第 11 课 图像处理——形态学处理

1.形态学概述

形态学是图像处理中应用最为广泛的技术之一，主要用于提取图像中对表达和描绘区域形状有意义的图像分量，使后续的识别工作能够抓住目标对象最为本质的形状特征，如边界和连通区域等。此外，细化、像素化和修剪毛刺等技术也常应用于图像的预处理和后处理中，成为图像增强技术的有力补充。

形态学的基本思想是利用一种特殊的结构元来测量或提取输入图像中相应的形状或特征，以便进一步进行图像分析和目标识别。

2.形态学转换

2.1 腐蚀和膨胀

腐蚀和膨胀是两种最基本、最重要的形态学运算， 它们是很多高级形态学处理的基础，有许多形态学算法是由这两种基本运算复合而成的。

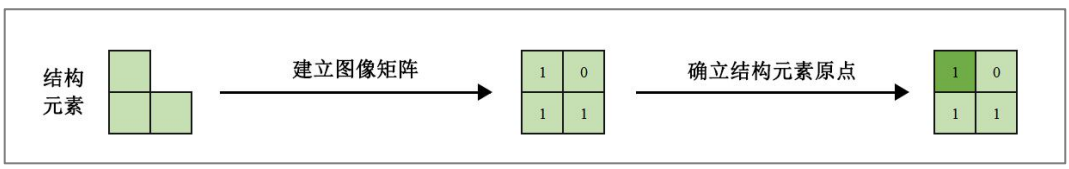
◆ **结构元素**

腐蚀与膨胀操作中都需要用到结构元素，可以将二维结构元素理解为一个二维矩阵矩阵元素是值为“**0**”或“**1**”。一般情况下，结构元素的尺寸小于待处理图像的尺寸。

