Pattern Recognition - image processing with OpenCV

Ziming Wang(19814)

CS455 - Algorithms & Structured Programming

1. Project subject

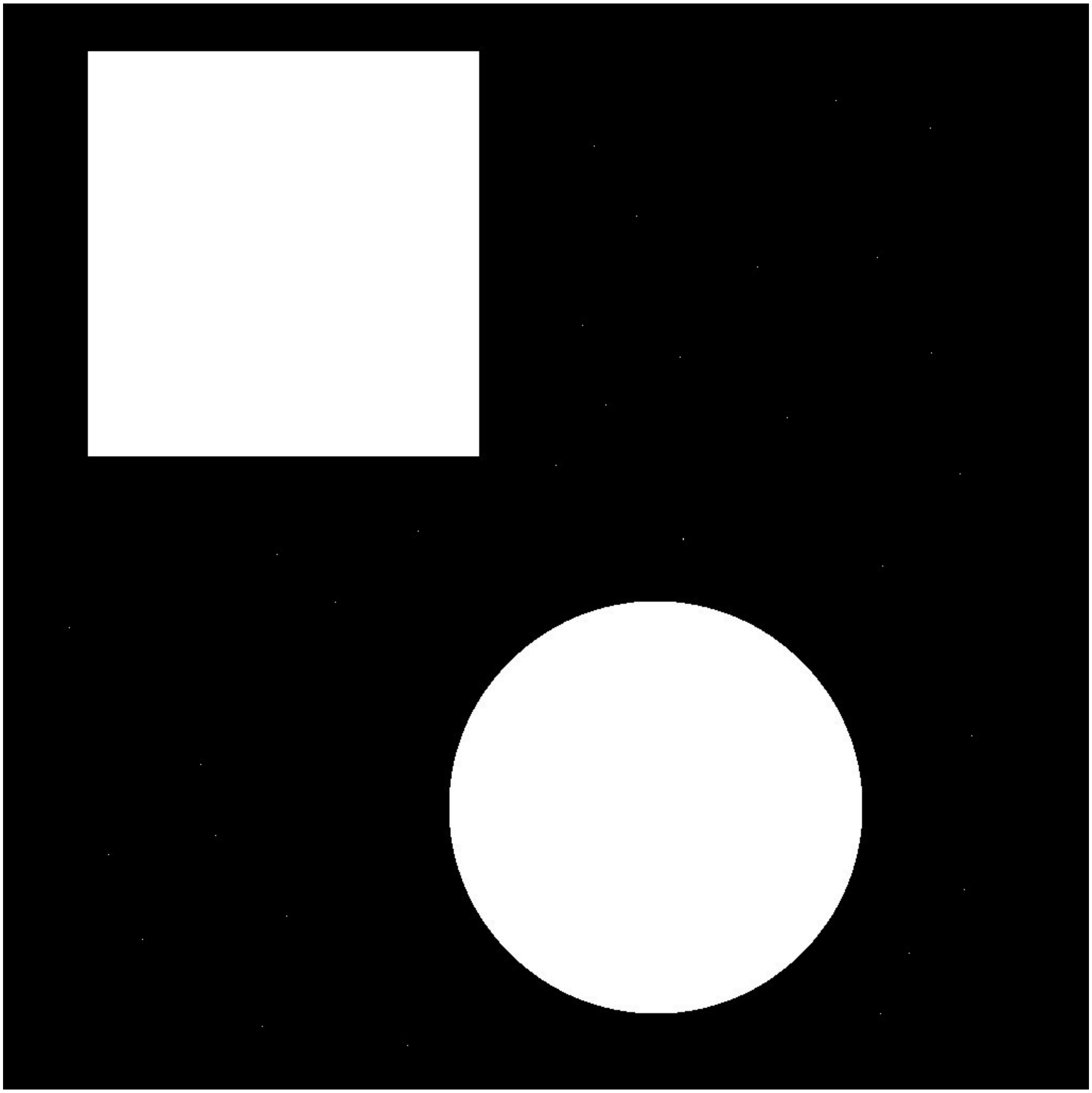
2. Design workflow

3. Code implementation

4. Conclusion

1. Project Subject:

Design and implement a program that reduce the noise following picture and recognize the circle and square shapes.



