Name: Raj Koyani

Reg no:21MIS1017

Subject: DIP

DA:3

Dataset:(26) Adaptive + Gaussian Thresholding

Mathematical Equation used:

For each pixel (x, y) in the image, I, the adaptive Gaussian thresholding calculates the threshold T(x, y) using the local statistics around that pixel:

T(x, y) = mean(x, y) - C

where:

mean(x, y) is the local mean calculated in a neighbourhood around the pixel (x, y).

C is a constant that can be adjusted to control the thresholding.

The local mean is calculated using a weighted sum of pixel intensities in a neighborhood around each pixel. Gaussian weighting is used to give more weight to the central pixels and less weight to the pixels farther away from the center. The neighborhood size is defined by the blockSize parameter in OpenCV's cv2.adaptiveThreshold function.

The formula for the local mean in the Gaussian weighted neighborhood is:

mean(x, y) = (1 / (blockSize^2)) * ∑(w(i, j) * I(i, j))

where:

(i, j) are the coordinates of each pixel in the neighborhood.

I(i, j) is the intensity of the pixel at position (i, j) in image I.

blockSize is the size of the neighborhood (block size) used for local mean calculation.

w(i, j) is the Gaussian weight for each pixel in the neighborhood.

The Gaussian weight w(i, j) for each pixel (i, j) in the neighborhood is computed as:

w(i, j) = exp(-((i - (block Size - 1) / 2)^2 + (j - (block Size - 1) / 2)^2) / (2 * sigma^2))

where:

sigma is a parameter that controls the spread of the Gaussian function. It can be adjusted based on the desired sensitivity of the thresholding.

By subtracting the constant C from the local mean, we obtain the adaptive threshold T(x, y) for each pixel. Finally, pixels with intensity values greater than or equal to the threshold T(x, y) are set to 255 (white) in the binary output image, while pixels with intensity values less than the threshold are set to 0 (black).

