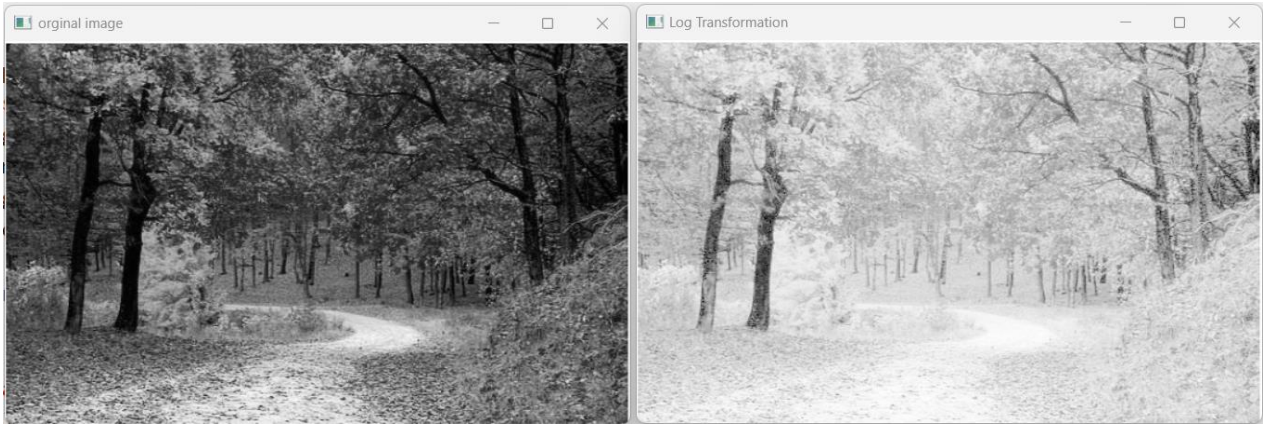
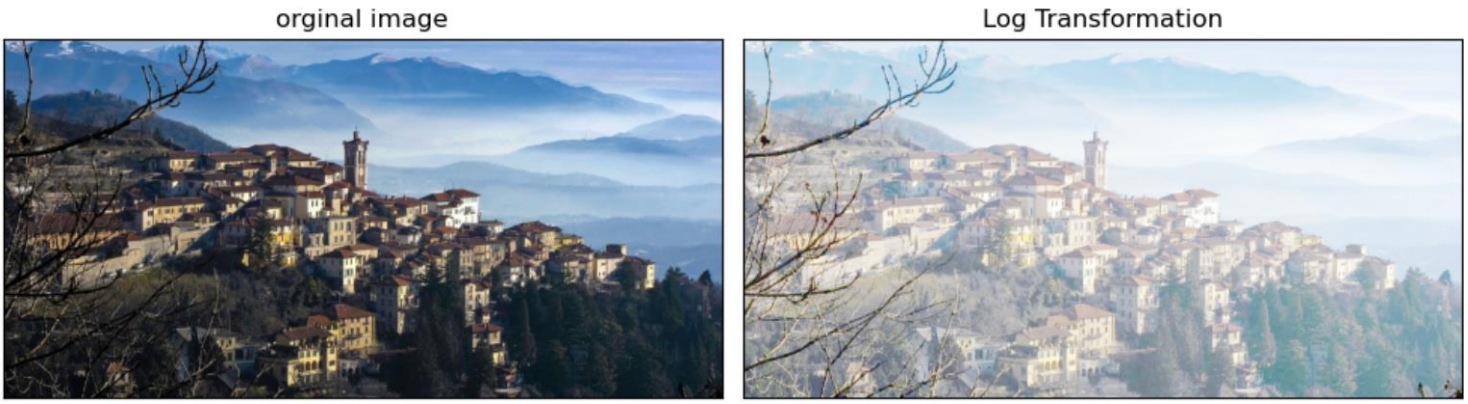
Log Transformation

```
#Log Transformation
import cv2
import numpy as np
from matplotlib import pyplot as plt
img = cv2.imread('images/Arithmetic.jpg',0)
#convert image into float32 type
# لتحسين دقة العمليات الحسابية
img_float=np.float32(img)
#(255-0) ضمان ان الصورة تبقى ضمن نطاق القيم
c=255/np.log(1+np.max(img_float))
#تطبيق اللوغاريتم
image_log=c*np.log(1+img_float)
image_log=np.uint8(image_log)
cv2.imshow('orginal image',img)
cv2.imshow('Log Transformation',image_log)
cv2.waitKey(0)
cv2.destroyAllWindows()
```