2. Design workflow

    2.1. Noise reduction by blurring (with gaussian filter)

    2.2. Histogram (determine the value of thresholding)

    2.3. Thresholding (with Otsu’s thresholding)

    2.4. Connectivity analysis

    2.5. Pattern recognition (with OpenCV API)

3. Code implementation

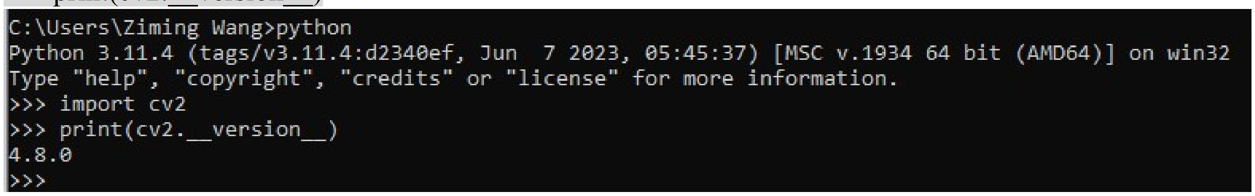
3.1 Install OpenCV package

a. Open terminal and type:

```
pip install numpy
pip install opencv-python
```

b. Check if OpenCV is correctly installed:

```
python
>>>import cv2
>>>print(cv2.__version__)
```



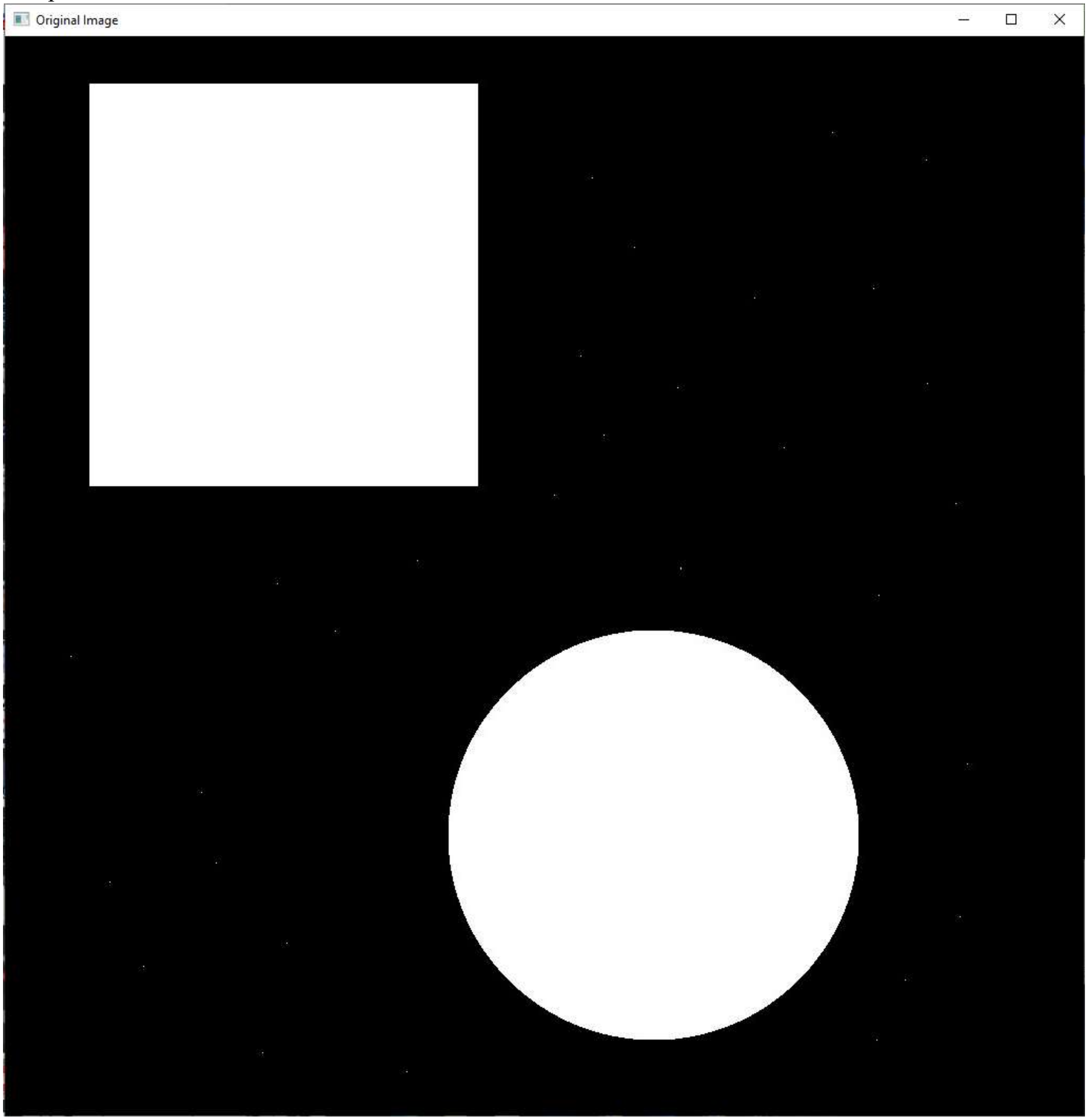
```
C:\Users\Ziming Wang>python
Python 3.11.4 (tags/v3.11.4:d2340ef, Jun  7 2023, 05:45:37) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import cv2
>>> print(cv2.__version__)
4.8.0
>>>
```

