Task-1:

Task-1:Q. What are the differences between linear and non-linear filter? In which cases one does perform better than the other?

Ans: Linear filter: A linear filter is a type of image filter where the output pixel values are a linear combination of the input pixel values within a neighborhood.

Non-linear filter: A non-linear filter is a type of image filter where the output pixel values are not necessarily a linear combination of the input pixel values.

The primary distinction between linear and non-linear filters is that linear filters are computationally efficient and have a well-defined mathematical framework, whereas non-linear filters can capture more complex patterns and structures in the image but are potentially more computationally expensive.

Linear & filters are ideal for image processing tasks requiring linear operations, such as image smoothing, edge detection and noice reduction.

Non-linear filters on the other hand, are best suited for complex image processing tasks such as texture analysis, object detection and image segmentation.

The choice of linear or non-linear filter is determined by the image processing task at hand. For simple image processing tasks, linear filters are fast and efficient for simple tasks, whereas non-linear filters are flexible and more powerful for more complex image processing tasks.

Task-2:

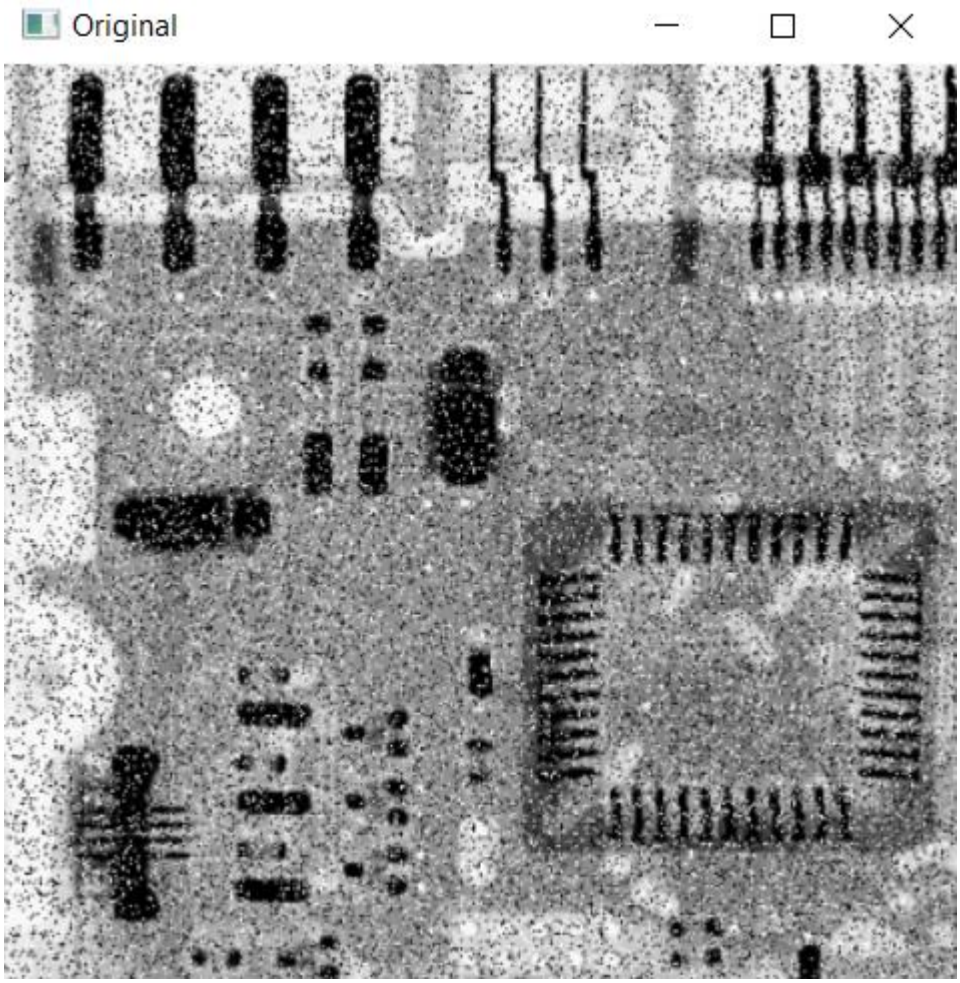
Task-2: The kind of noise which is induced in the input image is salt and pepper noise.