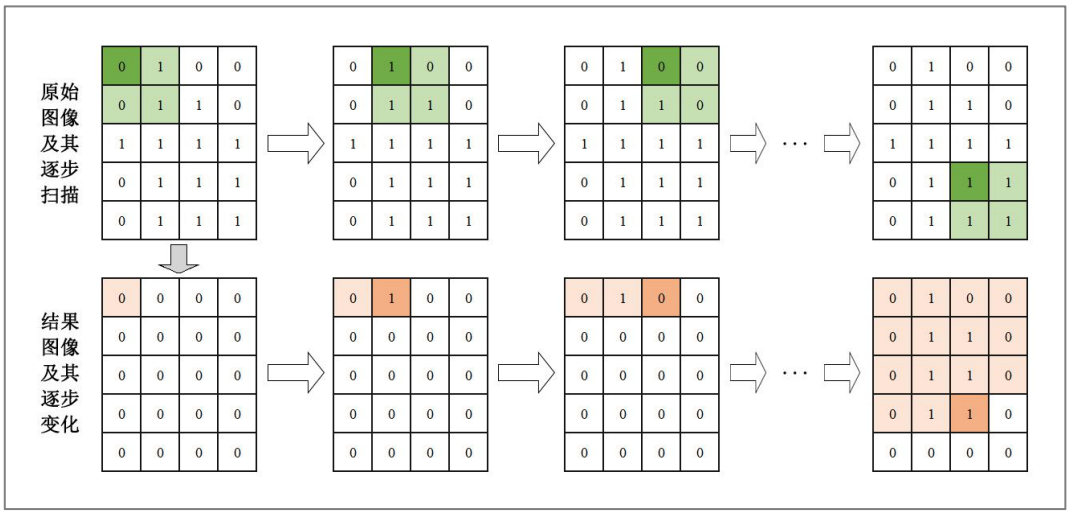
◆ **腐蚀（Erosion）**

腐蚀的作用是消除小且无意义的目标物，其具体实现步骤如下：

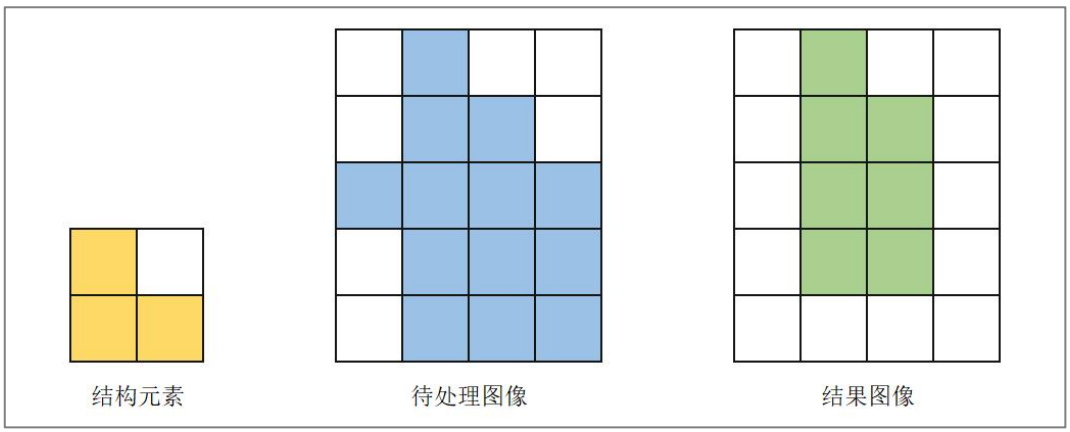
1） 基于结构元素建立一个图像矩阵，并确定其原点。此处以结构元素中左上角的元素为原点，并用较深的底色进行标记。



2） 将结构元素覆盖至待处理图像上。如果在待处理图像中，结构元素内所有值为“**1**”的元素对应的图像像素点的值也都为“**1**”，则在输出图像中将原点对应位置的像素点赋值为“**1**”，否则为“**0**”。



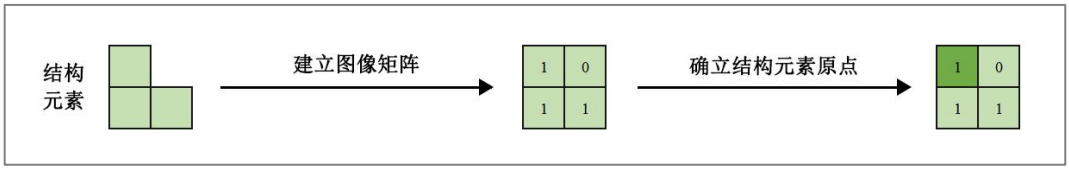
3） 令结构元素按顺序在待处理图像上进行移动，直至图像全部处理完成，得到的结果图像如下图所示：