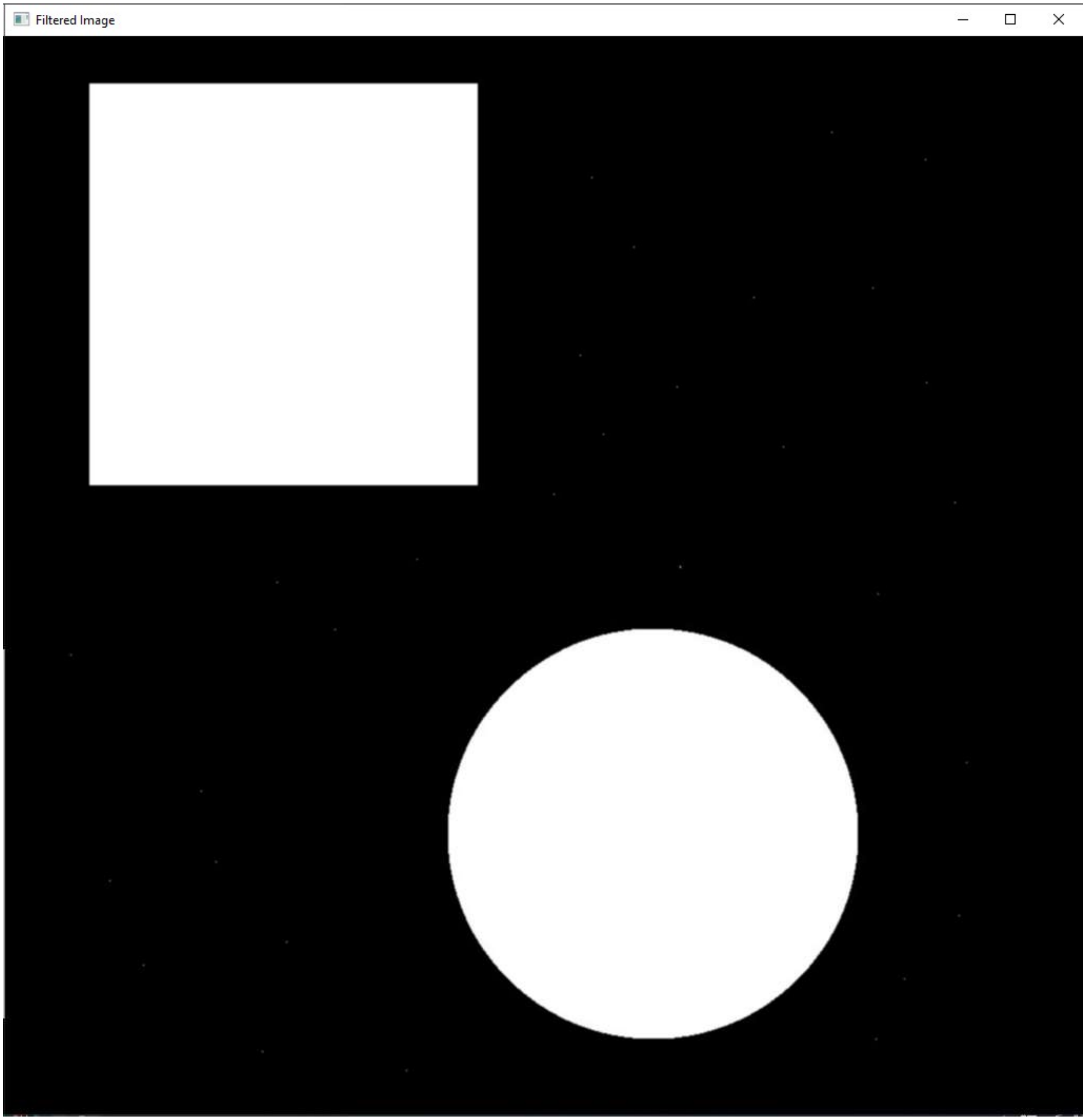
3.2 Download the image into the project folder.

