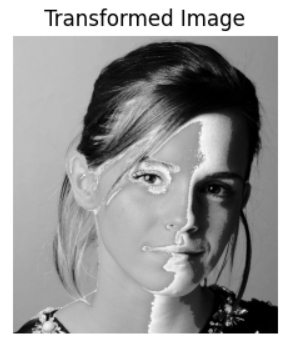
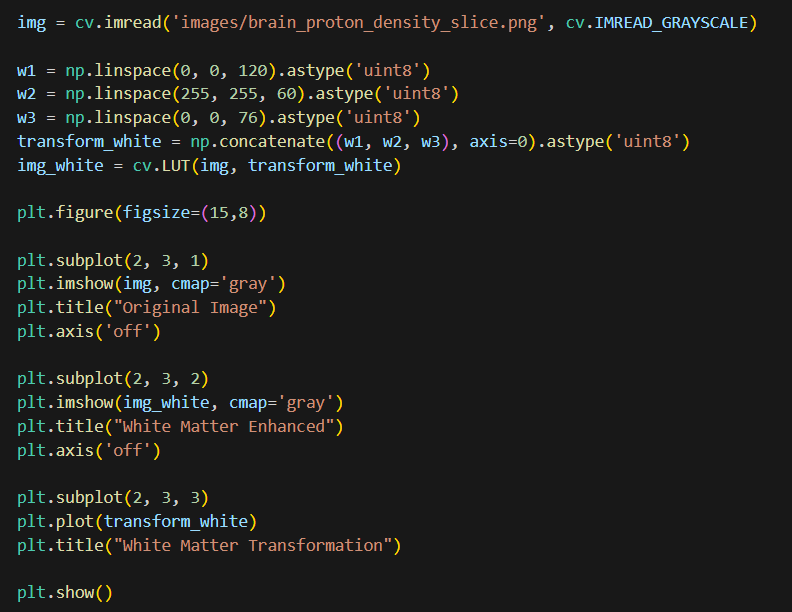
Index No: 249321V  
Name: HDPR Perera   
GitHub project link: <https://github.com/pramodiperera/IT5437>

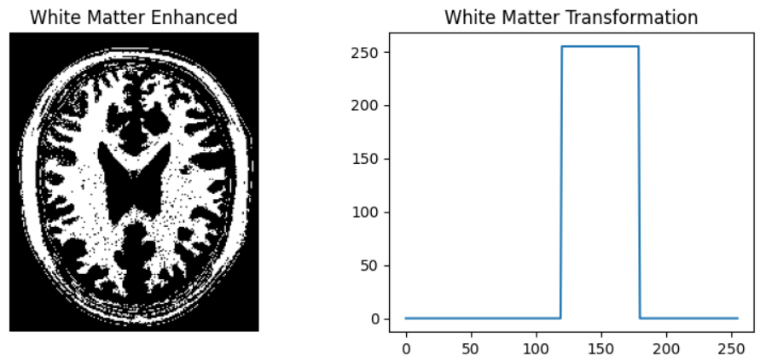
**Question 01**  
Input intensity pixels are grouped as (0 – 49 , 50 – 149, 150 - 255)



