

Digital Image Processing

المحاضرة الثامنة

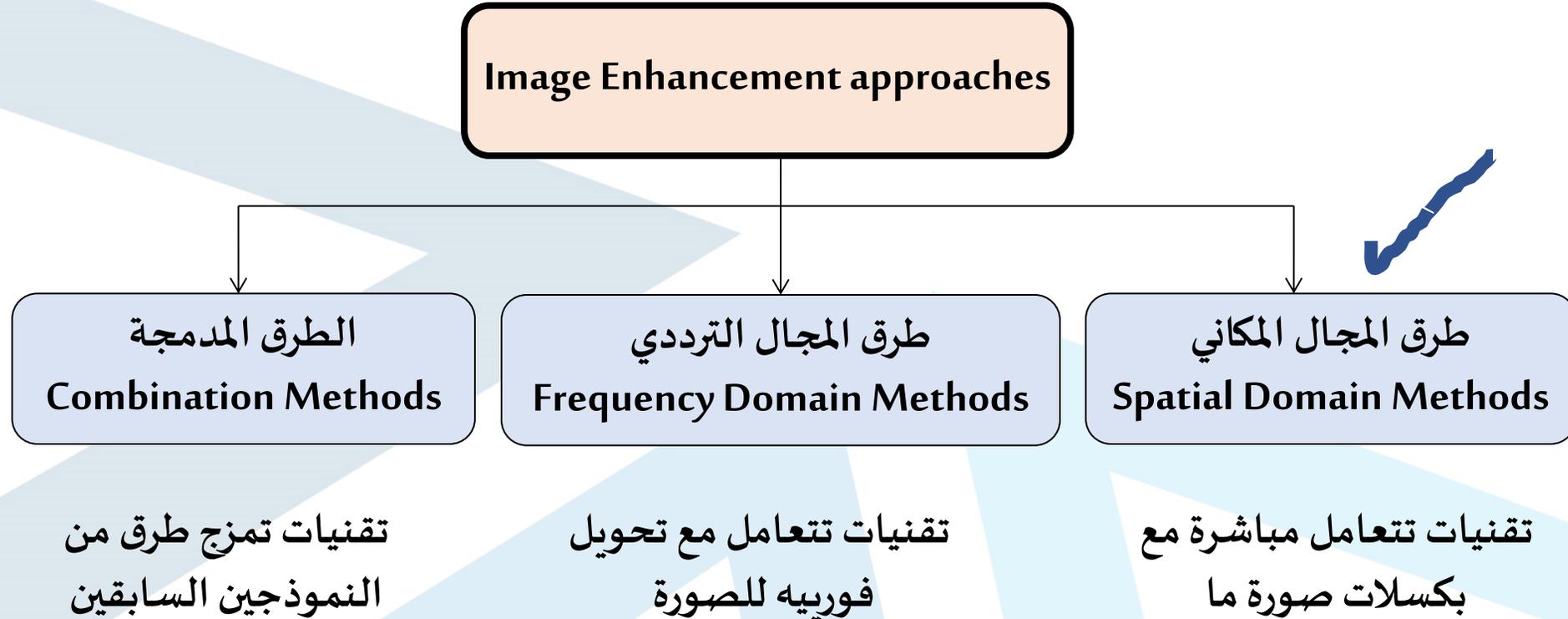
Image Enhancement

Spatial Domain Methods

العمليات على مستوى القناع أو جيران
البكسل باستخدام المرشحات المكانية

د. عيسى الغنام
2025





مفهوم ترشيح الصورة بالاعتماد على النوافذ المحلية
جيران البكسل التي تستخدم عادة في إزالة التشويش
أو للكشف عن الحواف واظهارها.

Spatial Domain Methods

مستوى القناع جيران البكسل Mask Level

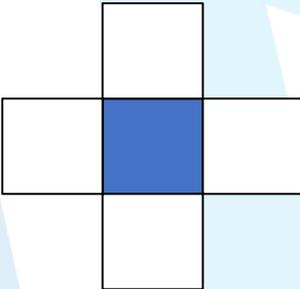
مستوى البكسل Pixel Level

العمليات على هيستوغرام الصورة

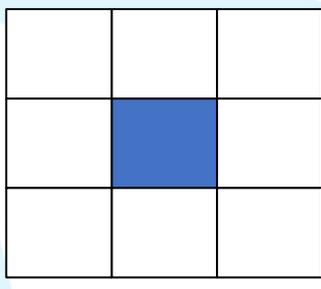
العمليات على البكسلات
باستخدام هيستوغرام الصورة
معالجة الرسم البياني (الهيستوغرام)
Histogram Processing مساواة
الهيستوغرام



$$N_0(x, y)$$

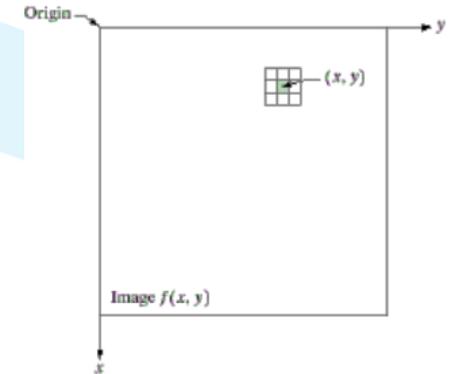


$$N_4(x, y)$$



$$N_8(x, y)$$

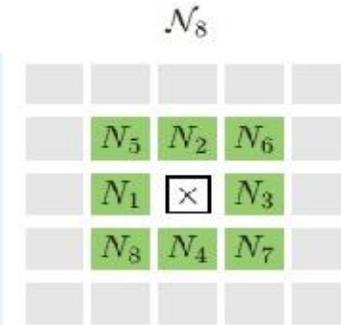
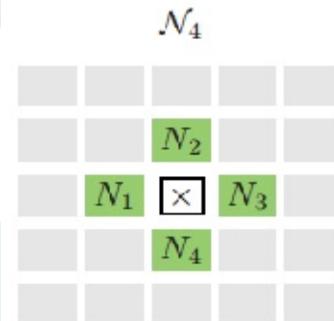
تحسين الصورة باستخدام المرشحات المكانية



□ جيران البكسل: البكسلات المحيطة به (أصغر مصفوفة يقع هذا البكسل في مركزها)

□ نافذة جيران البكسل (النافذة المحلية): هي مصفوفة مربعة ذات أبعاد فردية (قد تكون مستطيلة أو دائرية)

	174	66	115	174	190	21	123
البكسل الحالي	77	209	215	214	198	76	230
جيران البكسل 3X3	138	9	140	218	229	80	99
	38	199	89	177	180	113	213
	178	200	156	143	174	132	215



فإذا كانت إحداثيات البكسل المرجعي هي u, v فعندها تتراوح إحداثيات جوار هذا البكسل ضمن المجال $u, v \pm k$



التصنيف العام للمرشحات

- تُطبق عمليات الترشيح على كامل الصورة وفق مبدأ النافذة المنزلقة:
- تسمى النافذة المنزلقة بـ mask أو kernels أو window أو filter.
- تتم معالجة كل بكسل في الصورة بالاعتماد على عملية بكسلات هذه النافذة و تستخدم جيران هذا البكسل في الصورة الأصلية.
- تصنف عموماً (تبعاً للخصائص الرياضية لتابع المرشح) إلى مرشحات خطية وغير خطية.



