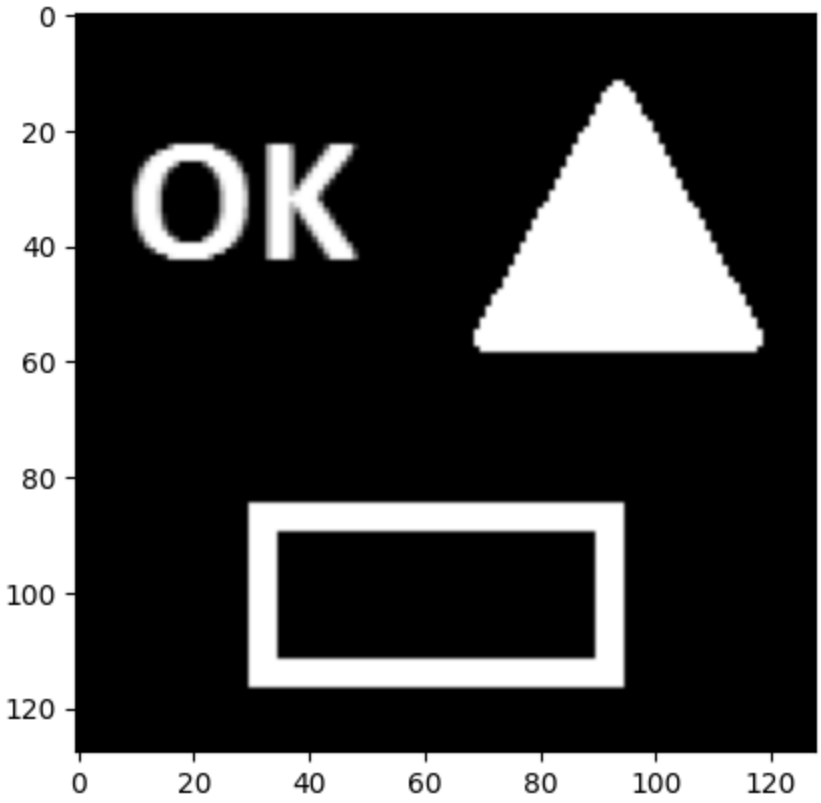
In [ ]:
```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from io import BytesIO
from PIL import Image
from google.colab.patches import cv2_imshow
from google.colab import files
# uploaded = files.upload()
```

In [ ]:
```
img = cv2.imread('OM.png',0)
plt.imshow(img,cmap=plt.cm.gray)
```

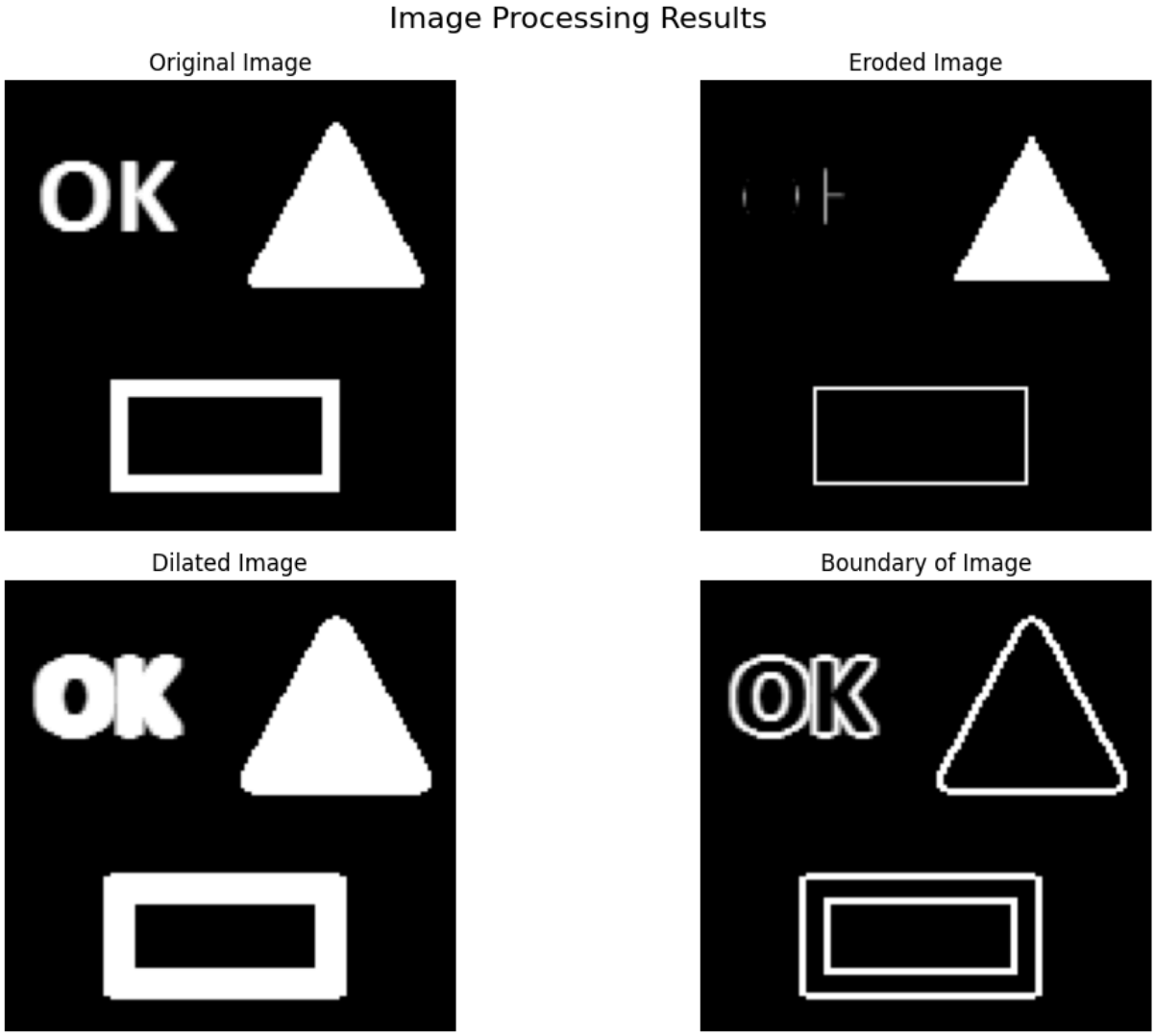
Out[ ]: <matplotlib.image.AxesImage at 0x7d8cd80b5540>