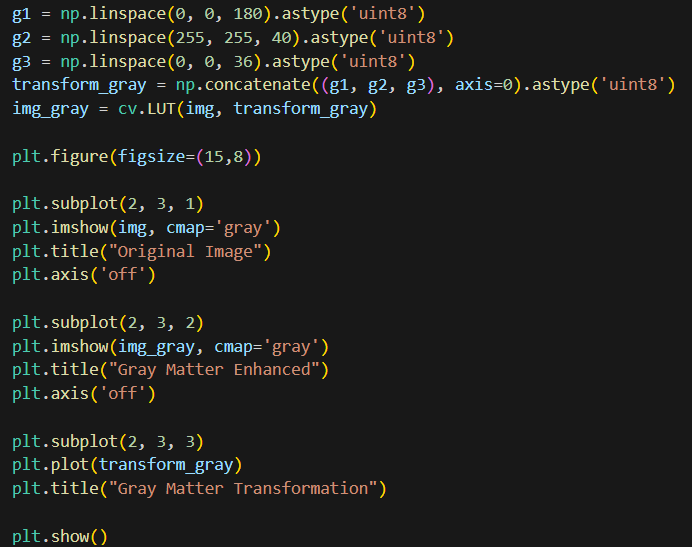
**Question 02**

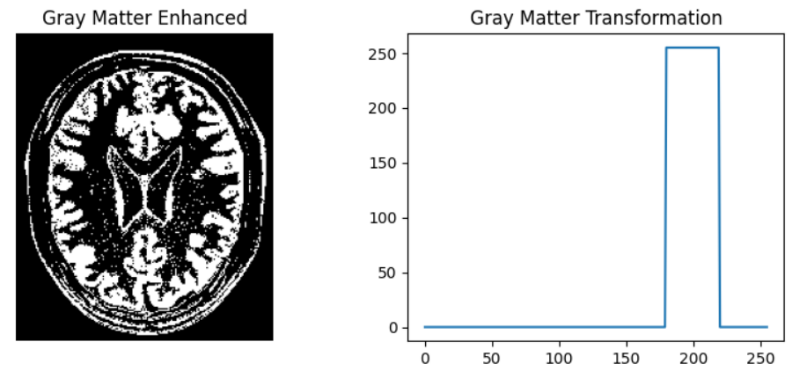
1. White matter

According to the original image, approximately pixel intensities from 120 to 180 belongs to white matter. To accentuate white matter, it is shown in white colour (maximum intensity) and other part pixels shown in black colour.

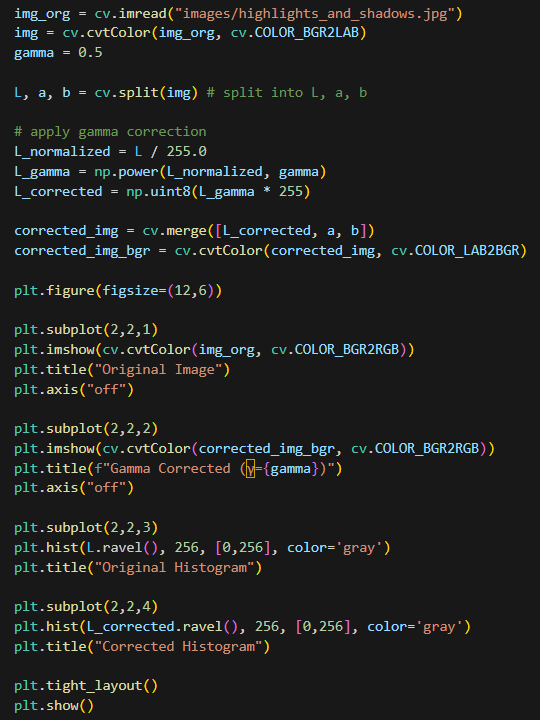


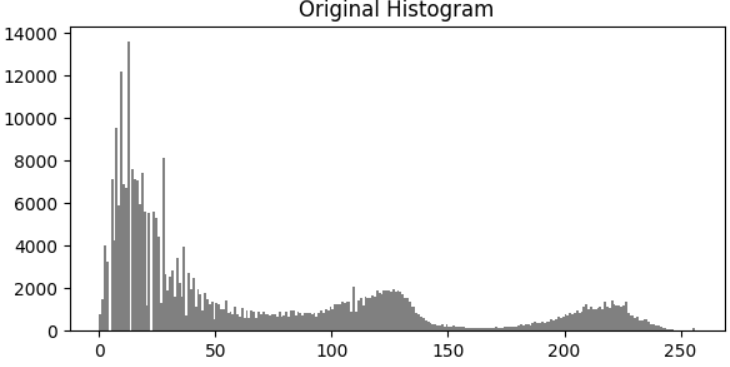
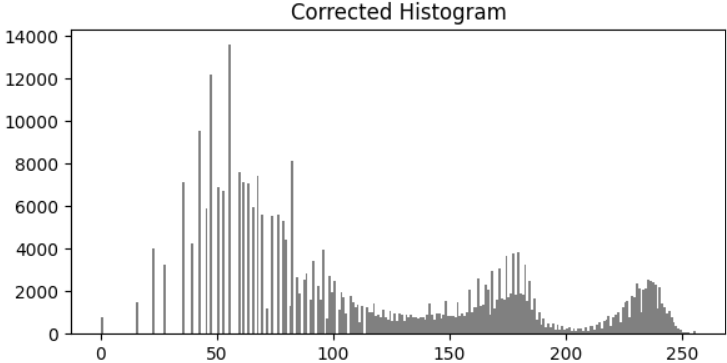
(b) Gray matter

 According to the original image, approximately pixel intensities from 180 to 220 belongs to gray matter. To accentuate gray matter, it is shown in white colour(maximum intensity) and other part pixels shown in black colour.

