

图像基本概念

什么是图像

- 用各种观测系统以不同形式和手段观测客观世界而获得的，可以直接或间接作用于人眼并进而产生视知觉的实体
- “图”是物体投射或反射光的分布，“像”是人的视觉系统对图的接受在大脑中形成的印像或反映。
- 图像是自然界景物的客观反映。

举例：人眼所见

照片

电视电影

图像的分类

(1) 从视觉特点 , 分为 :

可见图像

不可见图像 红外图像, SAR图像

(2) 从图像空间坐标和明暗程度的连续性 ,

可分为 :

模拟图像

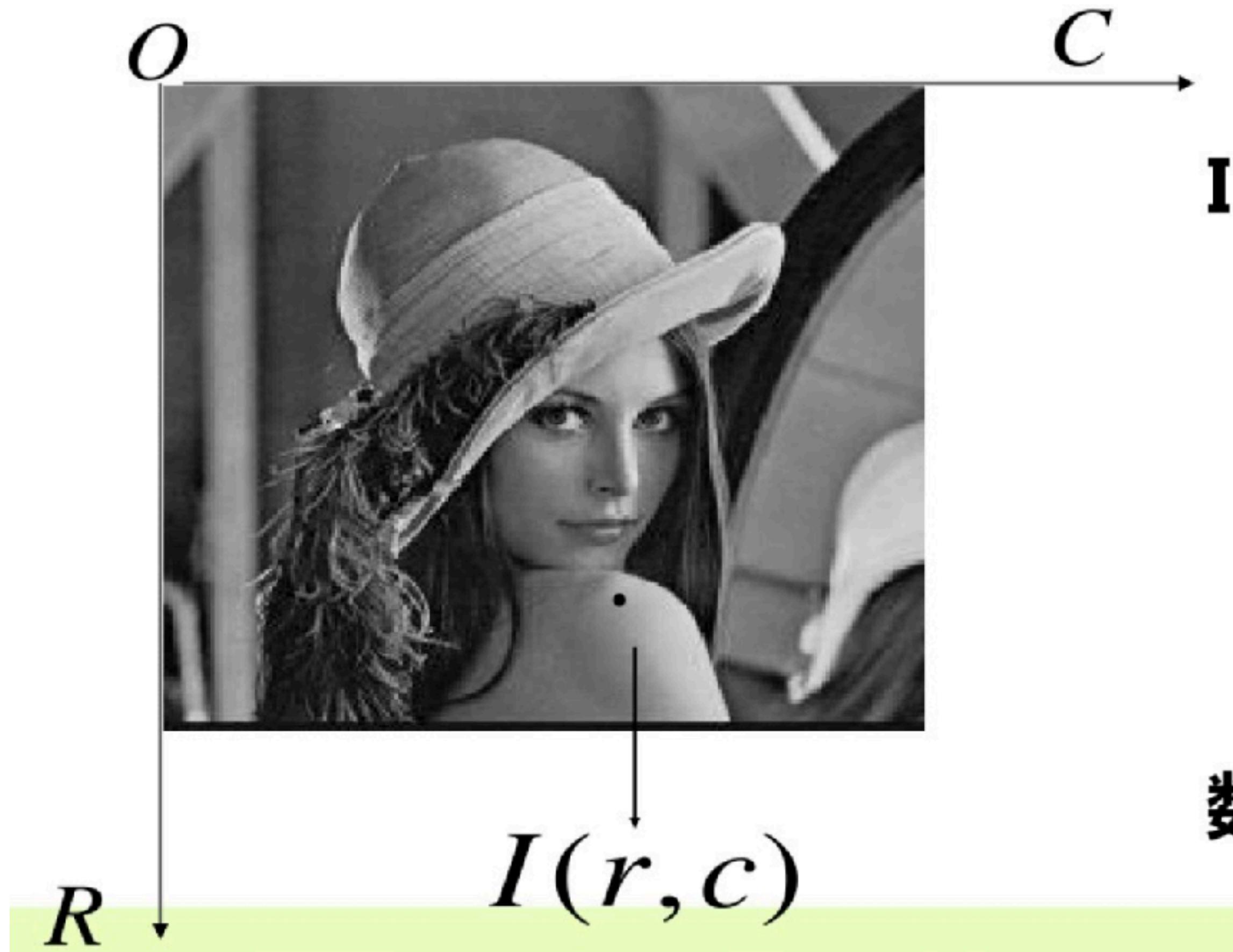
数字图像

数字图像怎么得到

- 连续图像、离散图像
- 图像数字化就是将连续图像离散化。
- 图像数字化包括取样和量化两个部分。
 - 取样：空间上的离散化
在空间上分割成 $M \times N$ 个网格（空间分辨率）
 - 量化：亮度上的离散化
把取样点上对应的亮度连续变化区间
转换为单个数码的过程

模拟图像是数字化得到数字图像

数字图像怎么得到



模拟图像数字化得到数字图像

$I(r, c)$ 是对 $f(x, y)$ 的离散化后的结果。

r : 表示图像的行(row) ;

c : 表示图像的列(column) ;

