```
import os
print(os.listdir())  # List all files in the current directory
```

```
['.config', 'sample_data']
```

```
from google.colab import files

uploaded = files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
import os
print(os.listdir())
```

```
['.config', 'sample_data', 'Flower.png']
```

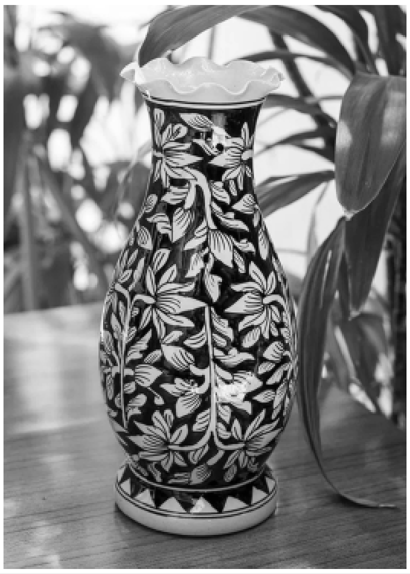
```
import cv2
import matplotlib.pyplot as plt

img = cv2.imread('Flower.png')

if img is None:
    print("Error: Image not found or unsupported format.")
else:

    plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
    plt.title('Loaded Image')
    plt.axis('off')
    plt.show()
```

Loaded Image

```
from google.colab import drive
drive.mount('/content/drive')
```

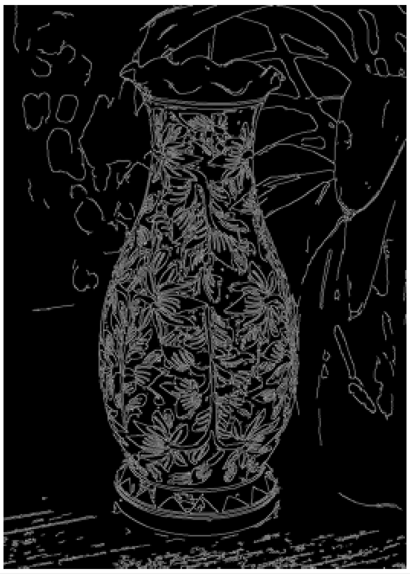
```
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
plt.imshow(gray, cmap='gray')
plt.title('Grayscale Image')
plt.axis('off')
plt.show()
```

Grayscale Image



```
edges = cv2.Canny(gray, threshold1=100, threshold2=200)
plt.imshow(edges, cmap='gray')
plt.title('Edges Detected (Canny)')
plt.axis('off')
plt.show()
```

Edges Detected (Canny)

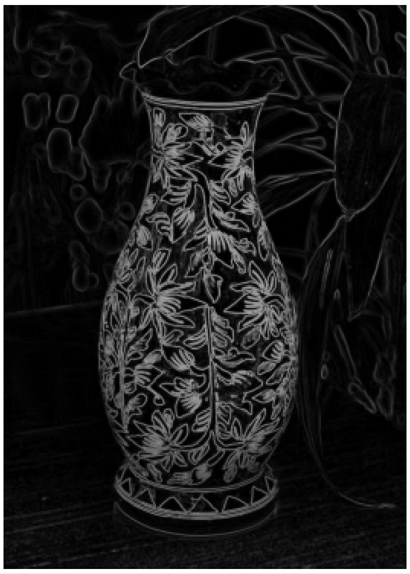


```
cv2.imwrite('edges_detected.png', edges)
```

```
True
```

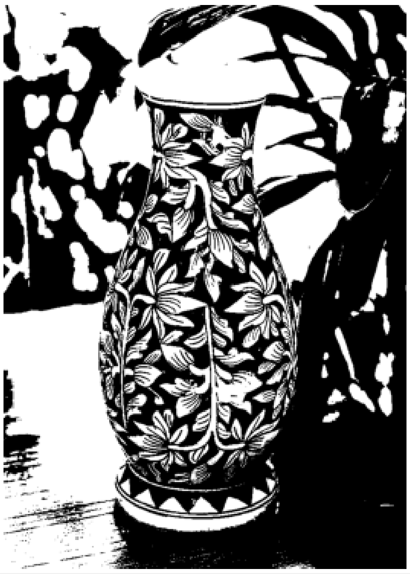
```
sobelx = cv2.Sobel(gray, cv2.CV_64F, 1, 0, ksize=3)  # Horizontal edges
sobely = cv2.Sobel(gray, cv2.CV_64F, 0, 1, ksize=3)  # Vertical edges
sobel_combined = cv2.magnitude(sobelx, sobely)
plt.imshow(sobel_combined, cmap='gray')
plt.title('Edges Detected (Sobel)')
plt.axis('off')
plt.show()
```

Edges Detected (Sobel)



```
ret, binary = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)
plt.imshow(binary, cmap='gray')
plt.title("Binary Image (Otsu's Thresholding)")
plt.axis('off')
plt.show()
```