□ معظم الأقنعة الشائعة هي مصفوفات متناظرة حول مركزها



التعامل مع حدود الصورة

□ هناك عدّة طرق للتعامل مع حدود الصورة:

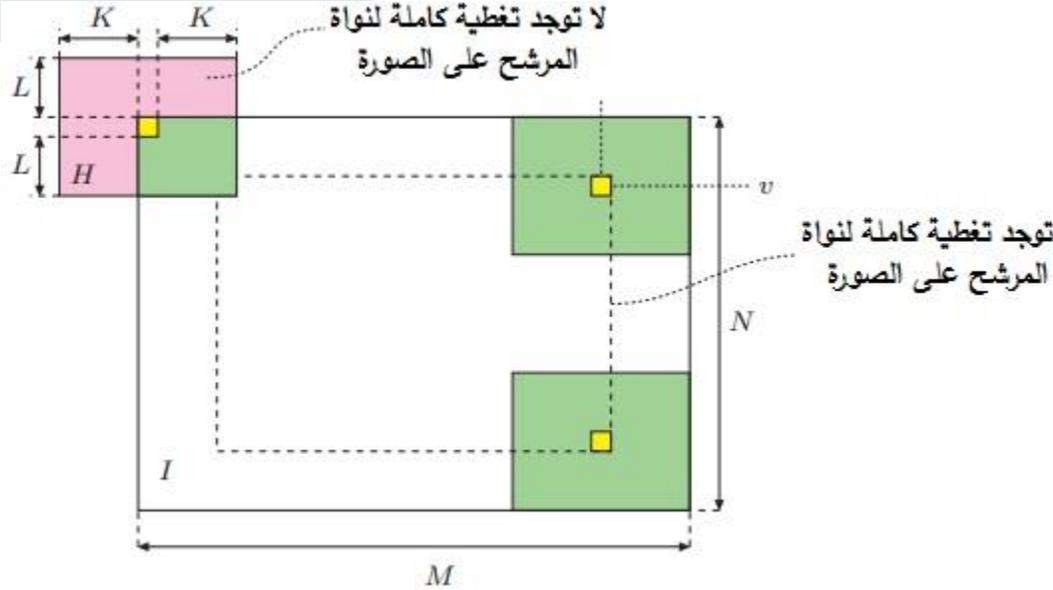
✓ تجاهل حدود الصورة

- الحفاظ على قيم البكسلات
- استبدال قيم البكسلات بقيمة ثابتة

✓ إضافة حدود صفيرية إضافية للصورة الأصلية

✓ تكرار حدود الصورة الأصلية

✓ التعامل مع حدود صورة الدخل على أنها تابع دوري ثنائي الأبعاد يكرر قيمه في كلّ من الاتجاهين العمودي والأفقي



التشويش

□ تنتج الكاميرات وقنوات بثّ الصور في بعض الأحيان أنواعاً معيّنة من التشويش والتي تتواجد على شكل بكسلات عشوائية ومعزولة

تكون قيمة السويات الرمادية فيها خارج نطاق السويات الرمادية أي إن قيمتها قد تكون أكبر بكثير أو أصغر بكثير من قيم السويات الرمادية لجيرانها ويدعى هذا التشويش بتشويش "الملح والفلفل" . salt and pepper

يعدّ التحدي الأكبر لإزالة التشويش هو التمييز بينها وبين التفاصيل الصغيرة في الصورة كالحواف أو الخطوط أو بعض الملامح الأخرى.

التشويش الغاوسي (: Gaussian noise) أو الطبيعي) هو ضوضاء أو تشويش له دالة كثافة احتمالية مساوية لدالة الكثافة الاحتمالية للتوزيع الطبيعي.

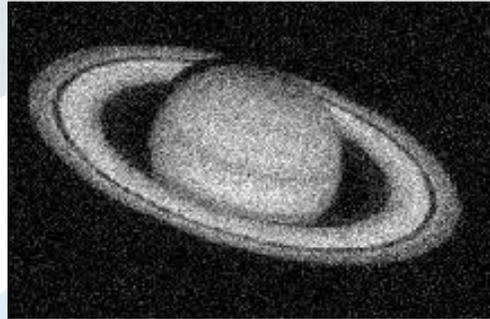


التشويش

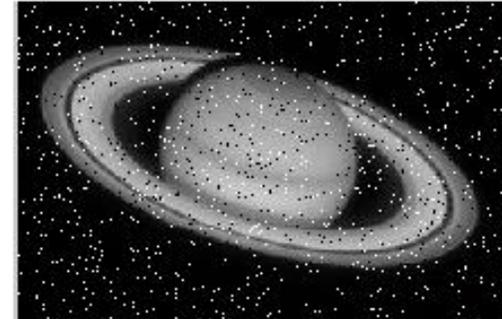
ولمقارنة وتحديد كفاءة المرشحات المختلفة نحتاج أولاً لإضافة التشويش إلى صور ومن ثم اختبار قدرة المرشحات على استعادة الصورة الأصلية.

Matlab code

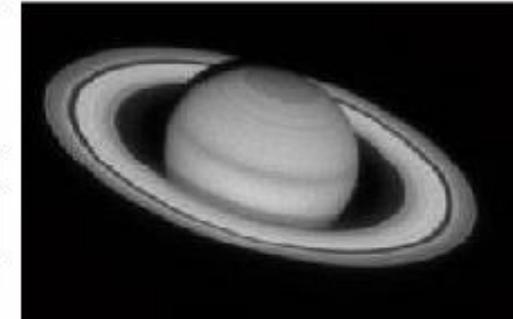
```
a=imread('saturn.jpg');  
figure, imshow(a)  
asp=imnoise(a, 'salt &  
pepper',0.05);  
figure, imshow(asp)  
ag=imnoise(a,'gaussian',0.02);
```



إضافة تشويش غاوص



إضافة تشويش الملح والفلفل



الصورة الأصلية



Adding Noise :



```
Noise = imnoise ( gray image , ' type ' );
```

'gaussian'	Gaussian white noise with constant mean and variance
'localvar'	Zero-mean Gaussian white noise with an intensity-dependent variance
'poisson'	Poisson noise
'salt & pepper'	"On and Off" pixels
'speckle'	Multiplicative noise





```
import cv2
import numpy as np
import os # Import the 'os' module

# Load the image (replace 'saturn.jpg' with your image path)
image_path = 'saturn.jpg'
a = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)

# Check if the image loaded successfully
if a is None:
    print("Error: Could not load image. Check the file path.")
    exit()

cv2.imshow('Original Image (Grayscale)', a)
cv2.waitKey(0)
```

```
# Add salt and pepper noise (intensity = 0.05)
asp = a.copy()
noise_prob = 0.05
for i in range(asp.shape[0]):
    for j in range(asp.shape[1]):
        rdn = np.random.random()
        if rdn < noise_prob:
            # Salt - set to white (255)
            asp[i, j] = 255
        elif rdn > 1 - noise_prob:
            # Pepper - set to black (0)
            asp[i, j] = 0
```

```
# Save the salt and pepper noisy image
output_path_sp =
os.path.join(os.path.dirname(image_path),
'npsaturn.jpg') # Construct path using original directory
cv2.imwrite(output_path_sp, asp)
print(f"Salt and pepper noise image saved to:
{output_path_sp}")
# Display the salt and pepper noisy image
cv2.imshow('Salt and Pepper Noise', asp)
cv2.waitKey(0)
```



```
# Add Gaussian noise (variance = 0.02)
```

```
ag = a.copy()
```

```
mean = 0
```

```
sigma = np.sqrt(0.02 * 255 * 255) # Standard deviation
```

```
gaussian_noise = np.random.normal(mean, sigma, ag.shape)
```

```
ag = np.clip(ag + gaussian_noise, 0, 255).astype(np.uint8)
```

```
# Save the Gaussian noisy image
```

```
output_path_gn = os.path.join(os.path.dirname(image_path), 'ngs saturn.jpg') # Construct path using original directory
```

```
cv2.imwrite(output_path_gn, ag)
```

```
print(f"Gaussian noise image saved to: {output_path_gn}")
```