In [ ]:
```
m,n= img.shape
```

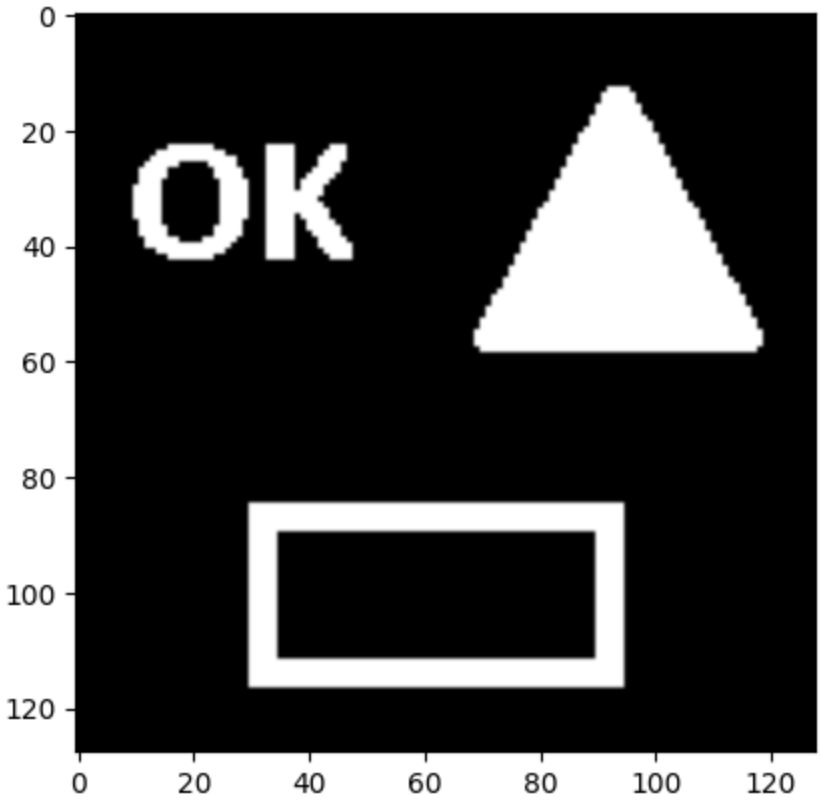
In [ ]:
```
kernel = np.ones((5,5),np.uint8)
kernel1=cv2.getStructuringElement(cv2.MORPH_RECT,(5,5))
kernel2 =cv2.getStructuringElement(cv2.MORPH_ELLIPSE,(5,5))
kernel3= cv2.getStructuringElement(cv2.MORPH_CROSS,(5,5))
img_erosion = cv2.erode(img, kernel, iterations=1)
img_dilate = cv2.dilate(img, kernel2, iterations=1)
img_boundary = img_dilate-img


fig = plt.figure(figsize=(12, 8))

# Set the title for the entire figure
fig.suptitle('Image Processing Results', fontsize=16)
```

```
# Add subplots
ax1 = fig.add_subplot(2, 2, 1)
ax2 = fig.add_subplot(2, 2, 2)
ax3 = fig.add_subplot(2, 2, 3)
ax4 = fig.add_subplot(2, 2, 4)

# Set titles for subplots
ax1.set_title('Original Image', fontsize=12)
ax2.set_title('Eroded Image', fontsize=12)
ax3.set_title('Dilated Image', fontsize=12)
ax4.set_title('Boundary of Image', fontsize=12)

# Display images on subplots
ax1.imshow(img, cmap='gray')
ax2.imshow(img_erosion, cmap='gray')
ax3.imshow(img_dilate, cmap='gray')
ax4.imshow(img_boundary, cmap='gray')

# Remove ticks from subplots
ax1.axis('off')
ax2.axis('off')
ax3.axis('off')
ax4.axis('off')

# Adjust layout
plt.tight_layout()

# Show the figure
plt.show()
```



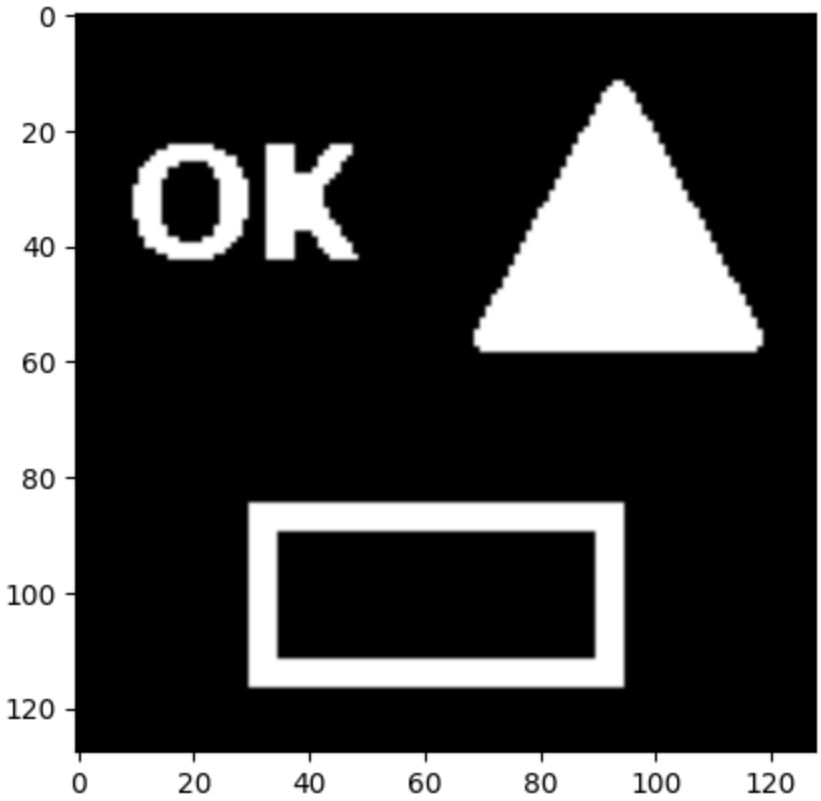
In [ ]:
```
#Opening & Closing
binr = cv2.threshold(img, 0, 255,
                     cv2.THRESH_BINARY+cv2.THRESH_OTSU)[1]

# define the kernel
kernel = np.ones((3, 3), np.uint8)

# opening the image
opening = cv2.morphologyEx(binr, cv2.MORPH_OPEN,
                           kernel, iterations=1)
# print the output
plt.imshow(opening, cmap='gray')
```