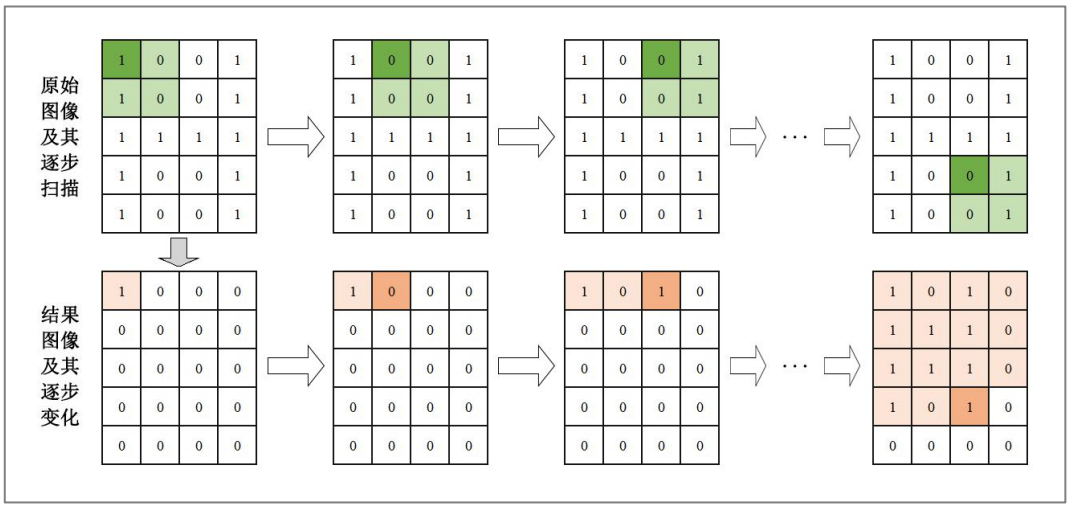
◆ **膨胀（Dilation）**

膨胀的作用是扩大图像边缘，填充目标物体边缘或内部的非目标像素点，其具体实现步骤如下：

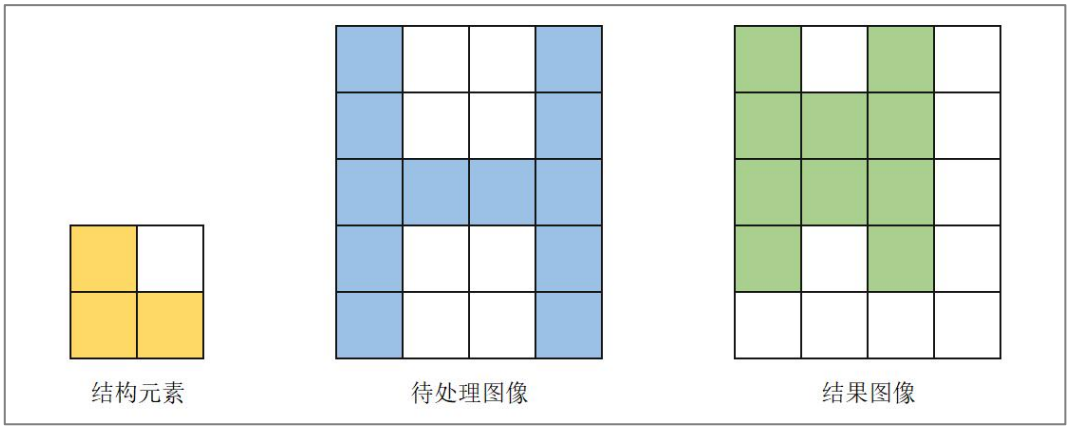
1） 基于结构元素建立一个图像矩阵，并确定其原点。此处以结构元素中左上角的元素为原点，并用较深的底色进行标记。



2） 将结构元素覆盖至待处理图像上。如果在待处理图像中，结构元素内所有值为“**1**”的元素对应的图像像素点的值至少有一个为“**1**”，则在输出图像中将原点对应位置的像素点赋值为“**1**”，否则为“**0**”。



3） 令结构元素按顺序在待处理图像上进行移动，直至图像全部处理完成，得到的结果图像如下图所示：



2.2 开运算和闭运算

开运算和闭运算就是按照一定顺序进行腐蚀和膨胀操作。

◆ **开运算**

开运算是先腐蚀后膨胀，其作用是分离物体、消除小区域、消除暗背景下的高亮区域。

◆ **闭运算**

闭运算是先膨胀后腐蚀，其作用是消除孔洞，即填充闭合区域、删除亮背景下的暗区域。

2.3 顶帽和底帽

◆ **顶帽运算（礼帽）**

顶帽运算也称作礼帽，是原始图像与开运算结果图之差（顶帽运算=原始图像-开运算结果图），可以获取原始图像中灰度较亮的区域。

◆ **底帽运算（黑帽）**

底帽运算也称作黑帽，是原始图像与闭运算结果图之差（底帽运算=原始图像-闭运算结果图），可以获取原始图像中灰度较暗的区域。

3.实验步骤

本节例程会**对指定图像分别进行腐蚀、膨胀、开运算、闭运算、顶帽运算和底帽运算处理**。

开始操作前，需要先将目录“**第 4 章 OpenCV 计算机视觉学习->图像处理进阶篇->第 11 课 图像处理——形态学处理->例程源码**”下的例程“**morphology_operations.py**”和示例图片“**example_org.jpg**”、“**example_noise.jpg**”、“**example_cave.jpg**”复制到共享文件夹。