The output image can be obtained by using median filter. So we'll be importing cv2 module and use the function medianBlur of cv2 on our ~~image~~ input image. And after that we'll ~~be~~ be getting output image closer to the attached right image i.e. the provided output.

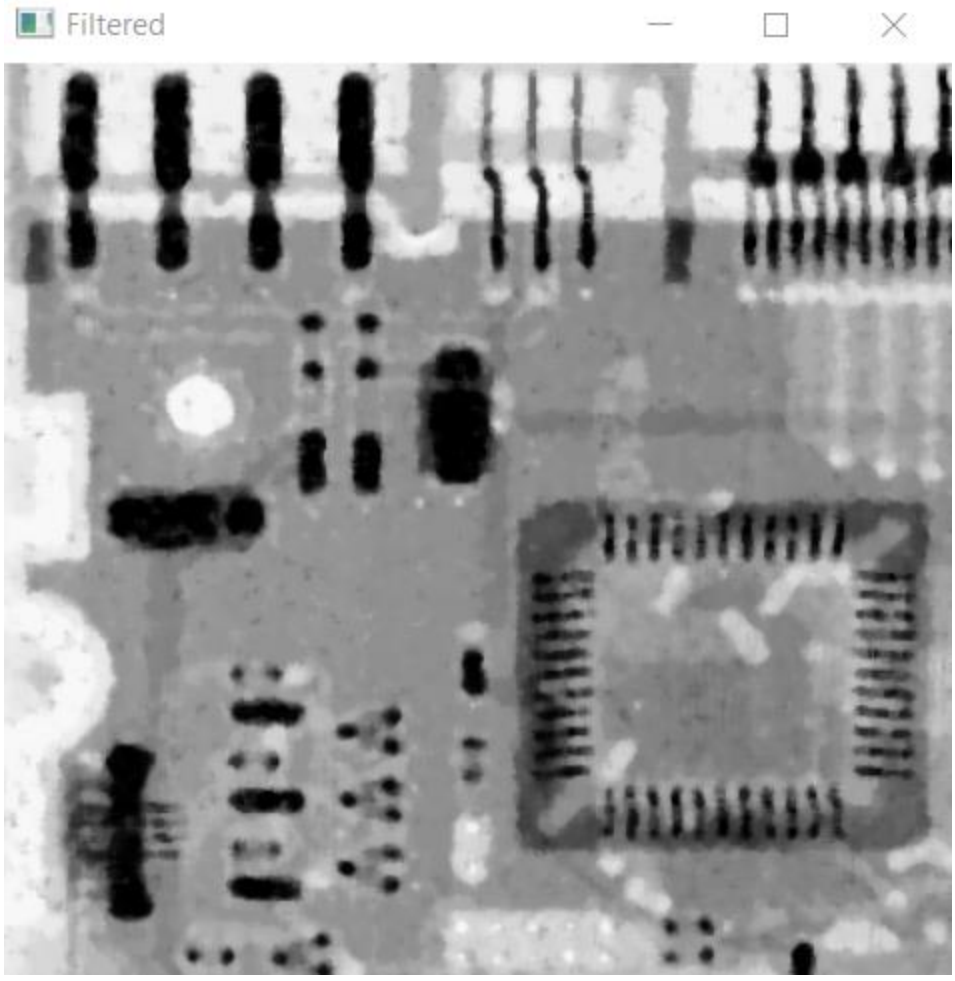
The code of the Task-2 is:

```
import cv2
import numpy as np

input_img = cv2.imread('Input-2.png')
grayed = cv2.cvtColor(input_img, cv2.COLOR_BGR2GRAY)
filtered_img = cv2.medianBlur(grayed, 5)

cv2.imshow('Original', grayed)
cv2.imshow('Filtered', filtered_img)
cv2.waitKey(0)
```

