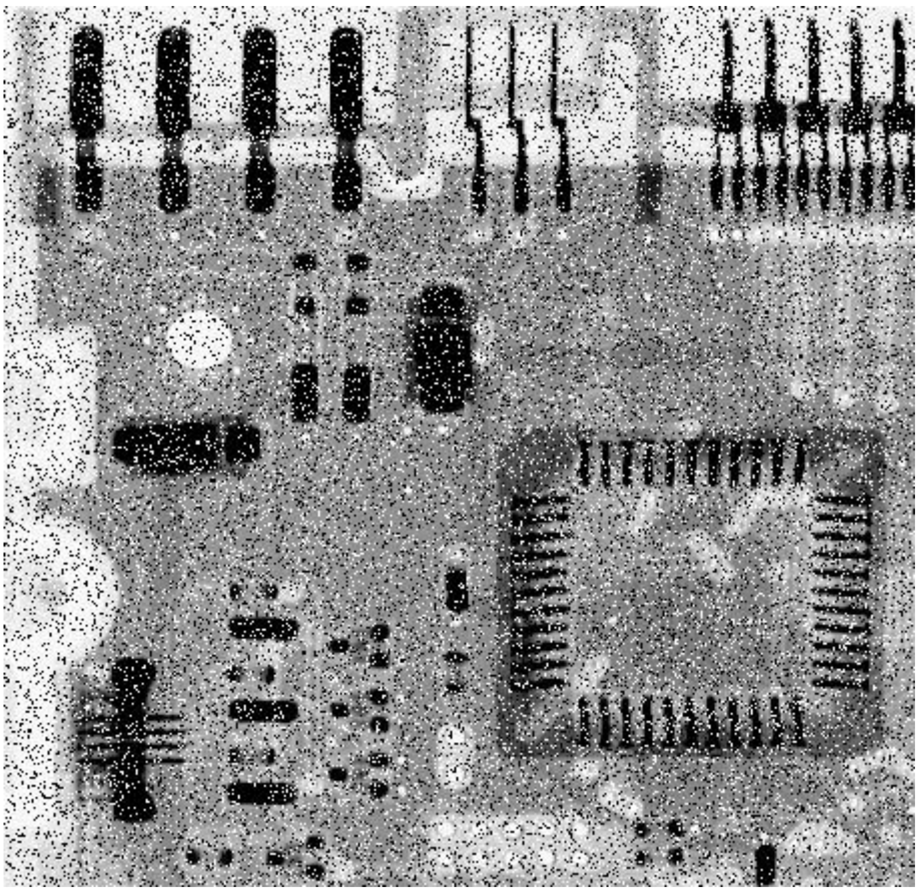
Experiment No. 7

- **Title:** To Implement Spatial Domain Filtering

- **Theory :**
  Spatial Filtering technique is used directly on pixels of an image. Mask is usually considered to be added in size so that it has a specific center pixel. This mask is moved on the image such that the center of the mask traverses all image pixels.
  Neighborhood processing in spatial domain: Here, to modify one pixel, we consider values of the immediate neighboring pixels also. For this purpose, 3X3, 5X5, or 7X7 neighborhood mask can be considered. An example of a 3X3 mask is shown below.

$$\begin{matrix} f(x-1, y-1) & f(x-1, y) & f(x-1, y+1) \\ f(x, y-1) & f(x, y) & f(x, y+1) \\ f(x+1, y-1) & f(x+1, y) & f(x+1, y+1) \end{matrix}$$