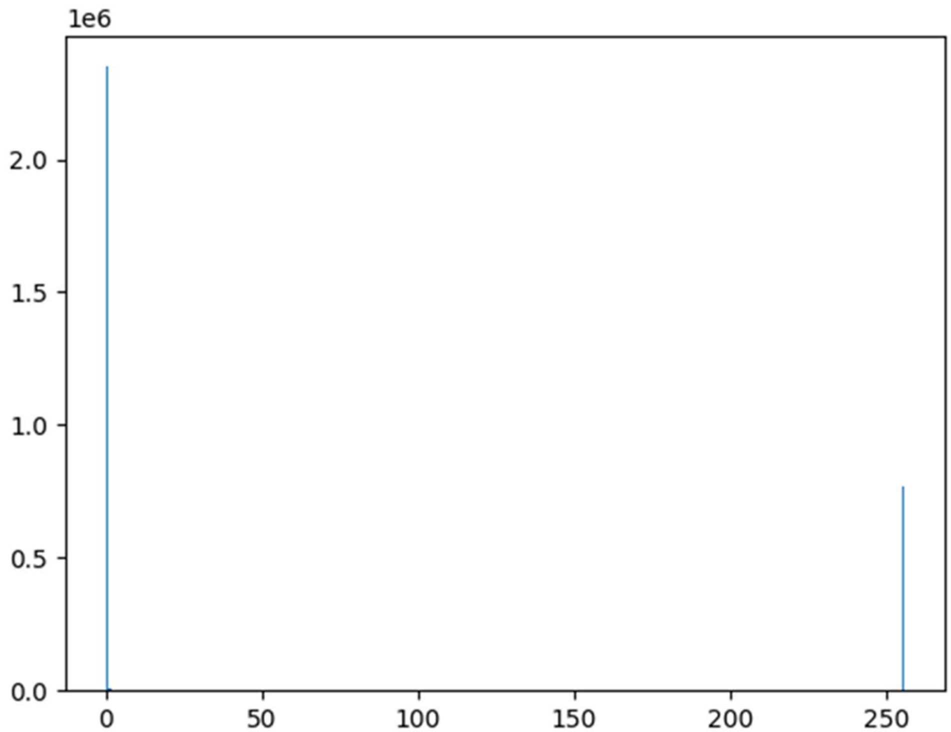
3.3 Create pattern.py and create a code to plot histogram with Thresholding.

```
import cv2
import numpy as np

# Load the image
image = cv2.imread('your_image.jpg')

# Apply a 3x3 Gaussian filter
filtered_image = cv2.GaussianBlur(image, (3, 3), 0)

# Show the histogram
plt.hist(image.ravel(),256,[0,256])
plt.show()

# Display the original and filtered images
cv2.imshow('Original Image', image)
cv2.imshow('Filtered Image', filtered_image)

# Wait for a key press and then close the windows
cv2.waitKey(0)
cv2.destroyAllWindows()
```