关于共享文件夹的配置方法，可参考目录“**第 2 章 Linux 系统简介及使用入门->Linux 基础课程->第 3 课 Linux 系统安装及换源方法**”下的文档。

注意：输入指令时需严格区分大小写，且可以使用“Tab”键补齐关键字。

1） 启动虚拟机，点击系统任务栏的图标  ，并点击图标  ，或使用快捷键“**Ctrl+Alt+T**”，打开命令行终端。

2） 输入指令“**cd /mnt/hgfs/Share/**”，并按下回车，进入共享文件夹。

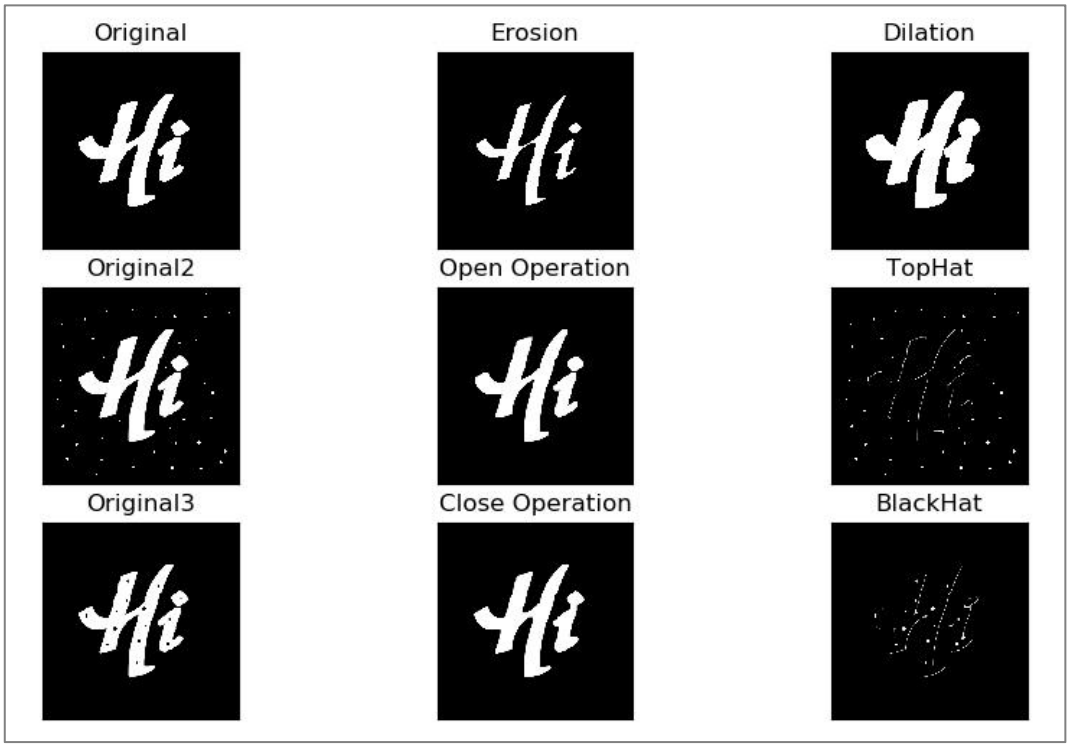
```
hiwonder@ubuntu:~$ cd /mnt/hgfs/Share/
```

3） 输入指令“**python3 morphology_operations.py**”，并按下回车，运行例程。

```
hiwonder@ubuntu:/mnt/hgfs/Share$ python3 morphology_operations.py
```