I : 表示离散后的 f ; 灰阶, 灰度级

I, r, c 的值只能是整数。

数字图像可用矩阵或数组进行描述。

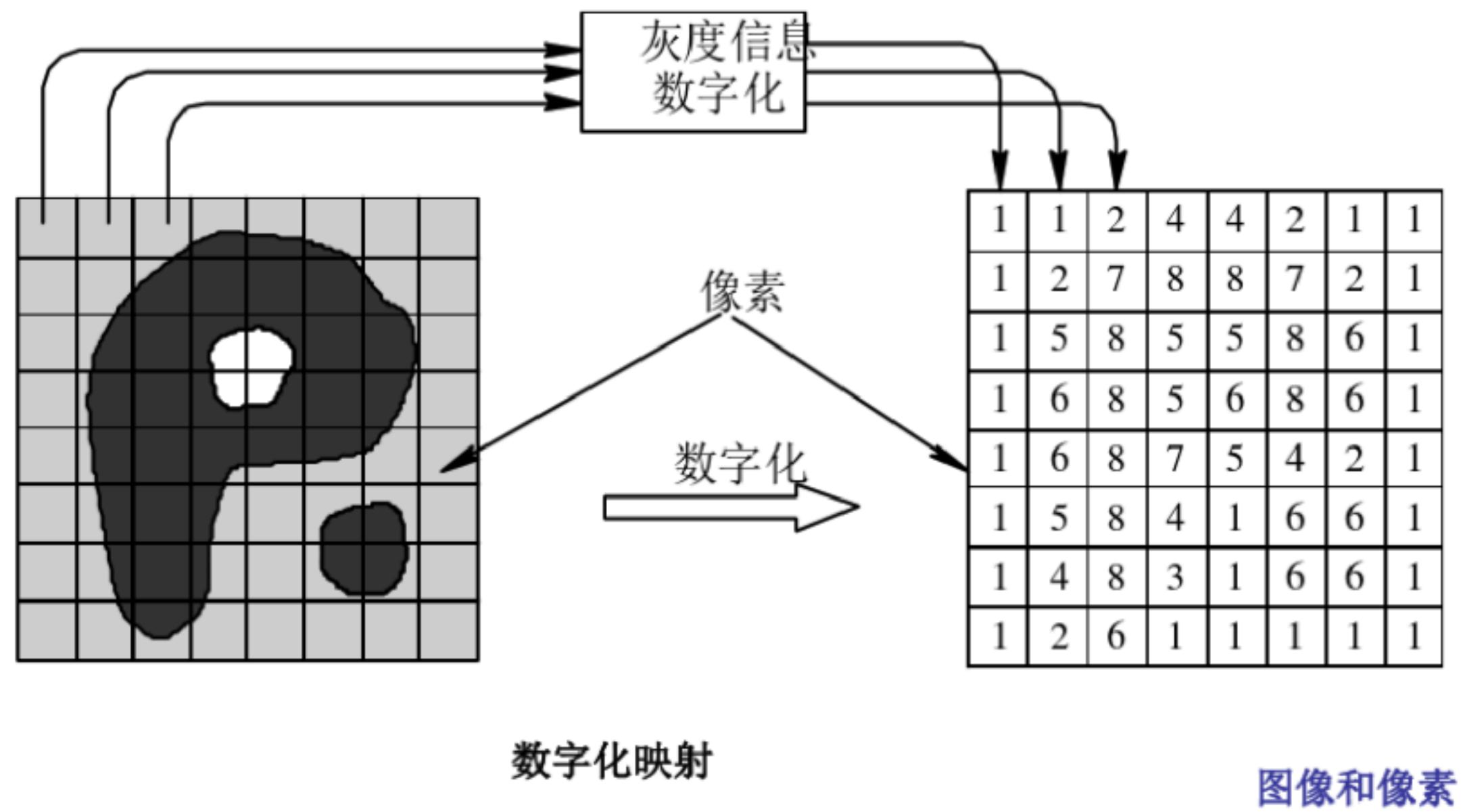
- 图像分辨率：分辨率越高，就是组成一幅图的像素越多，则图像文件越大。
- 像素深度：像素深度越深，就是表达单个像素的颜色和亮度的位数越多，图像文件越大。

数字图像的表达

- 图像分辨率：分辨率越高，就是组成一幅图的像素越多，则图像文件越大。
- 像素深度：像素深度越深，就是表达单个像素的颜色和亮度的位数越多，图像文件越大。灰阶，灰度级

理论上数字图像对模拟图像采样的图像分辨率越大，灰度级越高，反映的真实效果越好，但是给存储带来的压力就越大，所以合适最好

数字图像的邻域

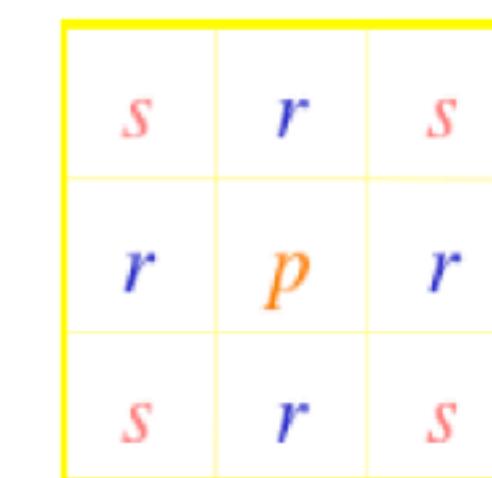
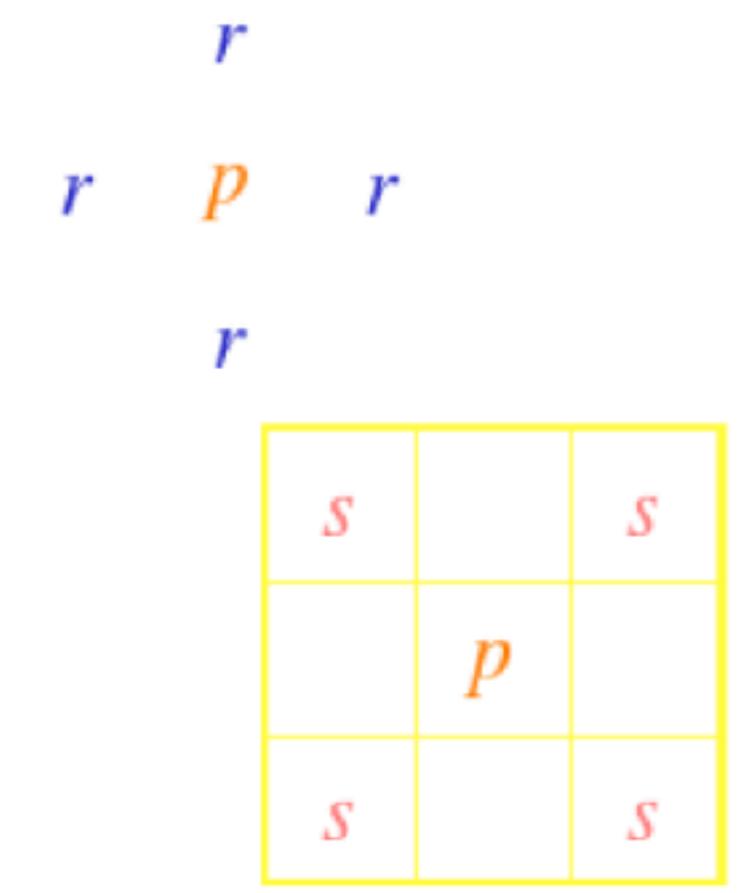