- **Input image:**



1) **Averaging filter**
   Code:

```
import cv2
import numpy as np
```

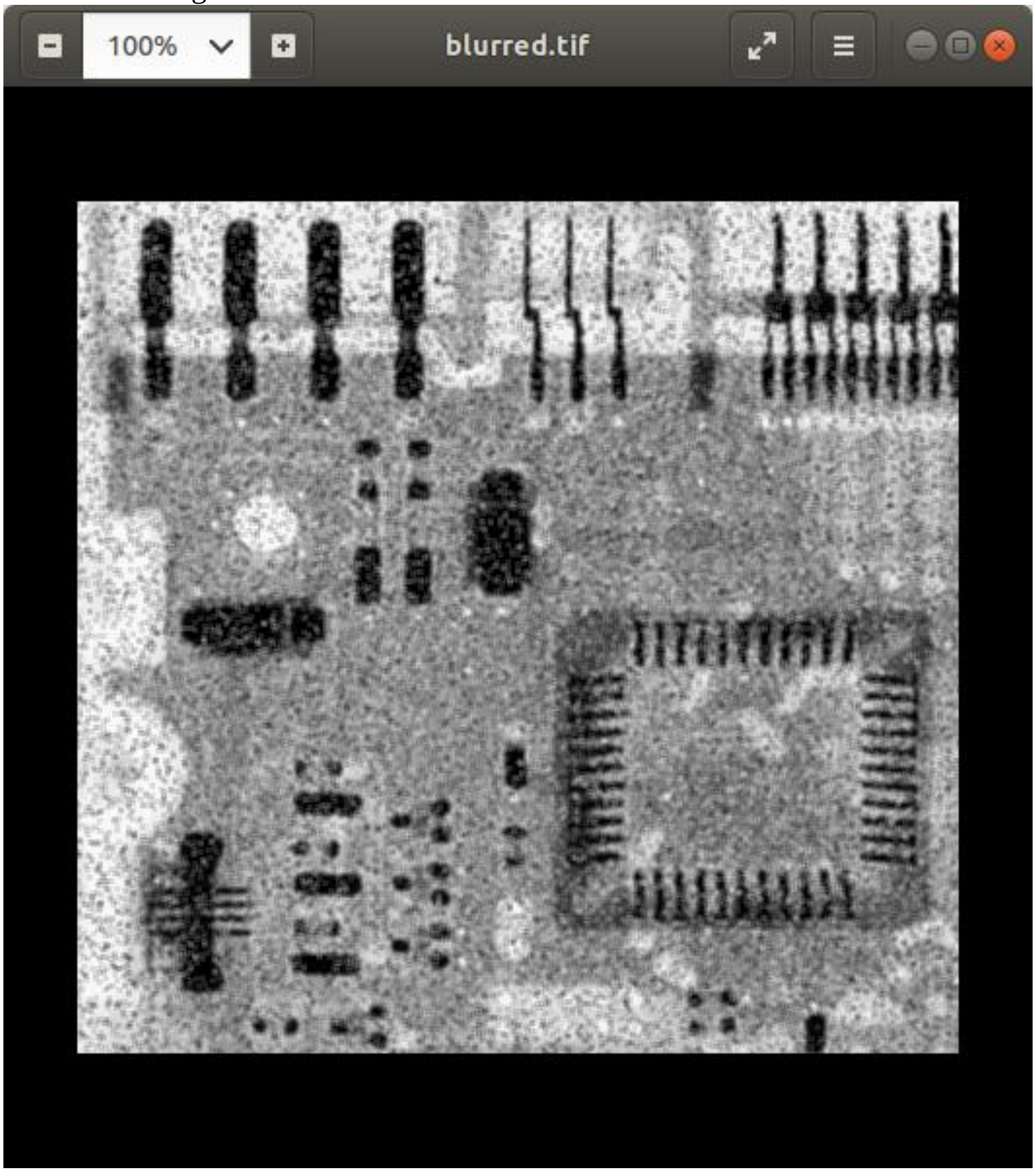
```
img = cv2.imread('sample.png', 0)
m, n = img.shape
# Develop Averaging filter(3, 3) mask
mask = np.ones([3, 3], dtype = int)
mask = mask / 9
# Convolve the 3X3 mask over the image
img_new = np.zeros([m, n])

for i in range(1, m-1):
    for j in range(1, n-1):
            temp = img[i-1, j-1]*mask[0, 0]+img[i-1, j]*mask[0, 1]+img[i-1, j +
            1]*mask[0, 2]+img[i, j-1]*mask[1, 0]+ img[i, j]*mask[1, 1]+img[i, j +
            1]*mask[1, 2]+img[i + 1, j-1]*mask[2, 0]+img[i + 1, j]*mask[2,
            1]+img[i + 1, j + 1]*mask[2, 2]

            img_new[i, j]= temp

img_new = img_new.astype(np.uint8)
cv2.imwrite('blurred.tif', img_new)
```

Output:

2) **Median filter:**
Code :