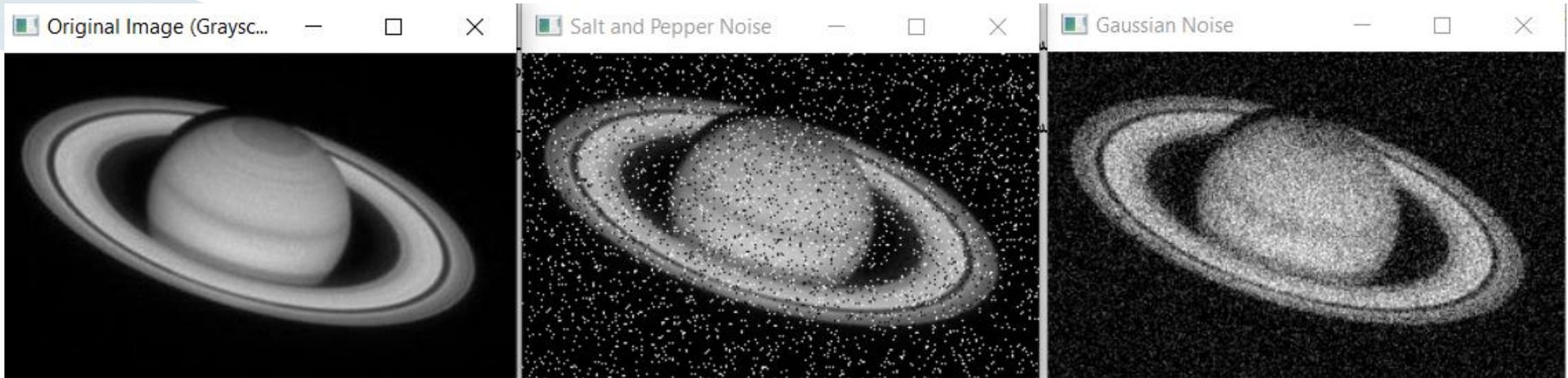
```
# Display the Gaussian noisy image
```

```
cv2.imshow('Gaussian Noise', ag)
```

```
cv2.waitKey(0)
```

```
cv2.destroyAllWindows()
```





$$I'(u,v) = (P_0 + P_1 + P_2 + P_3 + P_4 + P_5 + P_6 + P_7 + P_8) / 9$$

□ أبسط مرشح خطي على الإطلاق

$$I'(u,v) = \frac{1}{9} \cdot \sum_{i=-1}^{+1} \sum_{j=-1}^{+1} I(u+i, v+j) \quad \rightarrow \quad H(i,j) = \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

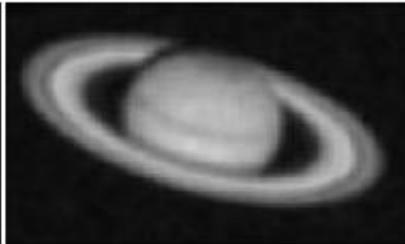
$$H(i,j) = \frac{1}{MN} \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & 1 & \dots & 1 \end{bmatrix}$$

- ويؤدي زيادة حجم المرشح إلى التقليل من تأثير تشويش غاوص ولكن على حساب جودة الصورة لأنه يقل تأثير قيمة البكسل الأصلي عن القيمة الجديدة له مقارنة بتأثير عدد أكبر من بكسلات الجوار في هذه القيمة .
- تضمن نسبة التقسيم $1/NM$ بقاء مجموع أوزان القناع مساو للواحد.

تأثير حجم المرشح في
التخلص من التشويش



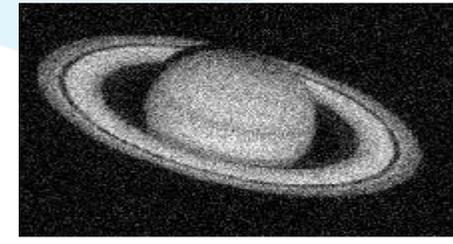
قناع مرشح 21X21



قناع مرشح 9X9



قناع مرشح 5X5



إضافة تشويش غاوص



$$\frac{1}{9} \times$$

1	1	1
1	1	1
1	1	1

$f(x, y)$

$g(x, y) = w(x, y) \otimes f(x, y)$

0	0	0	0	0	0	0	..
0	0	0	0	0	0	0	..
0	0	0	0	0	0	0	..
0	0	0	9	9	9	9	..
0	0	0	9	9	9	9	..
0	0	0	9	9	9	9	..
0	0	0	9	9	9	9	..
0	0	0	9	9	9	9	..
⋮	⋮	⋮	⋮	⋮	⋮	⋮	..

$f(x, y)$

0	0	0	0	0	0	0	0	..
0	0	0	0	0	0	0	0	..
0	0	0	0	0	0	0	0	..
0	0	0	9	9	9	9	9	..
0	0	0	9	9	9	9	9	..
0	0	0	9	9	9	9	9	..
0	0	0	9	9	9	9	9	..
0	0	0	9	9	9	9	9	..
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	..

$g(x, y) = w(x, y) \otimes f(x, y)$

0	0	0	0	0	0			
0	1	2	3	3	3			
0	2	4	6	6	6			
0	3	6	9	9	9			
0	3	6	9	9	9			
0	3	6	9	9	9			
0	3	6	9	9	9			
0								



Average Filter : Low pass filter

○ Averaging filters

mask values

$1/9$	$1/9$	$1/9$
$1/9$	$1/9$	$1/9$
$1/9$	$1/9$	$1/9$

mask values

$1/25$	$1/25$	$1/25$	$1/25$	$1/25$
$1/25$	$1/25$	$1/25$	$1/25$	$1/25$
$1/25$	$1/25$	$1/25$	$1/25$	$1/25$
$1/25$	$1/25$	$1/25$	$1/25$	$1/25$
$1/25$	$1/25$	$1/25$	$1/25$	$1/25$



Input

3X3

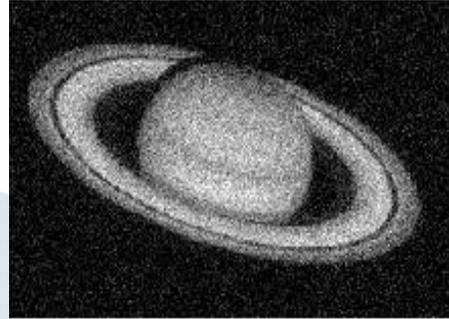
5X5

7X7

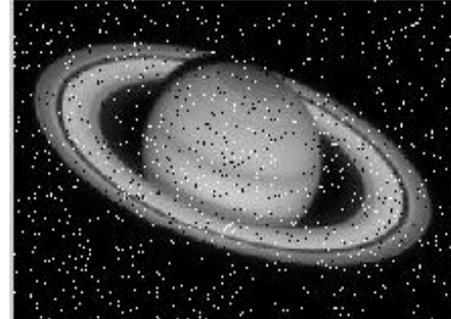


Matlab code

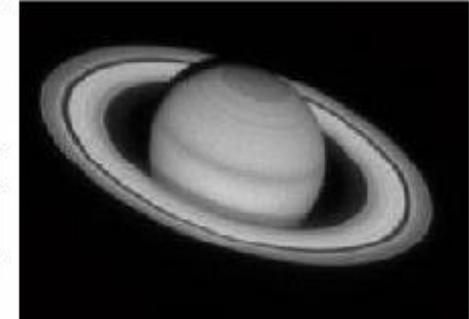
```
a=imread('saturn.jpg');  
b=imread('nsp saturn.jpg');  
c=imread('ngs saturn.jpg');  
;  
k=ones(5)/25;  
af=imfilter(a, k);  
figure, imshow(af)  
bf=imfilter(b, k);  
figure, imshow(bf)  
cf=imfilter(c, k);  
figure, imshow(cf)
```