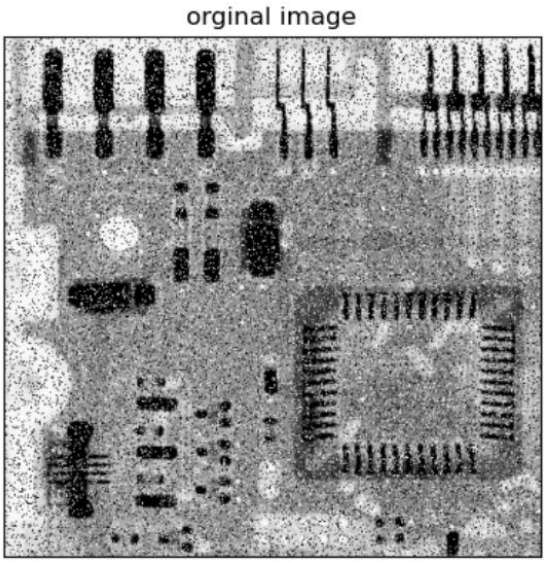
```python
#Log Transformation
import cv2
import numpy as np
from matplotlib import pyplot as plt
img = cv2.imread('images/varese.jpg')
img_float=np.float32(img)
c=255/np.log(1+np.max(img_float))
image_log=c*np.log(1+img_float)
image_log=np.uint8(image_log)
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
image_log = cv2.cvtColor(image_log, cv2.COLOR_BGR2RGB)
fig, axs = plt.subplots(1, 2, figsize=(10, 4))
axs[0].imshow(img)
axs[0].set_title('orginal image')
axs[1].imshow(image_log)
axs[1].set_title('Log Transformation')
for ax in axs:
    ax.set_xticks([])
    ax.set_yticks([])
plt.tight_layout()
plt.show()
```

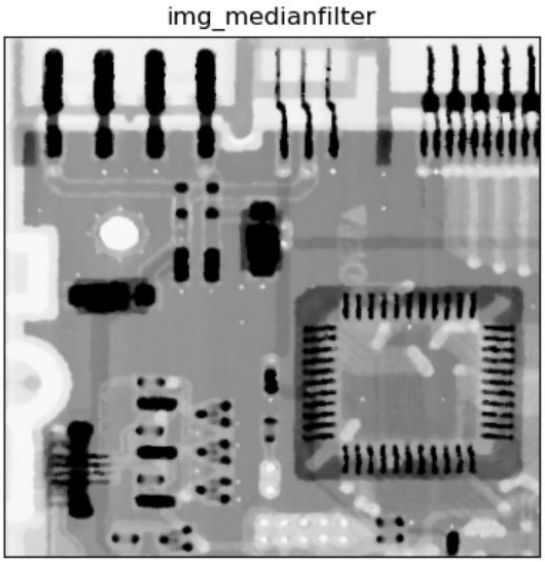


Spatial Domain- Smoothing Spatial Filters

```
#spatial domain Smoothing Linear Filters : mean filter,Gaussianfilter
import cv2
import numpy as np
from matplotlib import pyplot as plt
img = cv2.imread('images/sunflower.jpg')
img_meanfilter=cv2.blur(img,(5,5))
img_Gaussianfilter=cv2.GaussianBlur(img,(5,5),0)
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
img_meanfilter= cv2.cvtColor(img_meanfilter, cv2.COLOR_BGR2RGB)
img_Gaussianfilter= cv2.cvtColor(img_Gaussianfilter, cv2.COLOR_BGR2RGB)
fig, axs = plt.subplots(1, 3, figsize=(10, 4))
axs[0].imshow(img)
axs[0].set_title('orginal image')
axs[1].imshow(img_meanfilter)
axs[1].set_title('img_meanfilter')
axs[2].imshow(img_Gaussianfilter)
axs[2].set_title('img_Gaussianfilter')
for ax in axs:
    ax.set_xticks([])
    ax.set_yticks([])
plt.tight_layout()
plt.show()
```