$N_4(p)$ ——象素(x, y)的4邻域
 $(x+1, y), (x-1, y),$
 $(x, y+1), (x, y-1)$

$N_D(p)$ ——象素(x, y)的对角邻域
 $(x+1, y+1), (x+1, y-1),$
 $(x-1, y+1), (x-1, y-1)$

$N_8(p)$ ——象素(x, y)的8邻域

$$N_4(p) + N_D(p)$$



数字图像的分类

a. 黑白图像

- 定义：只有黑白两种颜色的图像称为黑白图像或单色图像，是指图像的每个像素只能是黑或者白，没有中间的过渡，故又称为二值图像。
- 特点：二值图像的像素值只能为**0**和**1**，图像中的每个像素值用**1位**存储。



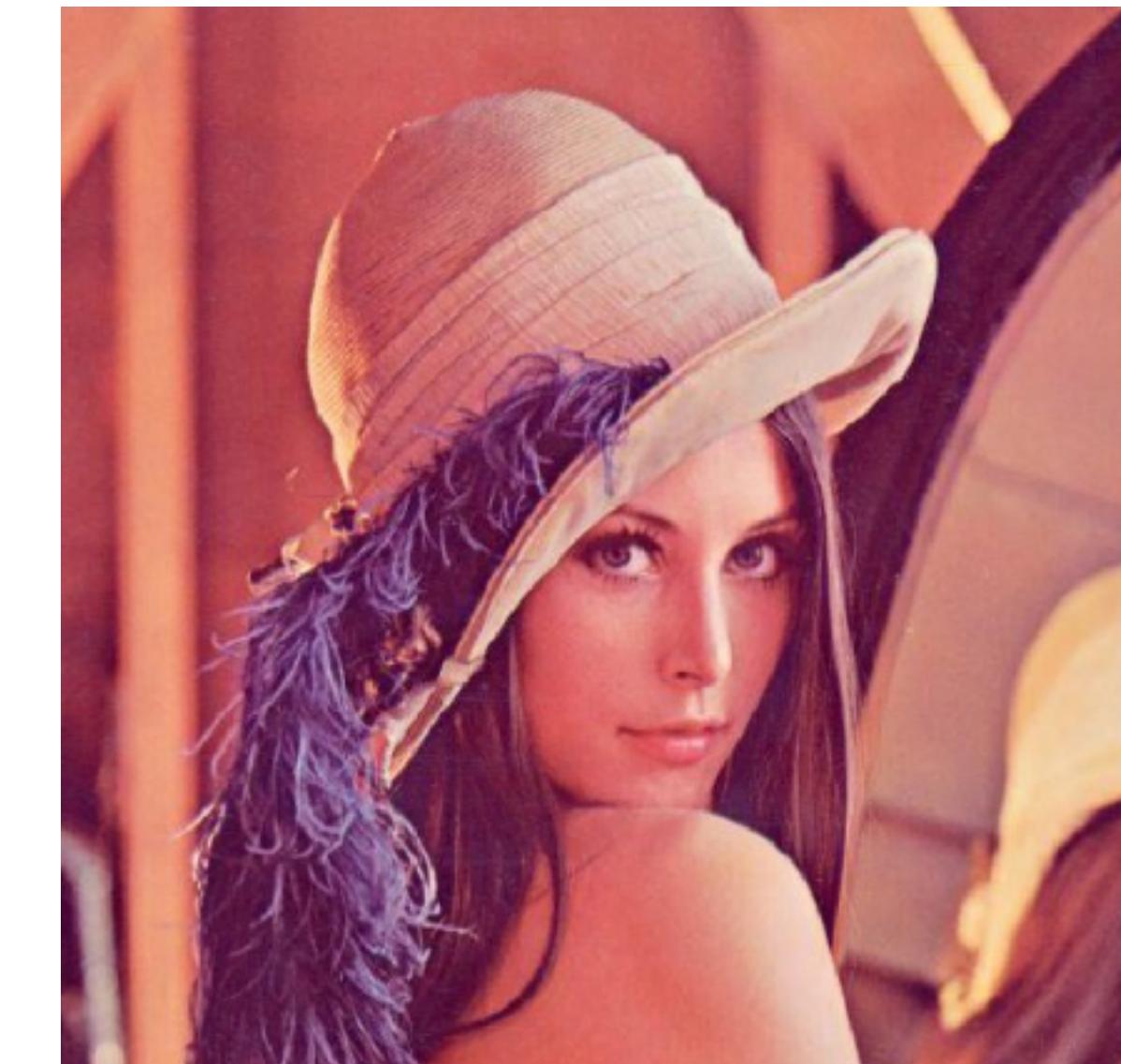
b. 灰度图像

- 定义：灰度图像是指每个像素的信息由一个量化的灰度级来描述的图像。如果每个像素的灰度值用一个字节表示，灰度值级数就等于**256**级，每个像素可以是**0~255**之间的任何一个数。
- 特点：它只有亮度信息，没有颜色信息。占据存储空间较黑白图像要大。



c. 彩色图像

- 定义：彩色图像除有亮度信息外，还包含有颜色信息。
- 特点：彩色图像的表示与所采用的彩色空间，即彩色的表示模型有关。同一幅彩色图像如果采用不同的彩色空间表示，对其的描述可能会有很大不同。



图像分割



`retval, dst=cv.threshold(src, thresh, maxval, type[, dst])`

阈值法 大津阈值 `cv.THRESH_OTSU`

`dst=cv.adaptiveThreshold(src, maxValue, adaptiveMethod, thresholdType, blockSize, C[, dst])`

`ADAPTIVE_THRESH_MEAN_C`
Python: `cv.ADAPTIVE_THRESH_MEAN_C`
the threshold value $T(x, y)$ is a mean of the $\text{blockSize} \times \text{blockSize}$ neighborhood of (x, y) minus C

`ADAPTIVE_THRESH_GAUSSIAN_C`
Python: `cv.ADAPTIVE_THRESH_GAUSSIAN_C`
the threshold value $T(x, y)$ is a weighted sum (cross-correlation with a Gaussian window) of the $\text{blockSize} \times \text{blockSize}$ neighborhood of (x, y) minus C. The default sigma (standard