#output




Filtered Image



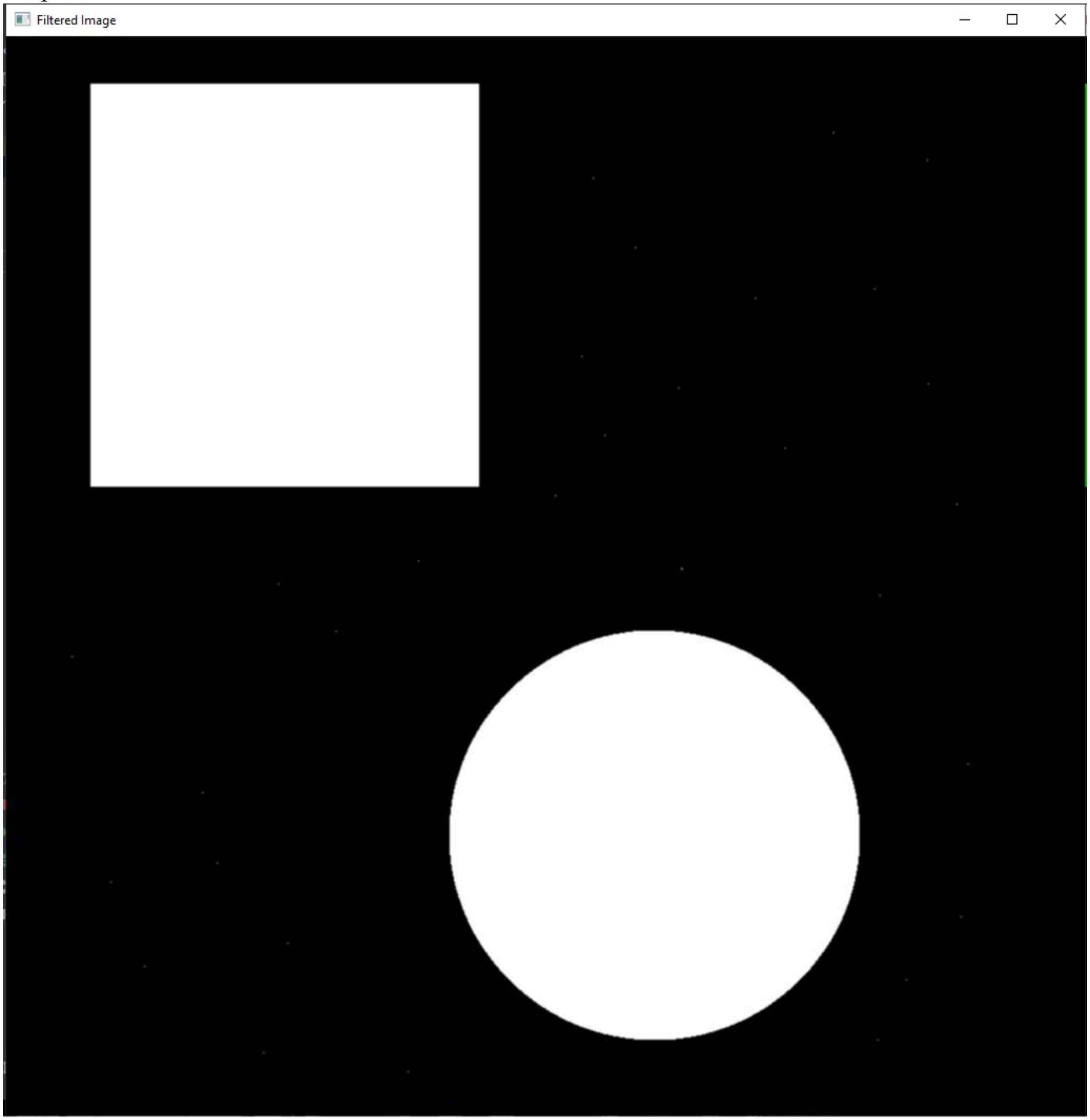
1e6
2.0
1.5
1.0
0.5
0.0
0
50
100
150
200
250