4.实验效果

执行程序后，图像处理结果如下图所示：



1） Original：原始图像；

2） Erosion：对原图进行腐蚀处理的结果图像；

3） Dilation：对原图进行膨胀处理的结果图像；

4） Original2：有噪点的原始图像；

5） Open Operation：对有噪点的原图进行开运算处理的结果图像；

6） TopHat：对有噪点的原图进行礼帽运算处理的结果图像；

7） Original3：有孔洞的原始图像；

8） Close Operation：对有孔洞的原图进行闭运算处理的结果图像；

9） BlackHat：对有孔洞的原图进行黑帽运算处理的结果图像。

5.程序分析

可以在目录“**第 4 章 OpenCV 计算机视觉学习->图像处理进阶篇->第 11 课 图像处理——形态学处理->例程源码**”下查看例程“**morphology_operations.py**”。

```python
import cv2
import numpy as np
import matplotlib.pyplot as plt

# 图像读取
img_org = cv2.imread('example_org.jpg')
img_noise = cv2.imread('example_noise.jpg')
img_cave = cv2.imread('example_cave.jpg')

# 创建核结构
kernel = np.ones((10, 10), np.uint8)  # 10*10的全1矩阵

# 形态学处理
erosion_img = cv2.erode(img_org, kernel)  # 腐蚀
dilate_img = cv2.dilate(img_org, kernel)  # 膨胀
open_img = cv2.morphologyEx(img_noise, cv2.MORPH_OPEN, kernel)  # 开运算
close_img = cv2.morphologyEx(img_cave, cv2.MORPH_CLOSE, kernel)  # 闭运算
top_hat_img = cv2.morphologyEx(img_noise, cv2.MORPH_TOPHAT, kernel)  # 礼帽运算
black_hat_img = cv2.morphologyEx(img_cave, cv2.MORPH_BLACKHAT, kernel)  # 黑帽运算

# 图像显示
plt.figure(figsize=(10, 6), dpi=100)
plt.rcParams['axes.unicode_minus'] = False

plt.subplot(331), plt.imshow(img_org), plt.title("Original")
plt.xticks([]), plt.yticks([])
plt.subplot(332), plt.imshow(erosion_img), plt.title("Erosion")
plt.xticks([]), plt.yticks([])
plt.subplot(333), plt.imshow(dilate_img), plt.title("Dilation")
plt.xticks([]), plt.yticks([])

plt.subplot(334), plt.imshow(img_noise), plt.title("Original2")
plt.xticks([]), plt.yticks([])
plt.subplot(335), plt.imshow(open_img), plt.title("Open Operation")
plt.xticks([]), plt.yticks([])
plt.subplot(336), plt.imshow(top_hat_img), plt.title("TopHat")
plt.xticks([]), plt.yticks([])

plt.subplot(337), plt.imshow(img_cave), plt.title("Original3")
plt.xticks([]), plt.yticks([])
plt.subplot(338), plt.imshow(close_img), plt.title("Close Operation")
plt.xticks([]), plt.yticks([])
plt.subplot(339), plt.imshow(black_hat_img), plt.title("BlackHat")
plt.xticks([]), plt.yticks([])

plt.show()
```

5.1 图像处理

- **导入模块**

首先，通过 import 语句导入所需模块。

```python
import cv2
import numpy as np
import matplotlib.pyplot as plt
```

◆ **读取图像**

通过调用 cv2 模块中的 imread()函数，读取需要处理的图像。

```
6  img_org = cv2.imread('example_org.jpg')
7  img_noise = cv2.imread('example_noise.jpg')
8  img_cave = cv2.imread('example_cave.jpg')
```

函数括号内的参数是图像名称。

◆ **创建核结构**

通过调用 numpy 模块中的 ones()函数，创建后续运算所需的核结构，即数组。

```
11  kernel = np.ones((10, 10), np.uint8)  # 10*10的全1矩阵
```

ones()函数的语法格式如下：

```
np.ones(shape, dtype=None, order='C')
```

第一个参数“**shape**”是一个整数类型或整数元组，用于定义数组大小。若指定整数类型变量，则返回一维数组。若指定整数元组，则返回给定形状的数组；

第二个参数“**dtype**”是数组的数据类型，默认值是“**float**”；

第三个参数“**order**”用于指定返回数组元素在内存的存储顺序。

◆ **腐蚀和膨胀**

通过调用 cv2 模块中的 erode()和 dilate()函数，对指定图像分别进行腐蚀和膨胀处理。

```
14  erosion_img = cv2.erode(img_org, kernel)  # 腐蚀
15  dilate_img = cv2.dilate(img_org, kernel)  # 膨胀
```