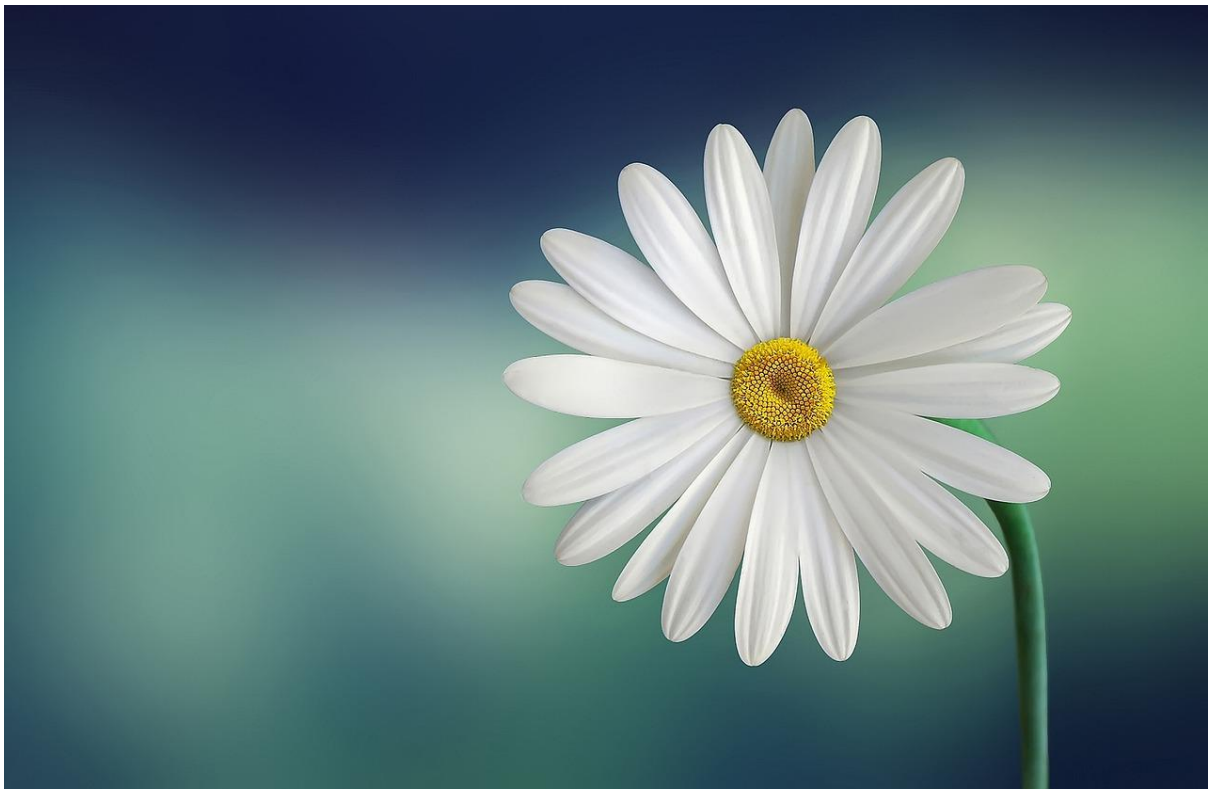
Code:

```
import cv2
import numpy as np

# Read the input image
image_path = ("C:/Users/rajko/Downloads/flower.jpg")
I = cv2.imread(image_path)

# Convert the image to grayscale if it is RGB
if I.shape[-1] == 3:
    I = cv2.cvtColor(I, cv2.COLOR_BGR2GRAY)

# Perform Adaptive Gaussian Thresholding
binaryImage = cv2.adaptiveThreshold(I, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
cv2.THRESH_BINARY, 11, 2)

# Display the original and binary images
cv2.imshow('Original Image', I)
cv2.imshow('Adaptive Gaussian Thresholding', binaryImage)

# Save the binary image if needed
cv2.imwrite('binary_image.png', binaryImage)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

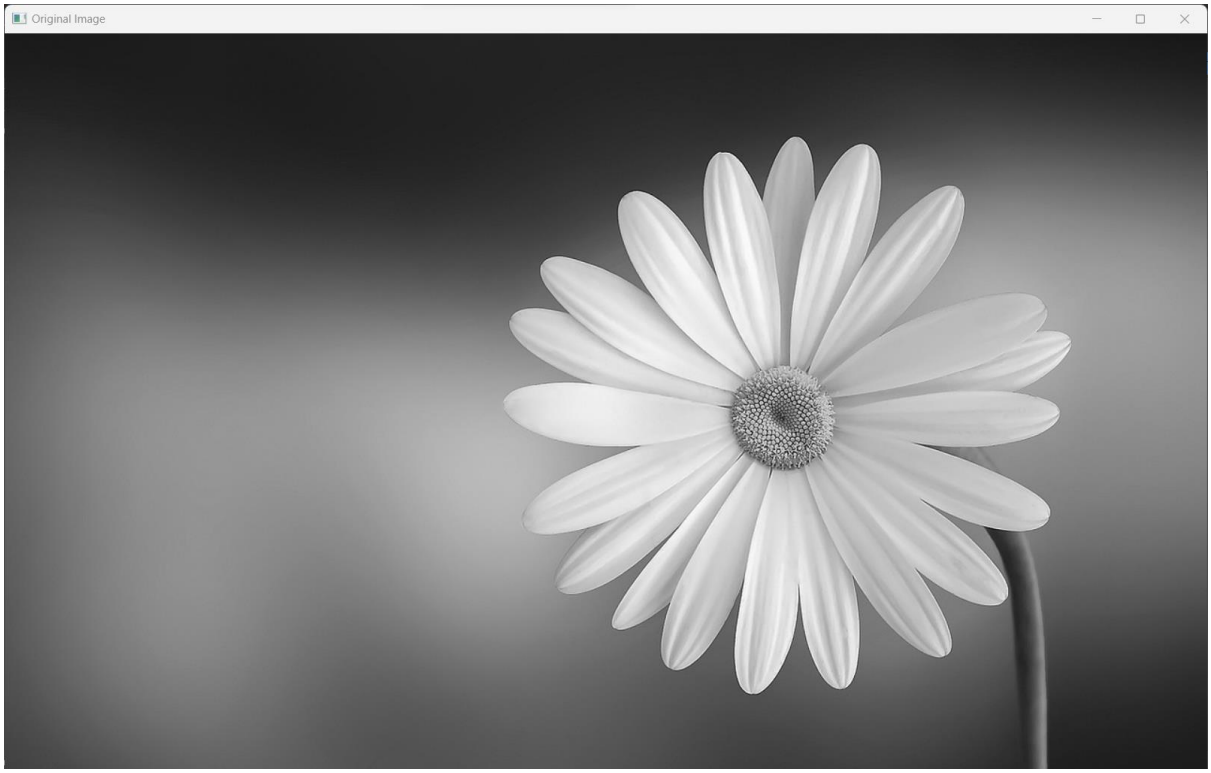
```python
import cv2
import numpy as np