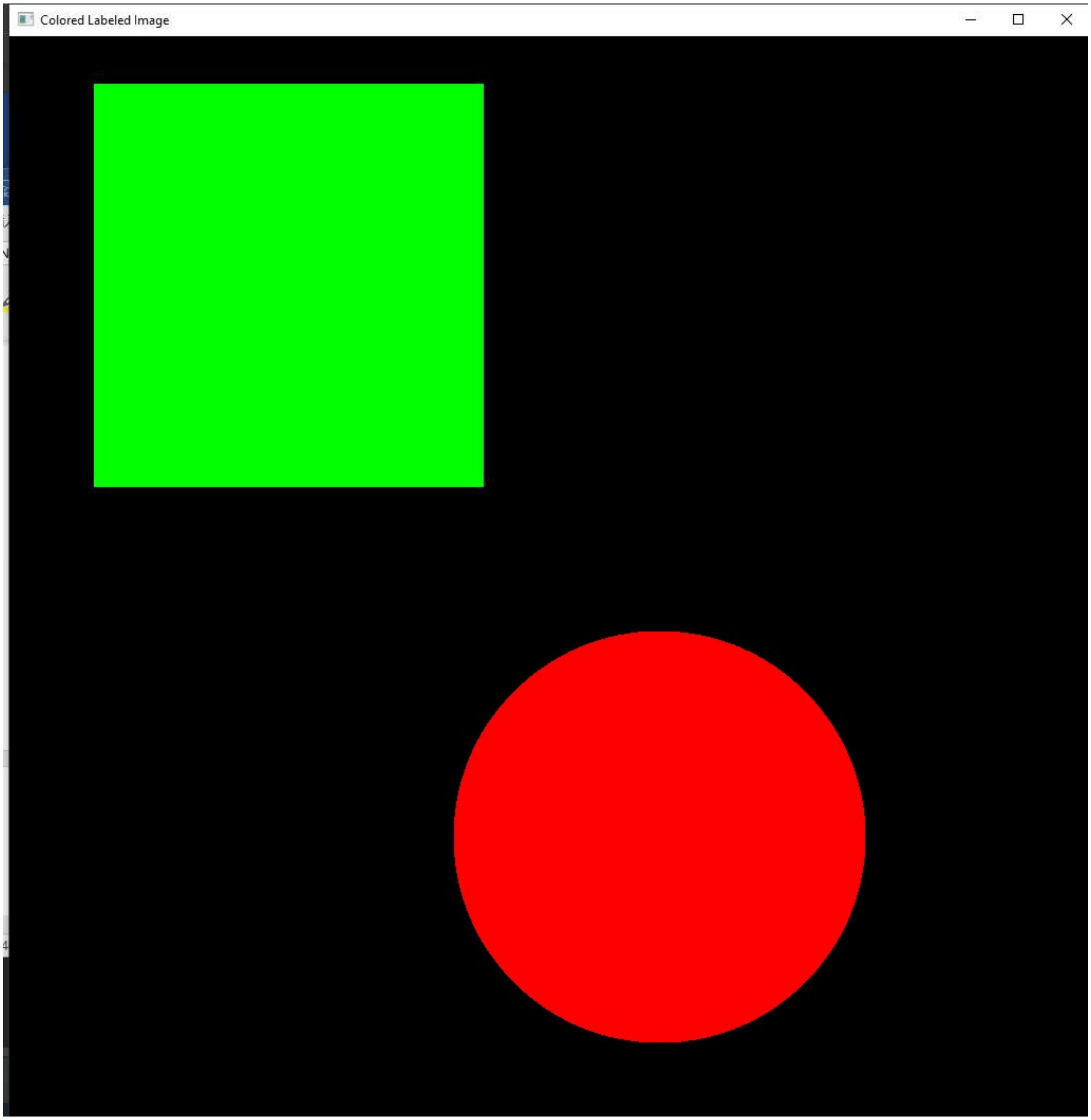
3.4 Connectivity analysis

This step distinguishes individual objects in the image by using 4- or 8-pixels connectivity. 4 pixels connect pixels with the same value along the edge. 8 pixels the same applies to adding edges and corners.

```
import cv2
import numpy as np

# Load the original image
original_image = cv2.imread('e_noise.jpg', cv2.IMREAD_GRAYSCALE)

# Apply a 3x3 Gaussian filter
filtered_image = cv2.GaussianBlur(original_image, (3, 3), 0)

# Threshold the filtered image to create a binary image
_, binary_image = cv2.threshold(filtered_image, 128, 255, cv2.THRESH_BINARY)

# Find connected components and label them
num_labels, labeled_image = cv2.connectedComponents(binary_image)

# Create a blank image with three channels to draw colored labels
colored_labels = np.zeros((labeled_image.shape[0], labeled_image.shape[1], 3), dtype=np.uint8)