❖ 有两种不同的自适应阈值计算方法，在这两种情况下，自适应阈值 $T(x,y)$ 在每个像素点都不同。

❖ 通过计算像素点周围**b*b**区域的加权平均，然后减去一个常数来得到该点的自适应阈值。

❖ b由参数**block_size**指定，常数由**param1**指定，加权方法由**adaptive_method**指定：

-CV_ADAPTIVE_THRESH_MEAN_C——区域像素平均加权

-CV_ADAPTIVE_THRESH_GAUSSIAN_C——区域像素高斯加权（离中心越近对阈值影响越大）

THRESH_BINARY Python: <code>cv.THRESH_BINARY</code>	$dst(x, y) = \begin{cases} \text{maxval} & \text{if } \text{src}(x, y) > \text{thresh} \\ 0 & \text{otherwise} \end{cases}$
THRESH_BINARY_INV Python: <code>cv.THRESH_BINARY_INV</code>	$dst(x, y) = \begin{cases} 0 & \text{if } \text{src}(x, y) > \text{thresh} \\ \text{maxval} & \text{otherwise} \end{cases}$
THRESH_TRUNC Python: <code>cv.THRESH_TRUNC</code>	$dst(x, y) = \begin{cases} \text{threshold} & \text{if } \text{src}(x, y) > \text{thresh} \\ \text{src}(x, y) & \text{otherwise} \end{cases}$
THRESH_TOZERO Python: <code>cv.THRESH_TOZERO</code>	$dst(x, y) = \begin{cases} \text{src}(x, y) & \text{if } \text{src}(x, y) > \text{thresh} \\ 0 & \text{otherwise} \end{cases}$
THRESH_TOZERO_INV Python: <code>cv.THRESH_TOZERO_INV</code>	$dst(x, y) = \begin{cases} 0 & \text{if } \text{src}(x, y) > \text{thresh} \\ \text{src}(x, y) & \text{otherwise} \end{cases}$
THRESH_MASK Python: <code>cv.THRESH_MASK</code>	
THRESH_OTSU Python: <code>cv.THRESH_OTSU</code>	flag, use Otsu algorithm to choose the optimal threshold value
THRESH_TRIANGLE Python: <code>cv.THRESH_TRIANGLE</code>	flag, use Triangle algorithm to choose the optimal threshold value

图像分割

阈值法

漫水法

水域法

金字塔分割

均值漂移分割

背景减除

漫水法 `retval, image, mask, rect=cv.floodFill(image, mask, seedPoint, newVal[, loDiff[, upDiff[, flags]]])`

分水岭算法 `markers=cv.watershed(image, markers)`

Grabcuts分割算法 `mask, bgdModel, fgdModel=cv.grabCut(img, mask, rect, bgdModel, fgdModel, iterCount[, mode])`

均值漂移分割算法 `dst=cv.pyrMeanShiftFiltering(src, sp, sr[, dst[, maxLevel[, termcrit]]])`

背景减除法 帧差法、平均背景法、高斯背景法、**Codebook**法

图像滤波（平滑处理）

均值滤波

加和滤波

中值滤波

高斯滤波

双边滤波

其它滤波

◆滤波是将信号中特定频段信号滤除的操作，是抑制和防止干扰的一项重要措施。

- 低通滤波器：允许低频率通过
- 高通滤波器：允许高频率通过
- 带通滤波器：允许一定范围内频率通过
- 带阻滤波器：阻止一定范围内频率通过
- 全通滤波器：允许所有频率通过，仅改变相位关系。

◆模糊：平滑，是一种低通滤波

$$f(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

0.0453542	0.0566406	0.0453542
0.0566406	0.0707355	0.0566406
0.0453542	0.0566406	0.0453542

0.0947416	0.118318	0.0947416
0.118318	0.147761	0.118318
0.0947416	0.118318	0.0947416

高斯滤波

前三种是线性滤波器，后两种是非线性滤波器。

都属于空间域滤波，即在一定的图像窗口上进行的直接运算。

均值滤波

$$\text{均值} = \frac{1}{ksize.width \cdot ksize.height} \sum_{i=1}^{ksize.height} \sum_{j=1}^{ksize.width} \text{src}(i, j)$$

盒子滤波

$$K = \alpha \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 & 1 \\ 1 & 1 & 1 & \cdots & 1 & 1 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & 1 & 1 & \cdots & 1 & 1 \end{bmatrix} \alpha = \begin{cases} \frac{1}{ksize.width * ksize.height} & \text{when } normalize=true \\ 1 & \text{otherwise} \end{cases}$$

空域核

$$d(i, j, k, l) = \exp\left(-\frac{(i-k)^2 + (j-l)^2}{2\sigma_d^2}\right)$$

值域核

$$r(i, j, k, l) = \exp\left(-\frac{\|f(i, j) - f(k, l)\|^2}{2\sigma_r^2}\right)$$

双边滤波权重

$$w(i, j, k, l) = \exp\left(-\frac{(i-k)^2 + (j-l)^2}{2\sigma_d^2} - \frac{\|f(i, j) - f(k, l)\|^2}{2\sigma_r^2}\right)$$

数学运算

函数

简单模糊

`blur(InputArray src, OutputArray dst, Size ksize)`

盒子滤波器

`boxFilter(InputArray src, OutputArray dst, int ddepth, Size ksize)`

高斯滤波器

`GaussianBlur(InputArray src, OutputArray dst, Size ksize, double sigmaX, double sigmaY=0,`

中值滤波器

`medianBlur(InputArray src, OutputArray dst, int ksize)`

双边滤波器

`bilateralFilter(InputArray src, OutputArray dst, int d, double sigmaColor, double sigmaSpace,`

双边滤波参考 <https://www.jianshu.com/p/8d11e26c9665>

边缘检测

Sobel边缘
检测

Scharr边缘
检测

拉普拉斯边
缘检测

Canny边缘
检测

形态学梯度
边缘检测

一阶微分最优边缘检测器 ——Canny算子

Canny(InputArray **image**, OutputArray **edges**,
double **threshold1**, double **threshold2**,
int **apertureSize**=3, bool **L2gradient**=false)

Sobel(InputArray **src**, OutputArray **dst**, int **ddepth**,
int **xorder**, int **yorder**, int **ksize**=3,
double **scale**=1, double **delta**=0,
int **borderType**=BORDER_DEFAULT)

$$s_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$$s_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

$$G_x = \begin{bmatrix} 0.0366 & 0.0821 & 0 & -0.0821 & -0.0366 \\ 0.1642 & 0.3679 & 0 & -0.3679 & -0.1642 \\ 0.2707 & 0.6065 & 0 & -0.6065 & -0.2707 \\ 0.1642 & 0.3679 & 0 & -0.3679 & -0.1642 \\ 0.0366 & 0.0821 & 0 & -0.0821 & -0.0366 \end{bmatrix}$$

$$G_y = \begin{bmatrix} 0.0366 & 0.1642 & 0.2707 & 0.16421 & 0.03666 \\ 0.0821 & 0.3679 & 0.6065 & 0.36799 & 0.0821 \\ 0 & 0 & 0 & 0 & 0 \\ -0.0821 & -0.3679 & -0.6065 & -0.3679 & -0.0821 \\ -0.0366 & -0.1642 & -0.2707 & -0.1642 & -0.0366 \end{bmatrix}$$

