

Lesson 11 Image Processing --- Morphological Processing

1.Morphology Introduction

Morphology is one of the most widely used techniques in image processing. It is mainly used to extract image components that are meaningful for describing the shape of an area, so that the most essential shape features of the target object can be captured in subsequent recognition, such as boundaries and connected areas. In addition, techniques such as thinning, pixelation, and burr trimming are often used in image preprocessing and postprocessing, which can greatly strengthen the image.

The basic idea of morphology is to use a special structural element to measure or extract the corresponding shape or feature in the input image for further image analysis and target recognition.

2.Morphological Transformation

2.1 Erosion and Dilation

Both erosion and dilation are the basic and important morphological operations, and also the foundations of multiple advanced morphological processing. Many morphological algorithms are composed of these two.

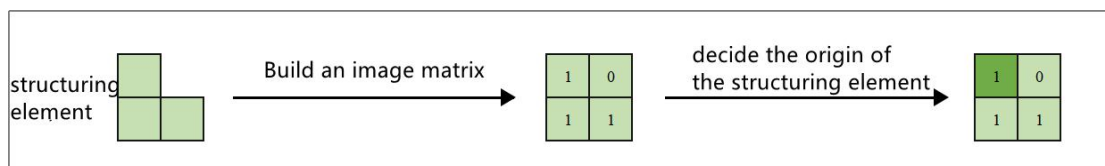
◆ Structuring Element

Structuring element is required in erosion and dilation. A two-dimensional structuring element can be seen as a two-dimensional matrix element which is “0” or “1”.

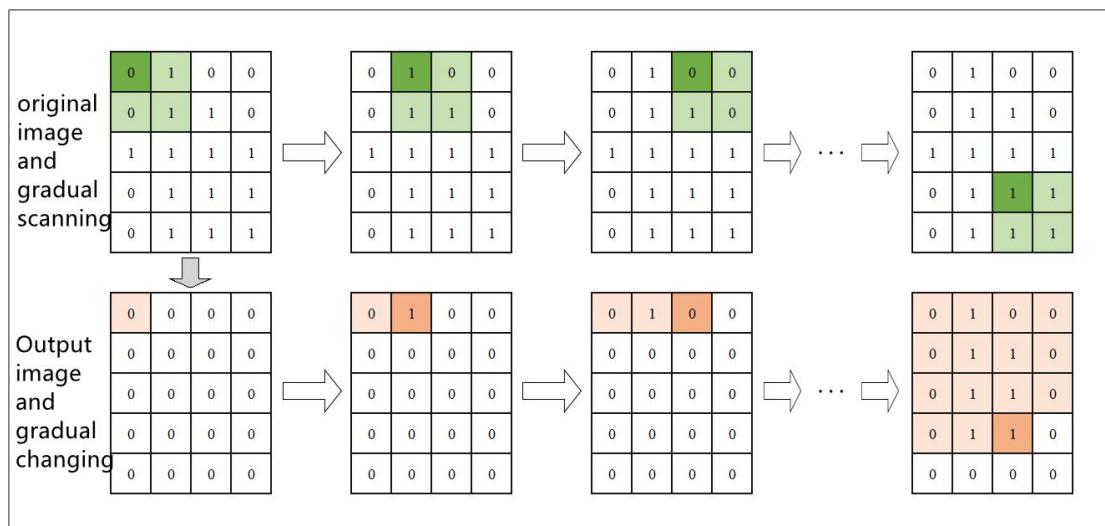
◆ Erosion

Erosion works to remove small and meaningless object, and the whole process is divided into three steps.

