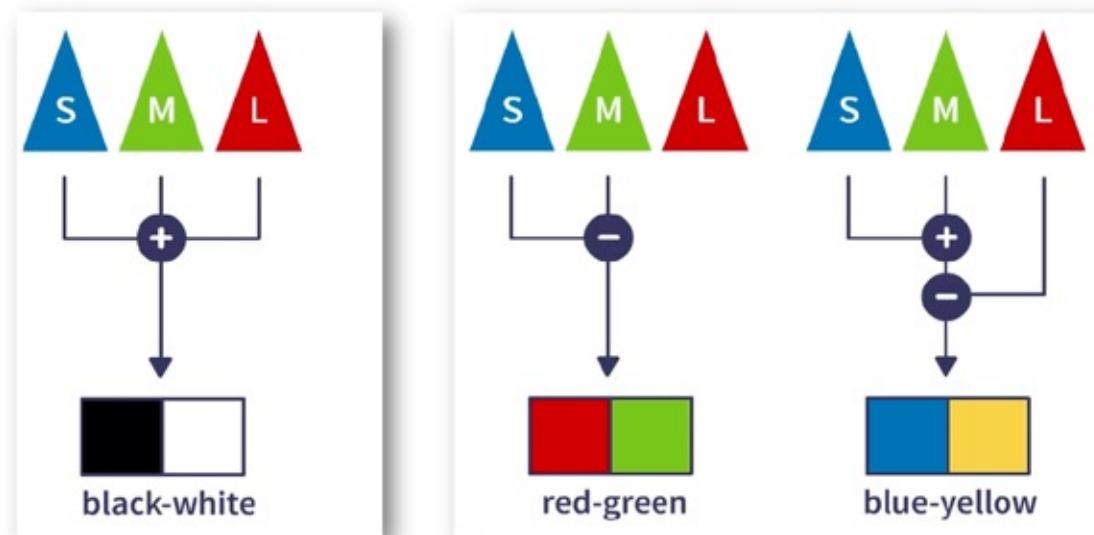
# 色彩空间：CIE-XYZ



三色视觉理论：人眼的三种锥状细胞分别会优先获得[相应敏感波长区域的](#)光信号刺激并进行合成，形成对颜色的感知。

# 色彩空间：CIE-XYZ

- 补色过程理论：人的视觉系统通过对立比较的方式获得对颜色的感知。
- 在受到某种颜色光的刺激时，锥状细胞分别会释放两种信号：1) 刺激、兴奋信号；2) 抑制信号。在视觉的三个色彩通道中，两种信号之间的强弱对比让视觉系统感知到颜色。
- 三个色彩通道：黑白、红绿、黄蓝



# 色彩空间：CIE-XYZ



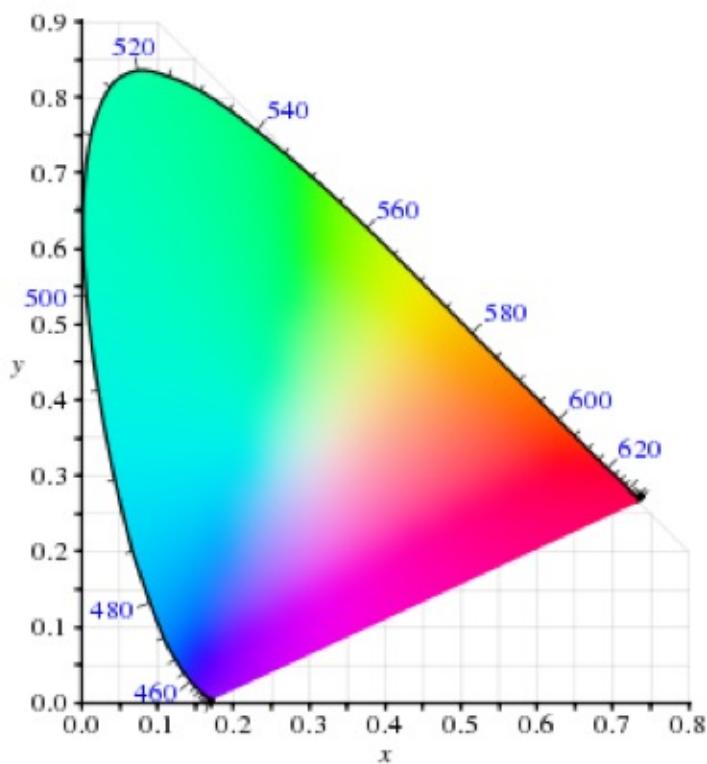
后像 (afterimage illusion)

# 色彩空间：CIE-XYZ

CIE-XYZ：由国际照明协会于1931年定义，它基于锥状细胞的感知能力。

- 短波锥状细胞（S细胞，刺激敏感范围420-440纳米）
- 中波锥状细胞（M细胞，刺激敏感范围530-540纳米）
- 长波锥状细胞（L细胞，刺激敏感范围560-580纳米）

三色刺激通道X、Y、Z大致对应红、绿、蓝。一种波的刺激值等于多种波的混合刺激。

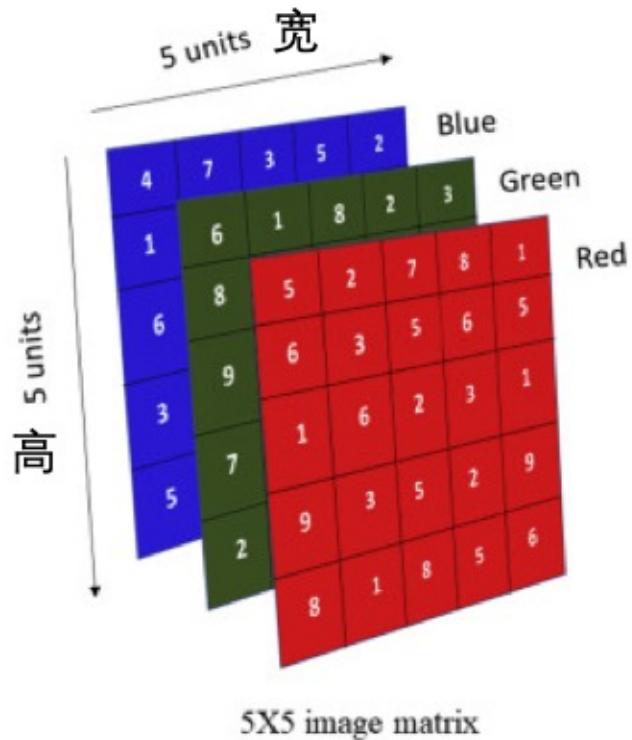
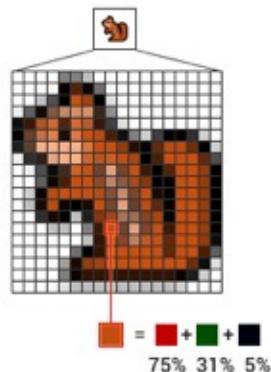


# 图像的储存

# 彩图和灰度图

## RGB三通道彩色图

- 图片储存为3维矩阵[宽度、高度、深度(RGB通道)]，每个像素取值范围[0,255]。
- 红、绿、蓝的亮度叠加混合后形成最终颜色

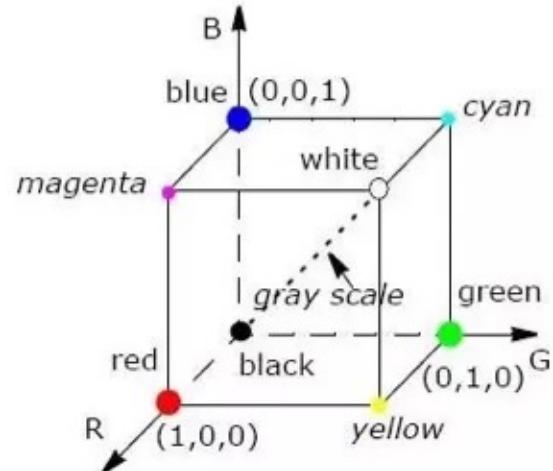


## 单通道灰度图

- 亮度信息取值范围[0,255]。

## 灰度化

- 将RGB彩图的3个通道的像素值分别加权叠加。
- 灰度 =  $0.3R + 0.59G + 0.11B$  (举例)
- 本质上是将三原色的亮度按一定规则投影到色彩空间中的黑白连线上。