Out[ ]:
```
<matplotlib.image.AxesImage at 0x7d8cd7c9d150>
```



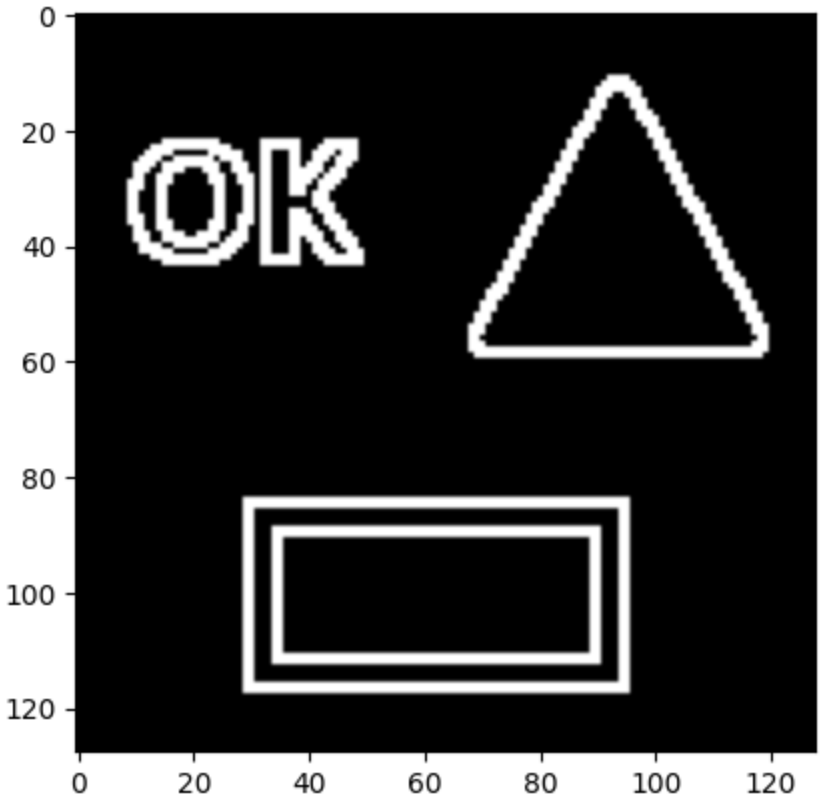
In [ ]:
```
binr = cv2.threshold(img, 0, 255, cv2.THRESH_BINARY+cv2.THRESH_OTSU)[1]

# define the kernel
kernel = np.ones((3, 3), np.uint8)

# opening the image
closing = cv2.morphologyEx(binr, cv2.MORPH_CLOSE, kernel, iterations=1)

# print the output
plt.imshow(closing, cmap='gray')
```

Out[ ]:
```
<matplotlib.image.AxesImage at 0x7d8cd7c9ee60>
```



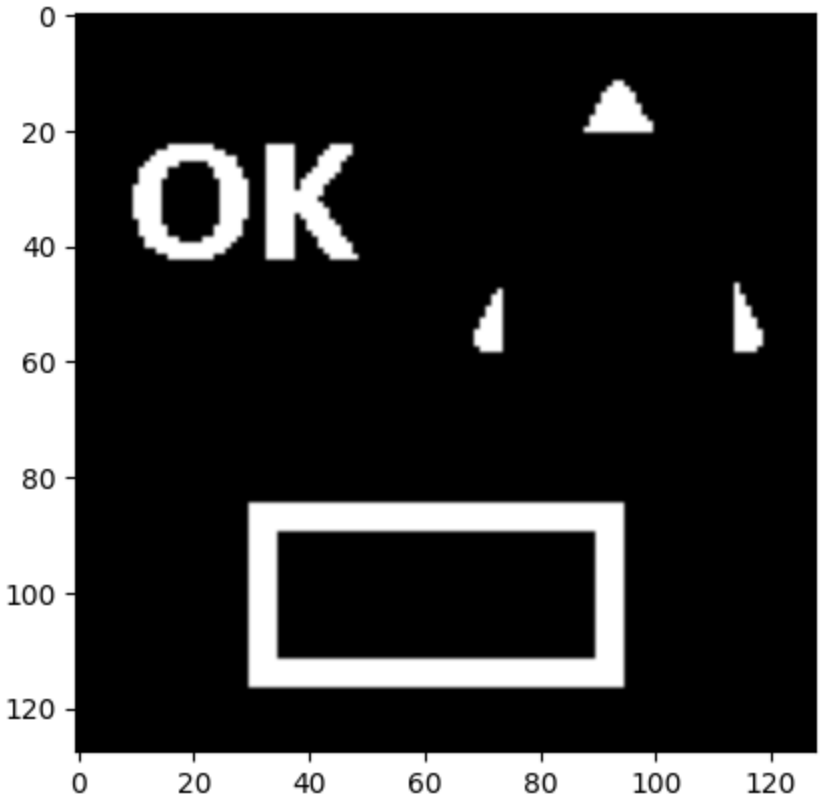
In [ ]:
```
#Morphological Gradient
binr = cv2.threshold(img, 0, 255, cv2.THRESH_BINARY+cv2.THRESH_OTSU)[1]

# define the kernel
kernel = np.ones((3, 3), np.uint8)

# invert the image
invert = cv2.bitwise_not(binr)

# use morph gradient
morph_gradient = cv2.morphologyEx(invert,
                                  cv2.MORPH_GRADIENT,
                                  kernel)


# print the output
plt.imshow(morph_gradient, cmap='gray')
```