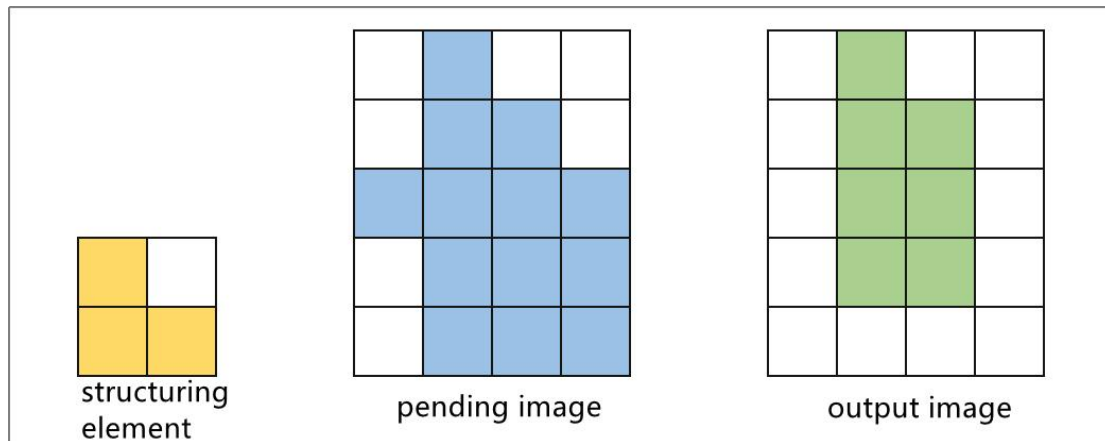
1) Build an image matrix upon the structuring element and determine its origin. Take the element at the upper left corner as the origin, and mark it with dark color.



2) Overlay the structuring element on the pending image. If the value of the pixel in the image corresponding to the elements whose value is "1" in the structuring elements are all "1", the pixel at the corresponding position of the origin is assigned as "1", otherwise it is "0".



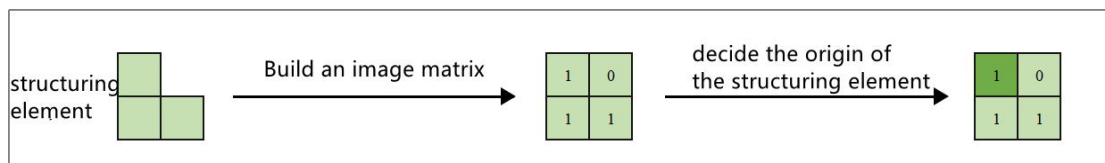
3) Make the structuring elements move on the pending image in order until all the images are processed completely.



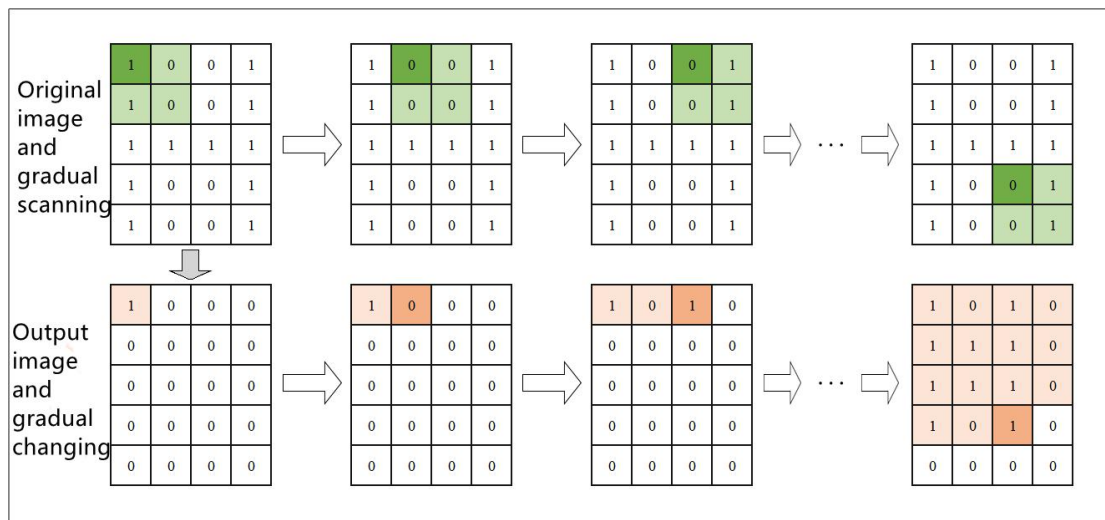
◆ Dilation

Dilation can enlarge the edge of the image and pad the edge of the target object or non-target pixel. The operations are divided into three steps.