Scharr

$$\begin{array}{c} \begin{bmatrix} -3 & 0 & 3 \\ -10 & 0 & 10 \\ -3 & 0 & 3 \end{bmatrix} \quad \begin{bmatrix} -3 & -10 & -3 \\ 0 & 0 & 0 \\ 3 & 10 & 3 \end{bmatrix} \end{array}$$

Laplacian(InputArray **src**, OutputArray **dst**, int **ddepth**,
int **ksize**=3, double **scale**=1, double **delta**=0,
int **borderType**=BORDER_DEFAULT)

$$\nabla^2 \approx \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

$$\nabla^2 \approx \begin{bmatrix} 1 & 4 & 1 \\ 4 & -20 & 4 \\ 1 & 4 & 1 \end{bmatrix}$$

数学形态学

❖图像形态学是一门建立在数学形态基础之上的图像分析学科，是数学形态学图像处理的基本理论。其**基本的运算包括**：二值腐蚀（Erosion）和膨胀（形态学）（Dilation）、二值开闭运算、骨架抽取、极限腐蚀、击中击不中变换、形态学梯度、Top-hat变换、颗粒分析、流域变换、灰值腐蚀和膨胀、灰值开闭运算、灰值形态学梯度等。

膨胀 (Dilation)

$$dilate(x, y) = \max_{(x',y') \in B} src(x+x', y+y')$$

腐蚀(Erosion)

$$erode(x, y) = \min_{(x',y') \in B} src(x+x', y+y');$$

```
dilate(InputArray src, OutputArray dst, InputArray element,  
       Point anchor=Point(-1,-1), int iterations=1,  
       int borderType=BORDER_CONSTANT, const Scalar  
&borderValue=morphologyDefaultBorderValue() )
```

```
erode(InputArray src, OutputArray dst, InputArray element,  
       Point anchor=Point(-1,-1), int iterations=1,  
       int borderType=BORDER_CONSTANT, const Scalar  
&borderValue=morphologyDefaultBorderValue() )
```

imdilate膨胀原始图像

Historically, certain computer programs were written using only two digits rather than four to define the applicable year. Accordingly, the company's software may recognize a date using "00" as 1900 rather than the year 2000.

使用B后1次膨胀后的图像

Historically, certain computer programs were written using only two digits rather than four to define the applicable year. Accordingly, the company's software may recognize a date using "00" as 1900 rather than the year 2000.

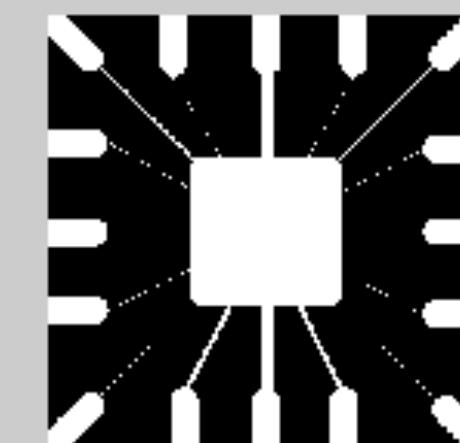
使用B后2次膨胀后的图像

Historically, certain computer programs were written using only two digits rather than four to define the applicable year. Accordingly, the company's software may recognize a date using "00" as 1900 rather than the year 2000.

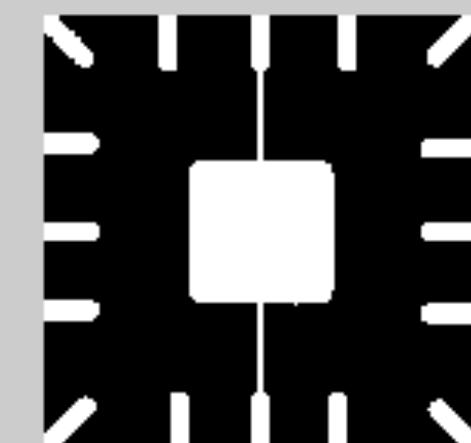
使用B后3次膨胀后的图像

Historically, certain computer programs were written using only two digits rather than four to define the applicable year. Accordingly, the company's software may recognize a date using "00" as 1900 rather than the year 2000.

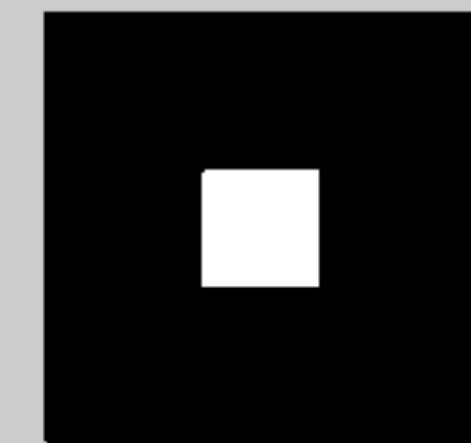
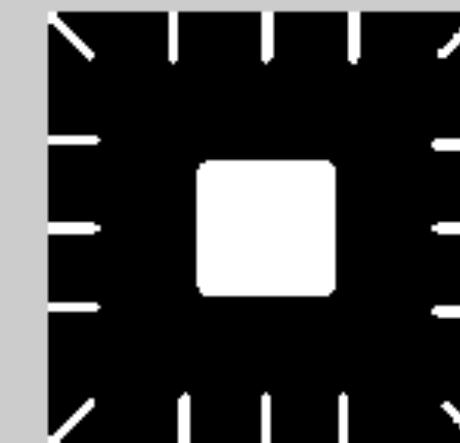
腐蚀原始图像



使用结构原始disk(5)腐蚀后的图像



使用结构原始disk(10)腐蚀后的图像 使用结构原始disk(20)腐蚀后的图像



轮廓检测

```
cv2.findContours(image, mode, method[, contours[, hierarchy[, offset ]]])
```