Binary Image (Otsu's Thresholding)



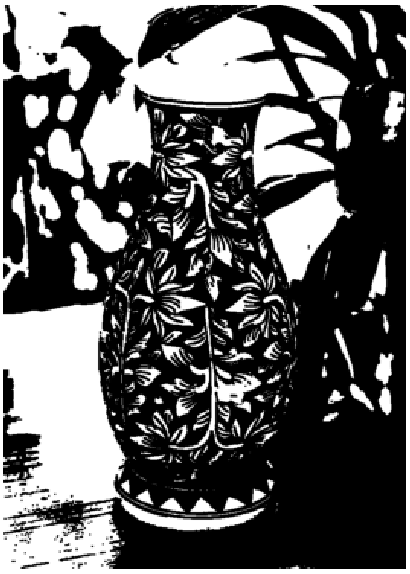
```
cv2.imwrite('binary_image_otsu.png', binary)
```

```
True
```

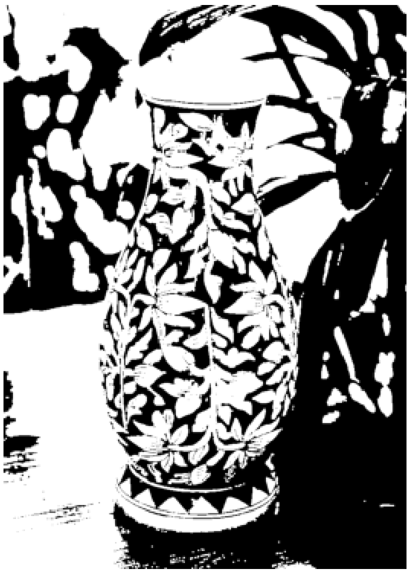
```
import numpy as np
# Define a kernel (3x3 matrix)
kernel = np.ones((3, 3), np.uint8)

# Apply erosion
erosion = cv2.erode(binary, kernel, iterations=1)

# Display the eroded image
plt.imshow(erosion, cmap='gray')
plt.title("Eroded Image")
plt.axis('off')
plt.show()
```

```
# Apply dilation
dilation = cv2.dilate(binary, kernel, iterations=1)

# Display the dilated image
plt.imshow(dilation, cmap='gray')
plt.title("Dilated Image")
plt.axis('off')
plt.show()

# Apply dilation after erosion (closing operation)
closing = cv2.dilate(erosion, kernel, iterations=1)

# Display the result
plt.imshow(closing, cmap='gray')
plt.title("Erosion followed by Dilation (Closing)")
plt.axis('off')
plt.show()
```

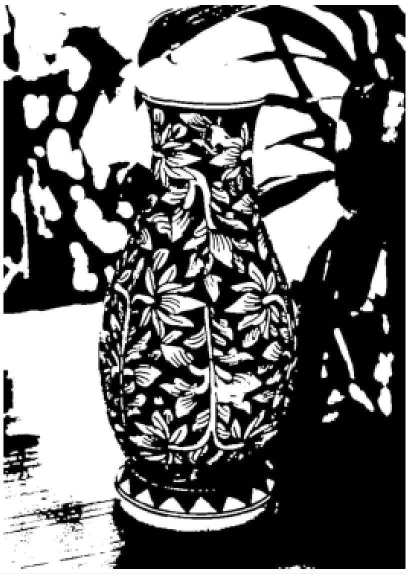
Eroded Image



Dilated Image



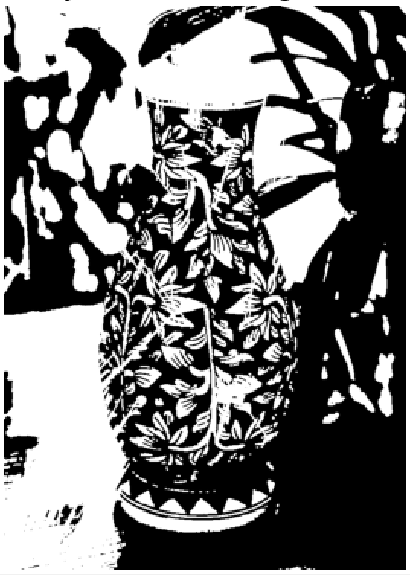
Erosion followed by Dilation (Closing)



```
lines = cv2.HoughLinesP(closing, 1, np.pi / 180, threshold=50, minLineLength=100, maxLineGap=10)
output_image = np.copy(closing)
if lines is not None:
    for line in lines:
        x1, y1, x2, y2 = line[0]
```

```
            cv2.line(output_image, (x1, y1), (x2, y2), (255, 0, 0), 2)  # Blue color with thickness=2
    plt.imshow(output_image, cmap='gray')
    plt.title("Boundary Detection (Hough Transform)")
    plt.axis('off')
    plt.show()
```

Boundary Detection (Hough Transform)



```
cv2.imwrite('boundary_detected.png', output_image)
```