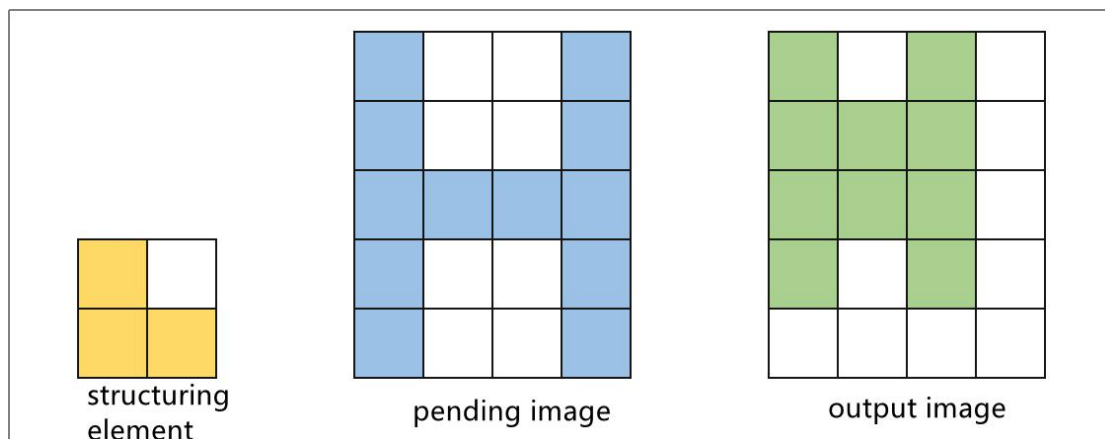
1) Build an image matrix upon the structuring element and determine its origin. Take the element at the upper left corner as the origin, and mark it with dark color.



2) Overlay the structuring element on the pending image. If at least one of the value of the pixel in the image corresponding to the elements whose value is "1" in the structuring elements is "1", the pixel at the corresponding position of the origin is assigned as "1", otherwise it is "0".



3) Make the structuring elements move on the pending image in order until all the images are processed completely.



2.2 Opening and Closing

In opening and closing, erosion and dilation are executed in sequence.

◆ Open Operation

Opening indicates that erosion is executed first and dilation follows. It is useful in separating objects, removing small area, removing highlight under dark background.

◆ Close Operation

In closing, dilation is executed first and erosion follows. It plays an

important role in eliminating holes, that is, filling closed areas and deleting dark areas under a bright background.

2.3 Top Hat and Bottom Hat

◆ Top Hat Operation

It is the difference between input image and the image after opening (Top hat operation= input image - image after opening), and it can obtain areas with brighter gray in the original image

◆ Bottom Hat Operation (Black Hat)

It is the difference between input image and the image after closing (Top hat operation= input image - image after closing), and it can obtain areas with darker gray in the original image


3.Operation Steps

This routine will perform erosion, dilation, opening, closing, top hat operation and bottom hat operation on the designated image.

Before operation, please copy the routine “**example_org.jpg**” in “4.OpenCV->Lesson 11 Image Processing --- Geometric Transformation->Routine Code” to the shared folder.

For how to configure the shared folder, please refer to the file in “**2. Linux Basic Lesson->Lesson 3 Linux Installation and Source Replacement**”.

Note: the input command should be case sensitive and the keywords can be complemented by “Tab” key.

- 1) Open virtual machine and start the system. Click “

5

or press “**Ctrl+Alt+T**” to open command line terminal.

2) Input command “**cd /mnt/hgfs/Share/**” and press Enter to enter the shared folder.