参数

第一个参数是寻找轮廓的图像；

第二个参数表示轮廓的检索模式，有四种（本文介绍的都是新的cv2接口）：

cv2.RETR_EXTERNAL表示只检测外轮廓

cv2.RETR_LIST检测的轮廓不建立等级关系

cv2.RETR_CCOMP建立两个等级的轮廓，上面的一层为外边界，里面的一层为内孔的边界信息。如果内孔内还有一个连通物体，这个物体的边界也在顶层。

cv2.RETR_TREE建立一个等级树结构的轮廓。

第三个参数method为轮廓的近似办法

cv2.CHAIN_APPROX_NONE存储所有的轮廓点，相邻的两个点的像素位置差不超过1，即 $\max(\text{abs}(x_1-x_2), \text{abs}(y_2-y_1)) == 1$

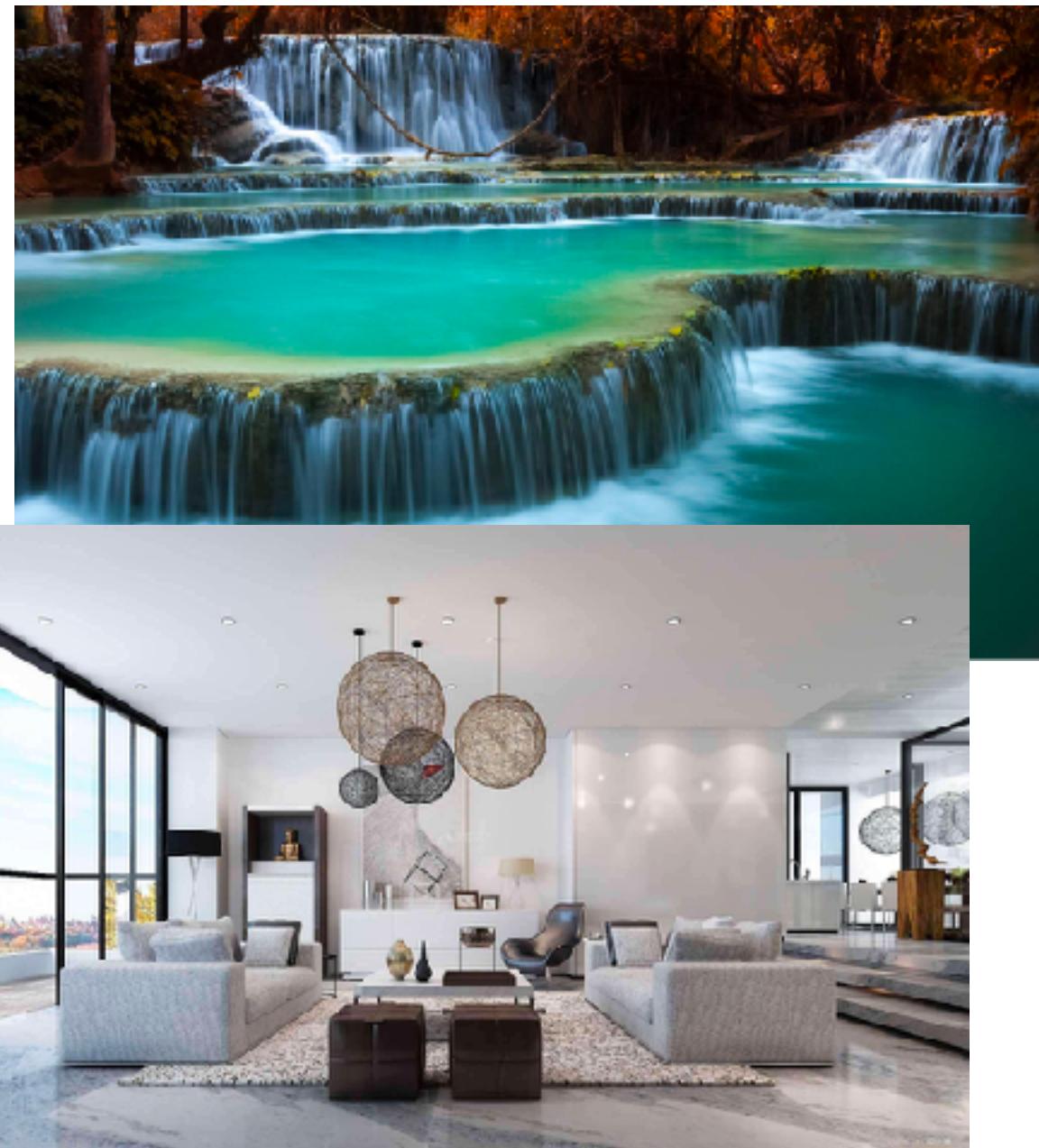
cv2.CHAIN_APPROX_SIMPLE压缩水平方向，垂直方向，对角线方向的元素，只保留该方向的终点坐标，例如一个矩形轮廓只需4个点来保存轮廓信息

cv2.CHAIN_APPROX_TC89_L1, CV_CHAIN_APPROX_TC89_KCOS使用teh-Chinl chain 近似[算法](#)