# 常见存储格式

- JPEG (joint photographic experts group) 是一种有损压缩格式，它能够将图像压缩在较小的储存空间。
- 支持极高的压缩率，同时能够处理16.8M种颜色，再现全彩色的图像。
- 允许自由地在最小文件尺寸（最低图像质量）和最大文件尺寸（最高图像质量）之间选择压缩比率。

igneous



# 常见存储格式

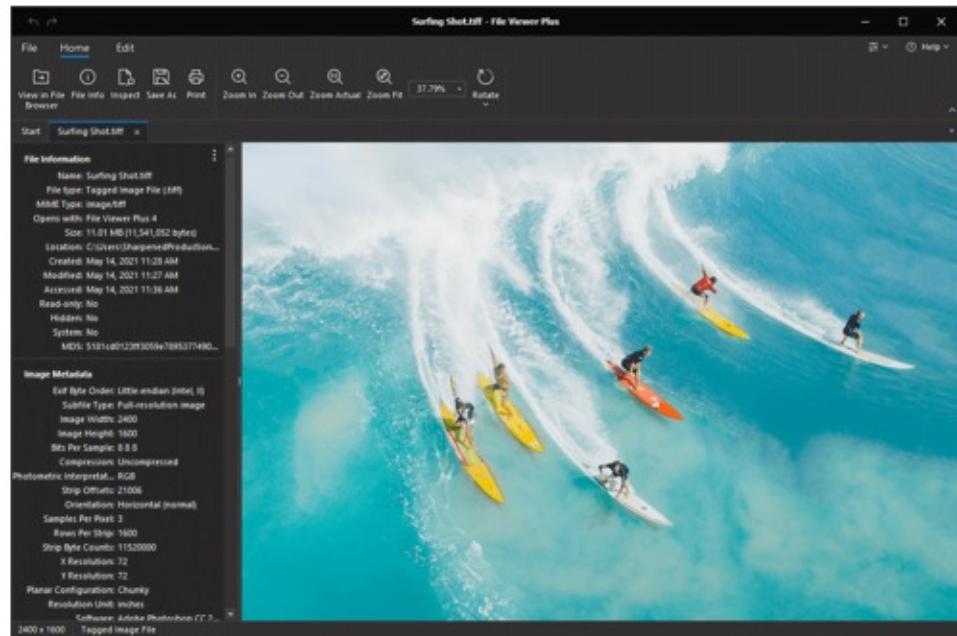
- PNG (portable network graphics) 是一种无损压缩格式，支持透明度和高品质图像。图像的颜色越单一（像素值种类越少），压缩率就越大。
- 最适合用于对精度和细节要求较高的图像。



支持透明度

# 常见存储格式

- TIFF (tagged image file format) 支持多种编码方法，包括无损压缩。
- 在压缩和解压缩过程中不会丢失图像数据，从而保持原始图像的质量。
- 适合用于专业图形处理、出版印刷等领域，支持RGB及CMYK。
- 可携带元数据（拍摄时间、地点、拍摄人等信息）



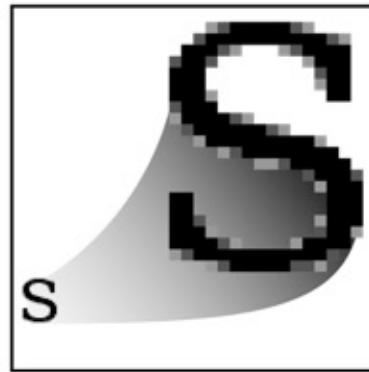
# 常见存储格式

- GIF (graphics interchange format) 是一种支持动画和透明度的图像格式。
- 常用于制作动图和图标 (icon) 。
- 对颜色的支持度低：它采用 8位压缩，最多只能处理256 种颜色。



# 常见存储格式

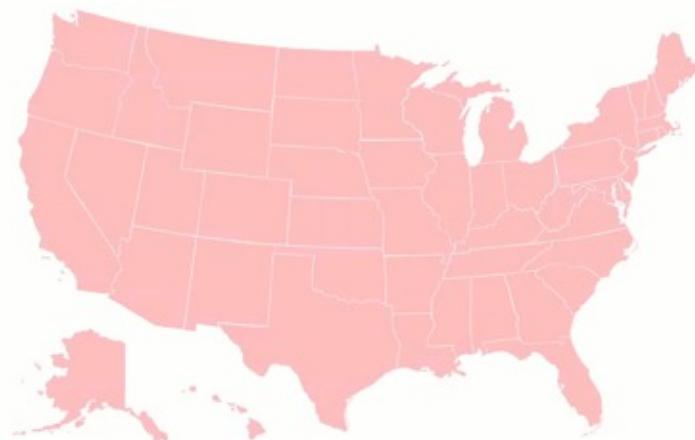
- SVG (scalable vector graphics) , 可以任意缩放而不失真。文件通常较小。
- 矢量图像技术不依赖于像素分辨率，因此可以在不同尺寸和分辨率的屏幕上显示，支持亚像素级渲染。
- 可以通过XML技术进行解析、生成和编辑；使用XML标签描述图像的形状、颜色和样式，易于与HTML和CSS集成，支持网页开发。
- 支持用户交互，可以通过JavaScript等技术实现动态效果。



Raster  
.jpeg .gif .png



Vector  
.svg



Supplements

Reset

# 常见存储格式

- BMP (bitmap) 是一种无压缩的图像格式，可以保持超高的图像质量。
- 支持透明度和颜色深度等功能。通用性好，普遍支持各类的软件和操作系统，便于不同技术栈使用。
- 文件体积通常很大，不利于网络传输。

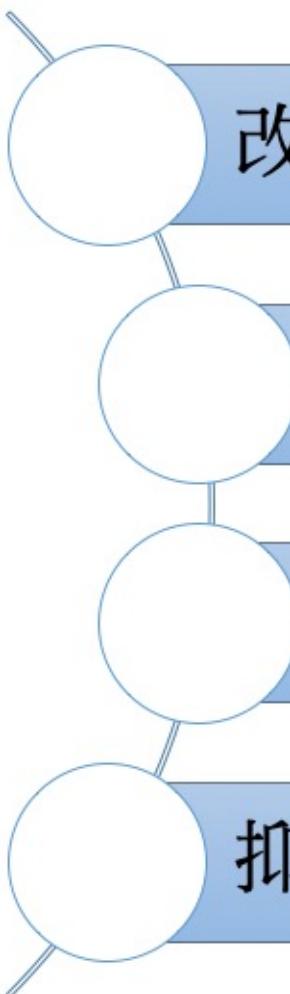


# 常见存储格式

| 格式   | 优点                                                                                                | 缺点                                                                  |
|------|---------------------------------------------------------------------------------------------------|---------------------------------------------------------------------|
| JPEG | 压缩率高，有利于网络传输；色彩支持度好（支持RGB所有颜色），可以如实再现全彩色的图像。                                                      | 有损压缩可能导致图像质量损失，尤其是使用过高的压缩比例时图像清晰度损失很大；并非所有浏览器都支持将JPEG图像插入网页。不支持透明度。 |
| PNG  | 无损压缩保持图像质量；支持透明度；适用于需要高精度和细节的图像。                                                                  | 文件体积可能较大，不利于网络传输。                                                   |
| TIFF | 无损压缩、色彩表现力强。灵活度好，支持多种图像类型、色彩空间和位深度，包括RGB无压缩、RLE压缩等，使其适用于多种应用场景。允许在图像文件中嵌入元数据，如拍摄日期、地点、作者信息等，便于检索。 | 文件通常较大、处理速度很慢。                                                      |
| GIF  | 支持动画和透明度；优秀的压缩算法使其在一定程度上保证了图像质量同时将体积变得很小。                                                         | 色彩表现有限，不适用于对颜色精度要求高的图片。                                             |
| SVG  | 可无限缩放而不失真；适用于网页与移动设备；文件体积相对较小；基于KML，可与HTML、CSS集成；可制作交互式可视化。                                       | 不适用于高质量的数码照片图像呈现；需要特殊的打开方式才能进行图像浏览；在某些版本过低的浏览器上无法呈现。                |
| BMP  | 图像质量高，没有压缩损失；支持透明度和颜色深度等功能；通用性好。                                                                  | 文件体积大，占用磁盘空间较多；不支持动画和透明图像等高级功能。                                     |