# Assign colors to different labels
for label in range(1, num_labels):
    color = (0, 0, 0)  # Initialize with black
    if label % 2 == 0:
        color = (0, 0, 255)  # Blue for even labels
    else:
        color = (0, 255, 0)  # Green for odd labels

    # Create a mask for the current label and set the color
    mask = labeled_image == label
    colored_labels[mask] = color

# Display the original image, filtered image, and the colored labeled image
cv2.imshow('Original Image', original_image)
cv2.imshow('Filtered Image', filtered_image)
cv2.imshow('Colored Labeled Image', colored_labels)

# Wait for a key press and then close the windows
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Output:




Colored Labeled Image

3.5 Pattern recognition

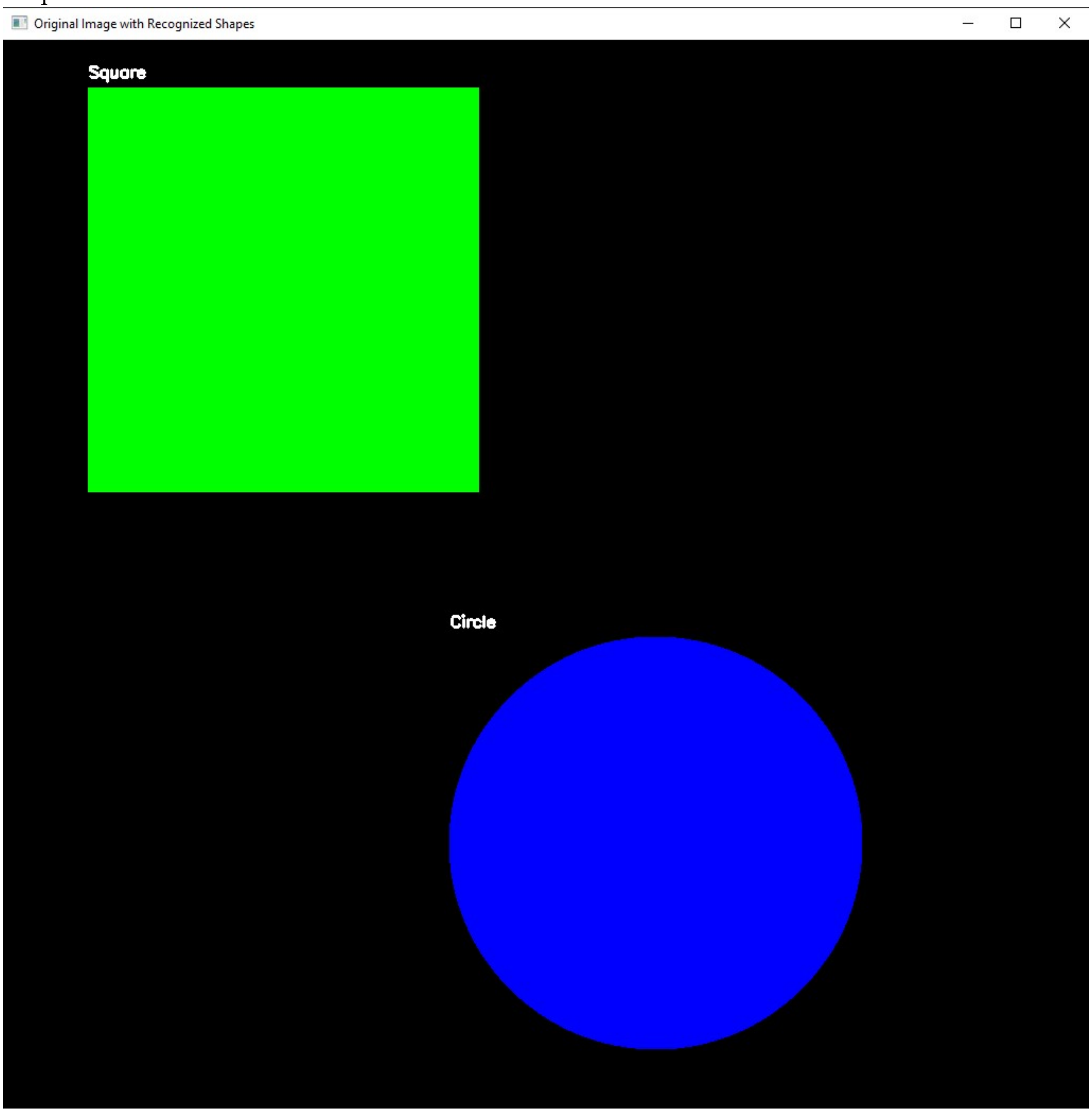
```
import cv2
import numpy as np