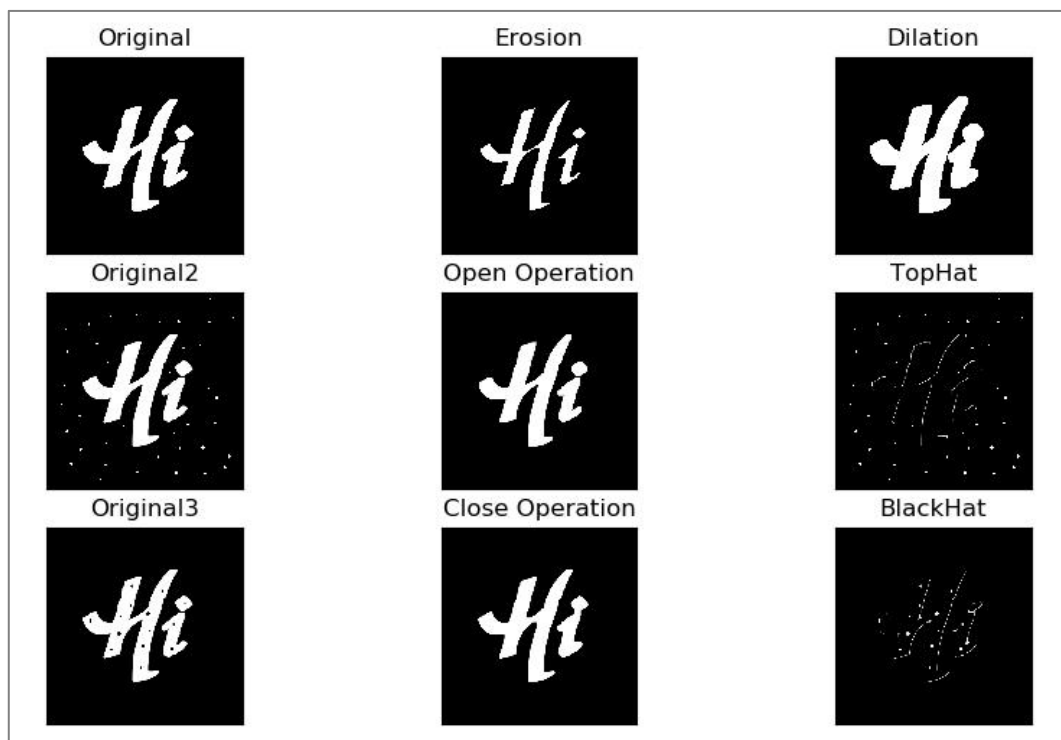
```
hiwonder@ubuntu:~$ cd /mnt/hgfs/Share/
```

3) Input command “**python3 morphology_operations.py**” and press Enter to run the routine.

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

4.Program Outcome

The final output image is as follow.



5.Program Analysis

The routine “**morphology_operations.py**” can be found in “4. OpenCV Computer Vision Lesson->Lesson 11 Image Processing---Morphological Processing->Routine Code”.

```

1  import cv2
2  import numpy as np
3  import matplotlib.pyplot as plt
4
5  # read the image
6  img_org = cv2.imread('example_org.jpg')
7  img_noise = cv2.imread('example_noise.jpg')
8  img_cave = cv2.imread('example_cave.jpg')
9
10 # build nuclear structure
11 kernel = np.ones((10, 10), np.uint8) # 10*10 all-one matrix
12
13 # Morphological processing
14 erosion_img = cv2.erode(img_org, kernel) # erosion
15 dilate_img = cv2.dilate(img_org, kernel) # dilation
16 open_img = cv2.morphologyEx(img_noise, cv2.MORPH_OPEN, kernel) # open operation
17 close_img = cv2.morphologyEx(img_cave, cv2.MORPH_CLOSE, kernel) # close operation
18 top_hat_img = cv2.morphologyEx(img_noise, cv2.MORPH_TOPHAT, kernel) # top hat operation
19 black_hat_img = cv2.morphologyEx(img_cave, cv2.MORPH_BLACKHAT, kernel) # bottom hat operation
20
21 # image display
22 plt.figure(figsize=(10, 6), dpi=100)
23 plt.rcParams['axes.unicode_minus'] = False
24
25 plt.subplot(331), plt.imshow(img_org), plt.title("Original")
26 plt.xticks([], plt.yticks([]))
27 plt.subplot(332), plt.imshow(erosion_img), plt.title("Erosion")
28 plt.xticks([], plt.yticks([]))
29 plt.subplot(333), plt.imshow(dilate_img), plt.title("Dilation")
30 plt.xticks([], plt.yticks([]))
31
32 plt.subplot(334), plt.imshow(img_noise), plt.title("Original2")
33 plt.xticks([], plt.yticks([]))
34 plt.subplot(335), plt.imshow(open_img), plt.title("Open Operation")
35 plt.xticks([], plt.yticks([]))
36 plt.subplot(336), plt.imshow(top_hat_img), plt.title("TopHat")
37 plt.xticks([], plt.yticks([]))
38
39 plt.subplot(337), plt.imshow(img_cave), plt.title("Original3")
40 plt.xticks([], plt.yticks([]))
41 plt.subplot(338), plt.imshow(close_img), plt.title("Close Operation")
42 plt.xticks([], plt.yticks([]))
43 plt.subplot(339), plt.imshow(black_hat_img), plt.title("BlackHat")
44 plt.xticks([], plt.yticks([]))
45
46 plt.show()

```

5.1 Image Processing

◆ Import Module

Firstly, import the required module through import statement.

```

1  import cv2
2  import numpy as np
3  import matplotlib.pyplot as plt

```

◆ Read the Image

Then call **imread()** function in cv2 module to read the pending image.

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

The parameter in the bracket is the name of the image

◆ Create Nuclear Structure

Call **ones()** function in numpy module to create the nuclear structure i.e. array required in the operation.

```
11 kernel = np.ones((10, 10), np.uint8) # 10*10 all-one matrix
```

The format of **ones()** function is as follow.

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

The first parameter “**shape**” is a integer or integer tuple used to define the size of the array. If it designates the variable of the integer, one--dimensional array will be returned. If it designate integer tuple, the array in given shape will be returned.

The second parameter “**dtype**” refers to the data type of the array, “**float**” by default.

The third parameter “**order**” is used to designate the storing order of the returned array elements in storage.

◆ Erosion and Dilation

Call **erode()** and **dilate()** function in cv2 module to perform erosion and dilation on the specific image.


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

The format of erode() function is as follow.

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

The first parameter “**src**” is the input image.

The second parameter “**kernel**” is the size of the kernel.

The third parameter is the umber of iteration.

The meaning of the parameter in dilate() function is the same as that of erode() function.

◆ Open Operation and Close Operation

Call morphologyEx() function in cv2 module to perform open operation, close operation, top hat operation and bottom hat operation on the specific image.

```
16 open_img = cv2.morphologyEx(img_noise, cv2.MORPH_OPEN, kernel) # open operation
17 close_img = cv2.morphologyEx(img_cave, cv2.MORPH_CLOSE, kernel) # close operation
18 top_hat_img = cv2.morphologyEx(img_noise, cv2.MORPH_TOPHAT, kernel) # top hat operation
19 black_hat_img = cv2.morphologyEx(img_cave, cv2.MORPH_BLACKHAT, kernel) # bottom hat operation
```

The format of morphologyEx() function is as follow.

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

The first parameter “**img**” indicates the input image

The second parameter “**op**”represents the operation type.

Operation Type	Meaning
cv2.MORPH_OPEN	open operation
cv2.MORPH_CLOSE	close operation

cv2.MORPH_GRADIENT	morphological gradient
cv2.MORPH_TOPHAT	top hat operation
cv2.MORPH_BLACKHAT	bottom hat operation

The third parameter indicates the size of the frame.

5.2 Image Display

◆ Create Custom Figure

Call figure() function in matplotlib.pyplot module to create a custom figure for displaying the final output image.

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

The format of the figure() function is as follow.

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

The first parameter “**num**” is the only identifier of the image i.e. the serial number of the picture (number) or the name (string).

The second parameter “**figsize**” is the width and height of the image in inch.

The third parameter “**dpi**” is the resolution of the image i.e. the number of pixels by inch

The fourth parameter “**facecolor**” is the background color.

The fifth parameter “**edgecolor**” is the frame color

The sixth parameter “**frameon**” determines whether to draw the picture, and it is “**True**” by default.

The seventh parameter “**FigureClass**” is used to select the custom figure when generating the image

The eighth parameter “**clear**” determines whether to clear all the original images.