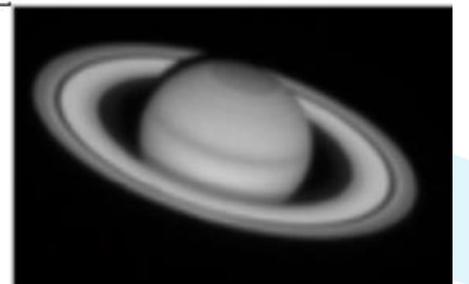
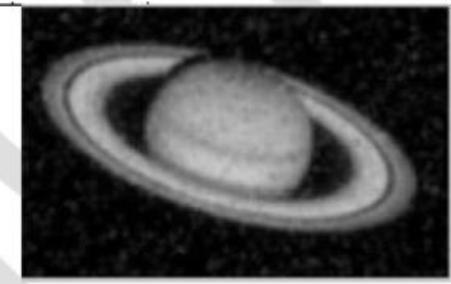
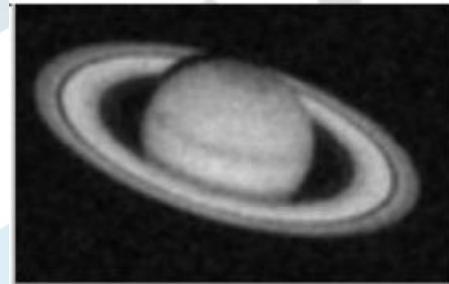
إضافة تشويش غاوص



إضافة تشويش الملح والفلفل



الصورة الأصلية



```
import cv2
import numpy as np
import os

# --- Function to apply a filter (imfilter in MATLAB) ---
def apply_filter(img, kernel):
    """Applies a 2D convolution (filtering) to an image."""
    filtered_img = cv2.filter2D(img, -1, kernel) # -1 for same depth/type as input
    return filtered_img

# Image paths (assuming they are in the current directory)
image_dir = '.'
original_image_name = 'saturn.jpg'
sp_image_name = 'npsaturn.jpg'
gn_image_name = 'ngsaturn.jpg'

# Construct full image paths
original_image_path = os.path.join(image_dir, original_image_name)
sp_image_path = os.path.join(image_dir, sp_image_name)
gn_image_path = os.path.join(image_dir, gn_image_name)
```



--- Load Images ---

Original Grayscale Image

```
a = cv2.imread(original_image_path, cv2.IMREAD_GRAYSCALE)
```

if a is None:

```
    print(f"Error: Could not load original image: {original_image_path}")
```

```
    exit()
```

Salt and Pepper Image

```
b = cv2.imread(sp_image_path, cv2.IMREAD_GRAYSCALE)
```

if b is None:

```
    print(f"Error: Could not load salt and pepper image: {sp_image_path}")
```

```
    exit()
```

Gaussian Noise Image

```
c = cv2.imread(gn_image_path, cv2.IMREAD_GRAYSCALE)
```

if c is None:

```
    print(f"Error: Could not load Gaussian noise image: {gn_image_path}")
```

```
    exit()
```



```
# --- Define the Kernel ('ones(5)/25' in MATLAB) ---  
k = np.ones((5, 5), np.float32) / 25 # Creates a 5x5 averaging kernel
```

```
# --- Apply Filters ---
```

```
# Filter the original image  
af = apply_filter(a, k)  
cv2.imshow('Filtered Original', af)  
cv2.waitKey(0)
```

```
# Filter the salt and pepper image  
bf = apply_filter(b, k)  
cv2.imshow('Filtered Salt & Pepper', bf)  
cv2.waitKey(0)
```

```
# Filter the Gaussian noise image  
cf = apply_filter(c, k)  
cv2.imshow('Filtered Gaussian', cf)  
cv2.waitKey(0)
```

```
cv2.destroyAllWindows()
```





Application : noise reduction and image smoothing
Disadvantage: lose sharp details



□ مرشح يأخذ بالحسبان مقدار قرب البكسل الجار من البكسل الأصلي ويعطي للبكسل الأصلي قيمة أكبر من جيرانه ليكون له وللجيران الأقرب نسب أكبر من المشاركة مقارنة مع البكسلات البعيدة

$$\frac{1}{16} \times$$

1	2	1
2	4	2
1	2	1



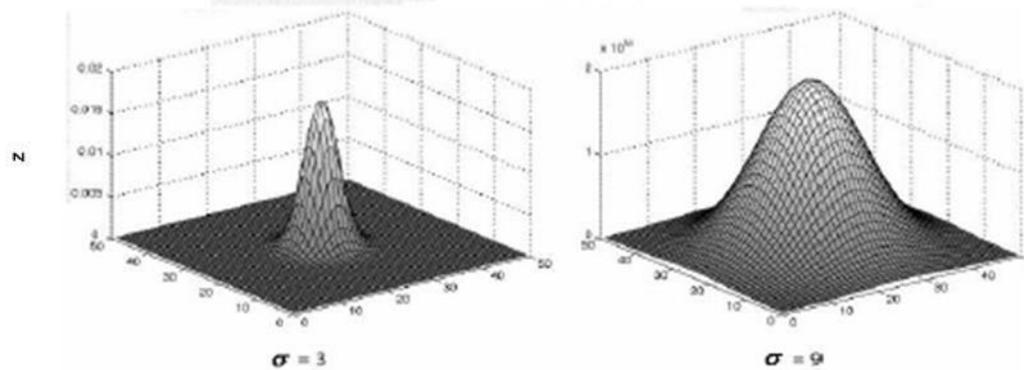
- يمثل مرشح قيمة متوسطة مثقل يزداد وزن البكسل فيه كلما اقتربنا من البكسل المركزي
- يستخدم في إزالة التشويش وتنعيم الصور ويستخدم قناعاً يمثل شكل توزيع غاوص



$$(1-D): g(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\bar{x})^2}{2\sigma^2}}$$

يمثل σ الانحراف المعياري لتوزيع غاوص

□ توزيع غاوص ثنائي البعد



$$(2-D): g(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$



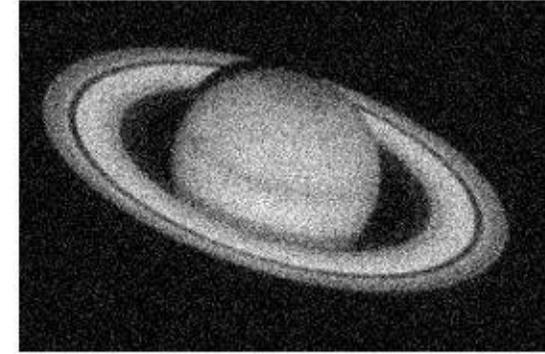
- يقوم مرشح غاوص بتنعيم الصورة وإزالة التشويش مثله مثل مرشح القيمة المتوسطة ولكن يختلف عنه بأنه يستخدم مرشح غاوصي أو قناعا له شكل توزع غاوص
- تحدد درجة التنعيم للصورة عن طريق تغيير قيمة الانحراف المعياري