Out[ ]:
```
<matplotlib.image.AxesImage at 0x7d8cd8387640>
```



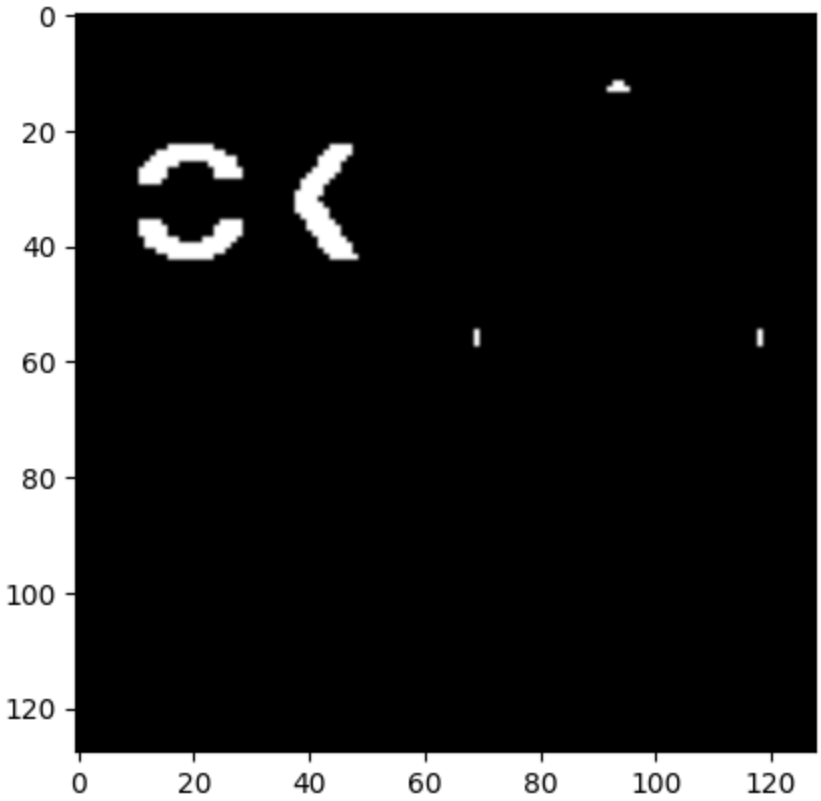
In [ ]:
```
#Top hat
binr = cv2.threshold(img, 0, 255, cv2.THRESH_BINARY+cv2.THRESH_OTSU)[1]

# define the kernel
kernel = np.ones((13, 13), np.uint8)

# use morph gradient
morph_gradient = cv2.morphologyEx(binr,
                                  cv2.MORPH_TOPHAT,
                                  kernel)
# print the output
plt.imshow(morph_gradient, cmap='gray')
```

Out[ ]:
```
<matplotlib.image.AxesImage at 0x7d8cd8d84400>
```

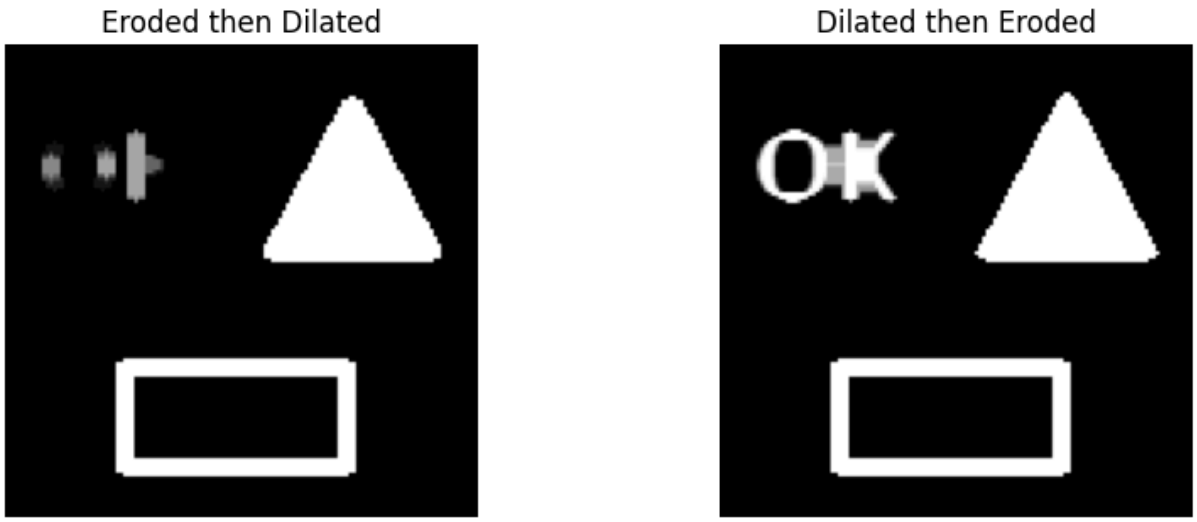


In [ ]:
```
# Black Hat
binr = cv2.threshold(img, 0, 255, cv2.THRESH_BINARY+cv2.THRESH_OTSU)[1]

# define the kernel
kernel = np.ones((5, 5), np.uint8)

# invert the image
invert = cv2.bitwise_not(binr)

# use morph gradient
morph_gradient = cv2.morphologyEx(invert,
                                  cv2.MORPH_BLACKHAT,
                                  kernel)
# print the output
plt.imshow(morph_gradient, cmap='gray')
```