# 图像增强

# 图像增强的目标

- 
- 改善图像的视觉效果
  - 转换为更适合计算机分析处理的格式
  - 突出对计算机视觉有意义的信息
  - 抑制无用的信息

# 图像增强的目标

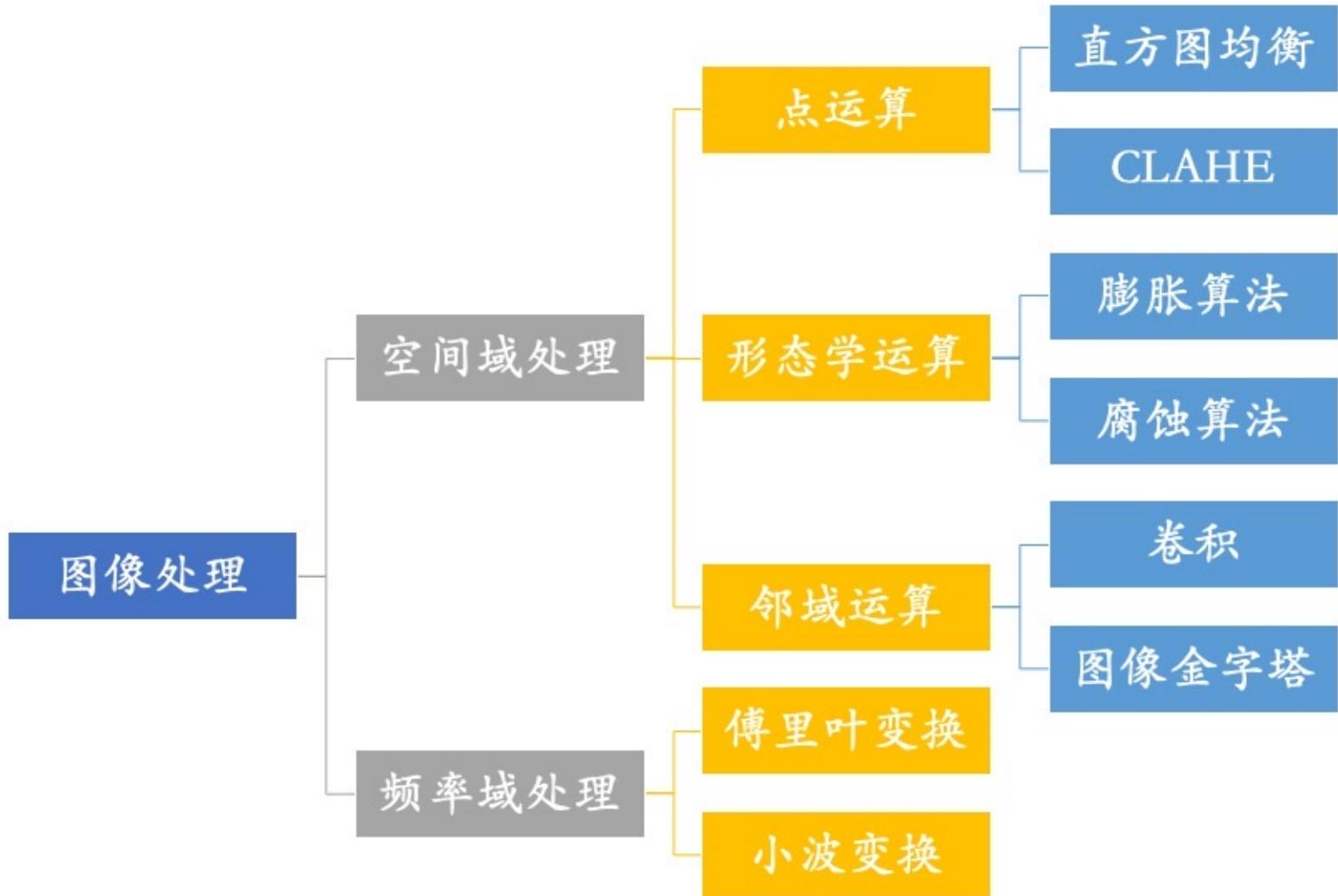
图像增强

图像锐化

平滑降噪

灰度调整

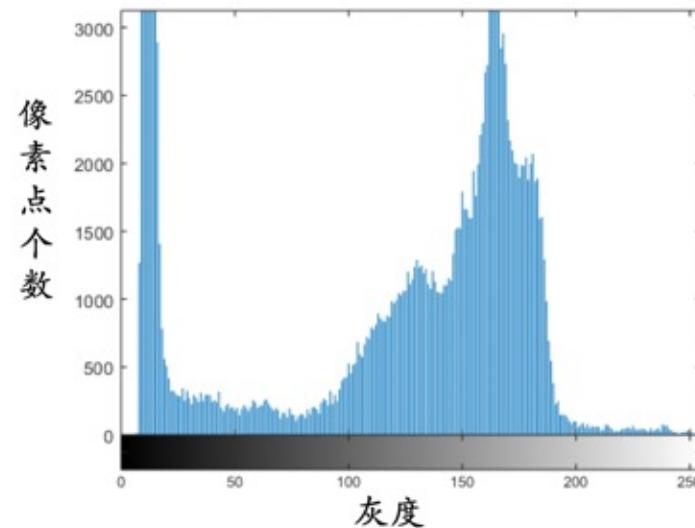
# 图像处理方法



# 直方图均衡化

## 直方图算法

- 直方图是对图片数据及特征分布的一种统计
  - 颜色、灰度
  - 边缘、形状、纹理
  - 局部特征的、视觉语义
- 区间/步长 (bin)
  - 应该具有一定的统计学或物理学意义，是一种数据或特征的代表
  - 需要预定义或机器学习产生



# 直方图均衡化

original image



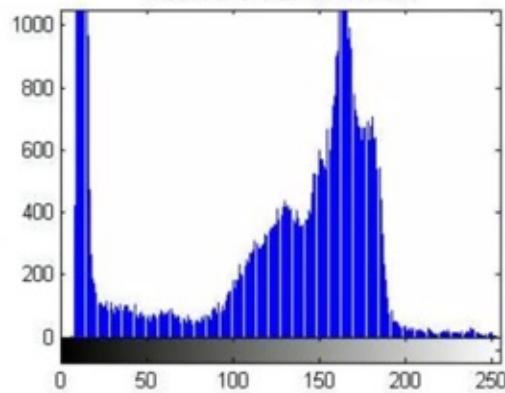
transformed image( $c=2.5$ )



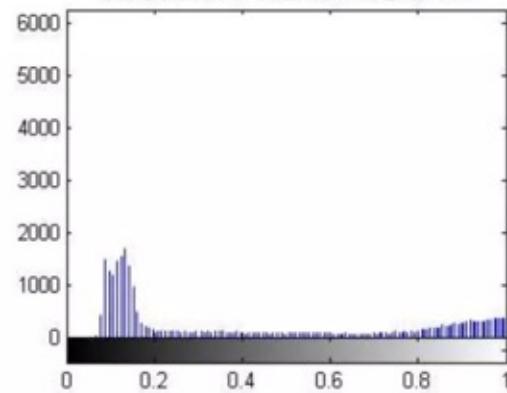
transformed image( $c=0.5$ )



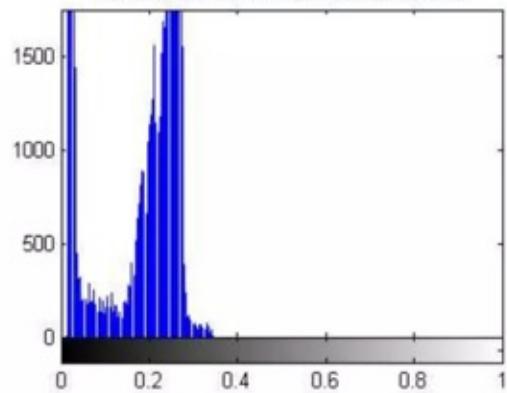
histogram of the original image



histogram of the transformed image( $c=2.5$ )



histogram of the transformed image( $c=0.5$ )