erode()函数的语法格式如下：

```
cv2.erode(src, kernel, iteration)
```

第一个参数“**src**”是输入图像；

第二个参数“**kernel**”是核大小；

第三个参数“**iteration**”是迭代次数。

dilate()函数的参数含义与 erode()函数的相同。

◆ **开运算和闭运算**

通过调用 cv2 模块中的 morphologyEx()函数，对指定图像分别进行开运算、闭运算、礼帽运算和黑帽运算处理。

```
16 open_img = cv2.morphologyEx(img_noise, cv2.MORPH_OPEN, kernel) # 开运算
17 close_img = cv2.morphologyEx(img_cave, cv2.MORPH_CLOSE, kernel) # 闭运算
18 top_hat_img = cv2.morphologyEx(img_noise, cv2.MORPH_TOPHAT, kernel) # 礼帽运算
19 black_hat_img = cv2.morphologyEx(img_cave, cv2.MORPH_BLACKHAT, kernel) # 黑帽运算
```

morphologyEx()函数的语法格式如下：

```
cv2.morphologyEx(img, op, kernel)
```

第一个参数“**img**”是输入图像；

第二个参数“**op**”是操作类型，下表是其取值范围：

取值范围	含义
cv2.MORPH_OPEN	开运算
cv2.MORPH_CLOSE	闭运算
cv2.MORPH_GRADIENT	形态学梯度
cv2.MORPH_TOPHAT	礼帽运算（顶帽运算）
cv2.MORPH_BLACKHAT	黑帽运算（底帽运算）

第三个参数“**kernel**”是方框大小。

5.2 图像显示

◆ **创建自定义图像**

通过调用 matplotlib.pyplot 模块中的 figure()函数，创建一个自定义画像（Figure），用

于显示滤波处理的结果图像。

```
22  plt.figure(figsize=(10, 6), dpi=100)
```

figure()函数的语法格式如下：

matplotlib.pyplot.figure(num=None, figsize=None, dpi=None, facecolor=None, edgecolor=None, frameon=True, FigureClass=<class 'matplotlib.figure.Figure'>, clear=False, **kwargs)

第一个参数“**num**”是画像的唯一标识符，即画像编号（数字）或名称（字符串）；

第二个参数“**figsize**”是画像的宽和高，单位为英寸；

第三个参数“**dpi**”是画像的分辨率，即每英寸的像素数；

第四个参数“**facecolor**”是画像的背景颜色；

第五个参数“**edgecolor**”是画像的边框颜色；

第六个参数“**frameon**”用于决定是否绘制画像，默认值为“**True**”；

第七个参数“**FigureClass**”是生成画像时选用的自定义 Figure；

第八个参数“**clear**”用于决定当画像存在时是否清除原有画像；

第九个参数“****kwargs**”是画像的其他属性。

◆ **修改 matplotlib 配置**

matplotlib 是 Python 的绘图库，用户可以通过参数字典“**rcParams**”访问与修改 matplotlib 的配置项。

```
23  plt.rcParams['axes.unicode_minus'] = False
```

上图所示代码用于控制正常字符显示。

◆ **设置图像显示参数**

通过调用 matplotlib.pyplot 模块中的 subplot()、imshow()和 title()函数，指定画像（Figure）

内的子图位置分布、子图颜色类型和子图标题。

```
25  plt.subplot(331), plt.imshow(img_org), plt.title("Original")
```

1） subplot()函数用于设置子图的位置分布，其语法格式如下：

```
matplotlib.pyplot.subplot(nrows, ncols, index, **kwargs)
```

