Name: Krish Sukhani

UID: 2018140059

Batch: D

Branch: IT

Smoothing: ¶

-Use Low Pass Averaging Filter (To eliminate Guassian Noise)

-Use Low Pass Median Filter (To eliminate Salt and Pepper Noise)

Sharpening:

-Use High Pass Filter

In [1]:

```
from PIL import Image
from PIL import ImageFilter
import cv2
import math
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/cont
ent/drive", force_remount=True).
```

In [3]:

```
img_gaussian_path = '/content/drive/MyDrive/Sem-7/DIP-Lab/SmoothSharp/DIP_3-1.jpg'
```

In [4]:

```
img_saltpepper_path = '/content/drive/MyDrive/Sem-7/DIP-Lab/SmoothSharp/DIP_3-2.png'
```

In [5]:

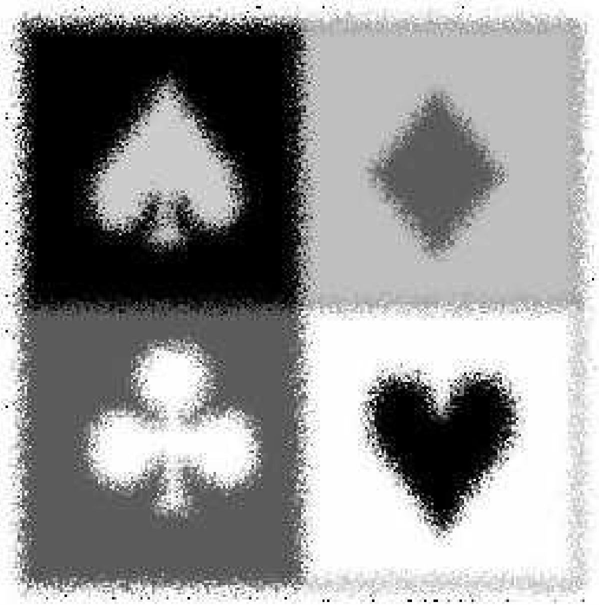
```
img_sharpening_path = '/content/drive/MyDrive/Sem-7/DIP-Lab/SmoothSharp/DIP_3-3.jpg'
```

Low Pass Averaging Filter

In [6]:

```
input_img = Image.open(img_gaussian_path)
input_img
```

Out[6]:



In [7]:

```
input_img = input_img.convert('L')
```

In [8]:

```
width, height = input_img.size
```

In [9]:

```
width
```

Out[9]:

```
257
```

In [10]:

```
height
```

Out[10]:

```
257
```

Mask

In [11]:
```
mask = np.array([[1, 1, 1],[1,  1, 1],[1, 1, 1]])
mask = mask / 9
```

In [12]:
```
mask
```

Out[12]:
```
array([[0.11111111, 0.11111111, 0.11111111],
       [0.11111111, 0.11111111, 0.11111111],
       [0.11111111, 0.11111111, 0.11111111]])
```

Adding Replication Padding to the Input Image

In [13]:
```
left = right = top = bottom = 1
new_width = width + right + left
new_height = height + top + bottom
```

In [14]:
```
new_height
new_width
```

Out[14]:
```
259
```

Convolve the 3X3 mask over the image

In [15]:
```
padded_input_img = Image.new(input_img.mode,(new_width,new_height),(0))
padded_input_img.paste(input_img,(left,top))
```

In [16]:

```
img = np.asarray(padded_input_img)
img
```

Out[16]:

```
array([[  0,   0,   0, ...,   0,   0,   0],
       [  0, 254, 254, ..., 255, 254,   0],
       [  0, 254, 254, ..., 255, 254,   0],
       ...,
       [  0, 255, 255, ..., 255, 254,   0],
       [  0, 254, 254, ..., 254, 254,   0],
       [  0,   0,   0, ...,   0,   0,   0]], dtype=uint8)
```