# 直方图均衡化

$$S_k = \frac{(L - 1) * (r_k - r_{km_{in}})}{r_{kmax} - r_{km_{in}}}$$

$$k = 1, 2, 3, \dots, L - 1$$

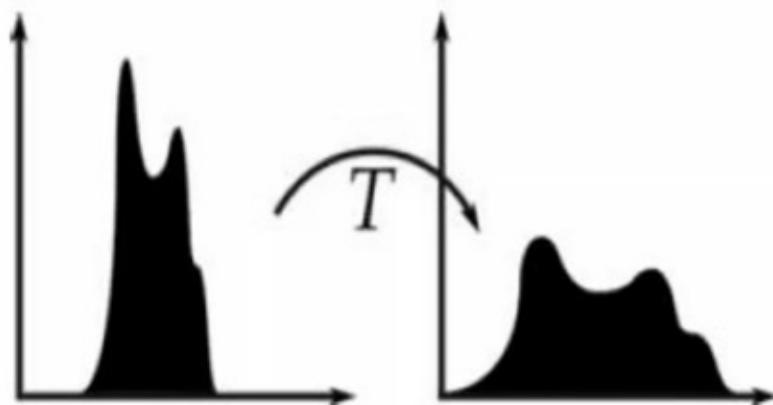
S: 输出的像素灰度值

r: 输入的像素灰度值

L: 图像中的灰度分为多少级别

$r_{kmax}$ : 输入图像的最大灰度值

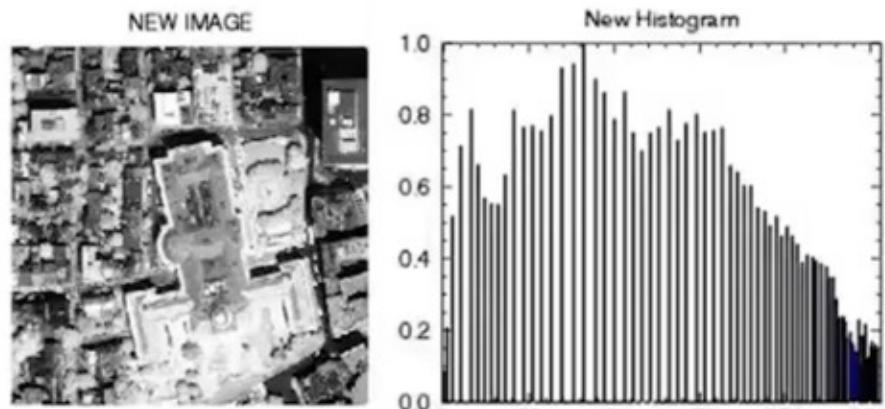
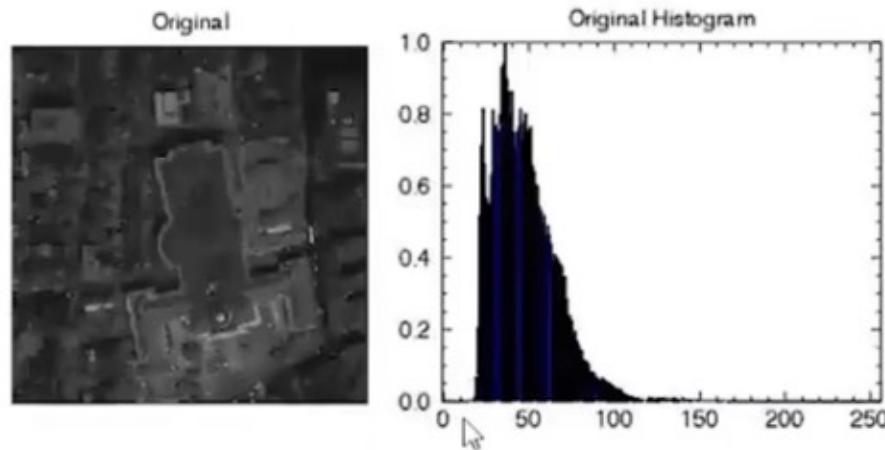
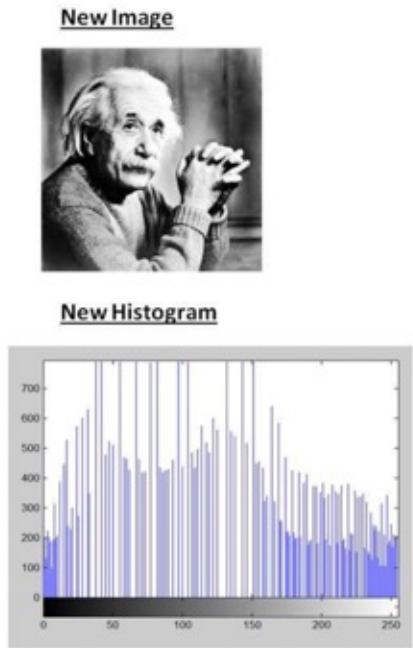
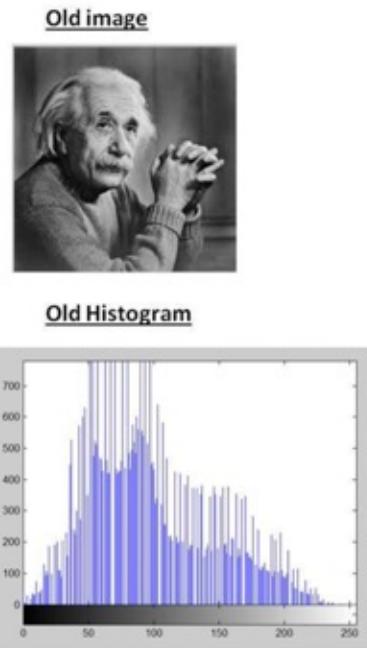
$r_{kmin}$ : 输入图像的最小灰度值



- Histogram equalization: 利用图像直方图调整对比度的方法。主要思想：通过累积分布函数，将图像的直方图分布变成一个近似均匀分布，从而增强图像的对比度。
- 通常用来增加图像的局部对比度，尤其在图像的重点信息部分对比度较弱时使用。
- 直方图均衡化以后，亮度可以更均匀地在图上分布（扩展常用的亮度），可增强局部的对比度，而不影响整体的对比度。

# 直方图均衡化

重新分配各个灰度bin中的像素点数量，使一定灰度范围的像素点数量大致相等。

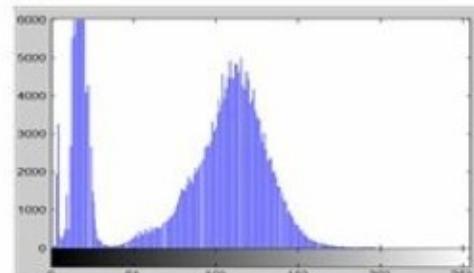
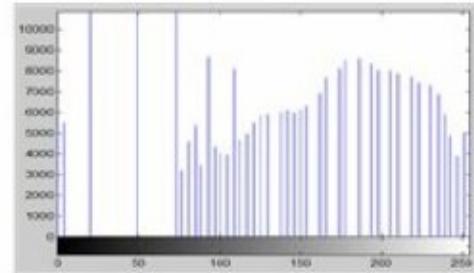
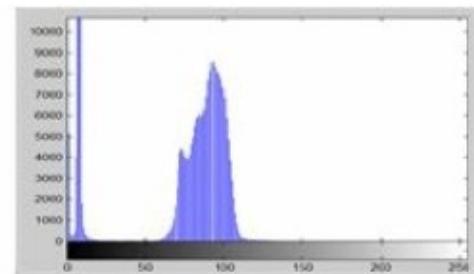
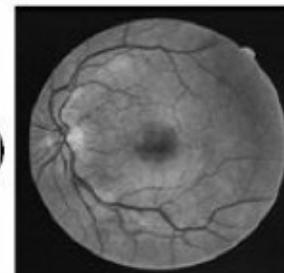
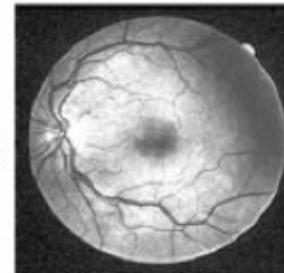
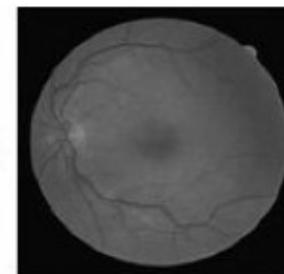
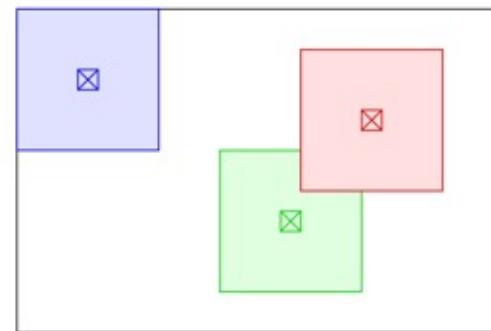


# 自适应直方图均衡化

直方图均衡的经典算法对整幅图像的像素使用相同的变换方法。如果图像中包括**明显的亮或暗区域**，则经典算法作用有限。

**自适应直方图均衡**（adaptive histogram equalization, AHE）通过对**局部区域**进行直方图均衡来解决上述问题。

- 移动模板在原图上按**特定步长**滑动；
- 滑动的每一步，模板区域内做直方图均衡，映射后的结果赋值给模板区域内所有点；
- 每个点会被多次赋值，最终取均值。

