medium.com

Morphological Transformations of Images using OpenCV | Image Processing Part-2

Ravjot Singh

4-5 minutos

Let’s start using OpenCV Library-

1. Importing the required libraries:

> *cv2*- For OpenCV (Used for Image Processing)
>
> *matplotlib*- For Matplotlib (Used fot Plotting and Visualization)
>
> *numpy*- For Numpy (Used for Scientific Computing)
>
> pandas- *For Pandas (Used for Data Analysis)*

Press enter or click to view image in full size

```
[15] import cv2
     import numpy as np
     import pandas as pd
     from matplotlib import pyplot as plt
```

Libraries required

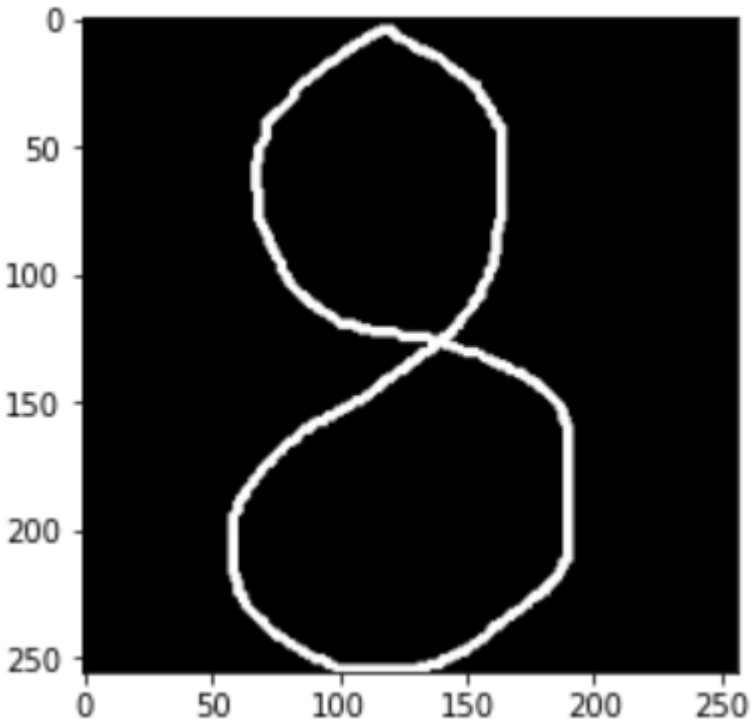
2. Reading an image using OpenCV library:

We can read an image by using OpenCV and Matplotlib library.

Press enter or click to view image in full size

```
img = cv2.imread('8.jpg')

plt.imshow(img)
```



Opening an image

3. Morphological Transformations on Image:

> ***Theory*** *-*
>
> *Morphological transformations are some simple operations based on the image shape. It is normally performed on binary images. It needs two inputs, one is our original image, second one is called* ***structuring element*** *or* ***kernel*** *which decides the nature of operation. Two basic morphological operators are Erosion and Dilation. Then its variant forms like Opening, Closing, Gradient etc also comes into play.*

Goal-

- We will learn different morphological operations like Erosion, Dilation, Opening, Closing etc.

- We will see different functions like : cv2.erode(), cv2.dilate(), cv2.morphologyEx() etc.