Matlab code

```
a=imread('ngs saturn.jpg');  
figure, imshow(a)  
k=fspecial('gaussian',[5 5],2);  
af=imfilter(a, k, 'conv');
```



الصورة بعد تطبيق مرشح غاوص



صورة تحتوي على تشويش غاوص

- يقوم برنامج الماتلاب بتوليد المصفوفة الخاصة بقناع مرشح غاوص باستخدام التعليمة fspecial



```
Mask = fspecial ( ' type ' );  
Image = imfilter ( gray image , mask ) ;
```

```
'average'   averaging filter  
'disk'     circular averaging filter  
'gaussian' Gaussian lowpass filter  
'laplacian' filter approximating the 2-D Laplacian operator  
'log'      Laplacian of Gaussian filter  
'motion'   motion filter  
'prewitt'  Prewitt horizontal edge-emphasizing filter  
'sobel'    Sobel horizontal edge-emphasizing filter  
'unsharp'  unsharp contrast enhancement filter
```



- يقوم مرشح غاوص بتنعيم الصورة على نحو مشابه لمرشح القيمة المتوسطة. ويتمّ تحديد درجة التنعيم للصورة عن طريق تغيير قيمة الانحراف المعياري؛
- إذ تؤدي زيادة الانحراف المعياري لتوزع غاوص إلى زيادة درجة التنعيم في الصورة.
- ويعدّ مرشح غاوص مرشح قيمة متوسطة مثقل يزداد وزن البكسل فيه كلما اقترب من البكسل المركزي.
- لذلك ينتج عنه تنعيم سلس وحفاظ على تفاصيل الصورة على نحو أفضل من مرشح القيمة المتوسطة ويستخدم لهذا كخطوة أولى في خوارزميات الكشف عن الحواف) مثل كاشف الحواف كاني
- يستخدم مرشح غاوص عادةً كمرشح ترددات منخفضة إلا أنه يمكن تطبيقه بشكل مباشر على الصورة وذلك عن طريق توليد قناع تمثل أوزانه القيم التقريبية لتوزع غاوص.



Gaussian filters



$5 \times 5, \sigma = 0.5$



$11 \times 11, \sigma = 1$



$5 \times 5, \sigma = 2$



$11 \times 11, \sigma = 5$



```
import cv2
import numpy as np
import os

# --- Function to apply a filter (imfilter in MATLAB) ---
def apply_filter(img, kernel):
    """Applies a 2D convolution (filtering) to an image."""
    filtered_img = cv2.filter2D(img, -1, kernel) # -1 for same
    depth/type as input
    return filtered_img

# Image path
image_dir = '.' # Current directory
gn_image_name = 'ngsaturn.jpg'
gn_image_path = os.path.join(image_dir, gn_image_name)
```

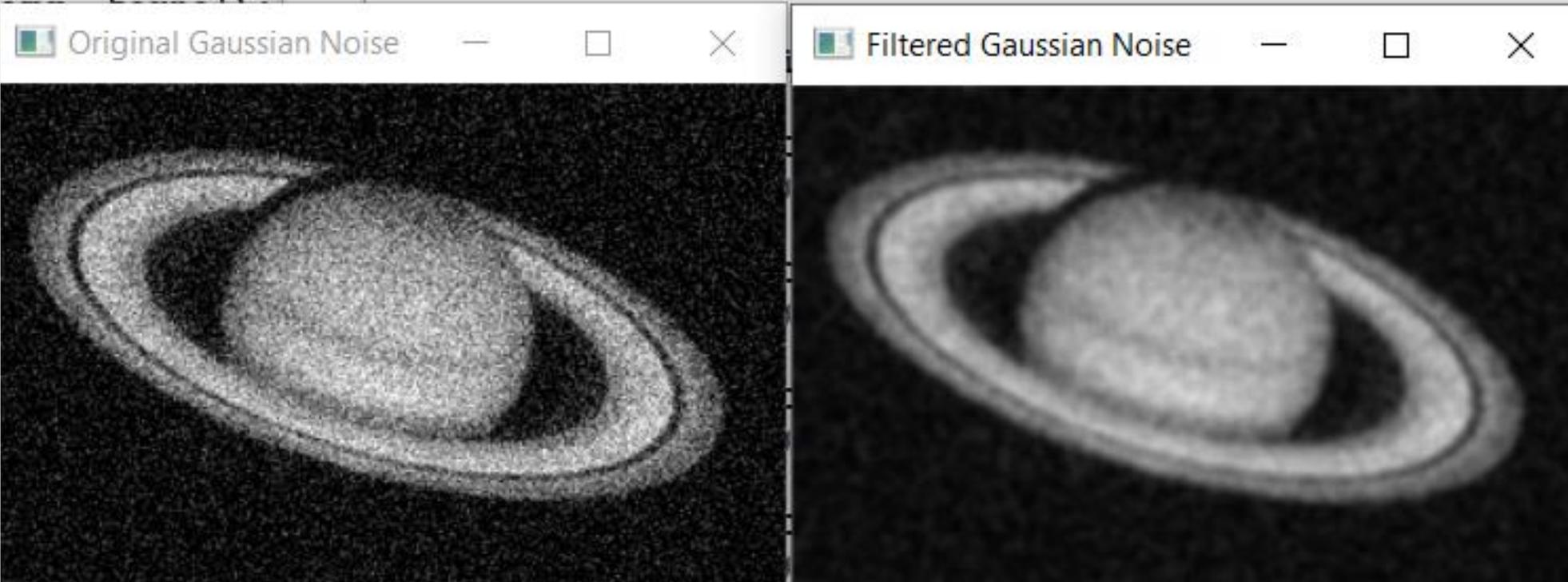
```
# --- Load Image ---
a = cv2.imread(gn_image_path, cv2.IMREAD_GRAYSCALE)
if a is None:
    print(f"Error: Could not load image: {gn_image_path}")
    exit()

# Display original image
cv2.imshow('Original Gaussian Noise', a)
cv2.waitKey(0)
```



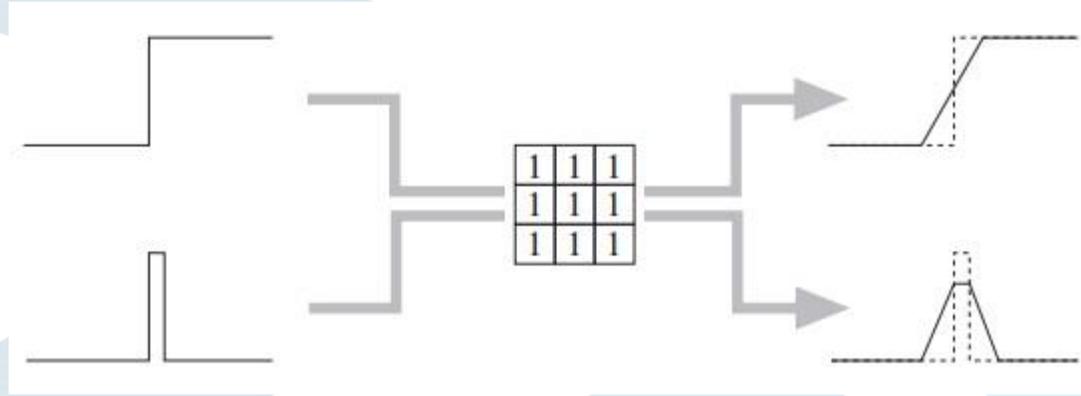
```
# --- Create Gaussian Kernel (fspecial('gaussian', [5 5], 2) in MATLAB) ---  
# Sigma = 2, kernel size [5,5]  
k = cv2.getGaussianKernel(5, 2) # Creates a 1D Gaussian kernel, then combines it.  
k = k @ k.T # Creates a 2D Gaussian kernel by outer product.  
  
# --- Apply Filter ---  
af = apply_filter(a, k) # Apply the Gaussian filter.  
  
# Display the filtered image  
cv2.imshow('Filtered Gaussian Noise', af)  
cv2.waitKey(0)  
  
cv2.destroyAllWindows()
```