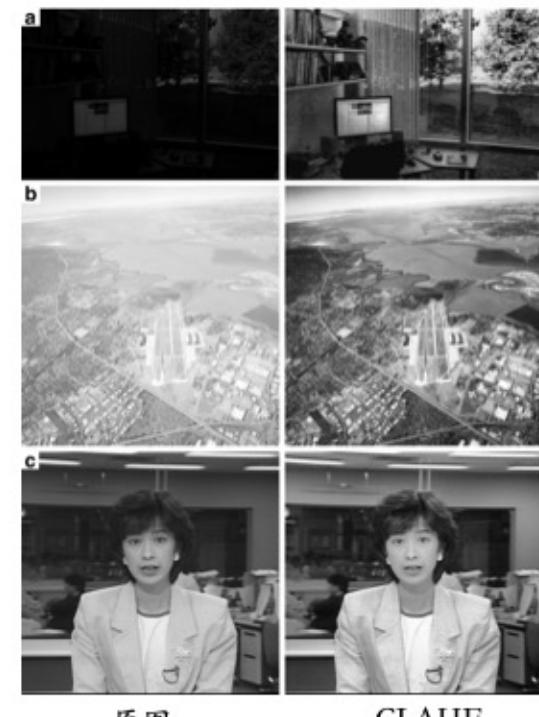
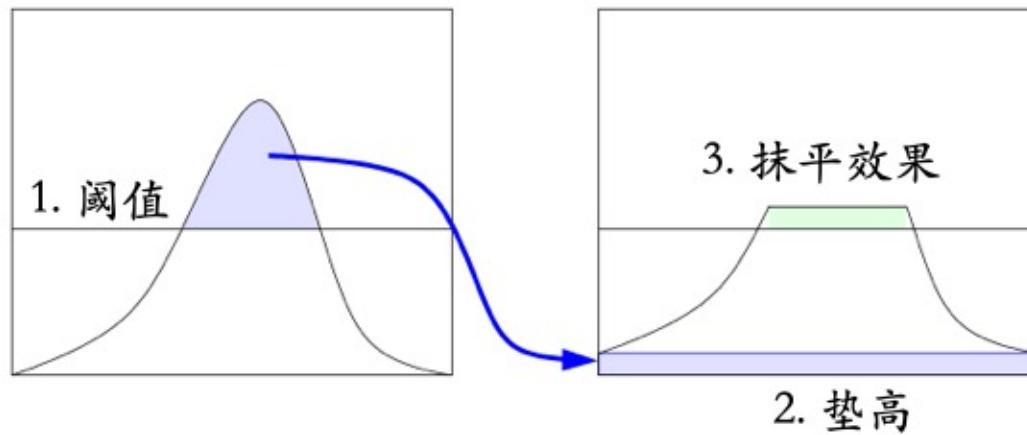


# CLAHE

AHE的缺点：会过度放大原图像中亮度相对均匀区域的噪音。

改进：限制对比度自适应直方图均衡（Contrast Limited AHE, CLAHE）。与普通的自适应直方图均衡相比，CLAHE会用修剪后的直方图进行图像均衡，最终输出图像的对比度更自然。

需要辅以双线性插值，让图像的灰度连续。

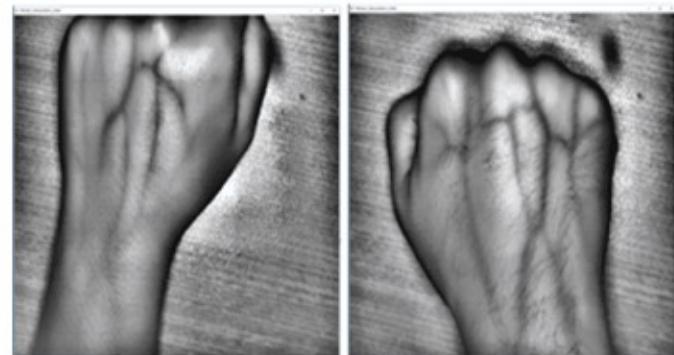
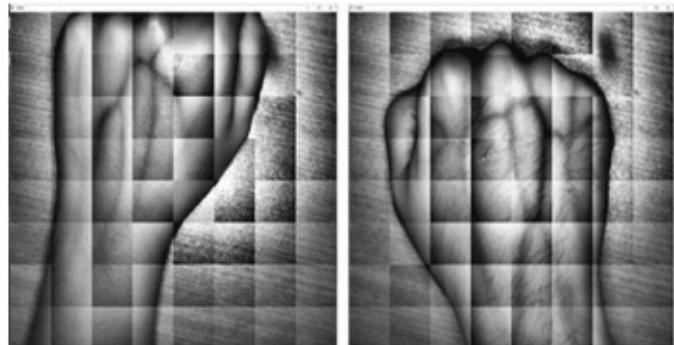
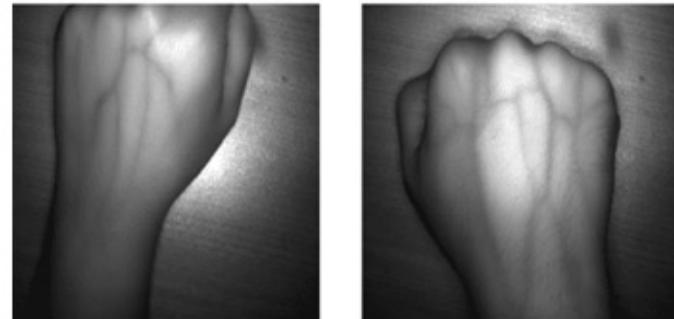
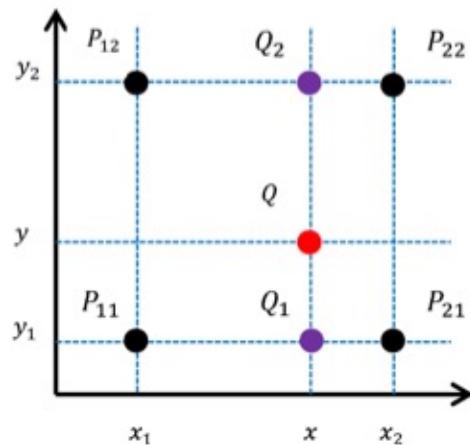


原图

CLAHE

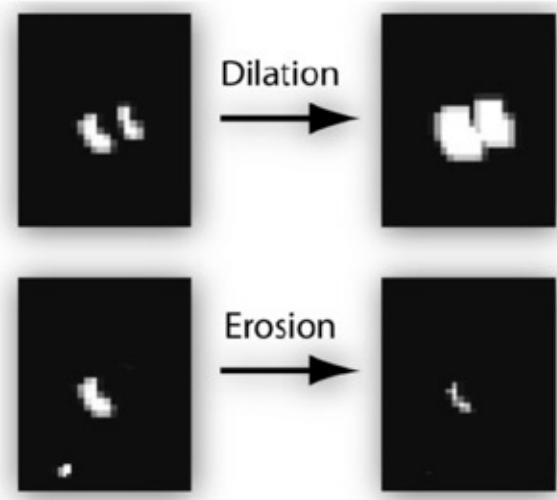
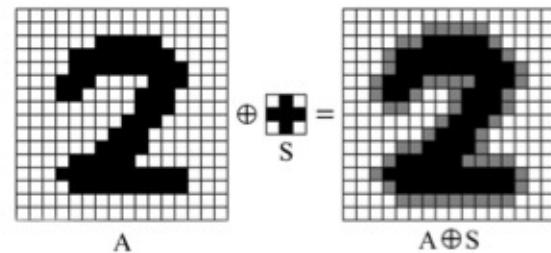
# CLAHE

1. 图像分块 (block)，以块为单位；
2. 每一块内，先计算直方图，然后设置阈值、进行切割和垫高操作、计算新的像素值；
3. 遍历操作各个图像块获得像素值；
4. 不同块间做双线性插值，从而调整像素值；
5. 必要时与原图做图层滤色混合操作。  
(可选)