第一个参数“**nrows**”和第二个参数“**ncols**”分别是 subplot 的行、列数；

第三个参数“**index**”是索引位置，从“**1**”开始累计（左上角为“**1**”），依次向右递增。

当行、列数都小于“**10**”，可以将两个数值缩写为一个整数，例如，代码“**subplot(3, 3, 1)**”和“**subplot(331)**”的含义相同，都表示将画像分为 3 行 3 列，当前子图排在第 1 位，即第 1 行的第 1 列。

2） imshow()函数用于设置子图颜色类型，其语法格式如下：

```
matplotlib.pyplot.imshow(X, cmap=None)
```

第一个参数“**X**”是图像数据；

第二个参数“**cmap**”是颜色图谱（colormap），默认为 RGB（A）颜色空间。

3） title()函数用于设置子图标题，函数括号内的参数是子图名称，其语法格式如下：

```
matplotlib.pyplot.title(label, fontdict=None, loc=None, pad=None, *, y=None, **kwargs)
```

第一个参数“**label**”是标题文本，类型为字符串；

第二个参数“**fontdict**”是标题文本的字体属性，类型为字典；

第三个参数“**loc**”是标题位置，可取值“**left**”、“**center**”或“**right**”，默认值为“**center**”；

第四个参数“**pad**”是标题与子图的填充距离（内边距），默认值为“**6.0**”；

第五个参数“**y**”是标题在子图中的垂直距离，单位为子图高度的百分比，默认值为“**None**”，即自动确定标题位置，避免与其他元素重叠。值为“**1.0**”表示在子图顶端；

第六个参数“**kwargs”是文本对象关键字属性，用于控制文本的外观属性，如字体、文本颜色等。

◆ **设置坐标轴刻度**

通过调用 matplotlib.pyplot 模块中的 xticks()和 yticks()函数，设置 X、Y 轴的刻度及标签，由于此例程在显示图像时无需使用坐标轴信息，此处将列表设为空，即不显示坐标轴。

```
26  plt.xticks([]), plt.yticks([])
```

xticks()函数的语法格式如下：

```
matplotlib.pyplot.xticks(ticks=None, labels=None, **kwargs)
```

当参数为空，函数会返回当前 X 轴的刻度及标签；否则，函数用于设置当前 X 轴的刻度及标签。

第一个参数“**ticks**”是 X 轴刻度的位置列表，若列表为空，X 轴刻度将清空；

第二个参数“**labels**”是 X 轴刻度的标签，当参数“**ticks**”不为空，此参数才会传递；

第三个参数“**kwargs”用于控制刻度标签的外观。

yticks()函数的语法格式和 xticks()函数的相同，区别在于，yticks()函数的作用对象是 Y 轴。

◆ **显示图像**

通过调用 matplotlib.pyplot 模块中的 show()函数，在窗口显示图像。

```
46  plt.show()
```

图像显示部分的完整代码如下：

```
plt.subplot(331), plt.imshow(img_org), plt.title("Original")
plt.xticks([]), plt.yticks([])
plt.subplot(332), plt.imshow(erosion_img), plt.title("Erosion")
plt.xticks([]), plt.yticks([])
plt.subplot(333), plt.imshow(dilate_img), plt.title("Dilation")
plt.xticks([]), plt.yticks([])

plt.subplot(334), plt.imshow(img_noise), plt.title("Original2")
plt.xticks([]), plt.yticks([])
plt.subplot(335), plt.imshow(open_img), plt.title("Open Operation")
plt.xticks([]), plt.yticks([])
plt.subplot(336), plt.imshow(top_hat_img), plt.title("TopHat")
plt.xticks([]), plt.yticks([])

plt.subplot(337), plt.imshow(img_cave), plt.title("Original3")
plt.xticks([]), plt.yticks([])
plt.subplot(338), plt.imshow(close_img), plt.title("Close Operation")
plt.xticks([]), plt.yticks([])
plt.subplot(339), plt.imshow(black_hat_img), plt.title("BlackHat")
plt.xticks([]), plt.yticks([])

plt.show()
```