The ninth parameter “****kwargs**” represents other properties of the image.

◆ Modify matplotlib Configuration

matplotlib is plotting library of Python. User can access and modify matplotlib configuration options through parameter dictionary “**rcParams**”.

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

The codes above are used to manipulate the display of the normal characters.

◆ Set the Parameter of Image Display

Call subplot(), imshow() and title() functions in matplotlib.pyplot modules to designate the position, color and headline of the subplot in the Figure.

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

1) subplot() function is used to set the position of the subplot, and the function format is as follow.

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

The first parameter “**nrows**” and the second parameter “**ncols**” respectively are the number of row and column of subplot.

The third parameter “**index**” is the index position. Index starts at 1 in the upper left corner and increases to the right.

When both the row and column are less than “10”, these two values can be abbreviated to an integer. For example, the meaning of “**subplot(3, 3, 1)**” and “**subplot(331)**” are the same, both representing the image is divided into three rows and three columns, and the subplot is in the first place i.e. 1st row, 1st column.

2) `imshow()` function is used to set the color of subplot, and its format is as follow.

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

The first parameter “**X**” is the image data.

The second parameter “**cmap**” is the colormap, RGB(A) color space by default.

3) `title()` function is used to set the title of the subplot. The parameter in the bracket is the name of the subplot and the function format is as follow.

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

The first parameter “label” is the title composed of string.

The second parameter “fontdict” is the property of the font, and the current parameter refers to dictionary.

The third parameter “loc” is the position of the title. It can be “left”, “center” or “right”, and “center” by default.

The fourth parameter “pad” is the padding distance (inside margin) between the tile and the subplot, “6.0” by default.

The fifth parameter “y” is the vertical distance between the title and the subplot, and the unit is the percentage of the height of the subplot. The default value is “None”, that is, the position of the title is automatically determined to

avoid overlapping with other elements. "1.0" means the title is at the top of the subplot.

The sixth parameter **"**kwargs"** is the text object keyword property, which is used to determine the appearance of the text, such as font, text color, etc.

◆ Set Axis Tick

Call `xticks()` and `yticks()` function in `matplotlib.pyplot` module to set the tick and tag of X and Y axis. As the coordinate axis is not required in image display in this routine, the list is set as `none` that is the coordinate axis will not be displayed.

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

The format of `xticks()` function is as follow.

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

The first parameter **"ticks"** is a list of the positions of the X-axis ticks. If the list is empty, the X-axis ticks will be cleared.

The second parameter **"labels"** is the label of X-axis tick. Only when parameter **"ticks"** is not `none`, can this parameter be passed.

The third parameter **"**kwargs"** is used to control the appearance of the tick and label.

The format of `yticks()` is the same as that of `xticks()`. The difference lies in the controlled object.

◆ Display Image

Call `show()` function in `matplotlib.pyplot` module to display the image on the window.

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

The complete codes of image display part are as follow.

```
25 plt.subplot(331), plt.imshow(img_org), plt.title("Original")
26 plt.xticks([], plt.yticks([])
27 plt.subplot(332), plt.imshow(erosion_img), plt.title("Erosion")
28 plt.xticks([], plt.yticks([])
29 plt.subplot(333), plt.imshow(dilate_img), plt.title("Dilation")
30 plt.xticks([], plt.yticks([])
31
32 plt.subplot(334), plt.imshow(img_noise), plt.title("Original2")
33 plt.xticks([], plt.yticks([])
34 plt.subplot(335), plt.imshow(open_img), plt.title("Open Operation")
35 plt.xticks([], plt.yticks([])
36 plt.subplot(336), plt.imshow(top_hat_img), plt.title("TopHat")
37 plt.xticks([], plt.yticks([])
38
39 plt.subplot(337), plt.imshow(img_cave), plt.title("Original3")
40 plt.xticks([], plt.yticks([])
41 plt.subplot(338), plt.imshow(close_img), plt.title("Close Operation")
42 plt.xticks([], plt.yticks([])
43 plt.subplot(339), plt.imshow(black_hat_img), plt.title("BlackHat")
44 plt.xticks([], plt.yticks([])
45
46 plt.show()
```