المرشحات غير الخطية

□ من سيئات المرشحات الخطية أنها تقوم بتنعيم كل محتويات وتفصيل الصورة بما فيها النقاط والخطوط والحواف



□ بعض المرشحات غير الخطية التي تستخدم في التخلص من التشويش

✓ مرشحات القيمة العظمى والصغرى Max and Min Filters

✓ مرشح الوسيط الهندسي Median filter

✓ مرشح الوسيط الهندسي المثقل Weighted median filter



Matlab code

```
a=imread('nsp saturn.jpg');
```

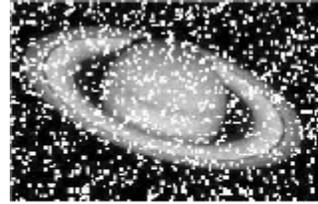
```
figure, imshow(a)
```

```
amin=uint8(colfilt(a,[3 3],'sliding', @min));
```

```
figure, imshow(amin)
```

```
amax=uint8(colfilt(a,[3 3],'sliding', @max));
```

```
figure, imshow(amax)
```



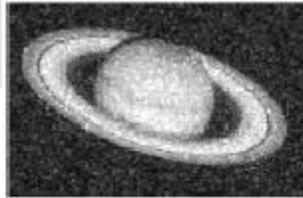
مرشح القيمة القصوى



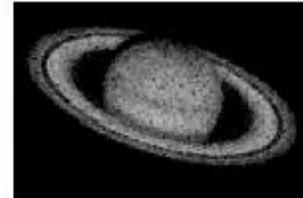
مرشح القيمة الصغرى



صورة ذات تشويش الملح والفلفل



مرشح القيمة القصوى



مرشح القيمة الصغرى



صورة ذات تشويش غاوسي

- مرشح القيمة الصغرى قد استطاع التخلص من النقاط البيضاء - في الصورة "الملح" لأنه سيتم استبدال أيّ بكسل أبيض وحيد في الصورة بقيمة بكسل آخر من جواره تكون قيمته أصغر ولكن هذا المرشح قد زاد في الوقت نفسه من النقاط السوداء في الصورة "الفلفل".
- ويظهر مرشح القيمة القصوى التأثير المعاكس لسابقه فوجود أيّ بكسل أبيض وحيد في نافذة المرشح سيعطي قيمته للبكسل وبالاتي سيزيد من تشويش الملح أما النقط السوداء في الصورة فستختفي تماماً.





```
import cv2
import numpy as np
import os
# --- Function to apply filter (using sliding window and min/max) ---
def apply_min_max_filter(img, kernel_size, operation):
    """Applies a min or max filter using a sliding window."""
    rows, cols = img.shape
    filtered_img = np.zeros_like(img)

    half_kernel = kernel_size // 2 # Integer division for kernel radius

    for r in range(1, rows-1):
        for c in range(1, cols-1):
            # Define the window boundaries
            row_start = max(0, r - half_kernel)
            row_end = min(rows, r + half_kernel)
            col_start = max(0, c - half_kernel)
            col_end = min(cols, c + half_kernel)

            # Extract the window (the kernel area)
            window = img[row_start:row_end, col_start:col_end]

            # Apply min or max operation
            if operation == 'min':
                filtered_img[r, c] = np.min(window)
            elif operation == 'max':
                filtered_img[r, c] = np.max(window)
            else:
                raise ValueError("Invalid operation. Choose 'min' or 'max'.")

    return filtered_img.astype(np.uint8) # Ensure uint8 output
```

```
# Image path
image_dir = '.'
#sp_image_name = 'nsp saturn.jpg'
sp_image_name = 'ngsaturn.jpg'
sp_image_path = os.path.join(image_dir, sp_image_name)

# --- Load Image ---
a = cv2.imread(sp_image_path, cv2.IMREAD_GRAYSCALE)
if a is None:
    print(f"Error: Could not load image: {sp_image_path}")
    exit()

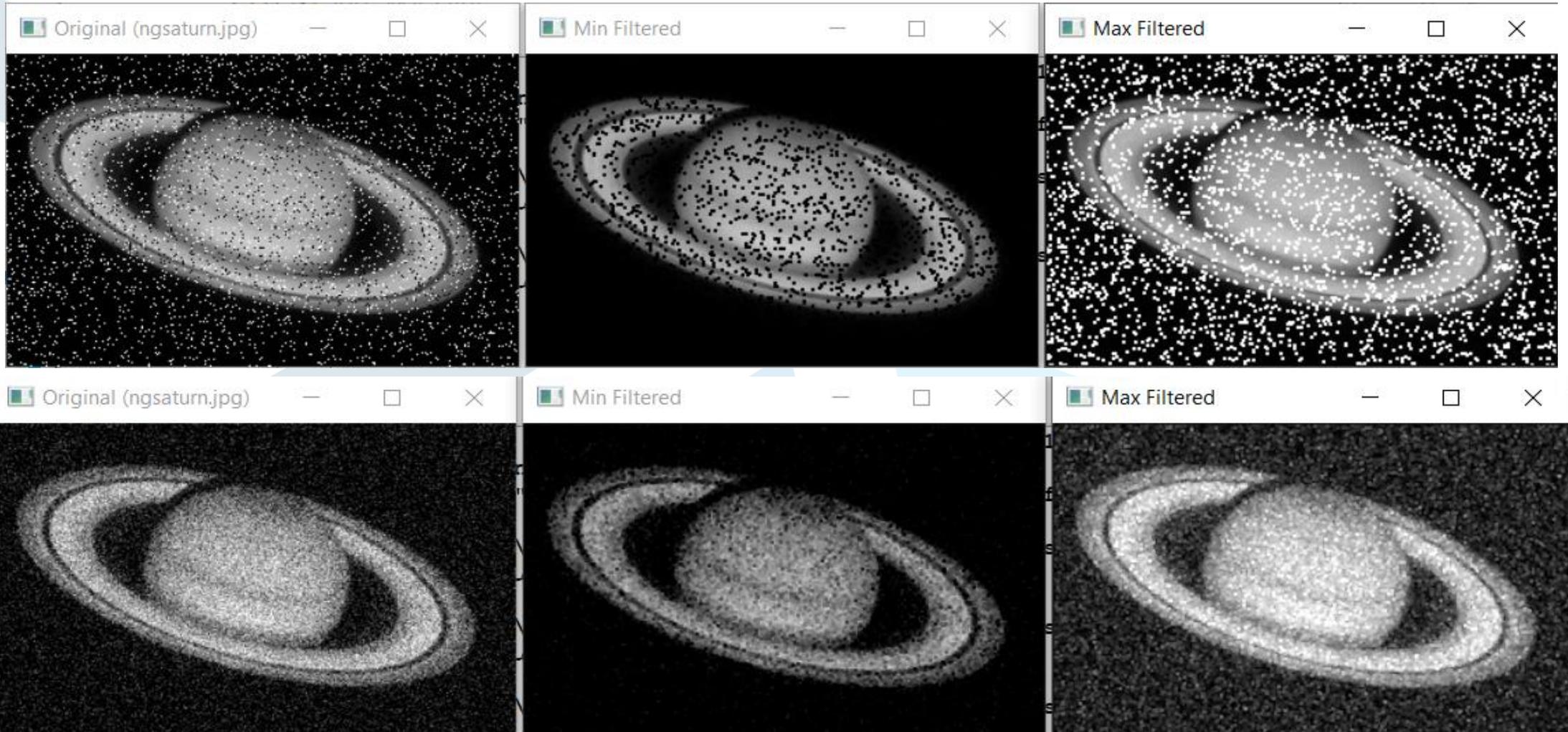
# Display Original Image
# cv2.imshow('Original (nsp saturn.jpg)', a)
cv2.imshow('Original (ngsaturn.jpg)', a)
cv2.waitKey(0)

# --- Apply Min Filter (3x3) ---
amin = apply_min_max_filter(a, 3, 'min')
cv2.imshow('Min Filtered', amin)
cv2.waitKey(0)

# --- Apply Max Filter (3x3) ---
amax = apply_min_max_filter(a, 3, 'max')
cv2.imshow('Max Filtered', amax)
cv2.waitKey(0)

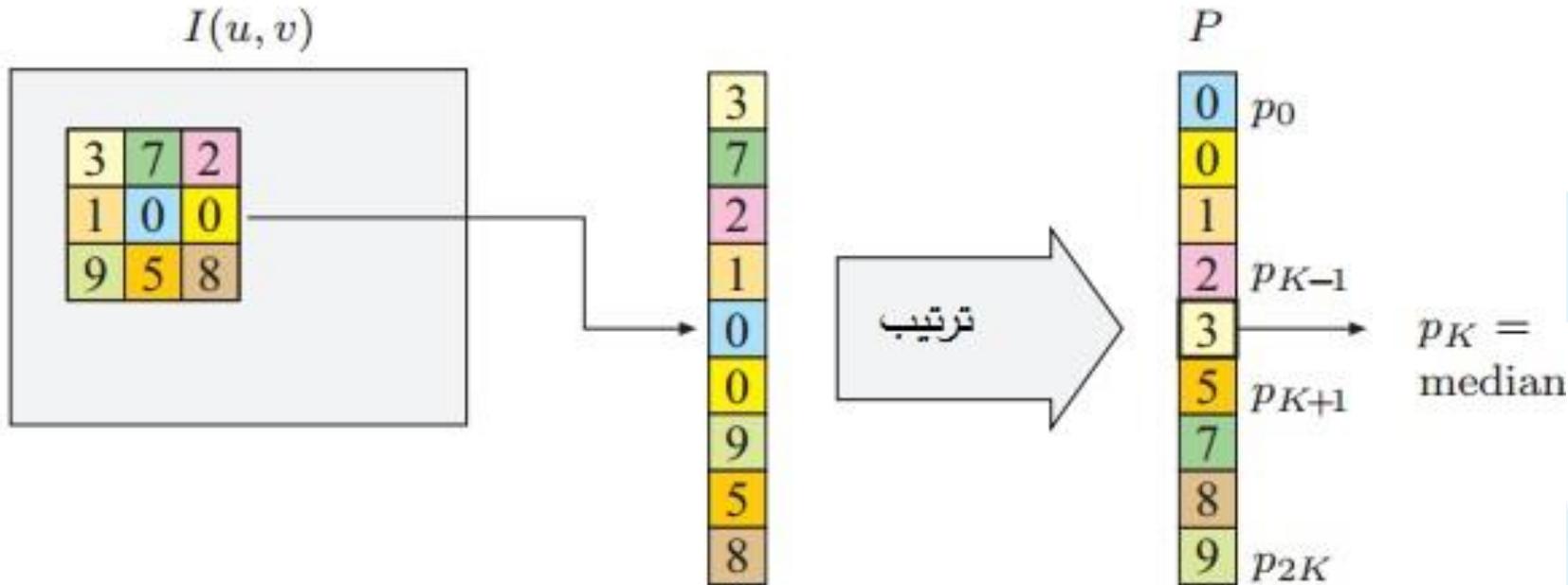
cv2.destroyAllWindows()
```