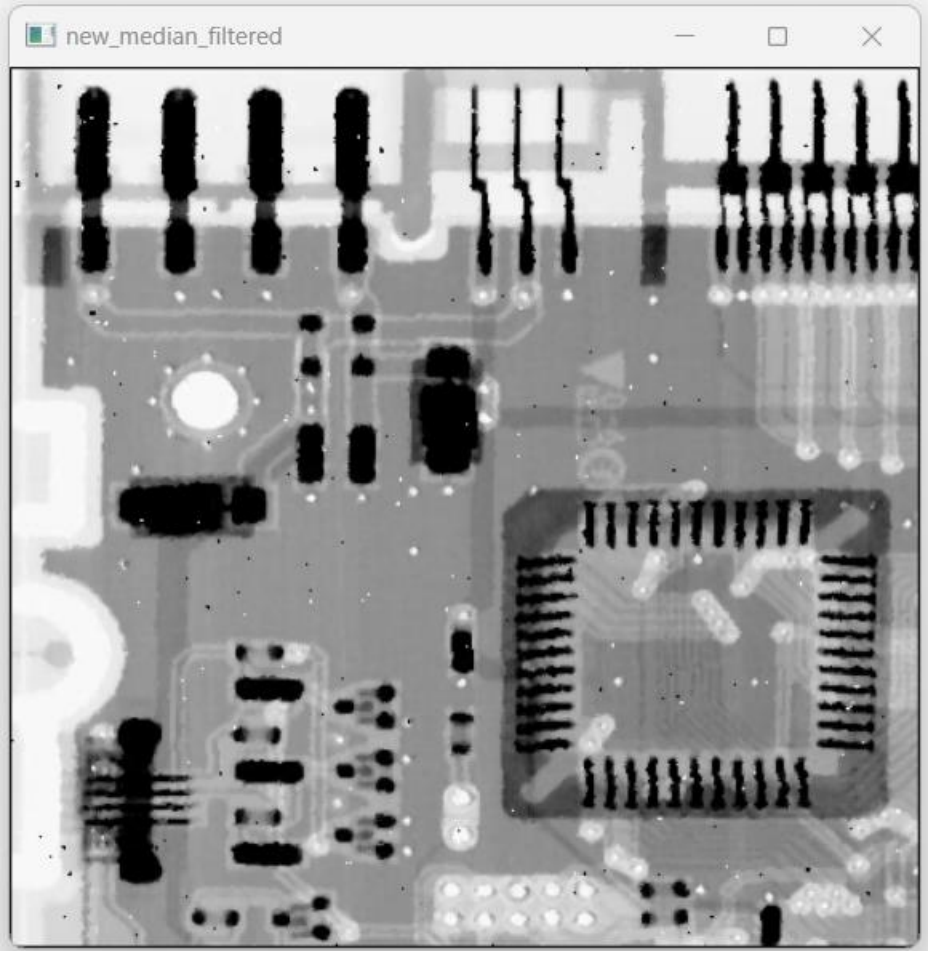
```python
#spatial domain Smoothing non-Linear Filters : median filter,
import cv2
import numpy as np
from matplotlib import pyplot as plt
#img = cv2.imread('images/sunflower.jpg')
img = cv2.imread('images/noisysalterpepper.png')
img_medianfilter=cv2.medianBlur(img,5)
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
img_medianfilter= cv2.cvtColor(img_medianfilter, cv2.COLOR_BGR2RGB)
fig, axs = plt.subplots(1, 2, figsize=(10, 4))
axs[0].imshow(img)
axs[0].set_title('orginal image')
axs[1].imshow(img_medianfilter)
axs[1].set_title('img_medianfilter')
for ax in axs:
    ax.set_xticks([])
    ax.set_yticks([])
plt.tight_layout()
plt.show()
```





```
# Median Spatial Domain Filtering
import cv2
import numpy as np
img_noisy1 = cv2.imread('images/noisysalterpepper.png', 0)
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)
cv2.imshow('new_median_filtered', img_new1)
cv2.waitKey(0)
cv2.destroyAllWindows()
```


new_median_filtered

```python
#spatial domain: filter2D
import cv2
import numpy as np
from matplotlib import pyplot as plt
img = cv2.imread('images/sunflower.jpg',)
mask = np.ones((5,5),np.float32)/25
#mask=np.array([
#    [1/9,1/9,1/9],
#    [1/9,1/9,1/9],
#    [1/9,1/9,1/9]
#])
#mask=np.array([
#    [-1,-1,-1],
#    [-1,8,-1],
#   [-1,-1,-1]
#])
img_filter=cv2.filter2D(src=img,
                        ddepth=-1,
                        kernel=mask)
cv2.imshow('image1',img)
cv2.imshow('image2',img_filter)
cv2.waitKey(0)
cv2.destroyAllWindows()
```