3A) EROSION

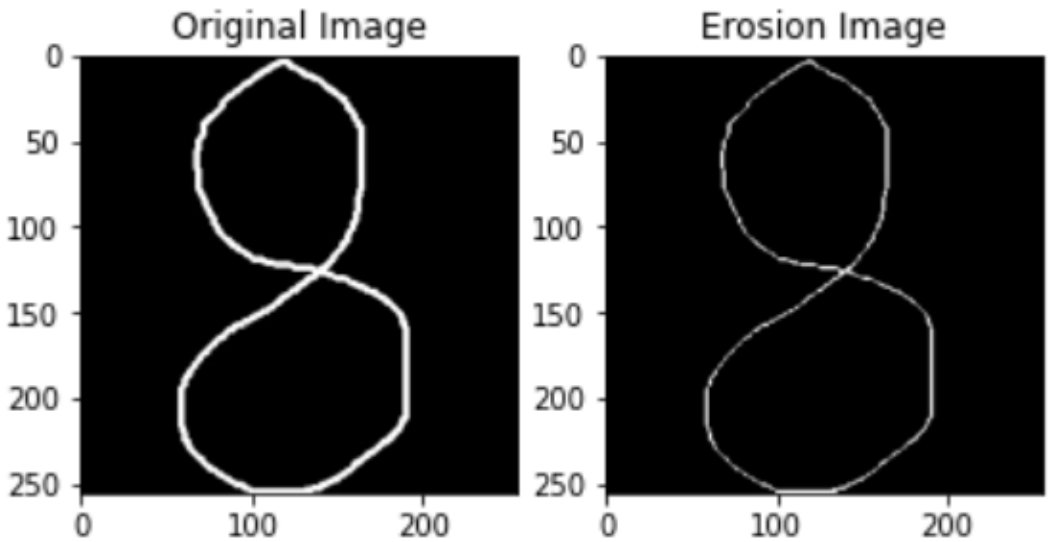
The basic idea of erosion is just like soil erosion only, it erodes away the boundaries of foreground object. The kernel slides through the image (as in 2D convolution). A pixel in the original image (either 1 or 0) will be considered 1 only if all the pixels under the kernel is 1, otherwise it is eroded (made to zero).

Here, as an example, I would use a 3x3 kernel with full of ones. Let's see it how it works:

Press enter or click to view image in full size

```
[17] kernel = np.ones((3,3),np.uint8)
     erosion = cv2.erode(img,kernel,iterations = 1)

     plt.subplot(121),plt.imshow(img),plt.title('Original Image')
     plt.subplot(122),plt.imshow(erosion),plt.title('Erosion Image')
```



3B) DILATION

Get Ravjot Singh's stories in your inbox

Join Medium for free to get updates from this writer.

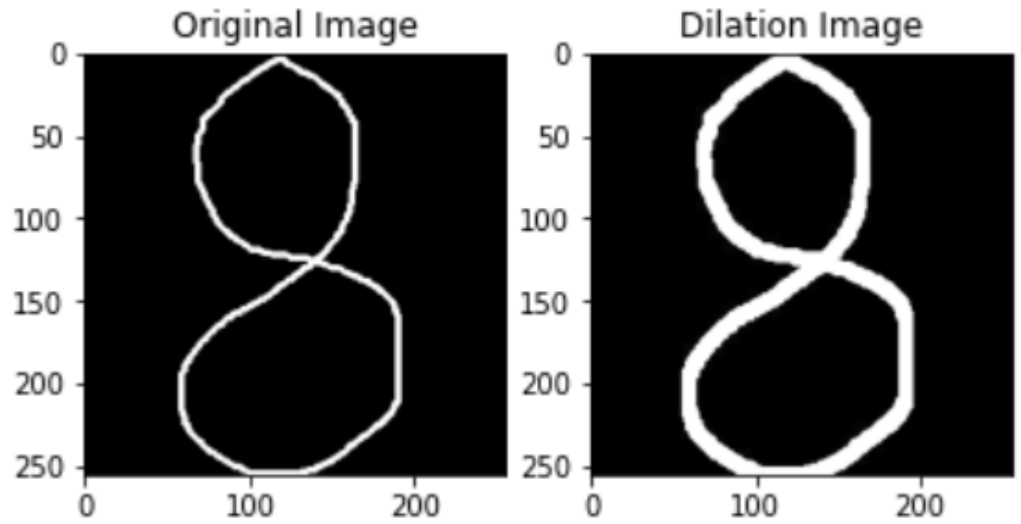
It is just opposite of erosion. Here, a pixel element is '1' if

atleast one pixel under the kernel is ‘1’. So it increases the white region in the image or size of foreground object increases. Normally, in cases like noise removal, erosion is followed by dilation. Because, erosion removes white noises, but it also shrinks our object. So we dilate it. Since noise is gone, they won’t come back, but our object area increases. It is also useful in joining broken parts of an object.

Press enter or click to view image in full size

```
[ ]  kernel = np.ones((6,6), np.uint8)
     dilation = cv2.dilate(img, kernel, iterations = 1)

     plt.subplot(121),plt.imshow(img),plt.title('Original Image')
     plt.subplot(122),plt.imshow(dilation),plt.title('Dilation Image')
```



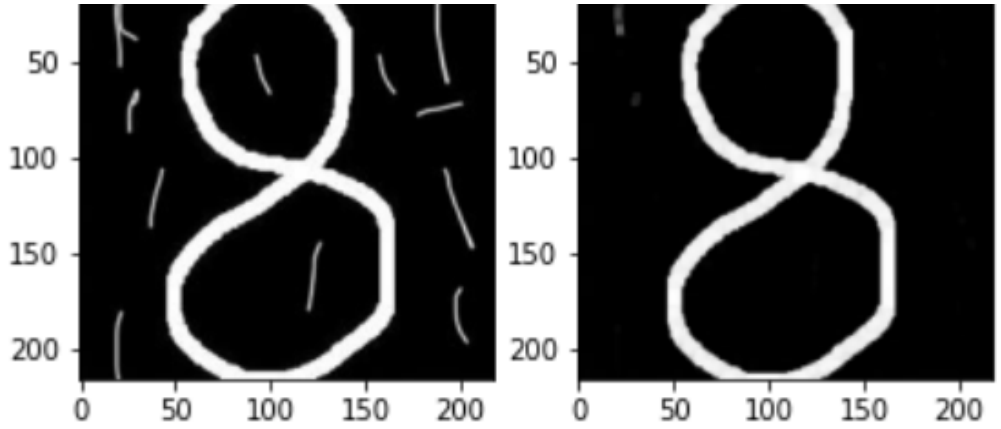
3C) OPENING

Opening is just another name of **erosion followed by dilation**. It is useful in removing noise, as we explained above. Here we use the function, **cv2.morphologyEx().**

Press enter or click to view image in full size

```
[19] img2 = cv2.imread('8a.jpeg')
     kernel = np.ones((4,4), np.uint8)
     opening = cv2.morphologyEx(img2, cv2.MORPH_OPEN, kernel)

     plt.subplot(121),plt.imshow(img2),plt.title('Original Image')
     plt.subplot(122),plt.imshow(opening),plt.title('Opening Image')
```

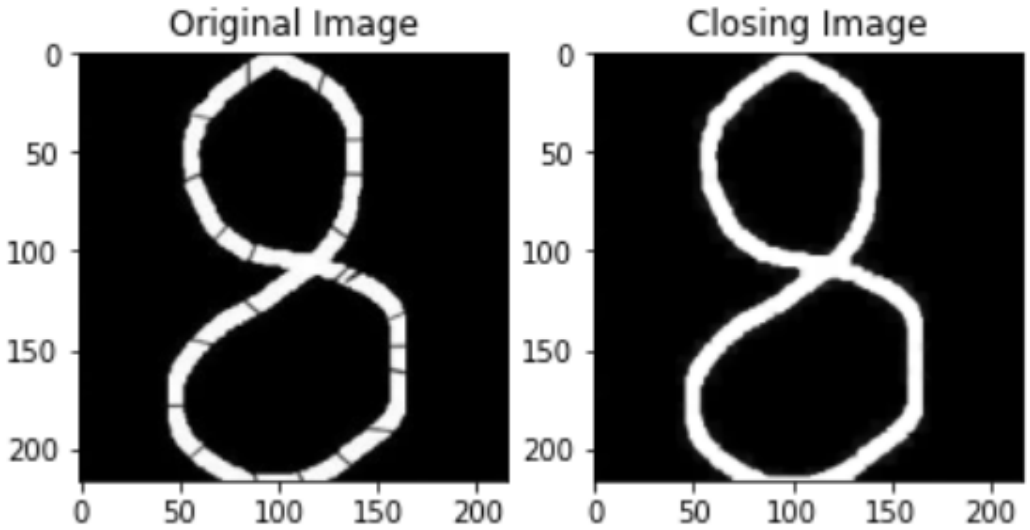


3D) CLOSING

Closing is reverse of Opening, **Dilation followed by Erosion**. It is useful in closing small holes inside the foreground objects, or small black points on the object.

Press enter or click to view image in full size

```
[20] img3 = cv2.imread('8b.jpeg')
     kernel = np.ones((4,4), np.uint8)
     closing = cv2.morphologyEx(img3, cv2.MORPH_CLOSE, kernel)
     plt.subplot(121),plt.imshow(img3),plt.title('Original Image')
     plt.subplot(122),plt.imshow(closing),plt.title('Closing Image')
```



3E) MORPHOLOGICAL GRADIENT

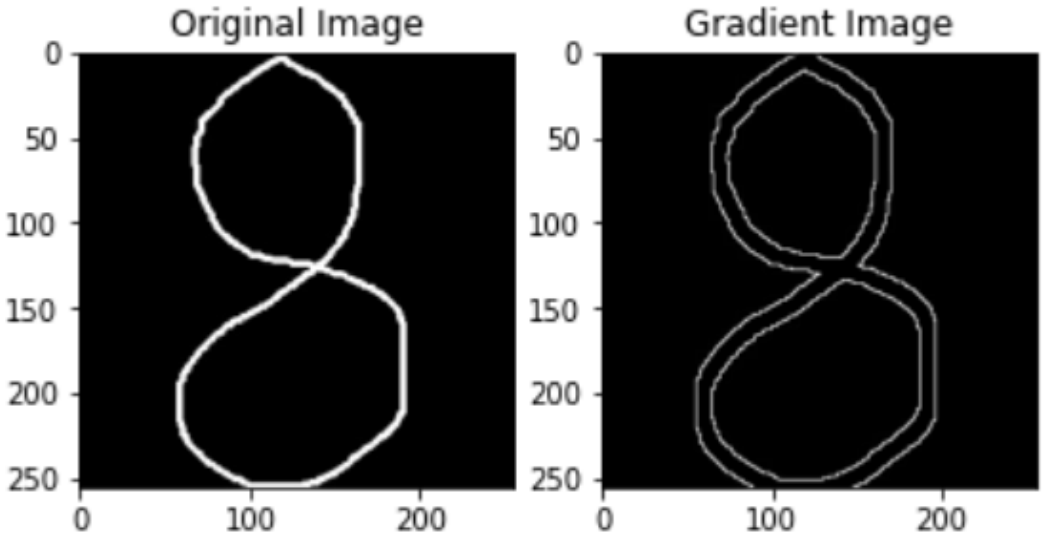
It is the difference between dilation and erosion of an image.

The result will look like the outline of the object.

Press enter or click to view image in full size

```
kernel = np.ones((2,2), np.uint8)
gradient = cv2.morphologyEx(dilation, cv2.MORPH_GRADIENT, kernel)

plt.subplot(121),plt.imshow(img),plt.title('Original Image')
plt.subplot(122),plt.imshow(gradient),plt.title('Gradient Image')
```