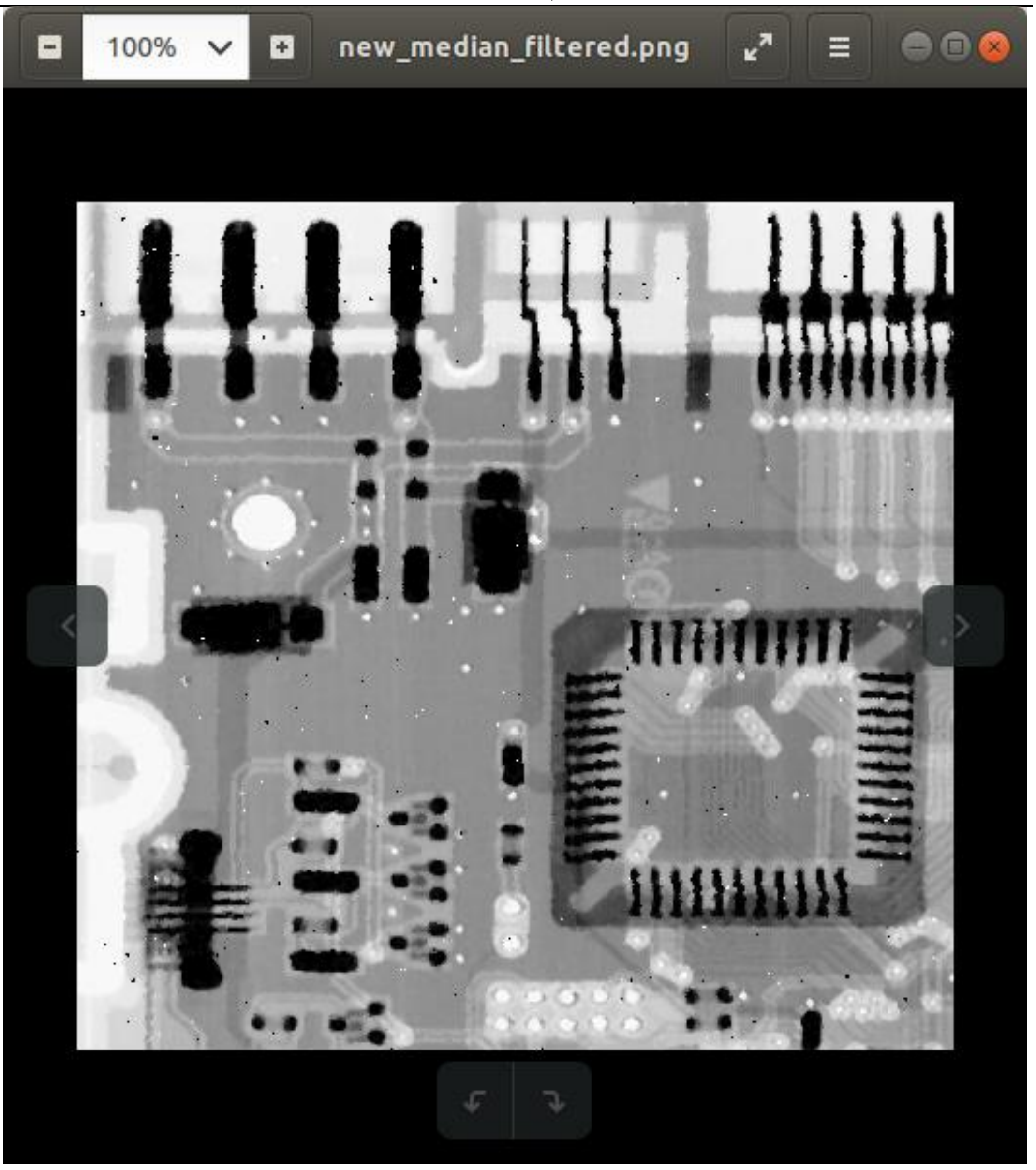
```
# Median Spatial Domain Filtering
import cv2
import numpy as np
```

```
# Read the image
```



```
img_noisy1 = cv2.imread('sample.png', 0)

# Obtain the number of rows and columns
# of the image
m, n = img_noisy1.shape

# Traverse the image. For every 3X3 area,
# find the median of the pixels and
# replace the center pixel by the median
img_new1 = np.zeros([m, n])

for i in range(1, m-1):
        for j in range(1, n-1):
                temp = [img_noisy1[i-1, j-1],
                        img_noisy1[i-1, j],
                        img_noisy1[i-1, j + 1],
                        img_noisy1[i, j-1],
                        img_noisy1[i, j],
                        img_noisy1[i, j + 1],
                        img_noisy1[i + 1, j-1],
                        img_noisy1[i + 1, j],
                        img_noisy1[i + 1, j + 1]]

                temp = sorted(temp)
                img_new1[i, j]= temp[4]

img_new1 = img_new1.astype(np.uint8)
cv2.imwrite('new_median_filtered.png', img_new1)
```

Output:


100%
new_median_filtered.png

Viva Questions:

1. What is Spatial Domain Filtering?
2. How does Spatial Filtering differ from Frequency Domain Filtering?
3. What is a kernel (mask) in spatial filtering?
4. What are the two types of Spatial Filtering?
5. What is Convolution in Spatial Filtering?
6. How do you perform linear filtering in OpenCV?
7. What is the difference between `cv2.blur()` and `cv2.GaussianBlur()`?
8. Write the OpenCV function for Gaussian filtering.
9. What is Median Filtering?
10. Write the OpenCV function for Median Filtering.
11. What is Bilateral Filtering?
12. Write the OpenCV function for Bilateral Filtering.
13. Where do we use Spatial Domain Filtering in real life?
14. Which filter is best for removing Gaussian noise?
15. Which filter is best for removing Salt-and-Pepper noise?
16. Which filter is best when we want smoothing but need to preserve edges?
17. How do you apply a custom 3×3 sharpening filter in OpenCV?
18. What is the effect of kernel size in spatial filtering?
19. What is the drawback of averaging (mean) filter?
20. How does OpenCV handle border pixels in filtering?