# 形态学运算

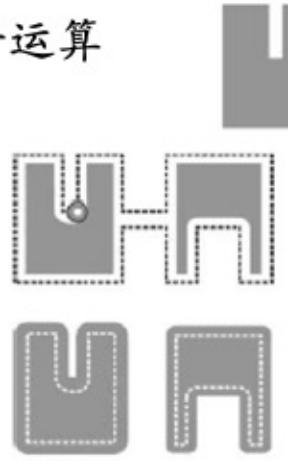
|      | 膨胀 dilation                                                        | 腐蚀 erosion                                                      |
|------|--------------------------------------------------------------------|-----------------------------------------------------------------|
| 定义   | 膨胀操作是通过结构元素（即“模板”，如卷积核）对图像进行局部区域进行的最大值操作，将核与图像进行卷积，用模板的最大值替代当前像素值。 | 腐蚀操作是通过结构元素与图像进行逐像素的比较，只有当结构元素完全覆盖对应图像区域时，输出图像对应像素才为前景。         |
| 效果   | 膨胀操作会使图像中的高亮区域（即前景或感兴趣的区域）扩张，因此可以用来填充高亮图像中的孔洞或断裂的区域。               | 与膨胀相反，腐蚀会使图像中的高亮区域缩小，从而去除细小的细节和孤立的高亮像素。                         |
| 应用场景 | 膨胀操作常用于特征增强、去噪、分割和边缘检测等任务。例如，在处理二值图像时，膨胀操作可以扩大高亮区域（即前景）。           | 腐蚀操作常用于去除图像中的小噪点和细小的连通区域，以及分离接触在一起的物体。例如，在处理二值图像时，腐蚀操作可以缩小高亮区域。 |



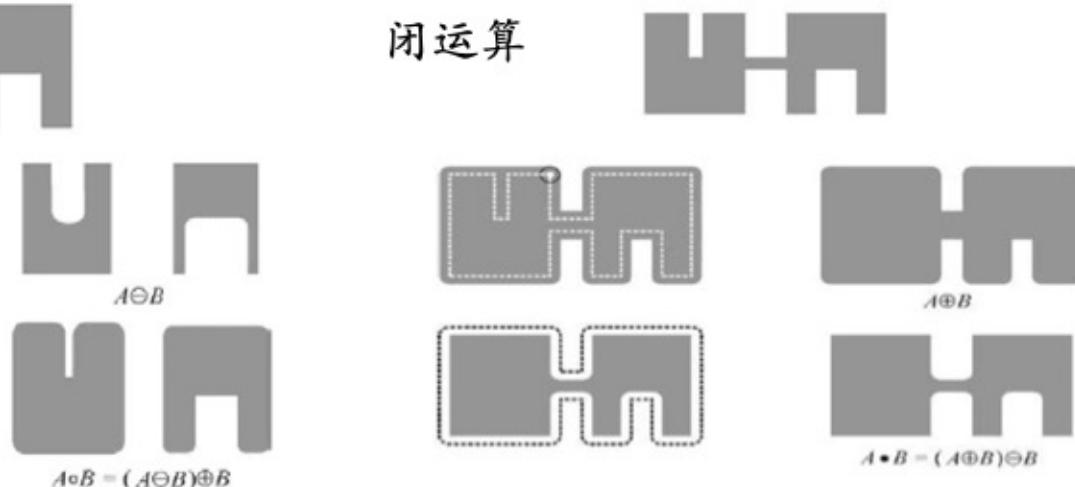
# 形态学运算

- 开运算：先进行腐蚀运算，再进行膨胀运算的组合。它能够除去孤立小点、物体边缘的毛刺，以及连接两个物体的桥状带，从而平滑边缘。
- 闭运算：先进行膨胀运算，再进行腐蚀运算的组合。这种运算可以填充物体内的小空洞、连接两个邻近的物体，从而平滑边缘。
- 它们可以有效地改善图像质量，为后续的图像分析和识别提供便利。