返回值

cv2.findContours()函数返回两个值，一个是轮廓本身，还有一个是每条轮廓对应的属性。

其他类型图像



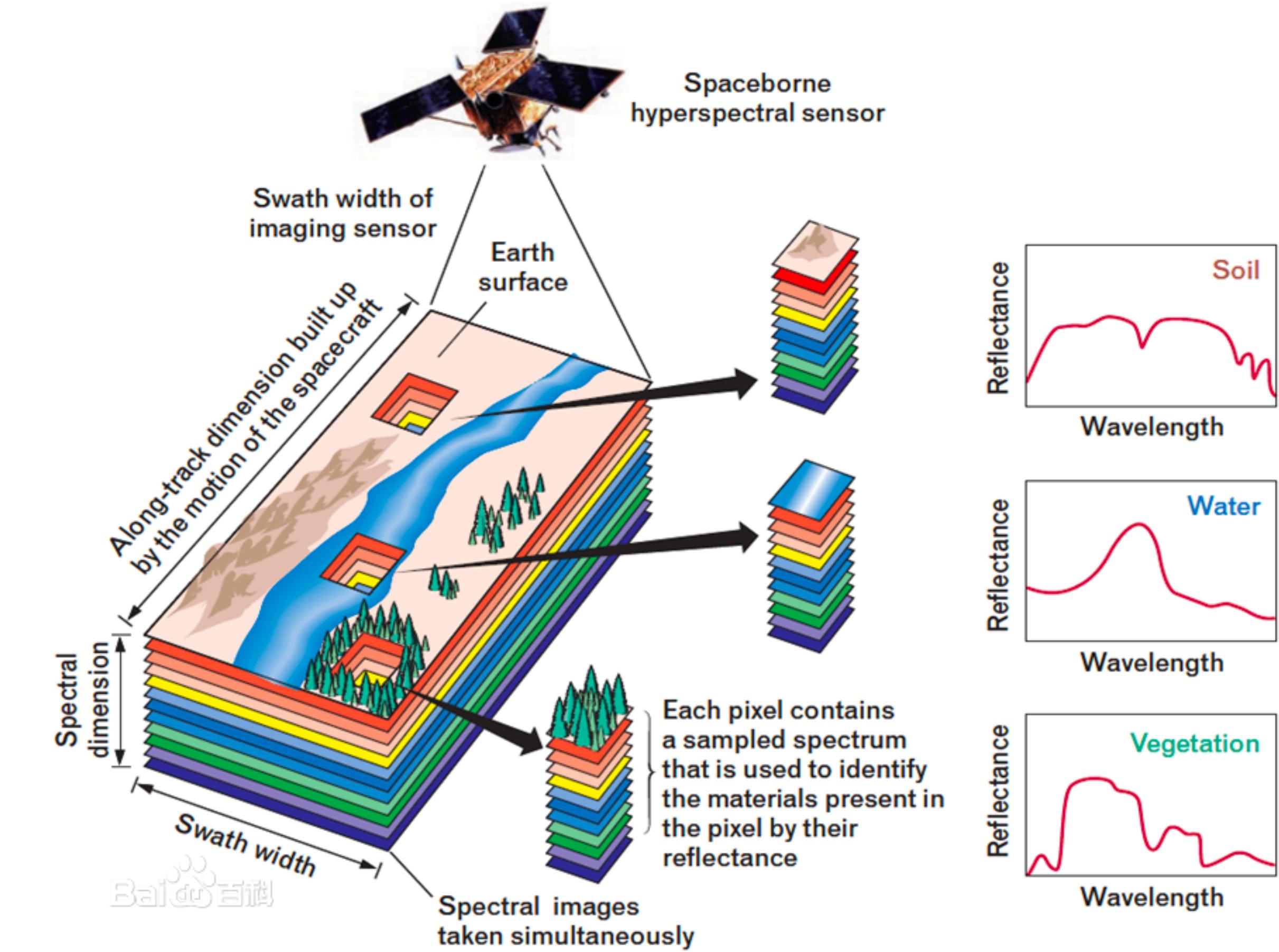
普通（自然）图像：OpenCV库等

遥感图像：GDAL库



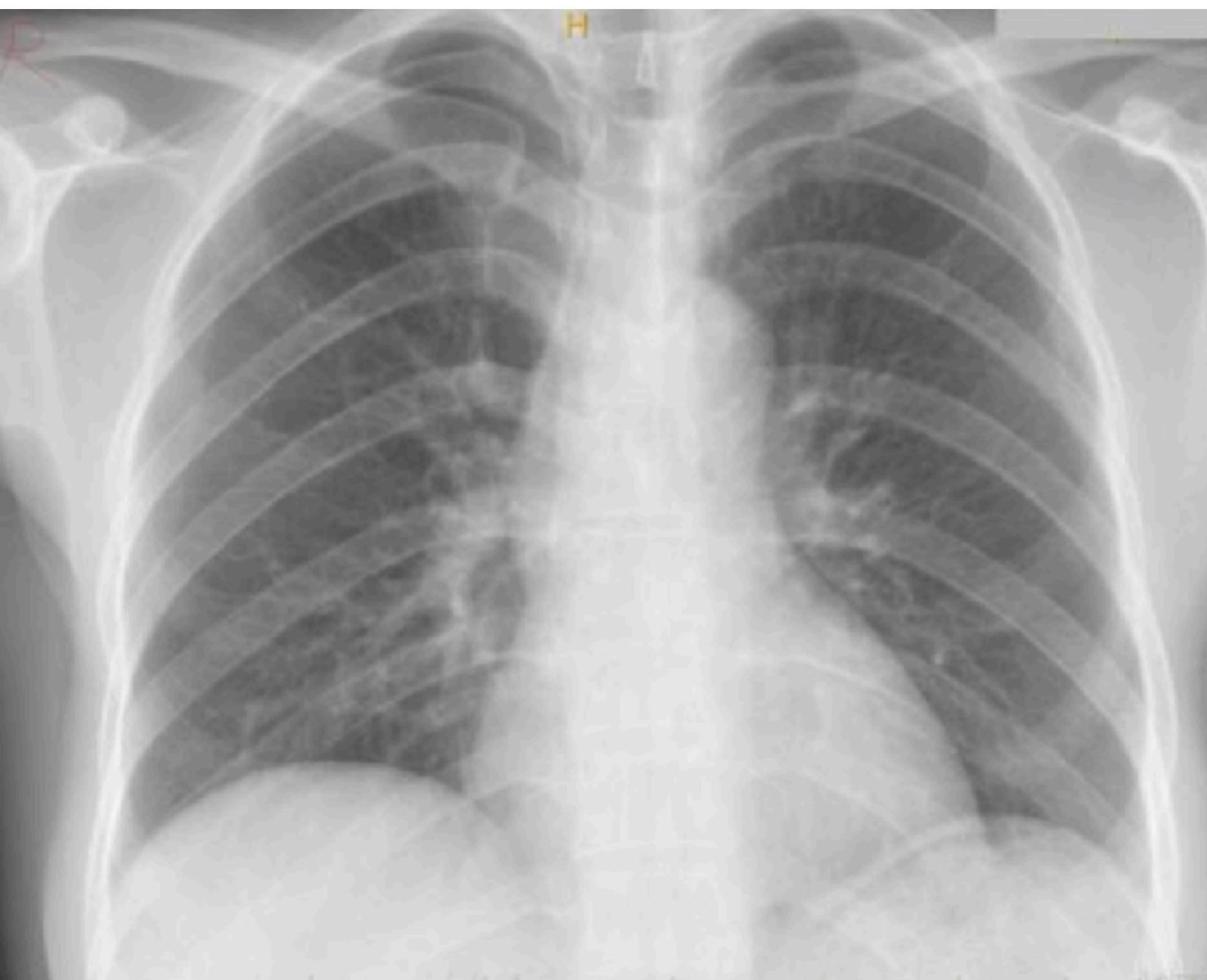
全色图和RGB图进行融合，分辨率重建

可见光图像、（近）红外图像
SAR图像、高光谱图像等



携带地理信息，需要进行矫正。
遥感图像数据是分级别的。其中，
零级：对应原始数据；
一级：经过了初步的辐射校正；
二级：经过了系统的几何校正。

其他类型图像



医学图像：Pydicom库，OpenCV等

呈层级数据，灰度级较高

彩色空间

也称彩色模型或彩色系统

●RGB

CCD (电荷耦合器件) 技术直接感知R,G,B三个分量，是图像成像、显示、打印等设备的基础

CCD成像原理<https://blog.csdn.net/waeceo/article/details/50580830>

●CMY(青、深红、黄)、CMYK(青、深红、黄、黑)