Out[ ]:
```
<matplotlib.image.AxesImage at 0x7d8cd8ef2fb0>
```

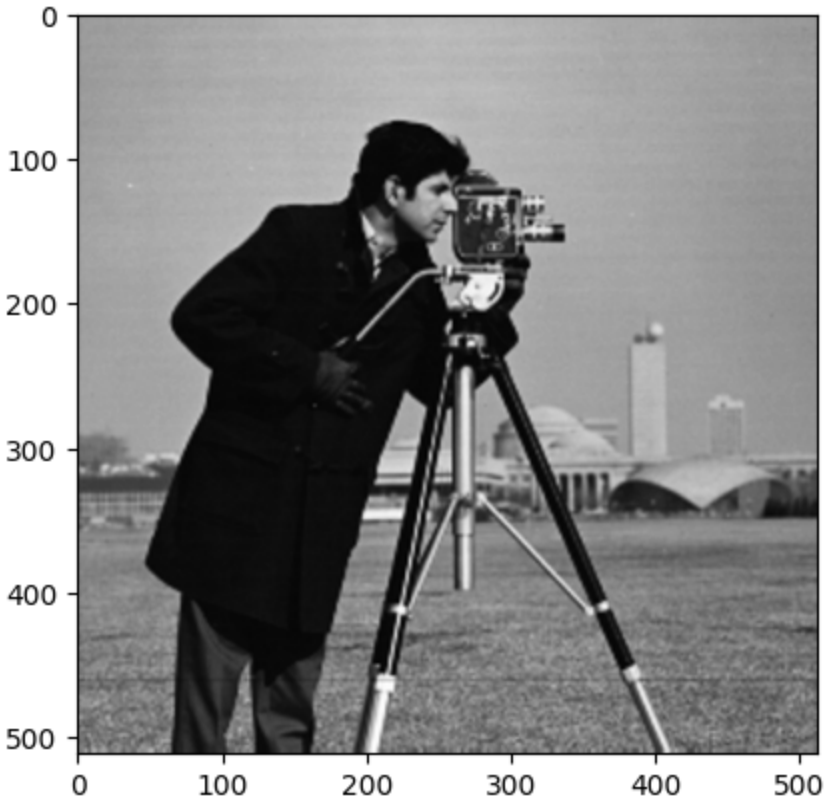


In [ ]:
```
img_eroded_dilate = cv2.dilate(img_erosion, kernel2, iterations=1)
img_dilated_erode = cv2.erode(img_dilate, kernel, iterations=1)


fig = plt.figure(figsize=(10, 8))

# Set the title for the entire figure
fig.suptitle('Opening & Closing Operations', fontsize=16)

# Add subplots
ax1 = fig.add_subplot(2, 2, 1)
ax2 = fig.add_subplot(2, 2, 2)

# Set titles for subplots
ax1.set_title('Eroded then Dilated', fontsize=12)
ax2.set_title('Dilated then Eroded', fontsize=12)

# Display images on subplots
ax1.imshow(img_eroded_dilate, cmap='gray')
ax2.imshow(img_dilated_erode, cmap='gray')

# Remove ticks from subplots
ax1.axis('off')
ax2.axis('off')

# Show the figure
plt.show()
```



In [ ]:
```
img = cv2.imread('cameraman.tif',0)
plt.imshow(img,cmap=plt.cm.gray)
```

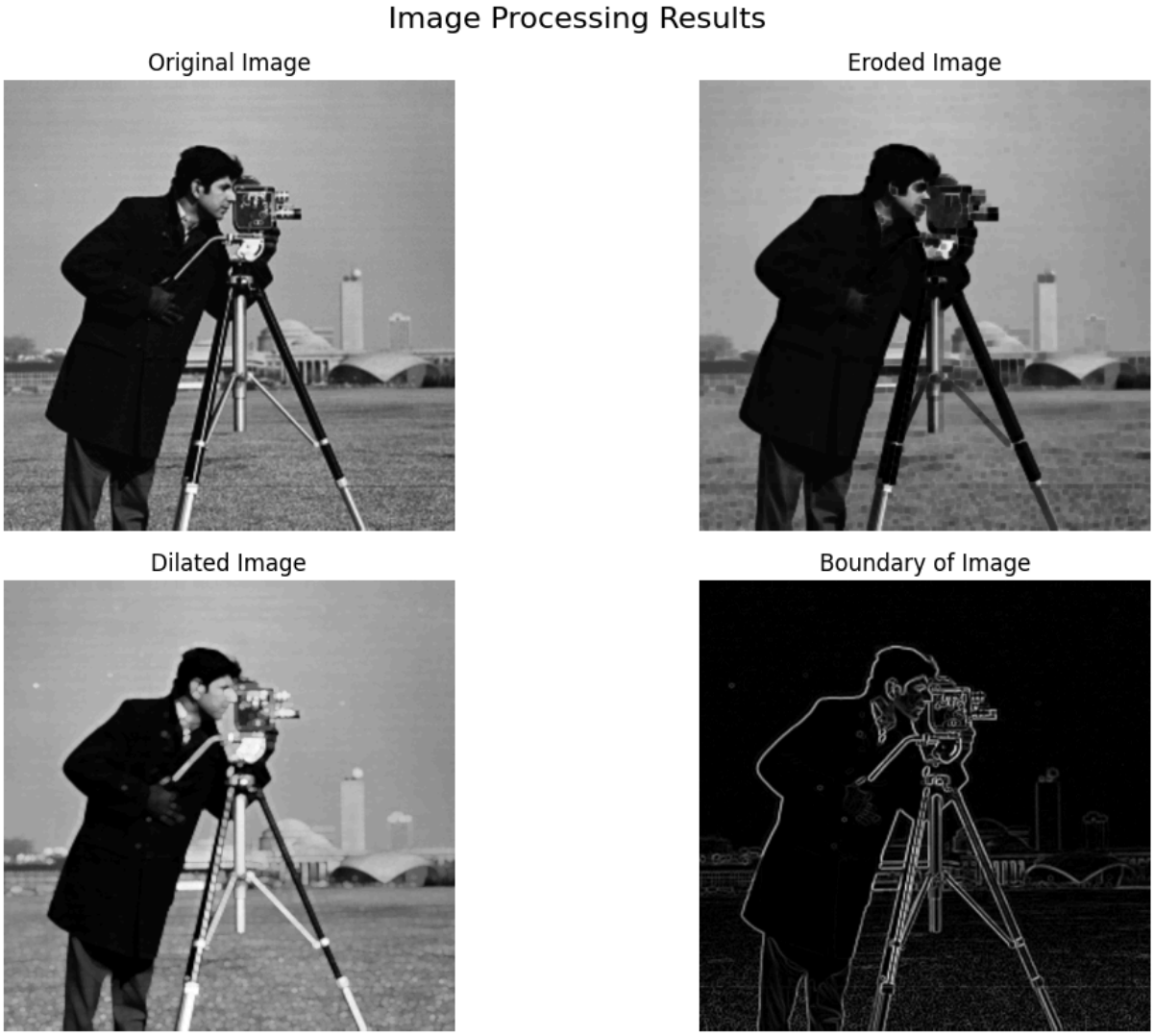
Out[ ]:
```
<matplotlib.image.AxesImage at 0x7d8cd8cecbe0>
```