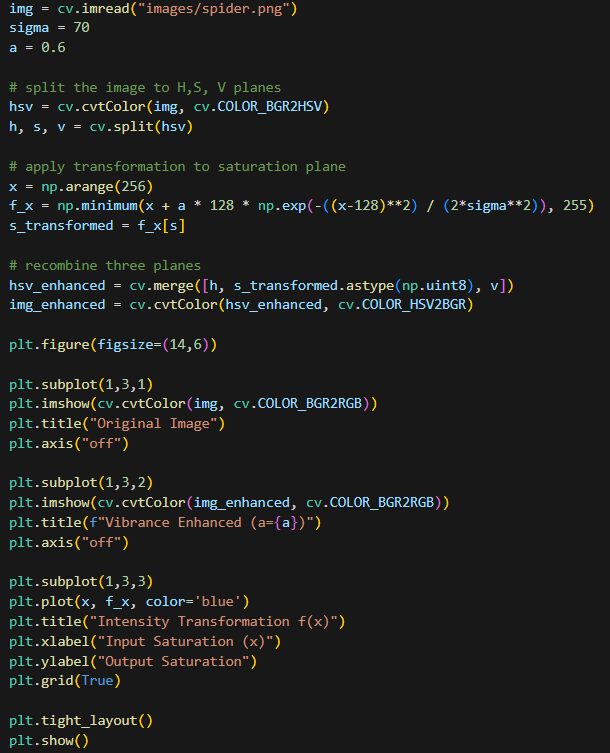


**Question 03**

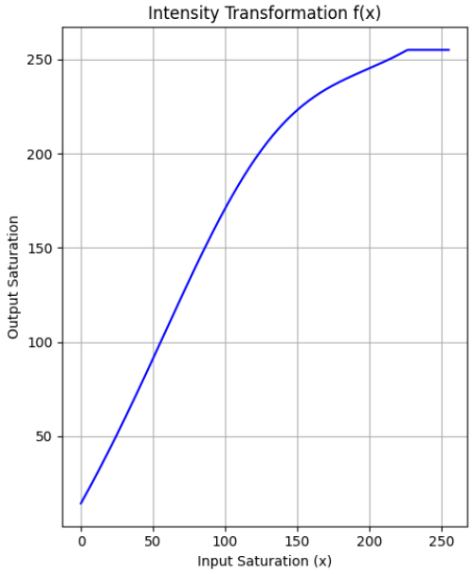




**Question 04**

****a = 0.6

****

****

**Question 05**

(a) Display hue, saturation, and values of the image in grayscale.

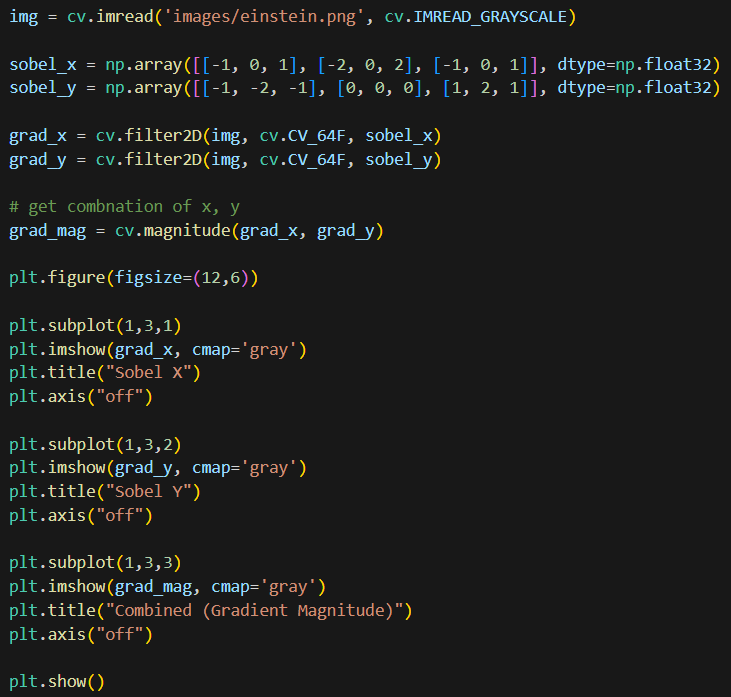
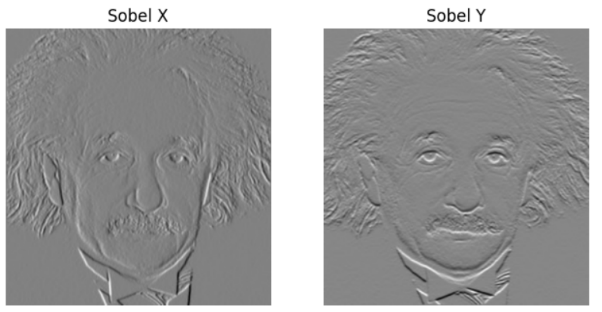


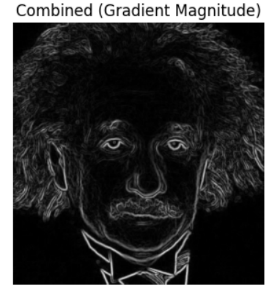
(b)