运用在大多数在纸上沉积彩色颜料的设备，如彩色打印机和复印机

- 1、打印中的主要颜色是黑色
- 2、等量的CMY原色产生黑色，但不纯
- 3、在CMY基础上，加入黑色，形成CMYK彩色空间

	黑	灰	白	红		橙	黄	绿	青	蓝	紫
hmin	0	0	0	0	156	11	26	35	78	100	125
hmax	180	180	180	10	180	25	34	77	99	124	155
smin	0	0	0	43		43	43	43	43	43	43
smax	255	43	30	255		255	255	255	255	255	255
vmin	0	46	221	46		46	46	46	46	46	46
vmax	46	220	255	255		255	255	255	255	255	255

●HSI(色调、饱和度、亮度) HSV

两个特点：

Ø I分量与图像的彩色信息无关

Ø H和S分量与人感受颜色的方式是紧密相连的

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

在OpenCV中的HSV模型

●YIQ 亮度(Brightness),即灰度值I和Q

指色调，描述色彩及饱和度

●YUV

●YCbCr

图像变换

❖ 图像的拉伸、缩小、Warp、旋转

- 一致调整变换或均匀调整变换
- 图像金字塔
- 非均匀映射
- 仿射变换
- 透视变换

❖ 一般映射

- 极坐标映射
- 对数极坐标
- 任意映射

1、平移变换

$$M = \begin{bmatrix} 1 & 0 & tx \\ 0 & 1 & ty \end{bmatrix}$$

$$\begin{pmatrix} m \\ n \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

`cv.warpAffine ()`

2、旋转变换

$$\begin{bmatrix} a(x, y) \\ b(x, y) \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

设旋转后新图像的左上角为原点，旋转前的中心坐标为 (a_0, b_0) ，旋转后的中心坐标为 (a_1, b_1) ，则旋转 α 角后的新坐标 (x_1, y_1) ，可由如下矩阵计算。

$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & a_1 \\ 0 & -1 & b_1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos \alpha & \sin \alpha & 0 \\ -\sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -a_0 \\ 0 & -1 & b_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \\ 1 \end{bmatrix}$$

为了找到这个转换矩阵，OpenCV提供了一个函数 `cv2.getRotationMatrix2D`.

`cv2.warpAffine`采用2x3变换矩阵，而`cv2.warpPerspective`采用3x3变换矩阵作为输入

图像变换

3、等距变换（欧式变换）

变换 (t) 和旋转变换 (R) 的复合

等距变换前后长度，面积，线线之间的角度都不变。

自由度 6 (3+3)

$$\begin{pmatrix} R & t \\ 0 & 1 \end{pmatrix}$$

4、相似变换

等距变换和均匀缩放 (S) 的一个复合

自由度 7

$$\begin{pmatrix} sR & t \\ 0 & 1 \end{pmatrix}$$

5、仿射变换（正交投影）

一个平移变换 (t) 和一个非均匀变换 (A) 的复合，
A是可逆矩阵，并不要求是正交矩阵。

仿射变换的不变量是：平行线，平行线的长度的比例，面积的比例

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

具体包括：平移 (Translation)、缩放 (Scale)、
旋转 (Rotation)、翻转 (Flip) 和错切 (Shear)

`cv.getAffineTransform` 得到变换矩阵，需要3组点

6、射影变换（透视变换）

当图像中的点的齐次坐标的一般非奇异线性变换

射影变换的不变量是：重合关系、长度的交比

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \begin{pmatrix} u \\ v \\ 1 \end{pmatrix}$$

`cv2.getPerspectiveTransform` 得到变换矩阵，需要4组点

常见问题解答

1、图像常见的传统特征描述有哪些？

LBP (**Local Binary Pattern**, 局部二值模式) 、

Gabor特征、

HOG (**Histogram of Oriented Gradient**, 方向梯度直方图) 、

SIFT (**Scale-invariant feature transform**, 尺度不变特征转换) 、

Hough (霍夫) 变换、

颜色直方图、

频谱特征提取、

圆周频率滤波、

视觉显著性特征、

矩特征 (几何矩、HU不变矩、Zernike矩和Legendre矩等) <https://blog.csdn.net/gdfsg/article/details/51015066>

FAST角点、

Haar-like特征 (积分图)

Harris角点检测等

角点检测：https://blog.csdn.net/qq_29828623/article/details/52403562

整理了个小文档，课后会发给大家

2、环境安装配置问题?

Anaconda

一个python环境中需要有一个解释器，和一个包集合。

Anaconda指的是一个开源的Python发行版本，其包含了conda、Python等180多个科学包及其依赖项。

解决了什么问题？

通过提供隔离环境，1、python版本互不干扰；2、各个依赖包之间的版本不受干扰

- **Anaconda Navigator**：用于管理工具包和环境的图形用户界面，后续涉及的众多管理命令也可以在 Navigator 中手工实现。
- **Jupyter notebook**：基于web的交互式计算环境，可以编辑易于人们阅读的文档，用于展示数据分析的过程。
- **qtconsole**：一个可执行 IPython 的仿终端图形界面程序，相比 Python Shell 界面，qtconsole 可以直接显示代码生成的图形，实现多行代码输入执行，以及内置许多有用的功能和函数。
- **spyder**：一个使用Python语言、跨平台的、科学运算集成开发环境。

见代码

3、Pycharm配置问题?

选择python环境?

演示

下载的jupyter文件无法运行?

编码方式的问题

现场提问kychen@kaikeba.com