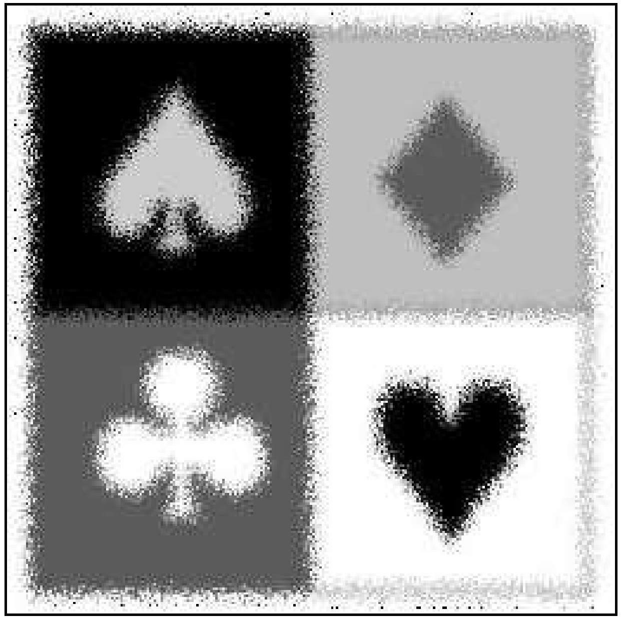
In [17]:

```
img_new = np.zeros([new_height, new_width])
for i in range(1, new_height-1):
    for j in range(1, new_width-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)
```

In [18]:

```
img_final_input = Image.fromarray(np.uint8(img)).convert('RGB')
img_final_input
```

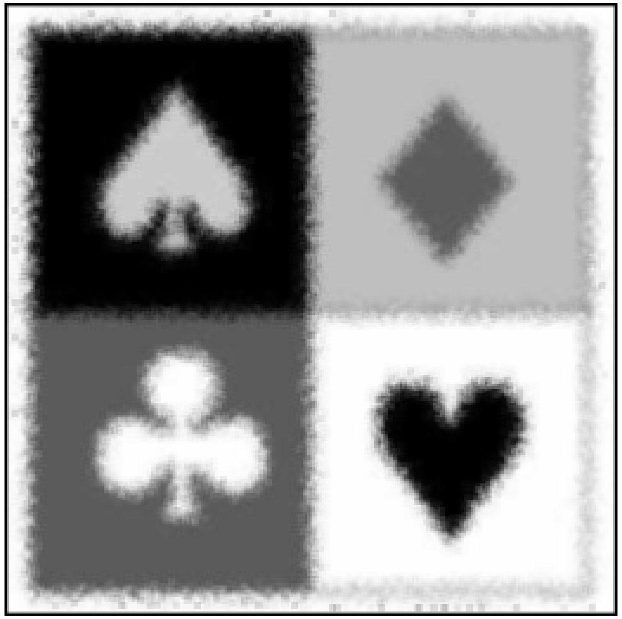
Out[18]:



In [19]:

```
img_final_output = Image.fromarray(np.uint8(img_new)).convert('RGB')
img_final_output
```

Out[19]:

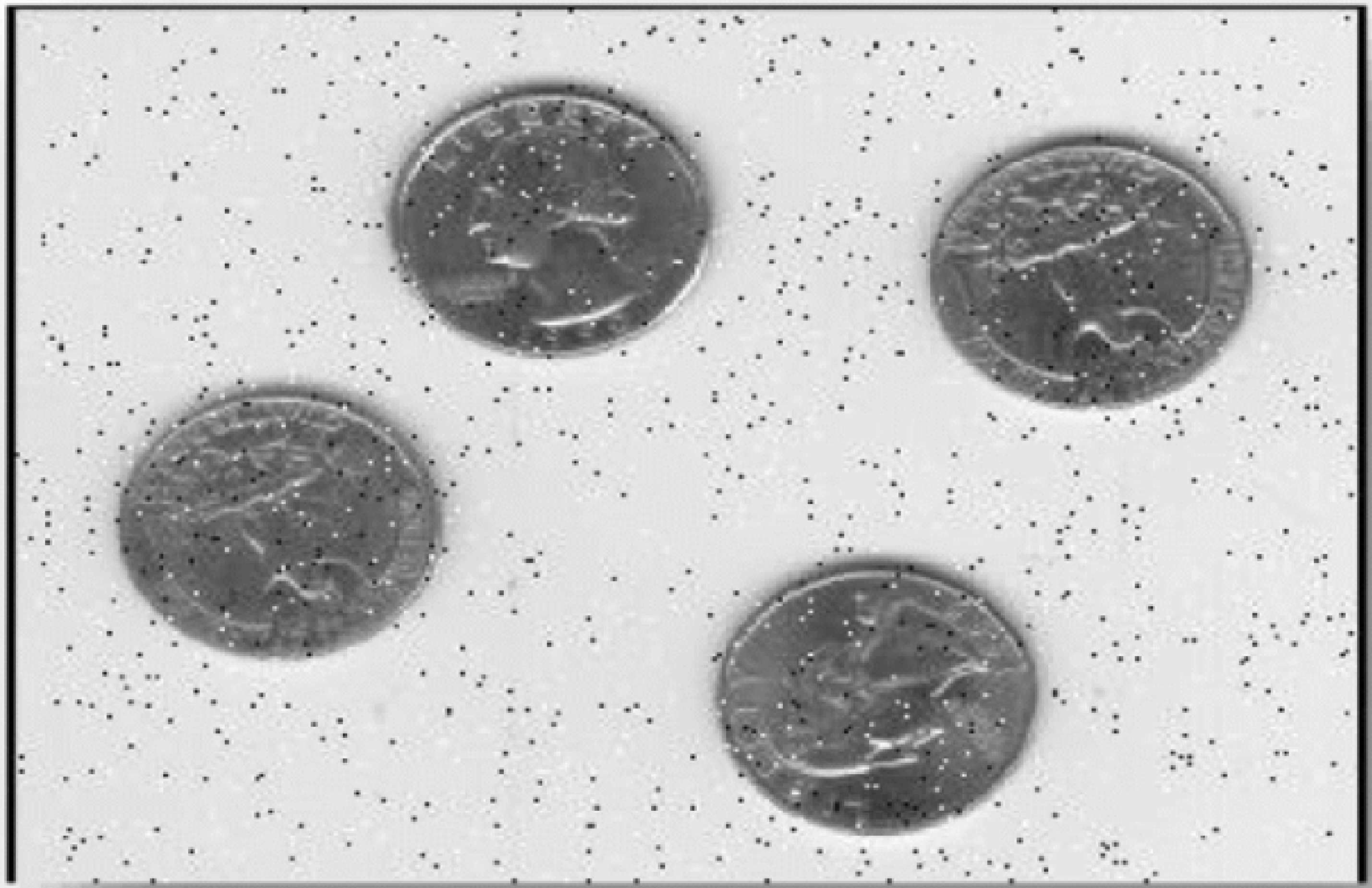


Observation: The image gets blurred - smoothened by eliminating gaussian noise

Low Pass Median Filter

In [20]:
```
input_img = Image.open(img_saltpepper_path)
input_img
```

Out[20]:



In [21]:
```
input_img = input_img.convert('L')
```

In [22]:

```
width, height = input_img.size
width
height
```

Out[22]:

```
479
```

In [23]:

```
mask = np.array([[1, 1, 1],[1,  1, 1],[1, 1, 1]])
```

In [24]:

```
mask
```

Out[24]:

```
array([[1, 1, 1],
       [1, 1, 1],
       [1, 1, 1]])
```

In [25]:

```
left = right = top = bottom = 1
new_width = width + right + left
new_height = height + top + bottom
```

In [26]:

```
new_height
new_width
```

Out[26]:

```
687
```

In [27]:

```
padded_input_img = Image.new(input_img.mode,(new_width,new_height),(0))
padded_input_img.paste(input_img,(left,top))
```

In [28]:

```
img = np.asarray(padded_input_img)
img
```

Out[28]:

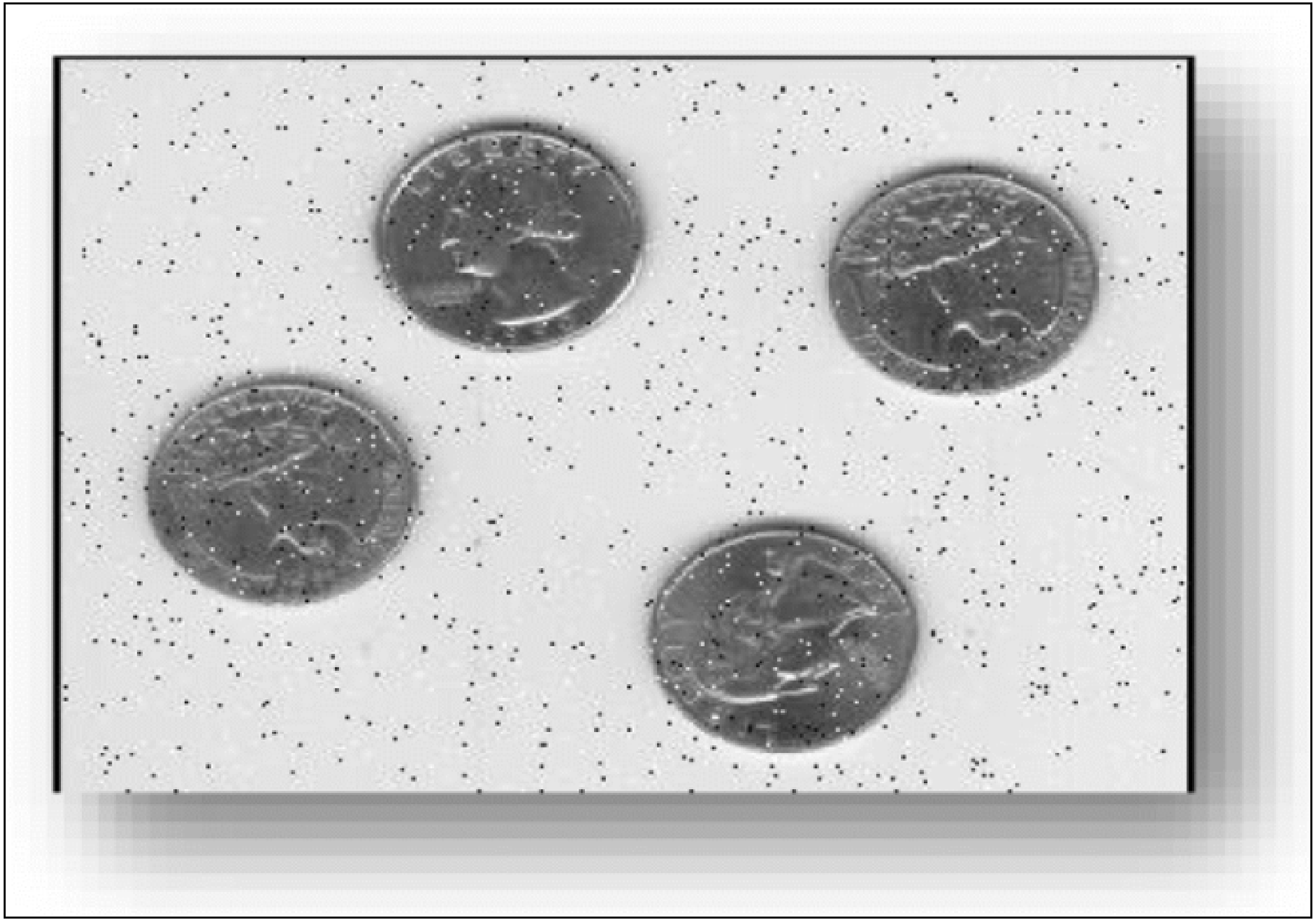
```
array([[  0,   0,   0, ...,   0,   0,   0],
       [  0, 255, 255, ..., 255, 255,   0],
       [  0, 255, 255, ..., 255, 255,   0],
       ...,
       [  0, 255, 255, ..., 252, 255,   0],
       [  0, 255, 255, ..., 255, 255,   0],
       [  0,   0,   0, ...,   0,   0,   0]], dtype=uint8)
```

In [29]:

```
img_new1 = np.zeros([new_height,new_width])

for i in range(1, new_height-1):
    for j in range(1, new_width-1):
        temp = [img[i-1, j-1],
               img[i-1, j],
               img[i-1, j + 1],
               img[i, j-1],
               img[i, j],
               img[i, j + 1],
               img[i + 1, j-1],
               img[i + 1, j],
               img[i + 1, j + 1]]

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

img_new1 = img_new1.astype(np.uint8)
```

In [30]:
```
img_final_input_1 = Image.fromarray(np.uint8(img)).convert('RGB')
img_final_input_1
```