In [ ]:
```
m,n= img.shape
```

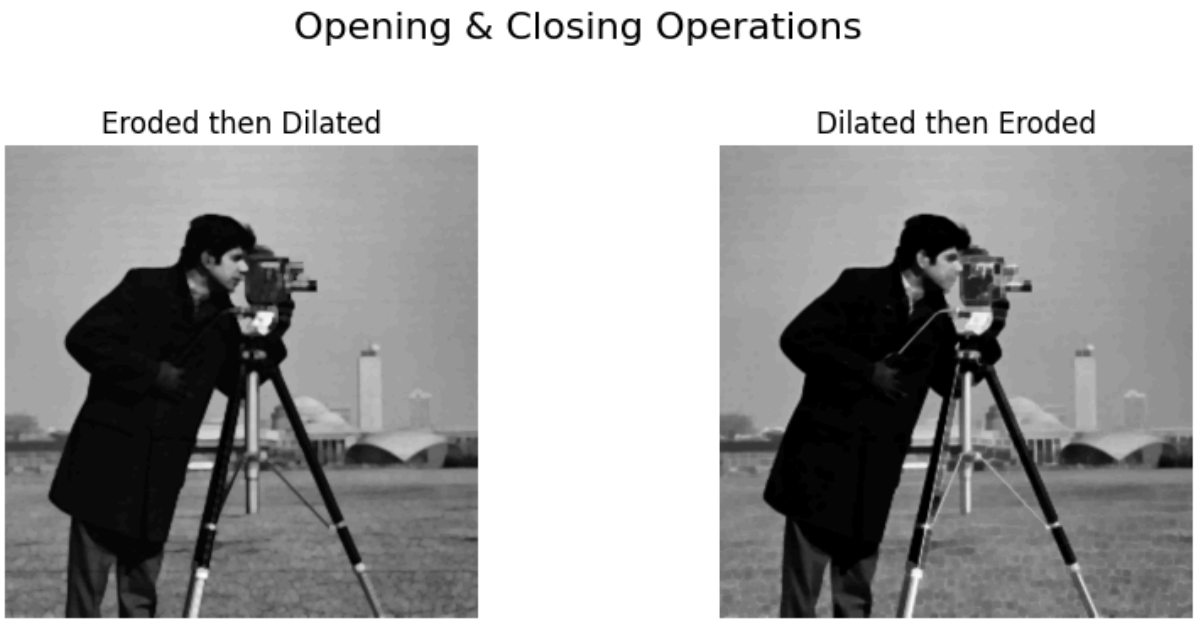
In [ ]:
```
img_erosion = cv2.erode(img, kernel, iterations=1)
img_dilate = cv2.dilate(img, kernel2, iterations=1)
img_boundary = img_dilate-img
```

```
fig = plt.figure(figsize=(12, 8))

# Set the title for the entire figure
fig.suptitle('Image Processing Results', fontsize=16)

# Add subplots
ax1 = fig.add_subplot(2, 2, 1)
ax2 = fig.add_subplot(2, 2, 2)
ax3 = fig.add_subplot(2, 2, 3)
ax4 = fig.add_subplot(2, 2, 4)

# Set titles for subplots
ax1.set_title('Original Image', fontsize=12)
ax2.set_title('Eroded Image', fontsize=12)
ax3.set_title('Dilated Image', fontsize=12)
ax4.set_title('Boundary of Image', fontsize=12)

# Display images on subplots
ax1.imshow(img, cmap='gray')
ax2.imshow(img_erosion, cmap='gray')
ax3.imshow(img_dilate, cmap='gray')
ax4.imshow(img_boundary, cmap='gray')

# Remove ticks from subplots
ax1.axis('off')
ax2.axis('off')
ax3.axis('off')
ax4.axis('off')

# Adjust layout
plt.tight_layout()

# Show the figure
plt.show()
```



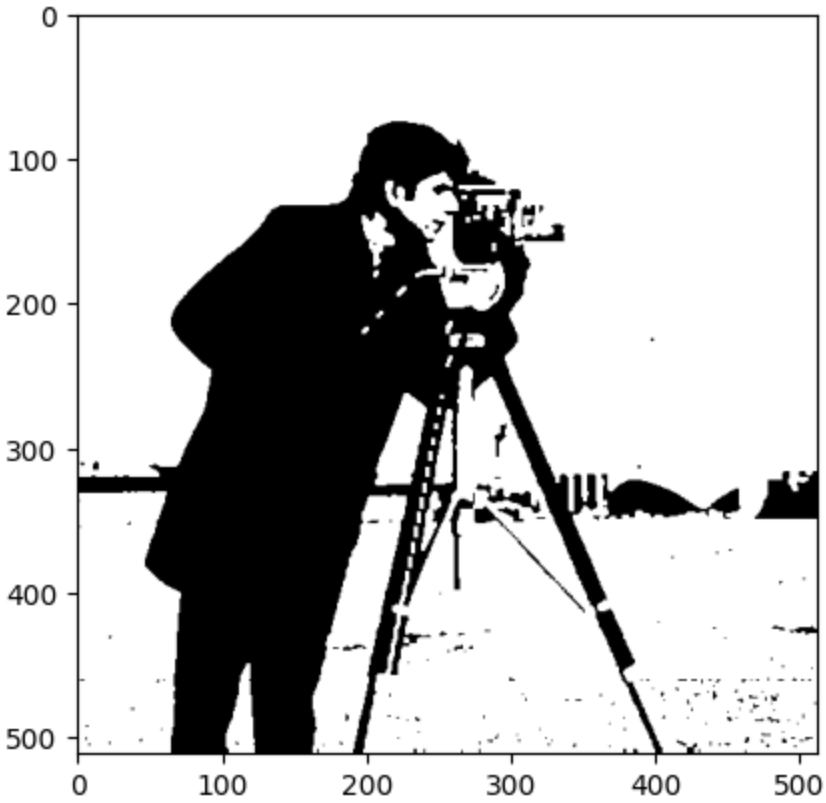
In [ ]:
```
img_eroded_dilate = cv2.dilate(img_erosion, kernel2, iterations=1)
img_dilated_erode = cv2.erode(img_dilate, kernel, iterations=1)


fig = plt.figure(figsize=(10, 8))

# Set the title for the entire figure
fig.suptitle('Opening & Closing Operations', fontsize=16)

# Add subplots
ax1 = fig.add_subplot(2, 2, 1)
ax2 = fig.add_subplot(2, 2, 2)

# Set titles for subplots
ax1.set_title('Eroded then Dilated', fontsize=12)
ax2.set_title('Dilated then Eroded', fontsize=12)

# Display images on subplots
ax1.imshow(img_eroded_dilate, cmap='gray')
ax2.imshow(img_dilated_erode, cmap='gray')

# Remove ticks from subplots
ax1.axis('off')
```

```
ax2.axis('off')

# Show the figure
plt.show()
```