**Question 06**

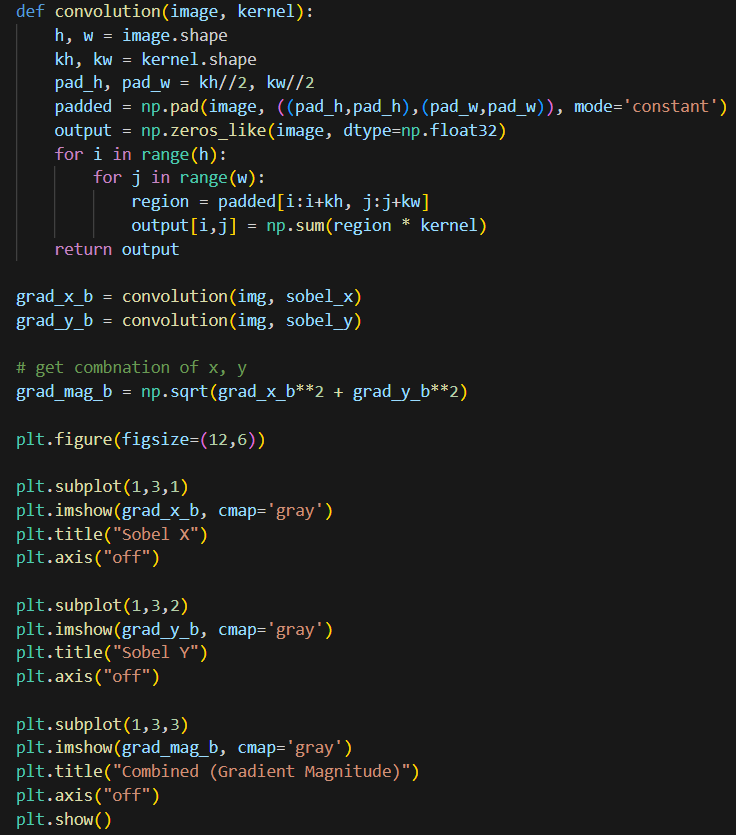
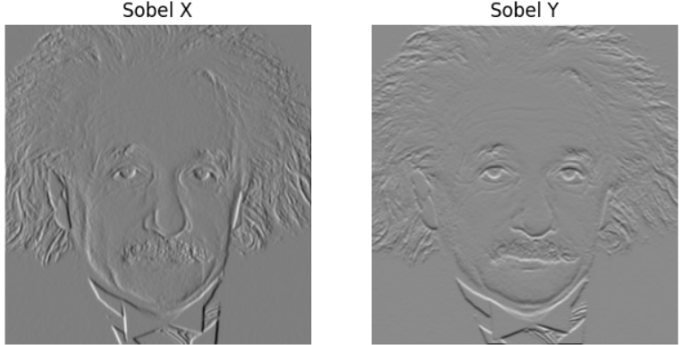
1. Using the existing filter2D to Sobel filter

Additionally, magnitude was calculated by combining gradient x and gradient y.   
magnitude (i, j) = ​



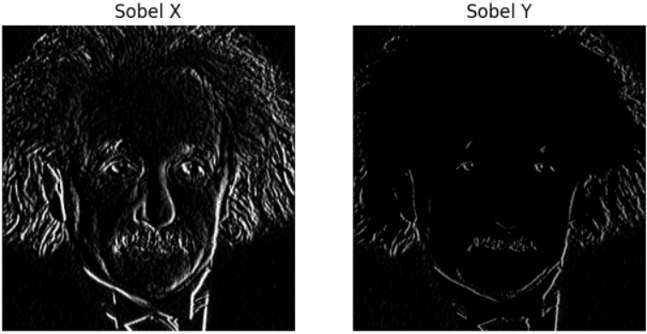
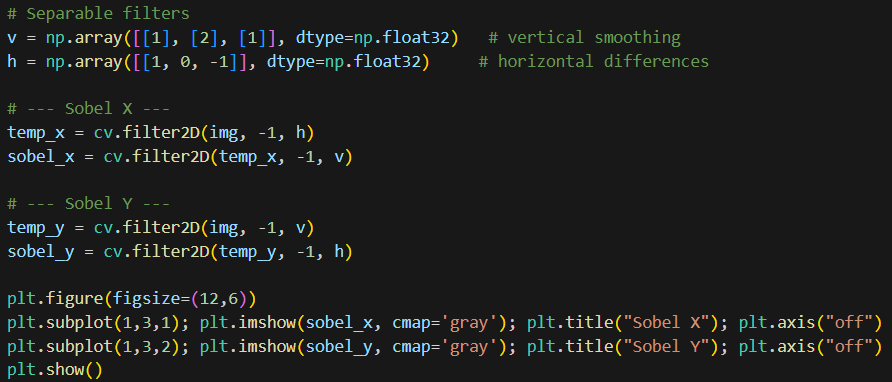


1. Using own code



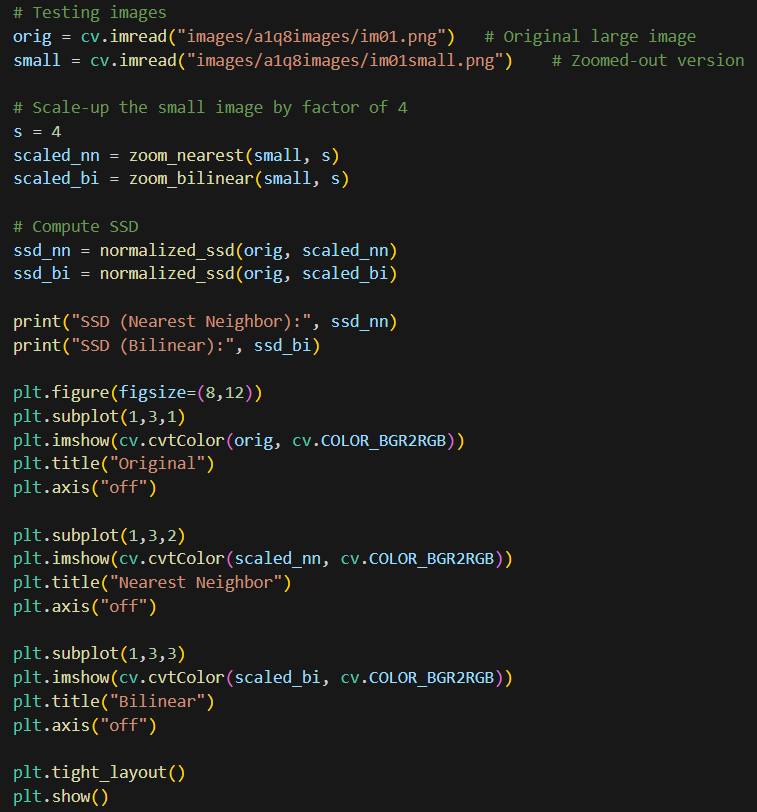
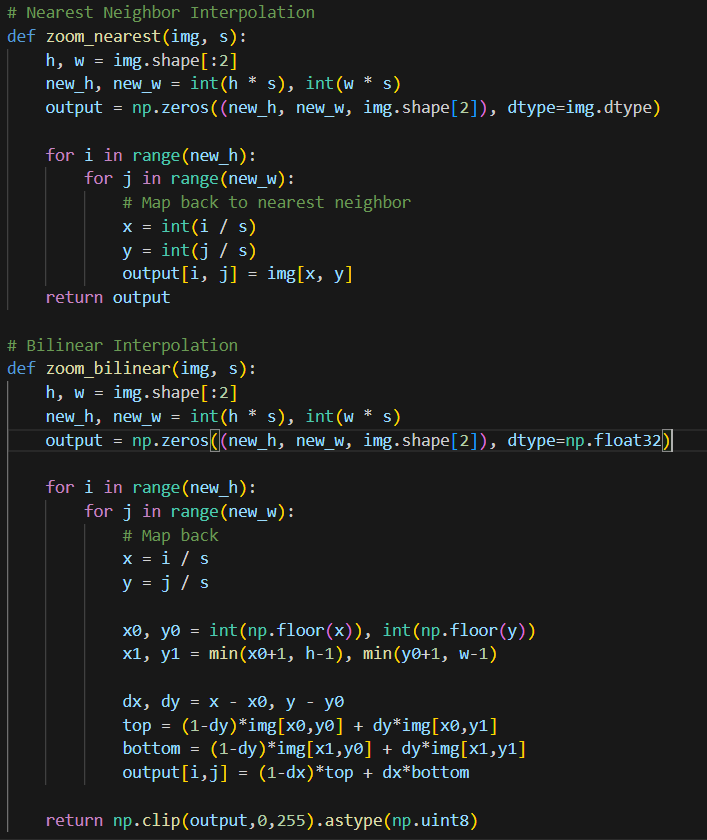


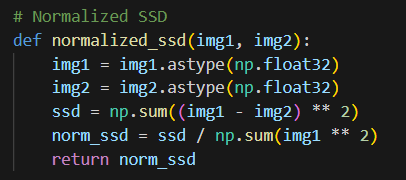
1. Using separability property

****The Sobel operator can be separated into a derivative filter ([1, 0, -1]) and a smoothing filter ([1, 2, 1]). This reduces the 2D convolution into two 1D steps.

**Question 07**

Left side code snippets includes functions to zoom images using nearest-neighbor, bilinear interpolation methods and find the normalized sum of squared difference (SSD). Right code snippet includes test images for zooming and calculate SSD.

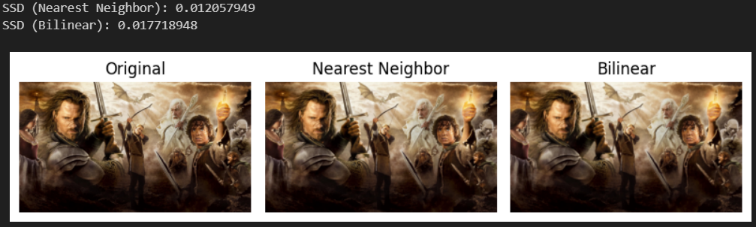
****



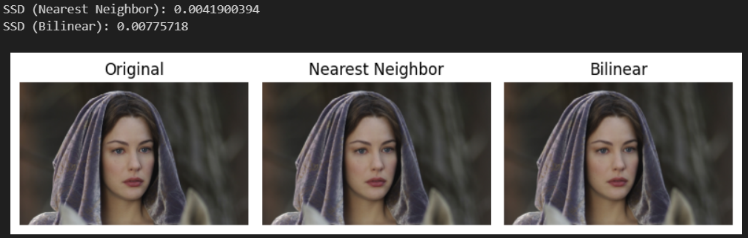
Outputs**:**

The normalized SSD values indicate how closely the upscaled images match the original.

Nearest-neighbor interpolation preserves exact pixel values but appears blocky, while bilinear interpolation produces smoother results with slightly higher error.

****

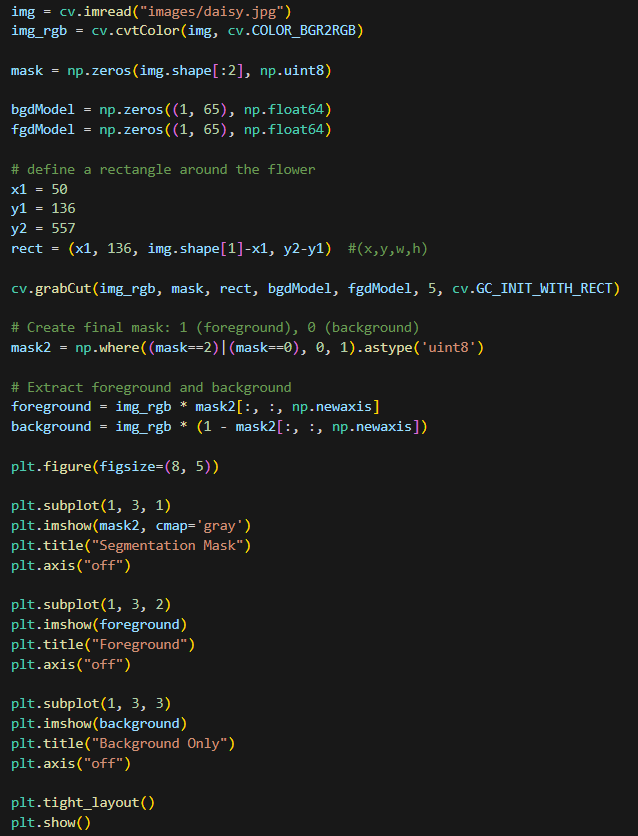
SSD (Nearest Neighbor): 0.012057949   
SSD (Bilinear): 0.017718948

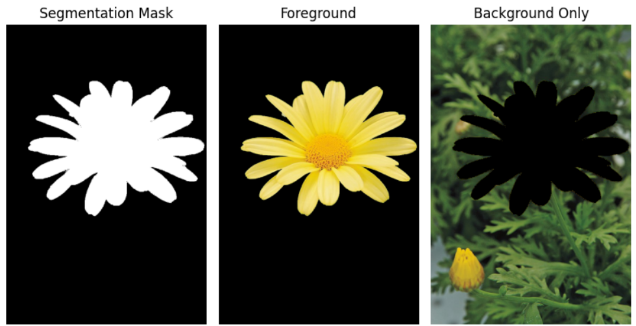
****

SSD (Nearest Neighbor): 0.0041900394  
SSD (Bilinear): 0.00775718

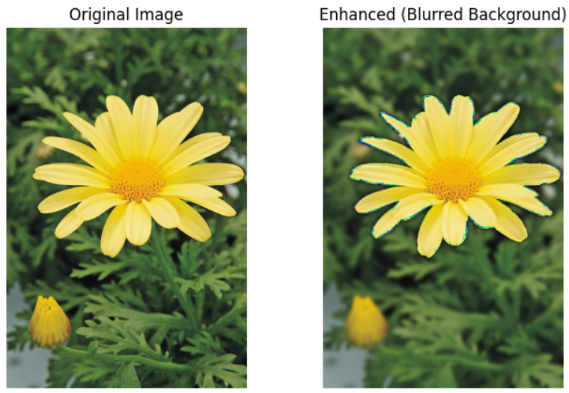
**Question 08**

1. Use grabCut to segment the image

The GrabCut algorithm was used with a bounding rectangle around the flower. Rectangle was defined approximately using the original image. This produced a segmentation mask, from which the foreground (flower) and background were extracted separately.



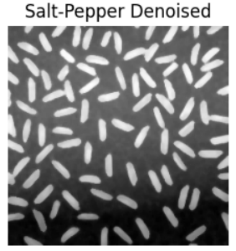
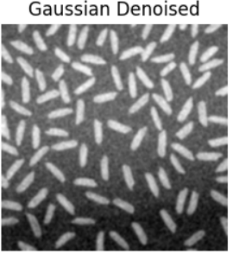
1. Produce an enhanced image with a blurred background

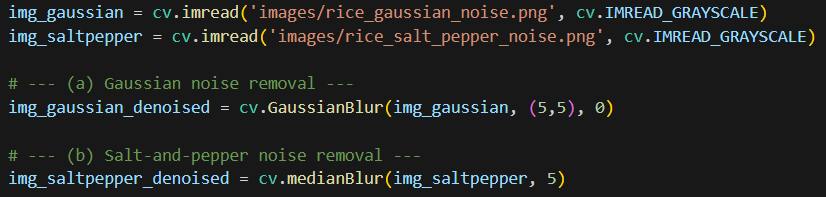
To generate a blurred background, gaussian blur was applied. Then, combined blurred background and original foreground.

1. Why is the background just beyond the edge of the flower quite dark in the enhanced image?

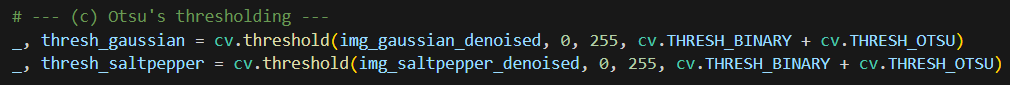
Because Grabcut doesn’t perfectly segment the flower from its background. There can be misclassified boundary pixels. So after it combined with the blurred background, we can see a dark outline around the flower.

**Question 09**

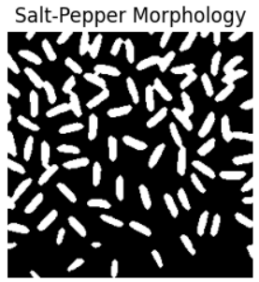
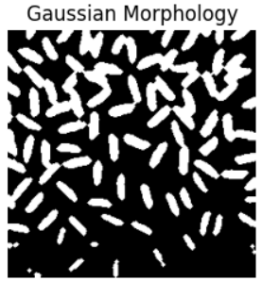
1. and (b) Preprocess the images to remove noise. Used kernel size is as 5.

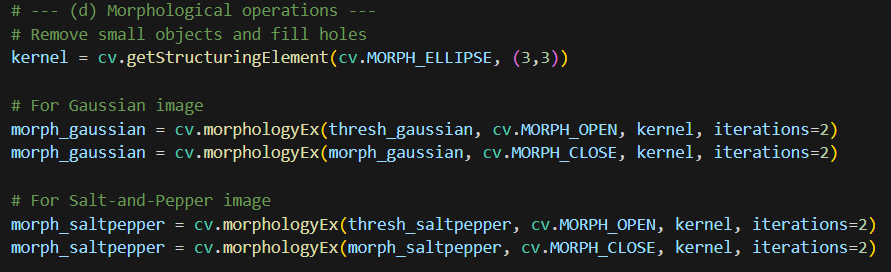


(c) Apply Otsu’s method to segment the image.

Convert denoised grayscale images to binary images to identify foreground and background. Otsu’s method automatically determines the best threshold.

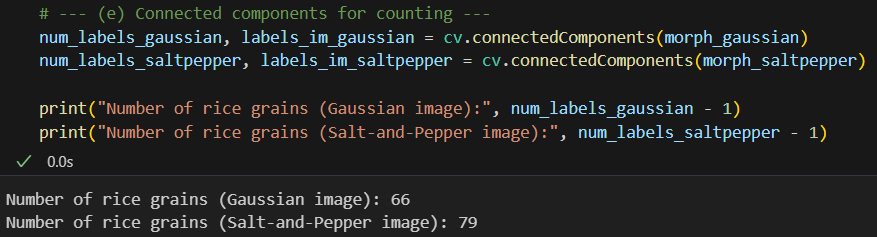
(d) Apply morphological operations

Clean the binary images using morphological operations to removes small white noise pixels and fills small black holes inside foreground objects.



(e) Count the number of rice grains.

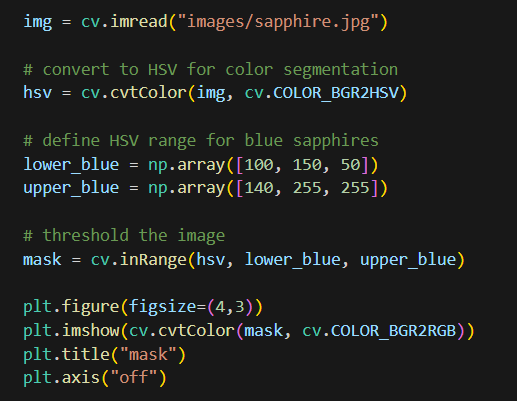
Identify connected regions (rice grains) in binary images. connectedComponents outputs num\_labels which has the number of connected regions including background. Therefore, we subtract 1 because background is also counted.

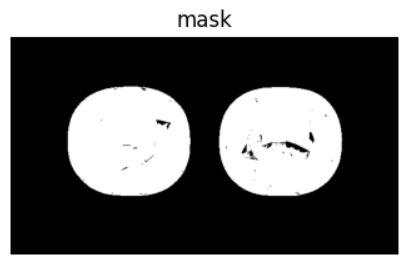


Number of rice grains (Gaussian image): 66  
Number of rice grains (Salt-and-Pepper image): 79

**Question 10**

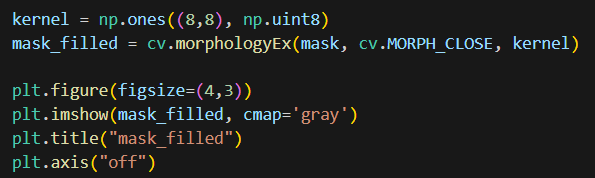
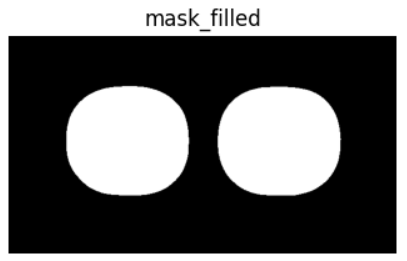
1. Obtain a binary mask for the sapphires.

A colour-based thresholding method was used to segment the image. The original image was converted to HSV color space. Then, the binary mask was created using a predefined HSV range for blue colour.



1. Apply a morphological operation to fill the holes

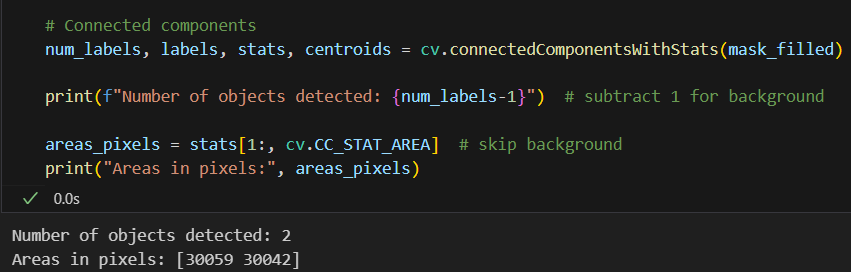
To remove holes inside the sapphires, a morphological operation was used with kernel size 8. Because small kernel sizes(< 8) resulted unfilled few holes.



1. Run connectedComponentsWithStats to obtain the areas in pixels.

**Connected component labeling** was used to detect each sapphire separately. The **area in pixels** for each connected component (excluding background) was extracted from stats[:, cv.CC\_STAT\_AREA].

Areas in pixels: 30059 , 30042

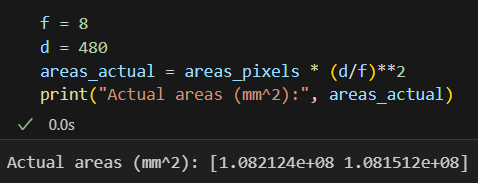


1. Compute the actual areas of the sapphires

The actual sapphire areas were computed using below equation.

Areas = 1.082124e+08 mm2, 1.081512e+08 mm2

= 108212400.00 mm2, 108151200.00 mm2

 Area actual​ = Area pixels​ × (​)2