开运算



闭运算



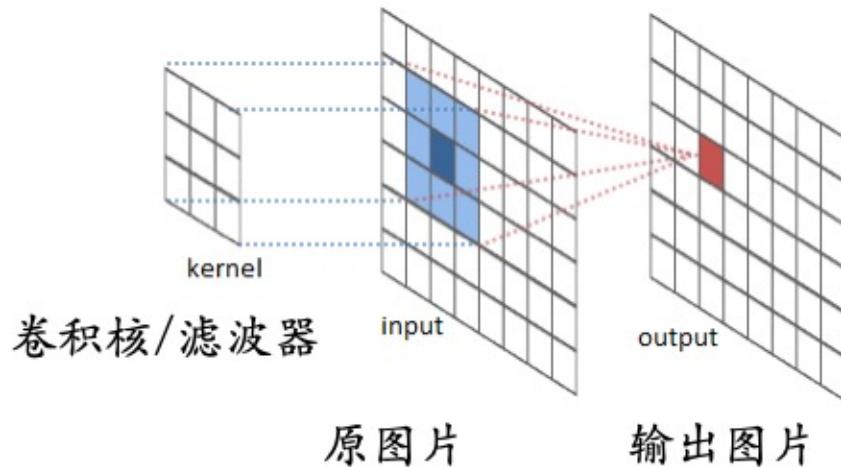
# 卷积操作

在图像的每个像素位置  $(x, y)$  上进行  
基于领域的函数计算。

- 卷积核、卷积模板
- 滤波器、滤波模板
- 扫描窗

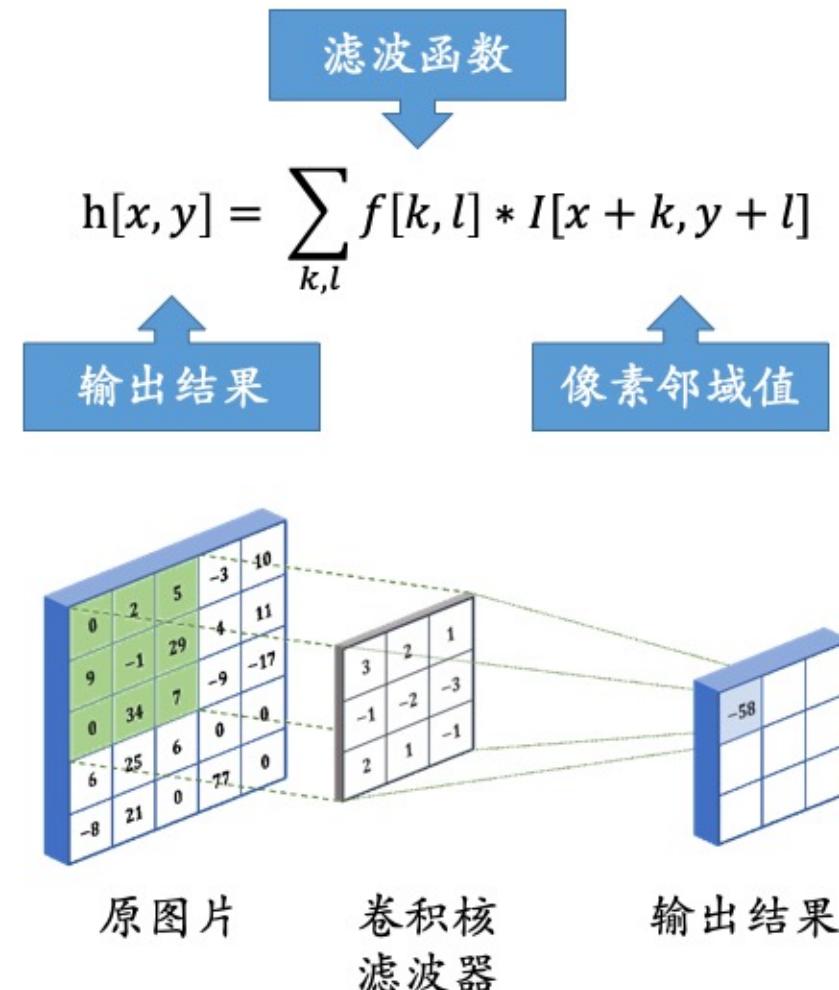
不同功能定义不同的函数：

- 平滑、降噪
- 梯度、锐化
- 边缘、斑点、纹理检测
- 模式识别

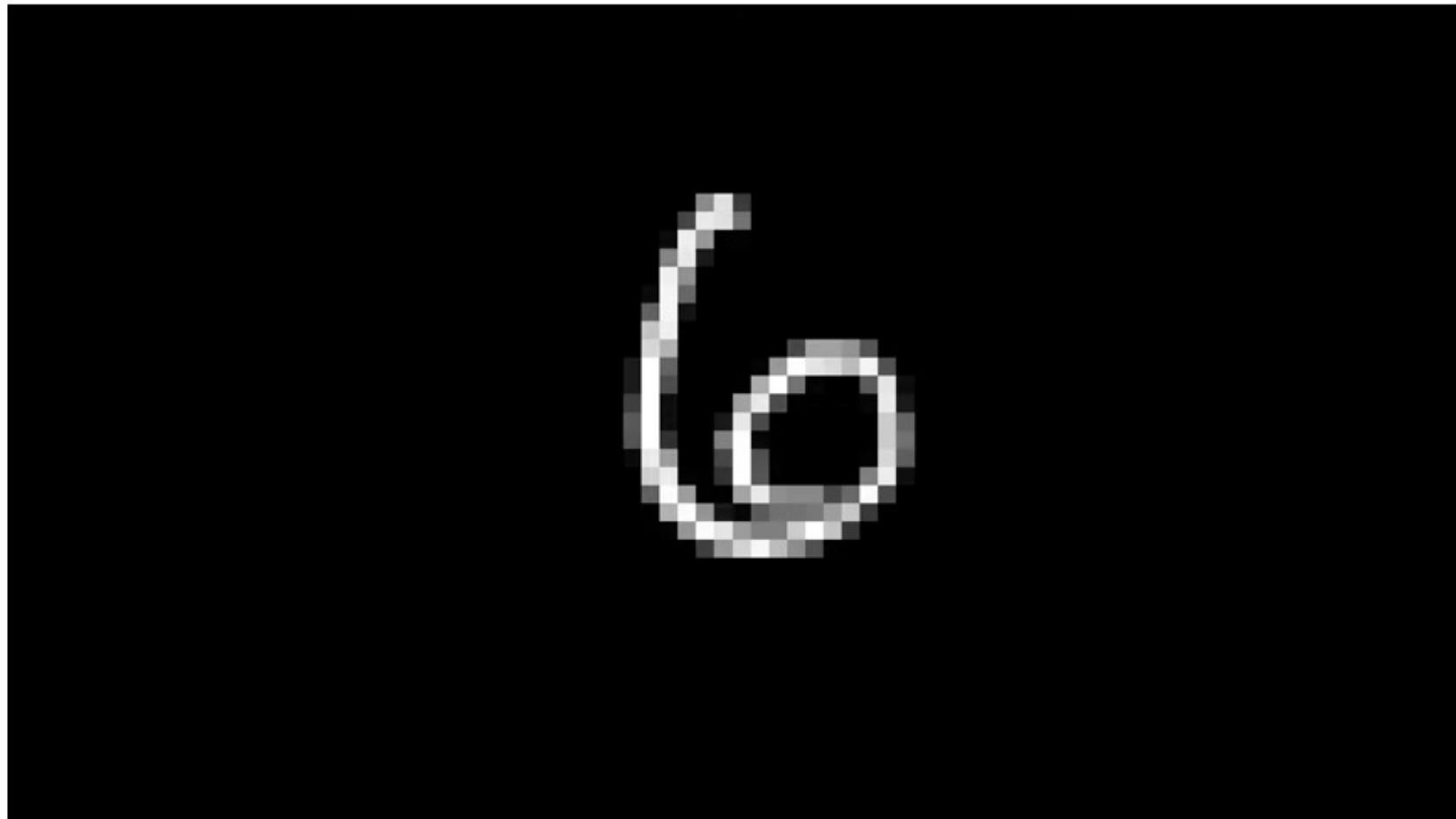


# 卷积操作

- $x, y$ : 像素在图像里的坐标。
- $k, l$ : 卷积核中的位置坐标，中心点坐标为 $(0, 0)$ 。
- $f[k, l]$ : 卷积核在 $(k, l)$ 位置上的权重参数。
- $I[x+k, y+l]$ : 图像中和 $f[k, l]$ 位置对应像素的值。
- $h[x, y]$ : 图片中 $(x, y)$ 像素的滤波/卷积结果。



# 卷积操作



# 卷积操作

原图像

|       |       |       |  |  |  |
|-------|-------|-------|--|--|--|
| $x_1$ | $x_2$ | $x_3$ |  |  |  |
| $x_4$ | $x_5$ | $x_6$ |  |  |  |
| $x_7$ | $x_8$ | $x_9$ |  |  |  |
|       |       |       |  |  |  |
|       |       |       |  |  |  |
|       |       |       |  |  |  |

卷积核

|       |       |       |
|-------|-------|-------|
| $w_1$ | $w_2$ | $w_3$ |
| $w_4$ | $w_5$ | $w_6$ |
| $w_7$ | $w_8$ | $w_9$ |

特征图

|     |  |  |  |
|-----|--|--|--|
| $y$ |  |  |  |
|     |  |  |  |
|     |  |  |  |
|     |  |  |  |
|     |  |  |  |

$$y = w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + \\ w_5x_5 + w_6x_6 + w_7x_7 + w_8x_8 + w_9x_9$$

使用 **卷积核** (convolution kernel, 也称为模板、算子或掩膜) 对原图像中的 **每个像素** 进行重新计算，形成“特征图”。具体来说，卷积核是一个矩阵，通常是一个**较小**的四方形网格结构，边长通常为奇数个像素（如3x3的矩阵）。在进行卷积计算时，卷积核的**中心会对准**原图像的某个像素点 (A)。然后，卷积核中的**每个元素**会与其覆盖的图像像素值**一一相乘**。这些**乘积求和**的结果，就是特征图中该相应位置 (A) 的新像素值。

# 卷积操作

卷积核和图像的对应位置分别相乘，再将每个位置的乘积相加。

|    |   |   |
|----|---|---|
| 2  | 0 | 1 |
| 3  | 1 | 0 |
| -1 | 4 | 1 |

|    |    |    |    |
|----|----|----|----|
| 5  | 6  | 11 | 13 |
| 12 | 0  | 7  | 9  |
| 8  | 11 | 2  | 6  |
| 3  | 3  | 4  | 2  |

|    |   |   |
|----|---|---|
| 2  | 0 | 1 |
| 3  | 1 | 0 |
| -1 | 4 | 1 |

|    |    |    |    |
|----|----|----|----|
| 5  | 6  | 11 | 13 |
| 12 | 0  | 7  | 9  |
| 8  | 11 | 2  | 6  |
| 3  | 3  | 4  | 2  |

|    |   |   |
|----|---|---|
| 2  | 0 | 1 |
| 3  | 1 | 0 |
| -1 | 4 | 1 |

|    |    |    |    |
|----|----|----|----|
| 5  | 6  | 11 | 13 |
| 12 | 0  | 7  | 9  |
| 8  | 11 | 2  | 6  |
| 3  | 3  | 4  | 2  |

|    |   |   |
|----|---|---|
| 2  | 0 | 1 |
| 3  | 1 | 0 |
| -1 | 4 | 1 |

|    |    |    |    |
|----|----|----|----|
| 5  | 6  | 11 | 13 |
| 12 | 0  | 7  | 9  |
| 8  | 11 | 2  | 6  |
| 3  | 3  | 4  | 2  |

卷积核

图像

卷积结果（特征图）

$$\begin{aligned} & 2 \times 5 + 0 \times 6 + 1 \times 11 \\ & + 3 \times 12 + 1 \times 0 + 0 \times 7 \\ & + (-1) \times 8 + 4 \times 11 + 1 \times 2 \\ & = 95 \end{aligned}$$

|    |  |
|----|--|
| 95 |  |
|    |  |

$$\begin{aligned} & 2 \times 6 + 0 \times 11 + 1 \times 13 \\ & + 3 \times 0 + 1 \times 7 + 0 \times 9 \\ & + (-1) \times 11 + 4 \times 2 + 1 \times 6 \\ & = 35 \end{aligned}$$

|    |    |
|----|----|
| 95 | 35 |
|    |    |

$$\begin{aligned} & 2 \times 12 + 0 \times 0 + 1 \times 7 \\ & + 3 \times 8 + 1 \times 11 + 0 \times 2 \\ & + (-1) \times 3 + 4 \times 3 + 1 \times 4 \\ & = 79 \end{aligned}$$

|    |    |
|----|----|
| 95 | 35 |
| 79 |    |

$$\begin{aligned} & 2 \times 0 + 0 \times 7 + 1 \times 9 \\ & + 3 \times 11 + 1 \times 2 + 0 \times 6 \\ & + (-1) \times 3 + 4 \times 4 + 1 \times 2 \\ & = 59 \end{aligned}$$

|    |    |
|----|----|
| 95 | 35 |
| 79 | 59 |

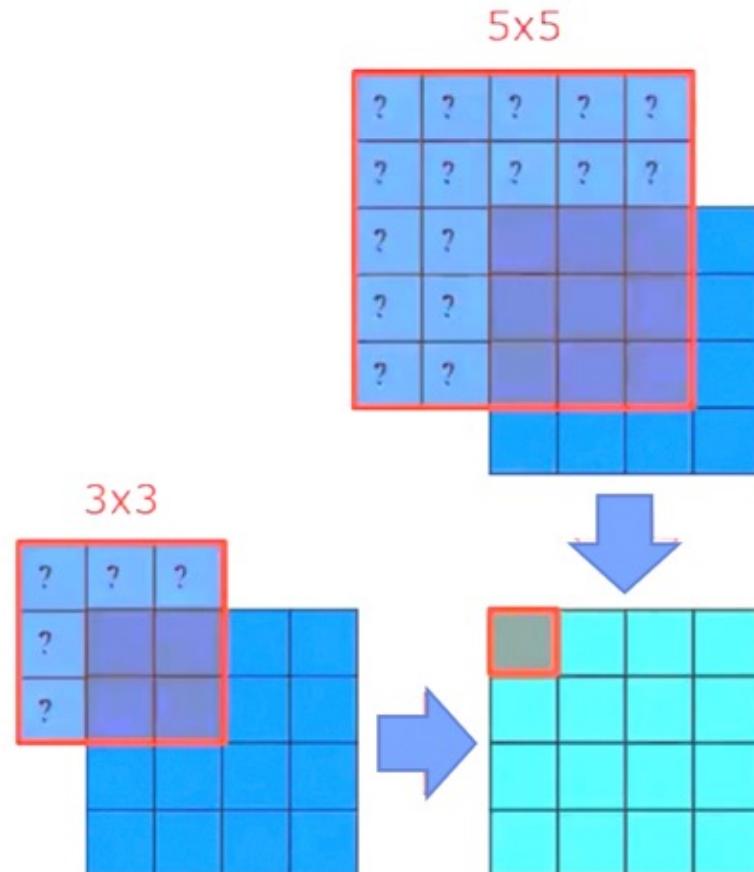
# 边缘填充

## 边缘填充 (padding)

- 如果需要获得和原图同尺寸的输出图像，需要进行边缘填充
- 卷积核越大，补充越多

### 边缘填充类型

- 补零，也称零填充 (zero padding)
- 边界复制 (replication)
- 镜像 (reflection)
- 块复制 (wrap-around)



# 边缘填充

补零

|   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 2 | 3 | 0 | 0 | 0 |
| 0 | 0 | 0 | 4 | 5 | 6 | 0 | 0 | 0 |
| 0 | 0 | 0 | 7 | 8 | 9 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

边界复制

|   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 2 | 3 | 3 | 3 | 3 |
| 1 | 1 | 1 | 1 | 2 | 3 | 3 | 3 | 3 |
| 1 | 1 | 1 | 1 | 2 | 3 | 3 | 3 | 3 |
| 1 | 1 | 1 | 1 | 2 | 3 | 3 | 3 | 3 |
| 4 | 4 | 4 | 4 | 5 | 6 | 6 | 6 | 6 |
| 7 | 7 | 7 | 7 | 8 | 9 | 9 | 9 | 9 |
| 7 | 7 | 7 | 7 | 8 | 9 | 9 | 9 | 9 |
| 7 | 7 | 7 | 7 | 8 | 9 | 9 | 9 | 9 |
| 7 | 7 | 7 | 7 | 8 | 9 | 9 | 9 | 9 |

# 边缘填充

镜像

|   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|
| 9 | 8 | 7 | 7 | 8 | 9 | 9 | 8 | 7 |
| 6 | 5 | 4 | 4 | 5 | 6 | 6 | 5 | 4 |
| 3 | 2 | 1 | 1 | 2 | 3 | 3 | 2 | 1 |
| 3 | 2 | 1 | 1 | 2 | 3 | 3 | 2 | 1 |
| 6 | 5 | 4 | 4 | 5 | 6 | 6 | 5 | 4 |
| 9 | 8 | 7 | 7 | 8 | 9 | 9 | 8 | 7 |
| 9 | 8 | 7 | 7 | 8 | 9 | 9 | 8 | 7 |
| 6 | 5 | 4 | 4 | 5 | 6 | 6 | 5 | 4 |
| 3 | 2 | 1 | 1 | 2 | 3 | 3 | 2 | 1 |

块复制

|   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 1 | 2 | 3 | 1 | 2 | 3 |
| 4 | 5 | 6 | 4 | 5 | 6 | 4 | 5 | 6 |
| 7 | 8 | 9 | 7 | 8 | 9 | 7 | 8 | 9 |
| 1 | 2 | 3 | 1 | 2 | 3 | 1 | 2 | 3 |
| 4 | 5 | 6 | 4 | 5 | 6 | 4 | 5 | 6 |
| 7 | 8 | 9 | 7 | 8 | 9 | 7 | 8 | 9 |
| 1 | 2 | 3 | 1 | 2 | 3 | 1 | 2 | 3 |
| 4 | 5 | 6 | 4 | 5 | 6 | 4 | 5 | 6 |
| 7 | 8 | 9 | 7 | 8 | 9 | 7 | 8 | 9 |

# 边缘填充

|   |     |     |     |     |     |
|---|-----|-----|-----|-----|-----|
| 0 | 0   | 0   | 0   | 0   | 0   |
| 0 | 105 | 102 | 100 | 97  | 96  |
| 0 | 103 | 99  | 103 | 101 | 102 |
| 0 | 101 | 98  | 104 | 102 | 100 |
| 0 | 99  | 101 | 106 | 104 | 99  |
| 0 | 104 | 104 | 104 | 100 | 98  |

## Image Matrix

$$0 * 0 + 0 * -1 + 0 * 0 \\ + 0 * -1 + 105 * 5 + 102 * -1 \\ + 0 * 0 + 103 * -1 + 99 * 0 = 320$$

| Kernel Matrix |    |    |
|---------------|----|----|
| 0             | -1 | 0  |
| -1            | 5  | -1 |
| 0             | -1 | 0  |

## Kernel Matrix

|     |  |  |  |
|-----|--|--|--|
| 320 |  |  |  |
|     |  |  |  |
|     |  |  |  |
|     |  |  |  |
|     |  |  |  |

### Output Matrix

## Convolution with horizontal and vertical strides = 1

# 均值滤波

卷积操作在深度学习中多用于提取特征。  
我们也可以用类似于卷积操作的手段，  
在预处理阶段对图像进行滤波。

效果不佳：

- 不能完全去除噪声
- 破坏了图像的有用细节

|   |    |    |    |    |    |    |   |   |
|---|----|----|----|----|----|----|---|---|
| 0 | 0  | 0  | 0  | 0  | 0  | 0  | 0 | 0 |
| 0 | 9  | 9  | 9  | 18 | 18 | 18 | 0 |   |
| 0 | 9  | 9  | 9  | 18 | 18 | 18 | 0 |   |
| 0 | 9  | 9  | 9  | 18 | 18 | 18 | 0 |   |
| 0 | 18 | 18 | 18 | 9  | 9  | 9  | 0 |   |
| 0 | 18 | 18 | 18 | 9  | 9  | 9  | 0 |   |
| 0 | 18 | 18 | 18 | 9  | 9  | 9  | 0 |   |
| 0 | 0  | 0  | 0  | 0  | 0  | 0  | 0 |   |



|    |    |    |    |    |    |
|----|----|----|----|----|----|
| 4  | 6  | 8  | 10 | 12 | 8  |
| 6  | 9  | 12 | 15 | 18 | 12 |
| 8  | 12 | 13 | 14 | 15 | 10 |
| 10 | 15 | 14 | 13 | 12 | 8  |
| 12 | 18 | 15 | 12 | 9  | 6  |
| 8  | 12 | 10 | 8  | 6  | 4  |

举例：3x3卷积核、零填充（滤波器）  
均值滤波：卷积核中的9个像素取平均值

# 中值滤波

|    |    |    |    |    |    |    |   |
|----|----|----|----|----|----|----|---|
| 11 | 7  | 4  | 5  | 3  | 3  | 2  | 2 |
| 38 | 22 | 10 | 7  | 4  | 3  | 3  | 2 |
| 73 | 60 | 29 | 13 | 7  | 5  | 3  | 2 |
| 69 | 69 | 52 | 29 | 12 | 7  | 4  | 3 |
| 62 | 66 | 66 | 59 | 27 | 11 | 7  | 3 |
| 66 | 60 | 60 | 66 | 62 | 25 | 8  | 4 |
| 58 | 54 | 56 | 62 | 74 | 42 | 13 | 6 |
| 49 | 49 | 51 | 54 | 58 | 50 | 25 | 9 |

Original image

min  
7  
10  
11  
22  
29  
38  
60  
73  
max

|  |  |  |  |  |  |  |  |  |
|--|--|--|--|--|--|--|--|--|
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |

Sort and rank

Median image