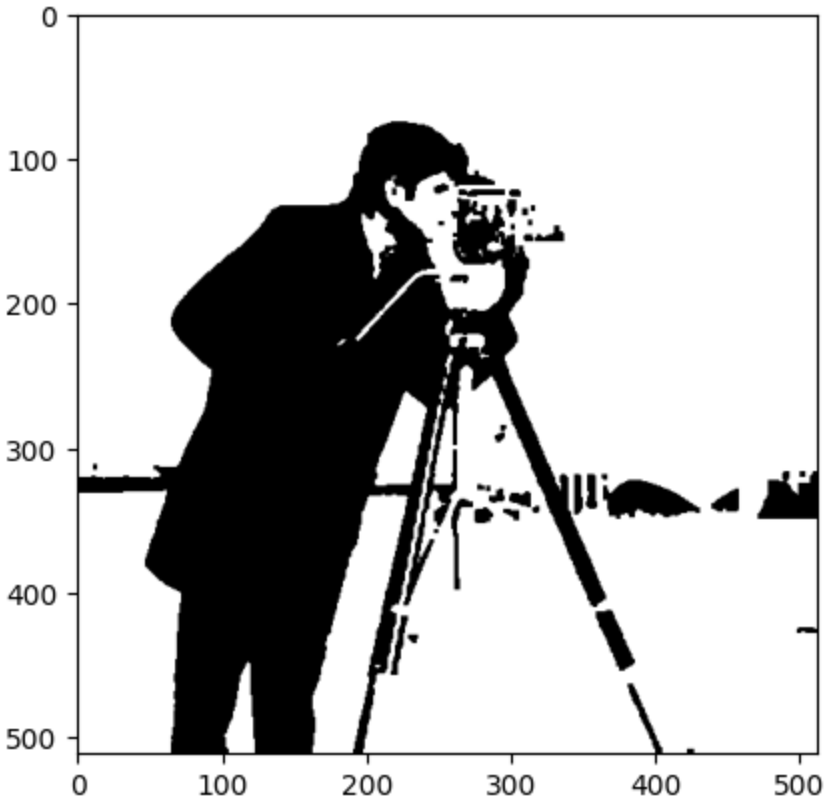
In [ ]:
```
#Opening & Closing
binr = cv2.threshold(img, 0, 255,
                     cv2.THRESH_BINARY+cv2.THRESH_OTSU)[1]

# define the kernel
kernel = np.ones((3, 3), np.uint8)

# opening the image
opening = cv2.morphologyEx(binr, cv2.MORPH_OPEN,
                           kernel, iterations=1)
# print the output
plt.imshow(opening, cmap='gray')
```

Out[ ]:
```
<matplotlib.image.AxesImage at 0x7d8cd732bee0>
```



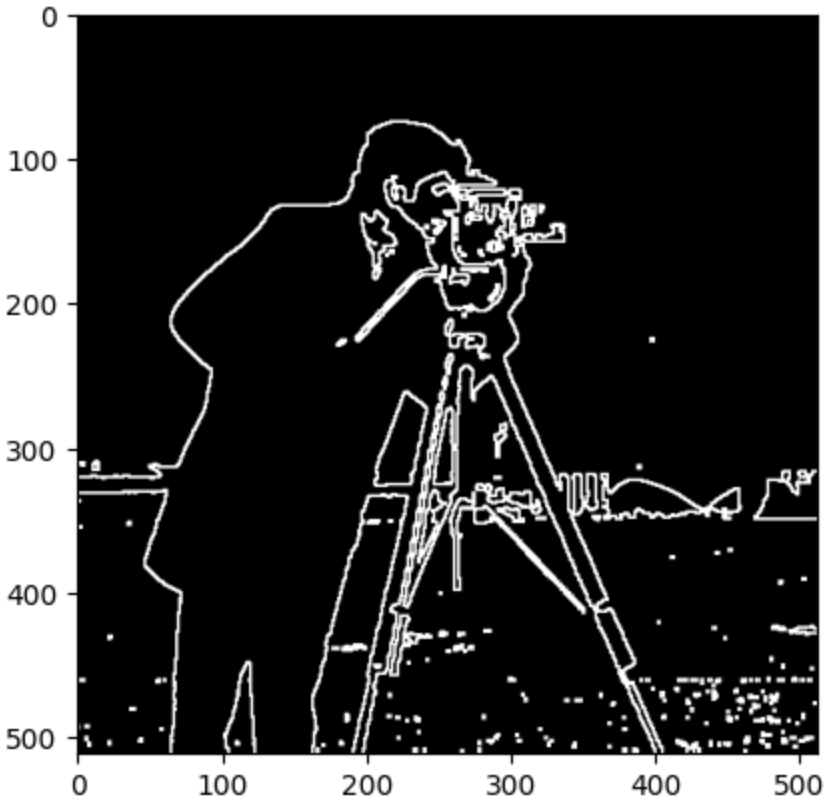
In [ ]:
```
binr = cv2.threshold(img, 0, 255, cv2.THRESH_BINARY+cv2.THRESH_OTSU)[1]

# define the kernel
kernel = np.ones((3, 3), np.uint8)

# opening the image
closing = cv2.morphologyEx(binr, cv2.MORPH_CLOSE, kernel, iterations=1)

# print the output
plt.imshow(closing, cmap='gray')
```