# Load the original image
original_image = cv2.imread('e_noise.jpg', cv2.IMREAD_COLOR)

# Apply a 3x3 Gaussian filter
filtered_image = cv2.GaussianBlur(original_image, (3, 3), 0)

# Convert the filtered image to grayscale for contour detection
filtered_gray = cv2.cvtColor(filtered_image, cv2.COLOR_BGR2GRAY)

# Threshold the filtered grayscale image to create a binary image
_, binary_image = cv2.threshold(filtered_gray, 128, 255, cv2.THRESH_BINARY)

# Find contours in the binary image
contours, _ = cv2.findContours(binary_image, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)

# Create a blank image with three channels to draw colored labels
colored_labels = np.zeros_like(original_image)

# Initialize a counter for different shapes
shape_counter = 0

# Process each contour
for contour in contours:
    # Approximate the contour to a polygon
    epsilon = 0.04 * cv2.arcLength(contour, True)
    approx = cv2.approxPolyDP(contour, epsilon, True)

    # Define colors for different shapes
    if len(approx) == 3:
        color = (0, 0, 255)  # Red for triangles
    elif len(approx) == 4:
        color = (0, 255, 0)  # Green for rectangles
    elif len(approx) >= 5:
        color = (255, 0, 0)  # Blue for circles and other shapes
    else:
        color = (0, 0, 0)    # Black (undefined shape)

    # Draw the contour and label with the corresponding color
    cv2.drawContours(colored_labels, [contour], -1, color, -1)

# Display the original image with recognized shapes
cv2.imshow('Original Image with Recognized Shapes', colored_labels)
```

```
# Wait for a key press and then close the window
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Explain:

This code loads an image, applies a Gaussian filter to reduce noise, and converts it to grayscale. It then detects shapes (triangles, squares, circles) through contour detection. Shapes are drawn with corresponding colors and labeled. The labeled image is displayed and waits for user interaction.

Output:



4. Conclusion

   This code leverages OpenCV, a powerful computer vision library, to perform image processing and shape recognition tasks. It utilizes techniques such as Gaussian filtering, contour detection, and shape classification to identify and label shapes within an image. OpenCV was chosen for its extensive capabilities in image processing, making it a robust choice for tasks like these, allowing for efficient and accurate shape recognition and labeling.

Reference

Histogram

https://opencv-pythontutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_histograms/py_histogram_begins/py_histogram_begins.html

https://docs.opencv.org/4.x/d5/de5/tutorial_py_setup_in_windows.html

Threshold

https://opencv-pythontutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_thresholding/py_thresholding.html

https://docs.opencv.org/master/d7/d1b/group__imgproc__misc.html#ggaa9e58d2860d4afa658ef70a9b1115576a147222a96556ebc1d948b372bcd7ac59

Connectivity analysis

https://iq.opengenus.org/connected-component-labeling/

https://github.com/yashml/OpenGenus_Articles_Code/tree/master/Connected%20Component%20

Labeling Pattern recognition

https://pysource.com/2018/09/25/simple-shape-detection-opencv-with-python-3/