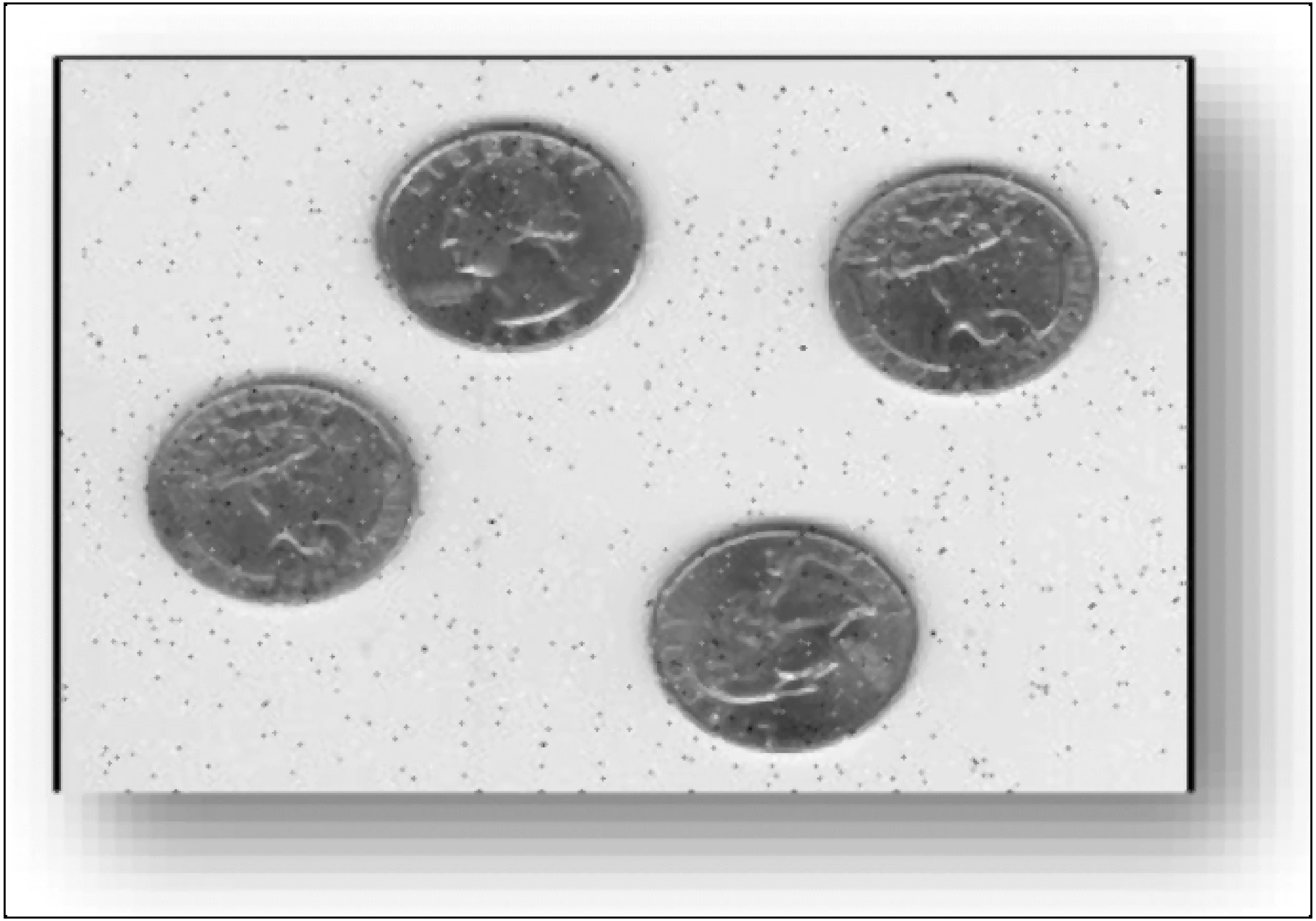
Out[30]:



In [31]:

```
img_final_output_1 = Image.fromarray(np.uint8(img_new1)).convert('RGB')
img_final_output_1
```

Out[31]:



Observation: The image gets smoothened by eliminating salt and pepper noise

High Pass Filter

In [32]:
```
input_img = Image.open(img_sharpening_path)
input_img
```

Out[32]:

In [33]:
```
input_img = input_img.convert('L')
```

In [34]:
```
width, height = input_img.size
width
height
```

Out[34]:
```
305
```

In [35]:
```
input_img
```

Out[35]:

In [36]:
```
mask = np.array([[-1, -1, -1],[-1,  8, -1],[-1, -1, -1]])
mask = mask/9
```

In [37]:

```
print(input_img)
```

```
<PIL.Image.Image image mode=L size=427x305 at 0x7F127B96DB50>
```

In [38]:

```
left = right = top = bottom = 1
new_width = width + right + left
new_height = height + top + bottom
```

In [39]:

```
new_height
new_width
```

Out[39]:

```
429
```

In [40]:

```
padded_input_img = Image.new(input_img.mode,(new_width,new_height),(0))
padded_input_img.paste(input_img,(left,top))
```

In [41]:

```
img = np.asarray(padded_input_img)
img
```

Out[41]:

```
array([[  0,   0,   0, ...,   0,   0,   0],
       [  0, 196, 193, ...,  35,  51,   0],
       [  0, 197, 198, ...,  57,  46,   0],
       ...,
       [  0, 177, 173, ..., 181, 181,   0],
       [  0, 123, 127, ..., 192, 192,   0],
       [  0,   0,   0, ...,   0,   0,   0]], dtype=uint8)
```

In [42]:

```
img_new = np.zeros([new_height, new_width])
for i in range(1, new_height-1):
    for j in range(1, new_width-1):
        # print(img[i-1, j-1]*mask[0, 0])
        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]
        # print(temp)
        img_new[i,j]= temp

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

In [43]:

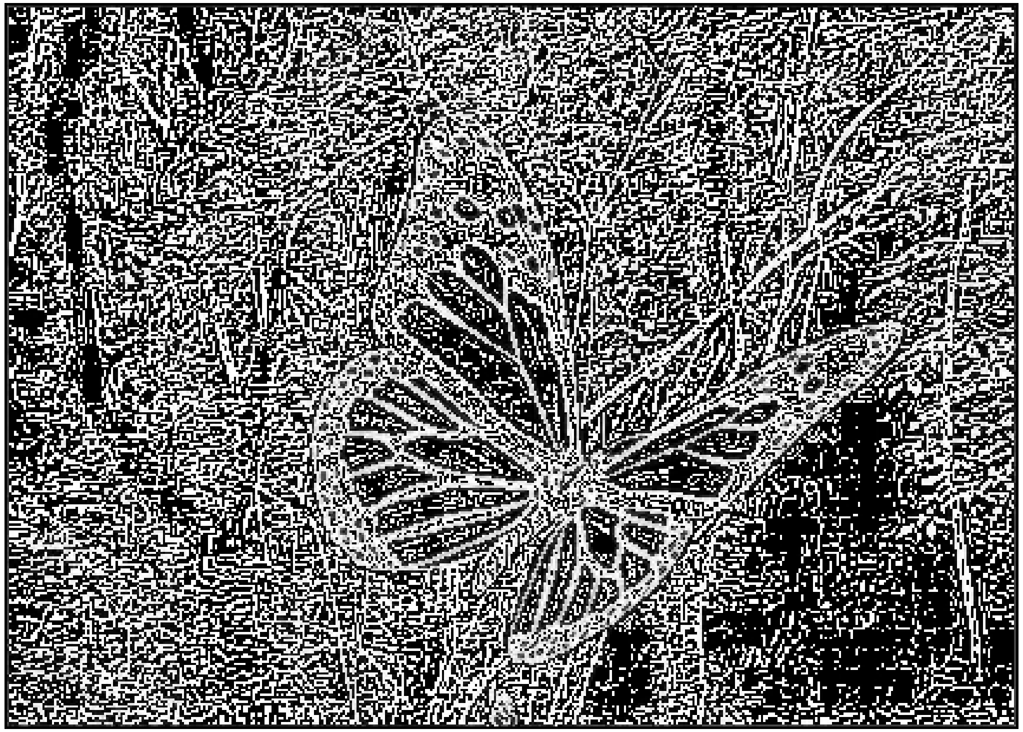
```
img_final_input_2 = Image.fromarray(np.uint8(img)).convert('RGB')
img_final_input_2
```

Out[43]:

In [44]:

```
img_final_output_2 = Image.fromarray(np.uint8(img_new)).convert('RGB')
img_final_output_2
```

Out[44]:



Observation: **The image gets sharpened by retaining the high frequency components**

Conclusion: **Smoothening and Sharpening were successfully performed**