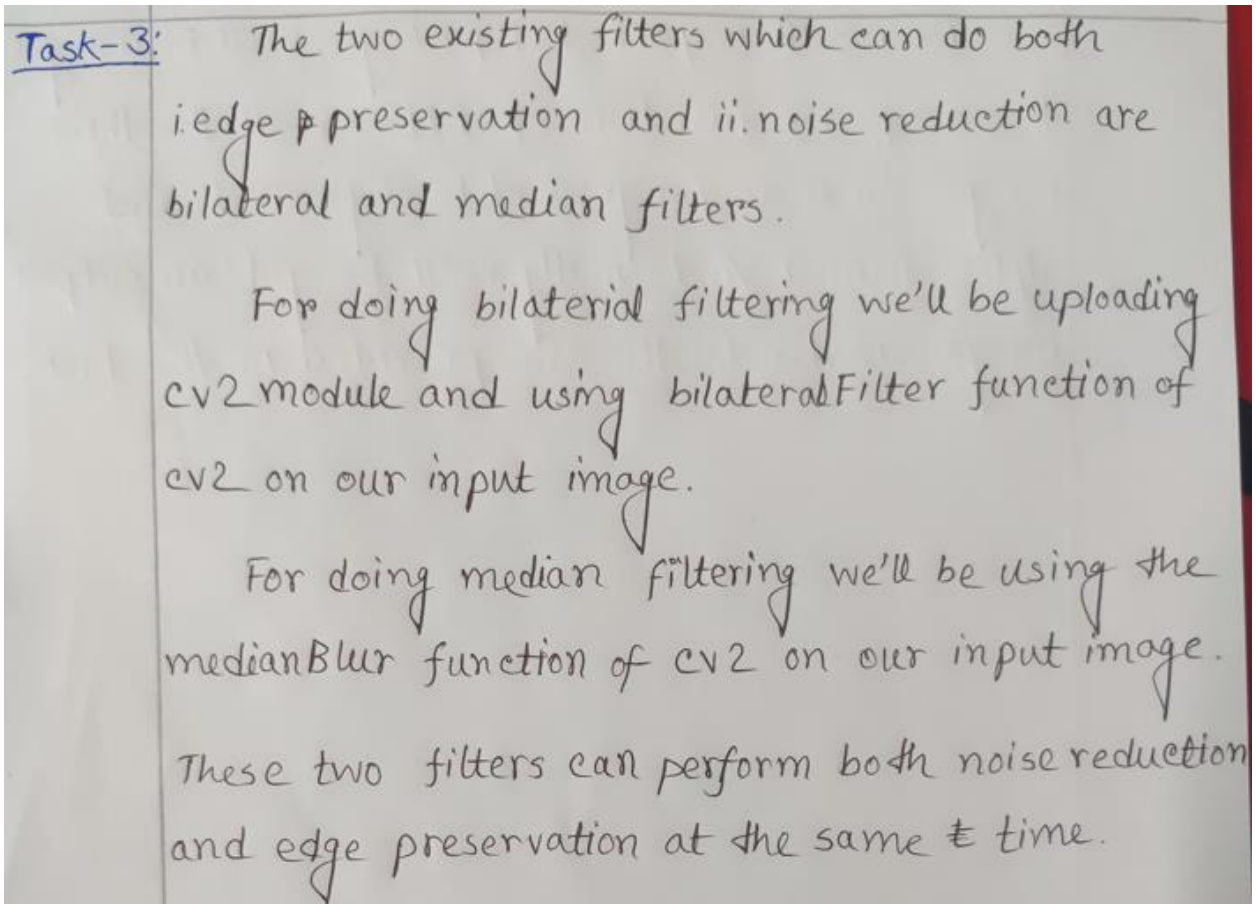
The Input:



The Output:



Task-3:

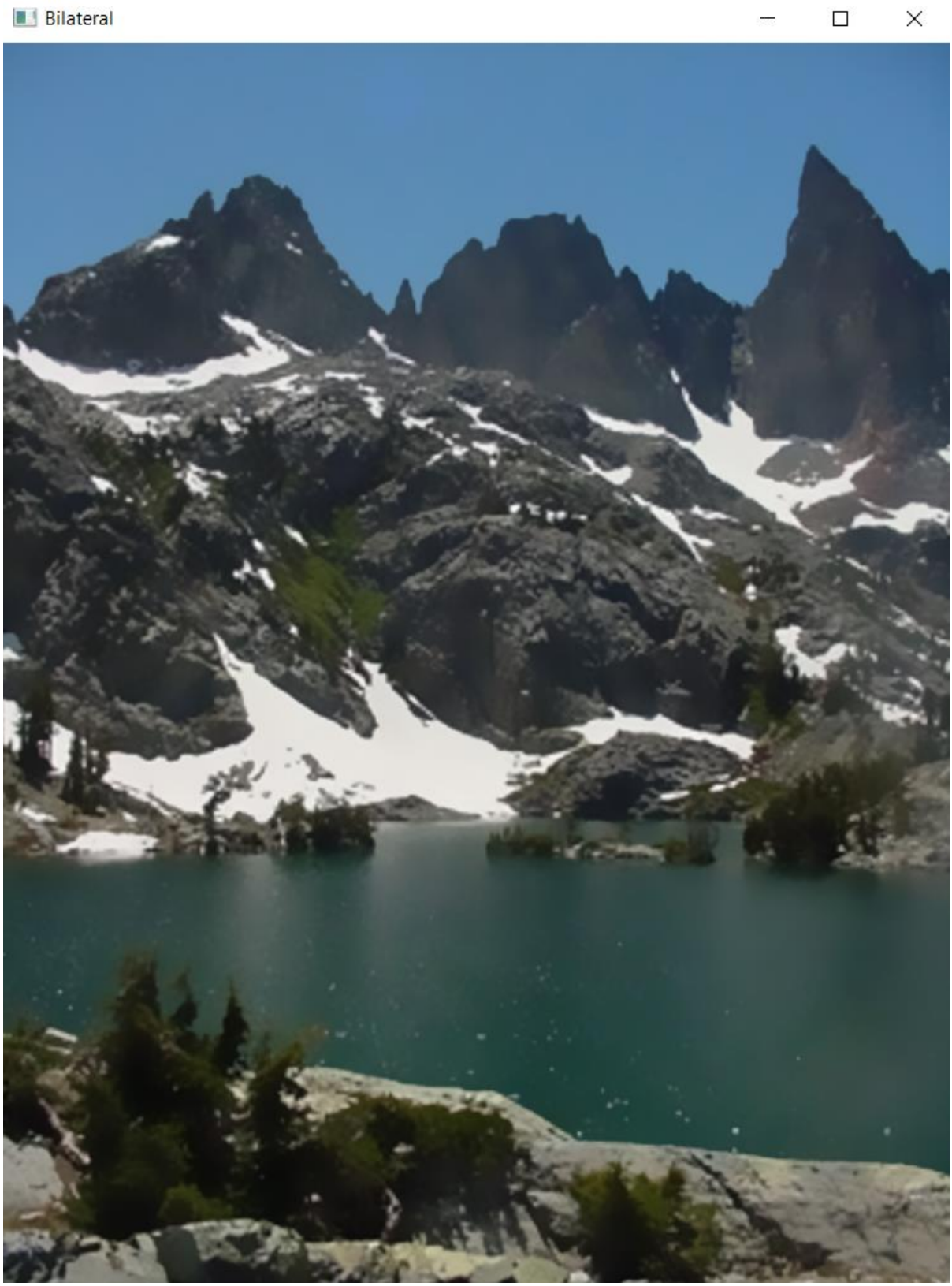


The code of Task-3(Bilateral) is:

```
import cv2
import numpy as np

input_noise_img = cv2.imread('input-3.png')
bilateral_img = cv2.bilateralFilter(input_noise_img, 20, 40, 100, borderType =
cv2.BORDER_CONSTANT)

cv2.imshow('Original', input_noise_img)
cv2.imshow('Bilateral', bilateral_img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

The Input:

The Output:



The code of Task-3(Median) is:

```
import cv2
import numpy as np

input_noise_img = cv2.imread('input-3.png')
median_img = cv2.medianBlur(input_noise_img, 7)

cv2.imshow('Original', input_noise_img)
cv2.imshow('Median', median_img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

The Input:

The Output:

As we can see from the input and output images, the filters performed both edge preservation and noise reduction effectively.

Task-4:

Task-4: We can achieve this kind of output image by certain operations.

At first we'll have to take foreground and background image. Then we'll have to take a alpha channel which will work as a mask.

Now we'll import PIL package. We'll be using the composite function of PIL on all the three images such as foreground, background and alpha channel. And finally we'll be get an output image similar to the one provided in the task.

The code of Task-4 is:

```
import cv2
import numpy as np

from PIL import Image

foregnd_img = Image.open('foregnd.png').convert('RGB').resize((600,600))
backgnd_img = Image.open('backgnd.png').convert('RGB').resize((600,600))

masking = Image.open('alpha_chnl.png').convert('L').resize((600,600))

Image.composite(foregnd_img, backgnd_img, masking).save('Output.png')

out_img = cv2.imread('Output.png')
cv2.imshow('Output Image', out_img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

The Inputs:

Background:

Foreground:

Alpha Channel:

The Output: