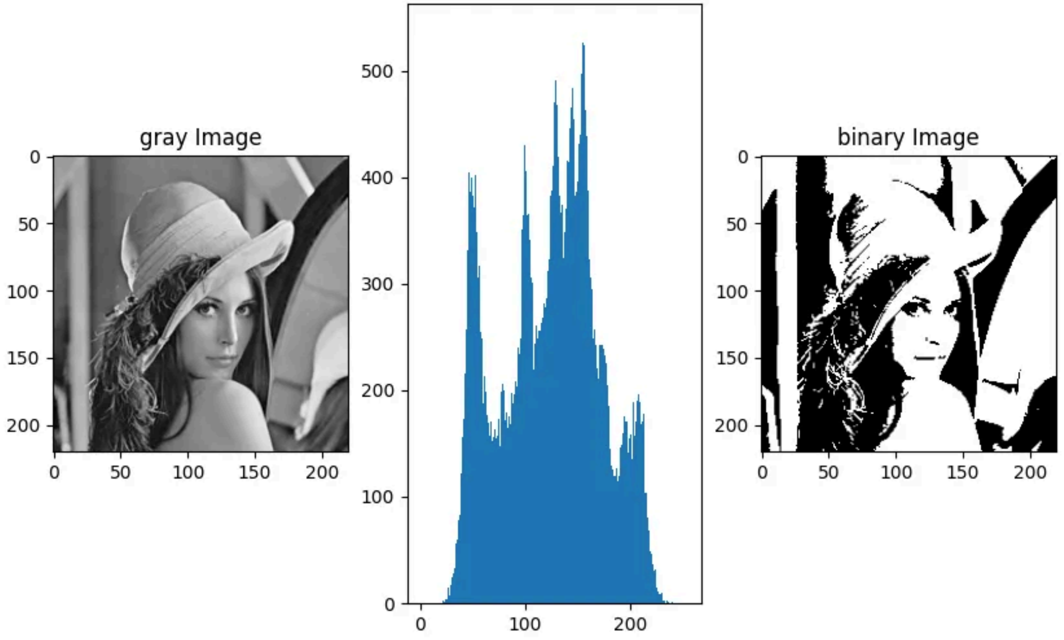
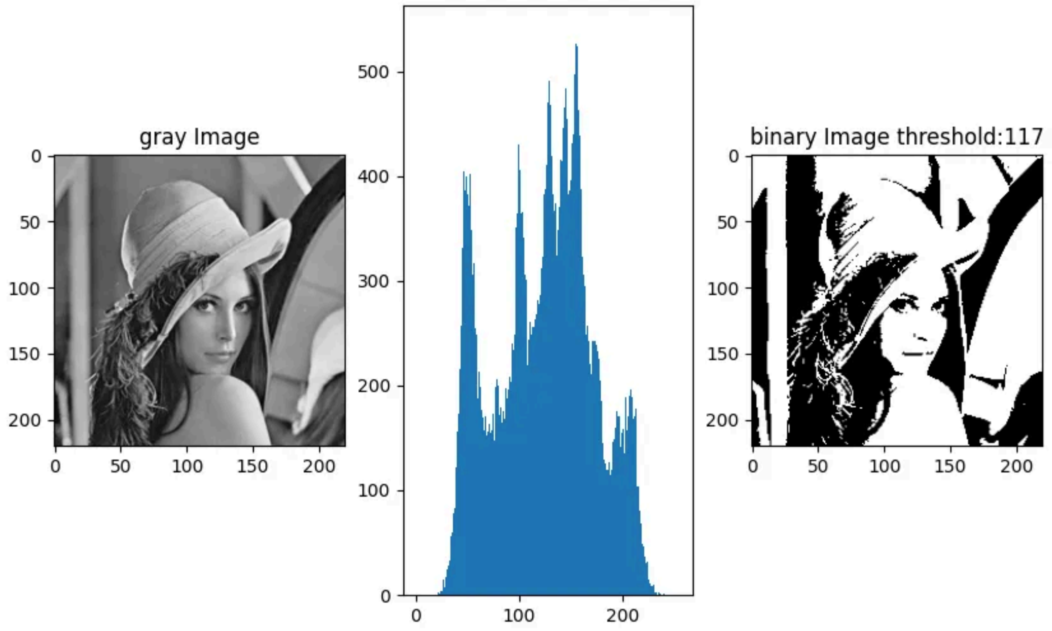
OpenCV3

```
import cv2
import matplotlib.pyplot as plt
image_path = 'image/lenna.png'
image = cv2.imread(image_path, 1)
gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
_, binary_image = cv2.threshold(gray_image, 120, 255, cv2.THRESH_BINARY)
plt.figure(figsize=(10, 6))
plt.subplot(1, 3, 1)
plt.imshow(gray_image, cmap='gray')
plt.title('gray Image')
plt.subplot(1, 3, 2)
plt.hist(gray_image.ravel(), 256, [0, 256])
plt.subplot(1, 3, 3)
plt.imshow(binary_image, cmap='gray')
plt.title('binary Image')
plt.show()
cv2.waitKey(0)
```



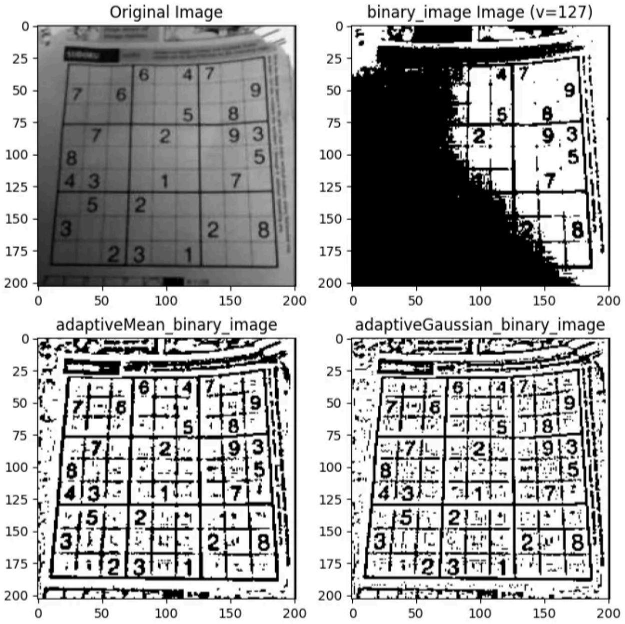
```
import cv2
import matplotlib.pyplot as plt
import numpy as np
image_path = 'image/lenna.png'
image = cv2.imread(image_path, 1)
gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
def otsu_binarization(image):
    hist, bins = np.histogram(image.flatten(), 256, [0, 256])
    total = image.shape[0] * image.shape[1]
    current_max = 0
    threshold = 0
    sum_total = 0
    for t in range(256):
        sum_total += t * hist[t]
    sumB = 0
    wB = 0
    wF = 0
    var_between = 0
    for t in range(256):
        wB += hist[t]
        if wB == 0:
            continue
        wF = total - wB
        if wF == 0:
            break
        sumB += t * hist[t]
        meanB = sumB / wB
        meanF = (sum_total - sumB) / wF
        var_between = wB * wF * (meanB - meanF) ** 2
        if var_between > current_max:
            current_max = var_between
            threshold = t
    return threshold
# _, binary_image = cv2.threshold(gray_image, 0, 255,
#                         cv2.THRESH_BINARY + cv2.THRESH_OTSU)
```

```python
# binary_image = cv2.adaptiveThreshold(gray_image, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
#               cv2.THRESH_BINARY, 11, 2)
threshold = otsu_binarization(gray_image)
_, binary_image = cv2.threshold(gray_image, threshold, 255, cv2.THRESH_BINARY)
plt.figure(figsize=(10, 6))
plt.subplot(1, 3, 1)
plt.imshow(gray_image, cmap='gray')
plt.title('gray Image')
plt.subplot(1, 3, 2)
plt.hist(gray_image.ravel(), 256, [0, 256])
plt.subplot(1, 3, 3)
plt.imshow(binary_image, cmap='gray')
plt.title(f'binary Image threshold:{threshold}')
plt.show()
cv2.waitKey(0)
```



```
import cv2
import matplotlib.pyplot as plt
import numpy as np
image_path = 'image/02.jpeg'
image = cv2.imread(image_path, 1)
gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
_, binary_image = cv2.threshold(gray_image, 127, 255, cv2.THRESH_BINARY)
# adaptivate Mean
adaptiveMean_binary_image = cv2.adaptiveThreshold(gray_image, 255,
                 cv2.ADAPTIVE_THRESH_MEAN_C, cv2.THRESH_BINARY, 11, 2)
```

```
# adaptivate Gaussian
adaptiveGaussian_binary_image = cv2.adaptiveThreshold(gray_image, 255,
                 cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY, 11, 2)
plt.figure(figsize=(8, 8))
plt.subplot(2, 2, 1)
plt.imshow(image, cmap='gray')
plt.title('Original Image')
plt.subplot(2, 2, 2)
plt.imshow(binary_image, cmap='gray')
plt.title('binary_image Image (v=127)')
plt.subplot(2, 2, 3)
plt.imshow(adaptiveMean_binary_image, cmap='gray')
plt.title('adaptiveMean_binary_image')
plt.subplot(2, 2, 4)
plt.imshow(adaptiveGaussian_binary_image, cmap='gray')
plt.title('adaptiveGaussian_binary_image')
plt.show()
```



Erosion

```
import cv2
import matplotlib.pyplot as plt
import numpy as np
image_path = 'image/03.png'
image = cv2.imread(image_path, 1)
gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
_, binary_image = cv2.threshold(gray_image, 127, 255, cv2.THRESH_BINARY)
kernel_4 = np.array([[0, 1, 0],
                     [1, 1, 1],
                     [0, 1, 0]], dtype=np.uint8)
eroded_image_4 = cv2.erode(binary_image, kernel_4, iterations=1)
kernel_8 = np.ones((3, 3), np.uint8)
eroded_image_8 = cv2.erode(binary_image, kernel_8, iterations=2)
cv2.imshow('binary_image', binary_image)
cv2.waitKey(0)
plt.figure(figsize=(8, 8))
plt.subplot(1, 3, 1)
plt.imshow(image, cmap='gray')
plt.title('Original Image')
plt.subplot(1, 3, 2)
plt.imshow(eroded_image_4, cmap='gray')
plt.title('eroded_image_4')
plt.subplot(1, 3, 3)
plt.imshow(eroded_image_8, cmap='gray')
plt.title('eroded_image_8')
plt.show()
```

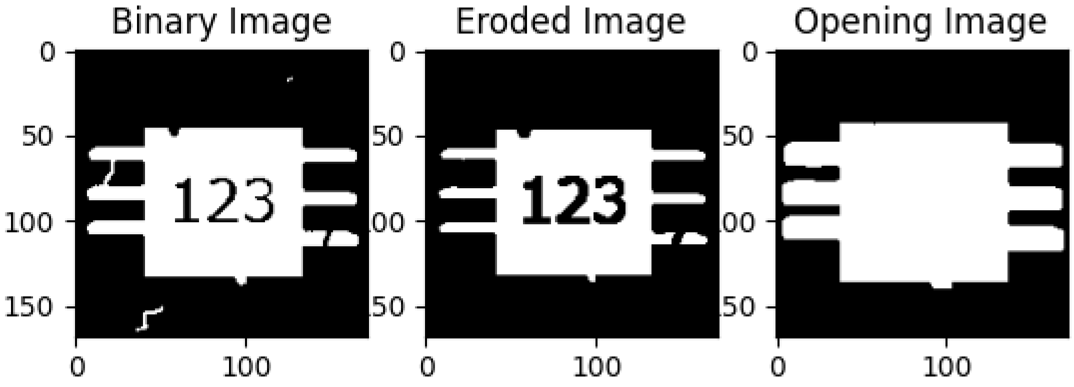
Dilation

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

```
image = cv2.imread('image/03.png', cv2.IMREAD_GRAYSCALE)
_, binary_image = cv2.threshold(image, 127, 255, cv2.THRESH_BINARY)
kernel = np.ones((3, 3), np.uint8)
dilated_image = cv2.dilate(binary_image, kernel, iterations=1)
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1), plt.imshow(binary_image, cmap='gray'), plt.title('原始二值图像')
plt.subplot(1, 2, 2), plt.imshow(dilated_image, cmap='gray'), plt.title('膨胀后的图像')
plt.show()
```

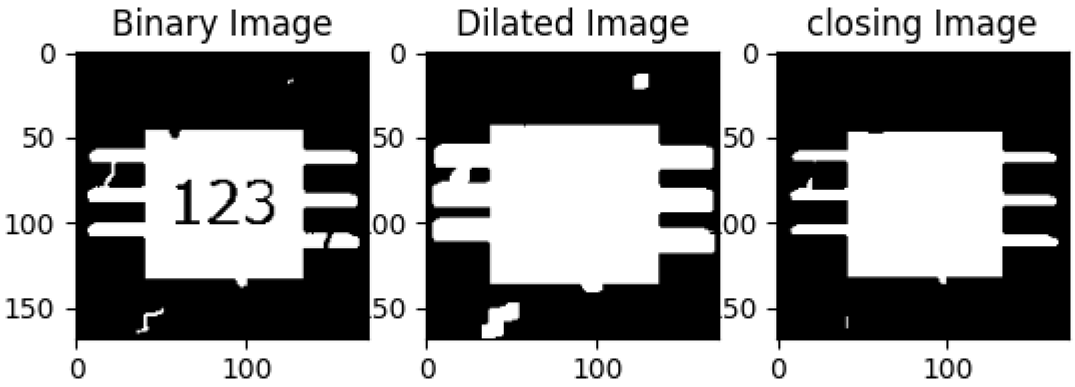
Opening

```
# erosion dilation
import cv2
import matplotlib.pyplot as plt
import numpy as np
image_path = 'image/04.png'
image = cv2.imread(image_path, 1)
gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
_, binary_image = cv2.threshold(gray_image, 127, 255, cv2.THRESH_BINARY)
kernel = np.ones((3, 3), np.uint8)
eroded_image_8 = cv2.erode(binary_image, kernel, iterations=1)
dilated_image = cv2.dilate(eroded_image_8, kernel, iterations=4)
plt.subplot(1, 3, 1)
plt.imshow(binary_image, cmap='gray')
plt.title('Binary Image')
plt.subplot(1, 3, 2)
plt.imshow(eroded_image_8, cmap='gray')
plt.title('Eroded Image')
plt.subplot(1, 3, 3)
plt.imshow(dilated_image, cmap='gray')
plt.title('Opening Image')
plt.show()
```



Closing

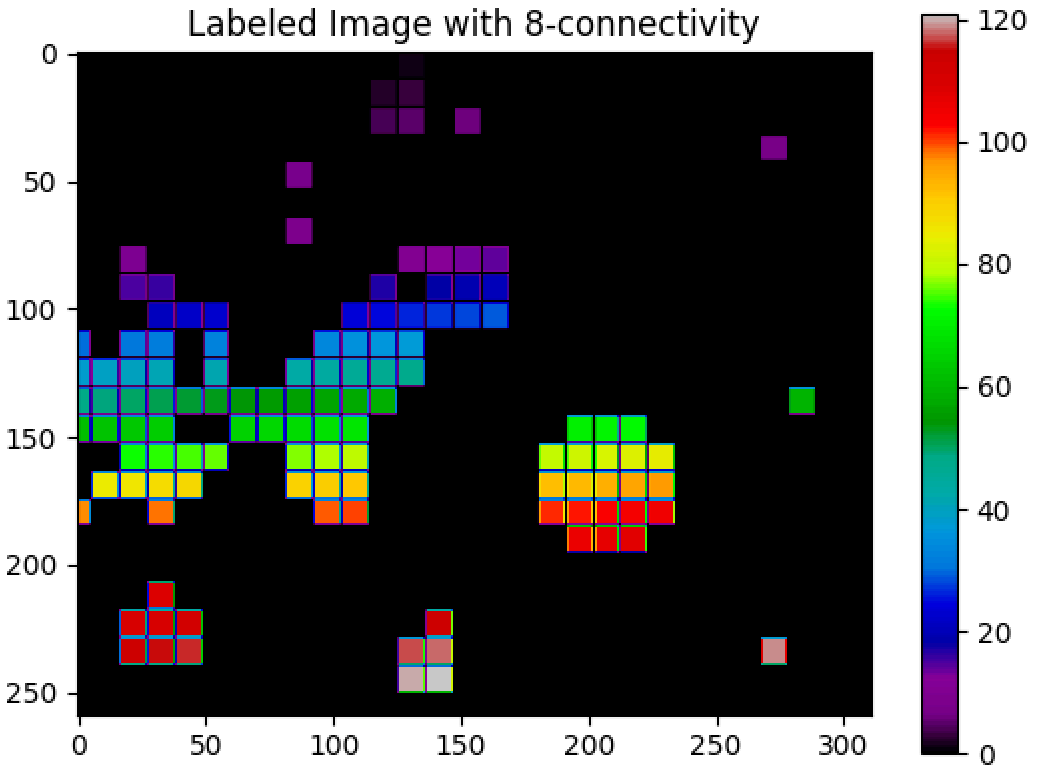
```
#  dilation erosion
import cv2
import matplotlib.pyplot as plt
import numpy as np
image_path = 'image/04.png'
image = cv2.imread(image_path, 1)
gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
_, binary_image = cv2.threshold(gray_image, 127, 255, cv2.THRESH_BINARY)
kernel = np.ones((3, 3), np.uint8)
dilated_image = cv2.dilate(binary_image, kernel, iterations=3)
eroded_image_8 = cv2.erode(dilated_image, kernel, iterations=4)
plt.subplot(1, 3, 1)
plt.imshow(binary_image, cmap='gray')
plt.title('Binary Image')
plt.subplot(1, 3, 2)
plt.imshow(dilated_image, cmap='gray')
plt.title('Dilated Image')
plt.subplot(1, 3, 3)
plt.imshow(eroded_image_8, cmap='gray')
plt.title('closing Image')
plt.show()
```



Connected Component Labeling - CCL

```
import numpy as np
import cv2
from matplotlib import pyplot as plt
# init
def find_neighbors(x, y, labels):
    neighbors = []
    for i in range(x-1, x+2):
        for j in range(y-1, y+2):
            if i == x and j == y:
                continue
            if 0 <= i < labels.shape[0] and 0 <= j < labels.shape[1]:
                neighbors.append(labels[i, j])
    return neighbors
def connected_component_labeling(binary_image):
    labels = np.zeros(binary_image.shape, dtype=int)
    current_label = 1
    for i in range(binary_image.shape[0]):
        for j in range(binary_image.shape[1]):
            if binary_image[i, j] == 255:
                neighbors = find_neighbors(i, j, labels)
                labeled_neighbors = [n for n in neighbors if n > 0]
                if len(labeled_neighbors) > 0:
                    labels[i, j] = min(labeled_neighbors)
                else:
                    labels[i, j] = current_label
                    current_label += 1
    return labels
img = cv2.imread('image/03.png', 0)
_, binary_image = cv2.threshold(img, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)

labeled_image = connected_component_labeling(binary_image)
plt.imshow(labeled_image, cmap='nipy_spectral')
plt.colorbar()
plt.title("Labeled Image with 8-connectivity")
plt.show()
```


Labeled Image with 8-connectivity
0
50
100
150
200
250
0
50
100
150
200
250
300
120
100
80
60
40
20
0