Out[ ]:
```
<matplotlib.image.AxesImage at 0x7d8cd705ca90>
```



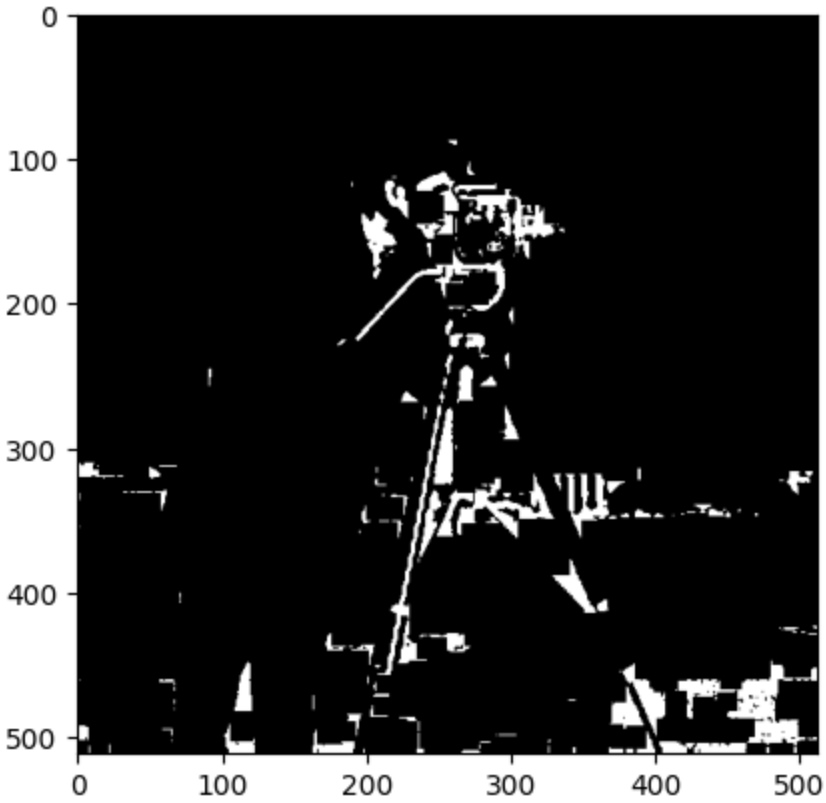
In [ ]:
```
#Morphological Gradient
binr = cv2.threshold(img, 0, 255, cv2.THRESH_BINARY+cv2.THRESH_OTSU)[1]

# define the kernel
kernel = np.ones((3, 3), np.uint8)

# invert the image
invert = cv2.bitwise_not(binr)

# use morph gradient
morph_gradient = cv2.morphologyEx(invert,
                                  cv2.MORPH_GRADIENT,
                                  kernel)

# print the output
plt.imshow(morph_gradient, cmap='gray')
```

Out[ ]:
```
<matplotlib.image.AxesImage at 0x7d8cd70bbee0>
```



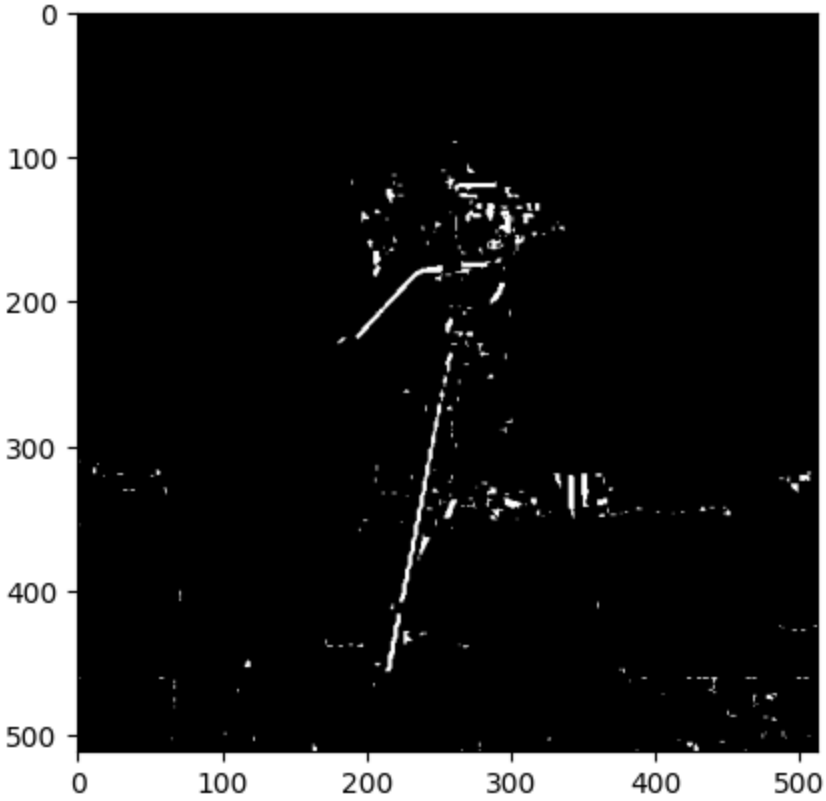
In [ ]:
```
#Top hat
binr = cv2.threshold(img, 0, 255, cv2.THRESH_BINARY+cv2.THRESH_OTSU)[1]

# define the kernel
kernel = np.ones((13, 13), np.uint8)

# use morph gradient
morph_gradient = cv2.morphologyEx(binr,
                                  cv2.MORPH_TOPHAT,
                                  kernel)
# print the output
plt.imshow(morph_gradient, cmap='gray')
```

Out[ ]: <matplotlib.image.AxesImage at 0x7d8cd7136020>



In [ ]:
```
# Black Hat
binr = cv2.threshold(img, 0, 255, cv2.THRESH_BINARY+cv2.THRESH_OTSU)[1]

# define the kernel
kernel = np.ones((5, 5), np.uint8)

# invert the image
invert = cv2.bitwise_not(binr)

# use morph gradient
morph_gradient = cv2.morphologyEx(invert,
                                  cv2.MORPH_BLACKHAT,
                                  kernel)
# print the output
plt.imshow(morph_gradient, cmap='gray')
```

Out[ ]:
```
<matplotlib.image.AxesImage at 0x7d8cd916cdf0>
```


0
100
200
300
400
500
0
100
200
300
400
500