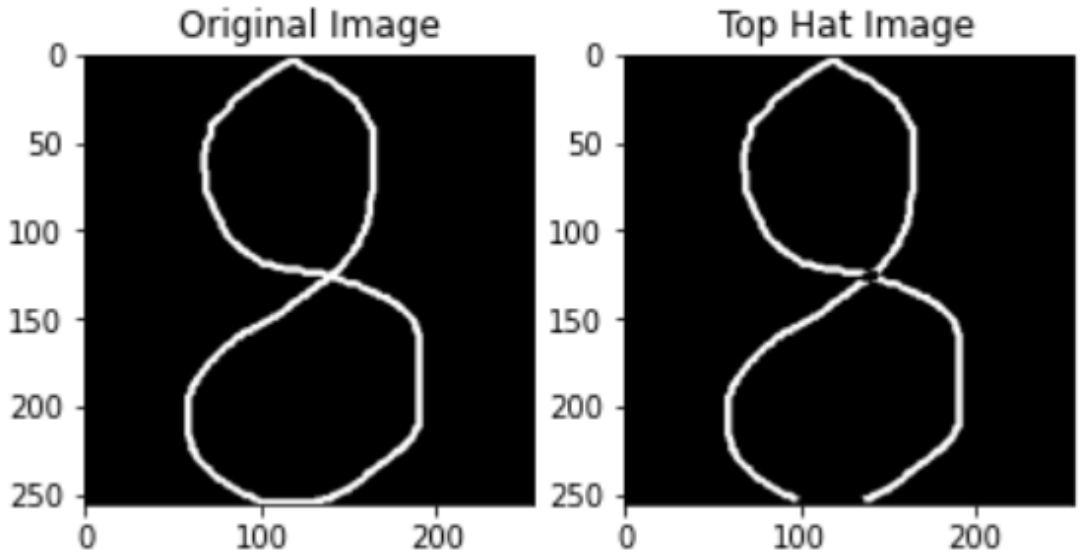
3F) TOP HAT

It is the difference between input image and Opening of the image. Below example is done for a 5x5 kernel.

Press enter or click to view image in full size

```
[21] kernel = np.ones((5,5), np.uint8)
     tophat = cv2.morphologyEx(img, cv2.MORPH_TOPHAT, kernel)

     plt.subplot(121),plt.imshow(img),plt.title('Original Image')
     plt.subplot(122),plt.imshow(tophat),plt.title('Top Hat Image')
```



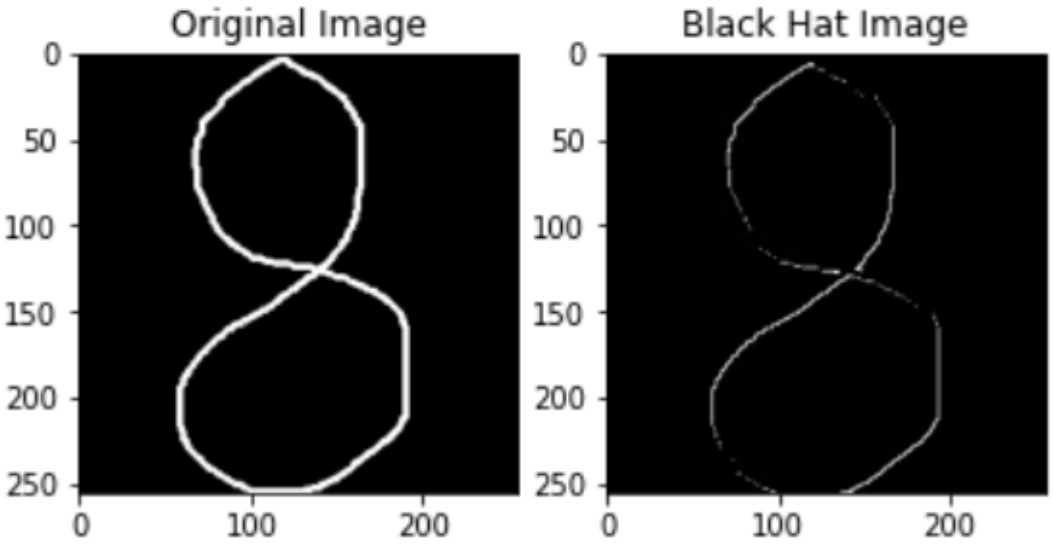
3G) BLACK HAT

It is the difference between the closing of the input image and input image.

Press enter or click to view image in full size

```
[ ]  kernel = np.ones((4,4), np.uint8)
     blackhat = cv2.morphologyEx(img, cv2.MORPH_BLACKHAT, kernel)

     plt.subplot(121),plt.imshow(img),plt.title('Original Image')
     plt.subplot(122),plt.imshow(blackhat),plt.title('Black Hat Image')
```



This brings us to the end of this article. I hope you have understood the basics of Image Processing with OpenCV in Python clearly. ***Make sure you practice as much as possible***.

Do look out for other Jupyter notebooks in the series which will explain the various other aspects of Image Processing with OpenCV in Python.

If you wish to check out more resources related to Data Science and Machine Learning you can refer to my Github account.

Hope you like the post.