# Read the input image
image_path = ("C:/Users/rajko/Downloads/flower.jpg")
I = cv2.imread(image_path)

# Convert the image to grayscale if it is RGB
if I.shape[-1] == 3:
    I = cv2.cvtColor(I, cv2.COLOR_BGR2GRAY)

# Perform Adaptive Gaussian Thresholding
binaryImage = cv2.adaptiveThreshold(I, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY, 11, 2)

# Display the original and binary images
cv2.imshow('Original Image', I)
cv2.imshow('Adaptive Gaussian Thresholding', binaryImage)

# Save the binary image if needed
cv2.imwrite('binary_image.png', binaryImage)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

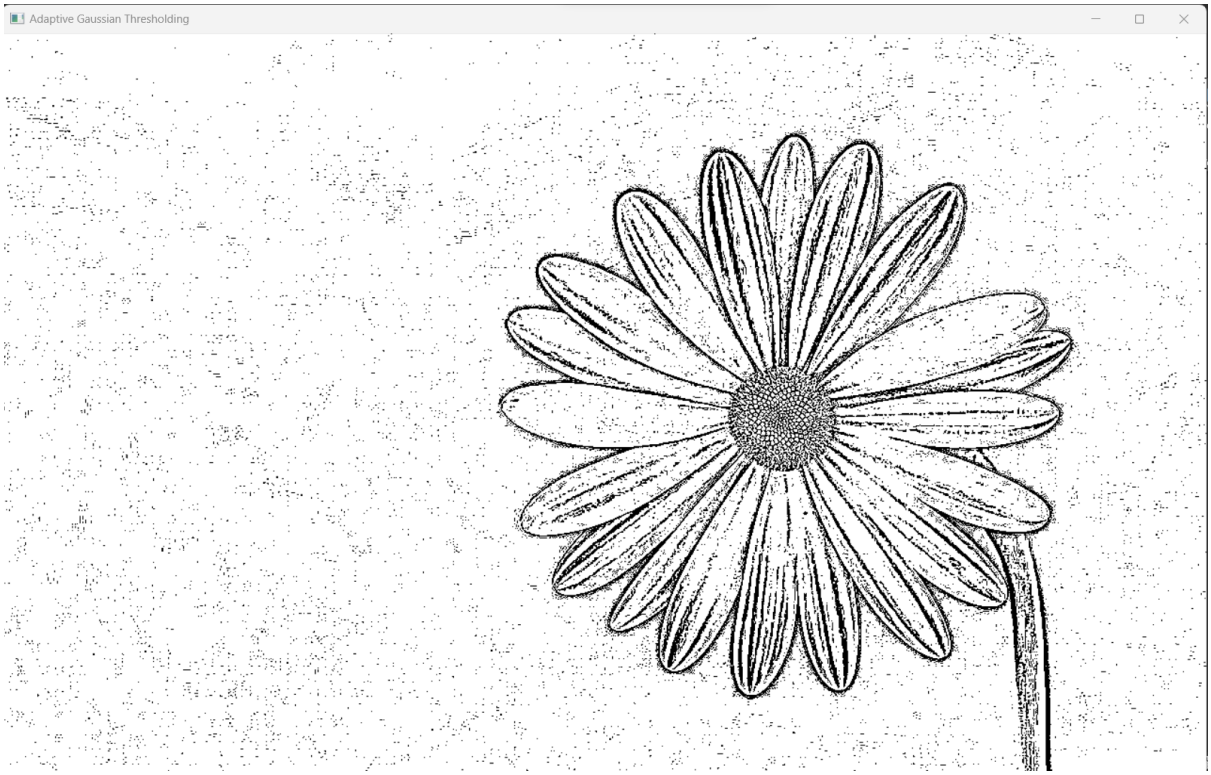
```
gpy\launcher' '51091' '--' 'C:\Users\rajko\Downloads\5a.py'
PS C:\Users\rajko\Downloads>  c:; cd 'c:\Users\rajko\Downloads'; & 'C:\Users\rajko\AppData\Local\Microsoft\WindowsApps\python3.11.exe' 'c:\Users\rajko\.vscod
e\extensions\ms-python.python-2023.12.0\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\launcher' '51843' '--' 'C:\Users\rajko\Downloads\4a.py'
PS C:\Users\rajko\Downloads>  c:; cd 'c:\Users\rajko\Downloads'; & 'C:\Users\rajko\AppData\Local\Microsoft\WindowsApps\python3.11.exe' 'c:\Users\rajko\.vscod
e\extensions\ms-python.python-2023.12.0\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\launcher' '51855' '--' 'C:\Users\rajko\Downloads\4a.py'
```

Input Image:



Output:


Original Image


Adaptive Gaussian Thresholding