المرشحات غير الخطية مرشح الوسيط الهندسي Median Filter or Order-Statistic Filter

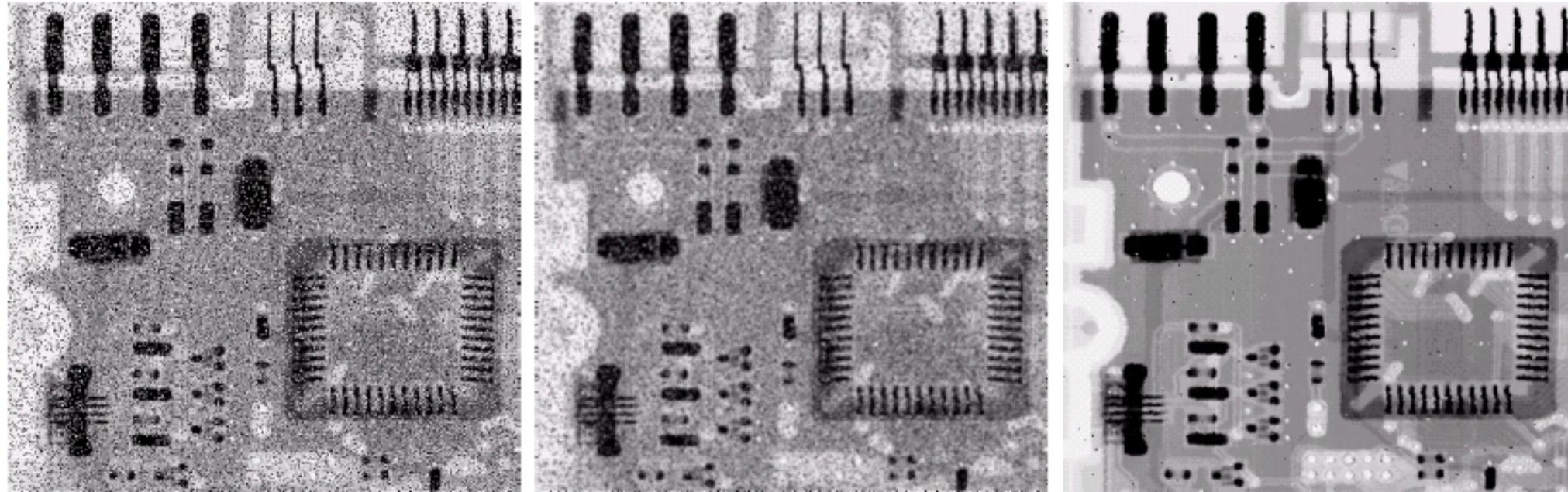
$$I'(u, v) = \text{median}\{I(u + i, v + j) \mid (i, j) \in R\}$$



- إن الهدف الرئيس من المرشحات هو التخلص من التشويش مع إبقاء كل تفاصيل الصورة ولكن هذا السبب صعب التحقق لأن المرشحات لا تستطيع تمييز بين محتويات الصورة المهمة وغير المهمة.
- ويعدّ مرشح الوسيط الهندسي من أكثر المرشحات شعبية في مجال التخلص من التشويش.
- عند تطبيق مرشح الوسيط الحسابي يتم استبدال كل بكسل في الصورة بقيمة الوسيط الهندسي لنافذة المرشح R وفق المعادلة الآتية:



Order-Statistic Filters or Median Filter



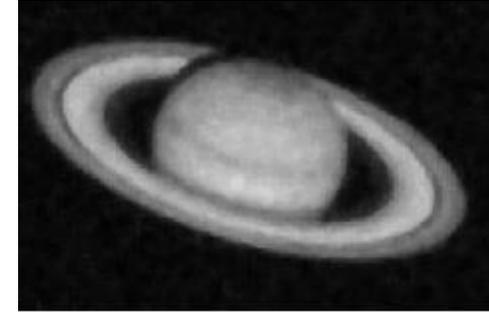
a b c

FIGURE 3.37 (a) X-ray image of circuit board corrupted by salt-and-pepper noise. (b) Noise reduction with a 3×3 averaging mask. (c) Noise reduction with a 3×3 median filter. (Original image courtesy of Mr. Joseph E. Pascente, Lixi, Inc.)



```
a=imread('ngsaturn.jpg');  
figure, imshow(a)  
af=medfilt2(a,[7 7]);  
figure, imshow(af)
```

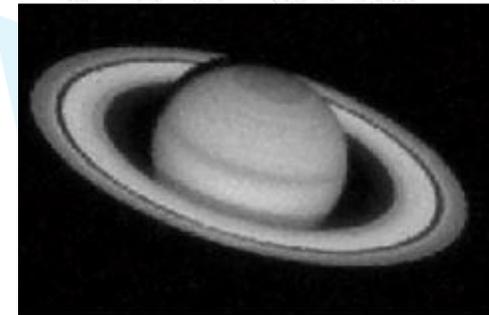
- أن مرشح الوسيط الهندسي قد استطاع التغلب على - بقية المرشحات السابقة في الحفاظ على تفاصيل الصورة مع القضاء على التشويش
- وخصوصاً عندما تكون بكسلات التشويش معزولة وذات قيمة مرتفعة جداً أو منخفضة جداً (كتشويش الملح والفلل) أما في حالة تشويش غاوص فقد استطاع هذا المرشح إزالة التشويش ولكن على حساب التقليل من جودة الصورة.



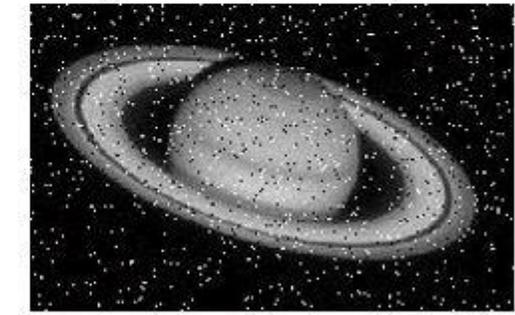
تطبيق مرشح الوسيط الهندسي



صورة ذات تشويش غاوصي



تطبيق مرشح الوسيط الهندسي



صورة ذات تشويش الملح والفلل



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

```
# Read the image
```

```
image = cv2.imread('ngs saturn.jpg')  
#image = cv2.imread('nsp saturn.jpg')
```

```
# Display the original image
```

```
cv2.imshow('Original Image', image)
```

```
# Apply median filtering with a 7x7 kernel
```

```
filtered_image = cv2.medianBlur(image, 7)
```

```
# Display the filtered image
```

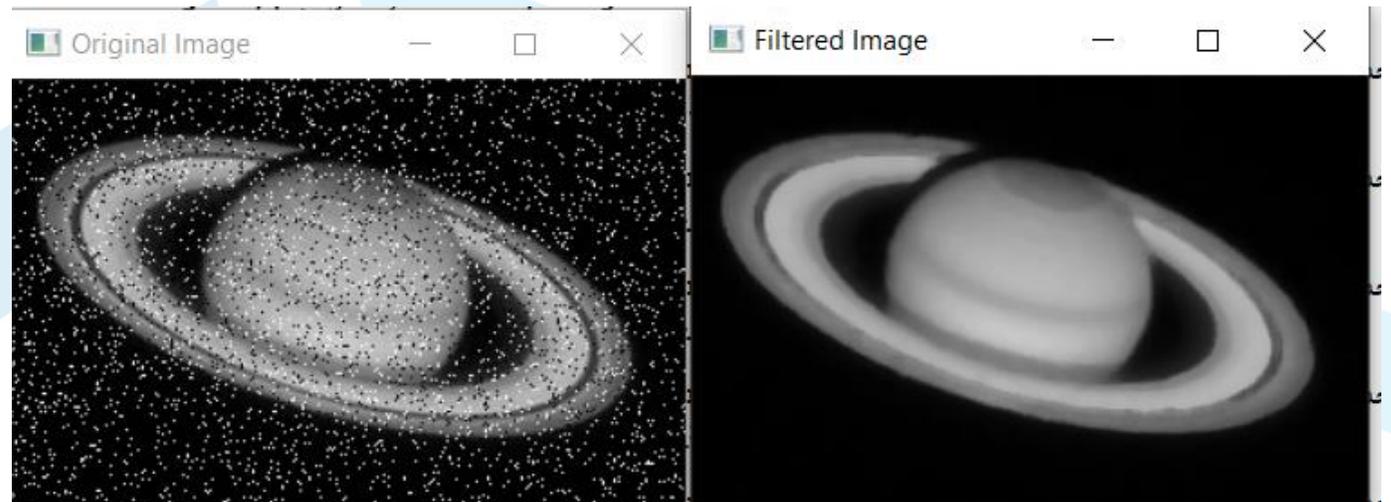
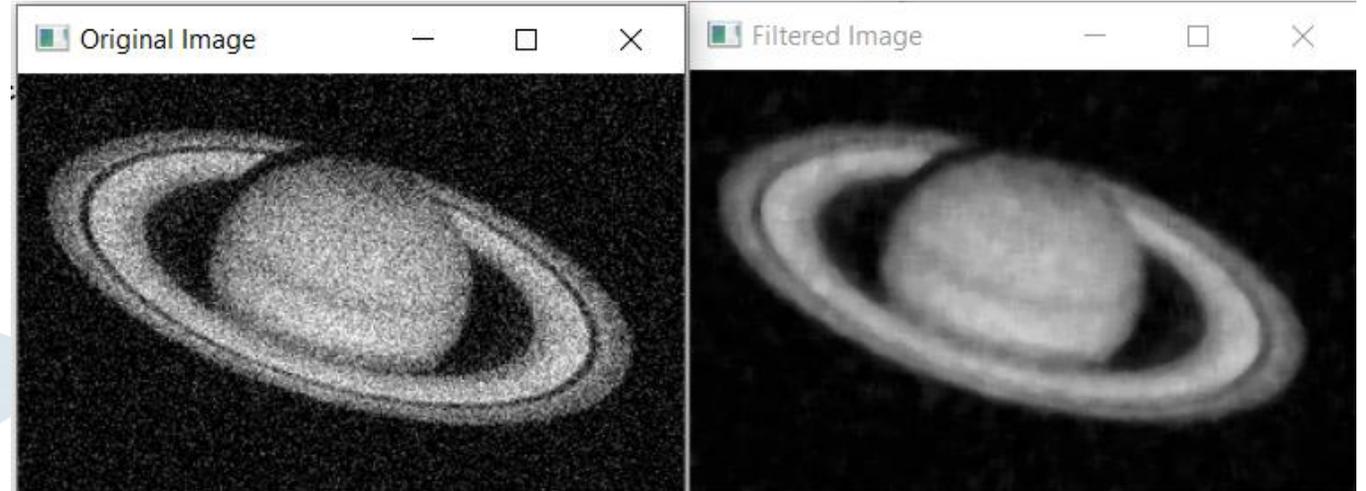
```
cv2.imshow('Filtered Image', filtered_image)
```

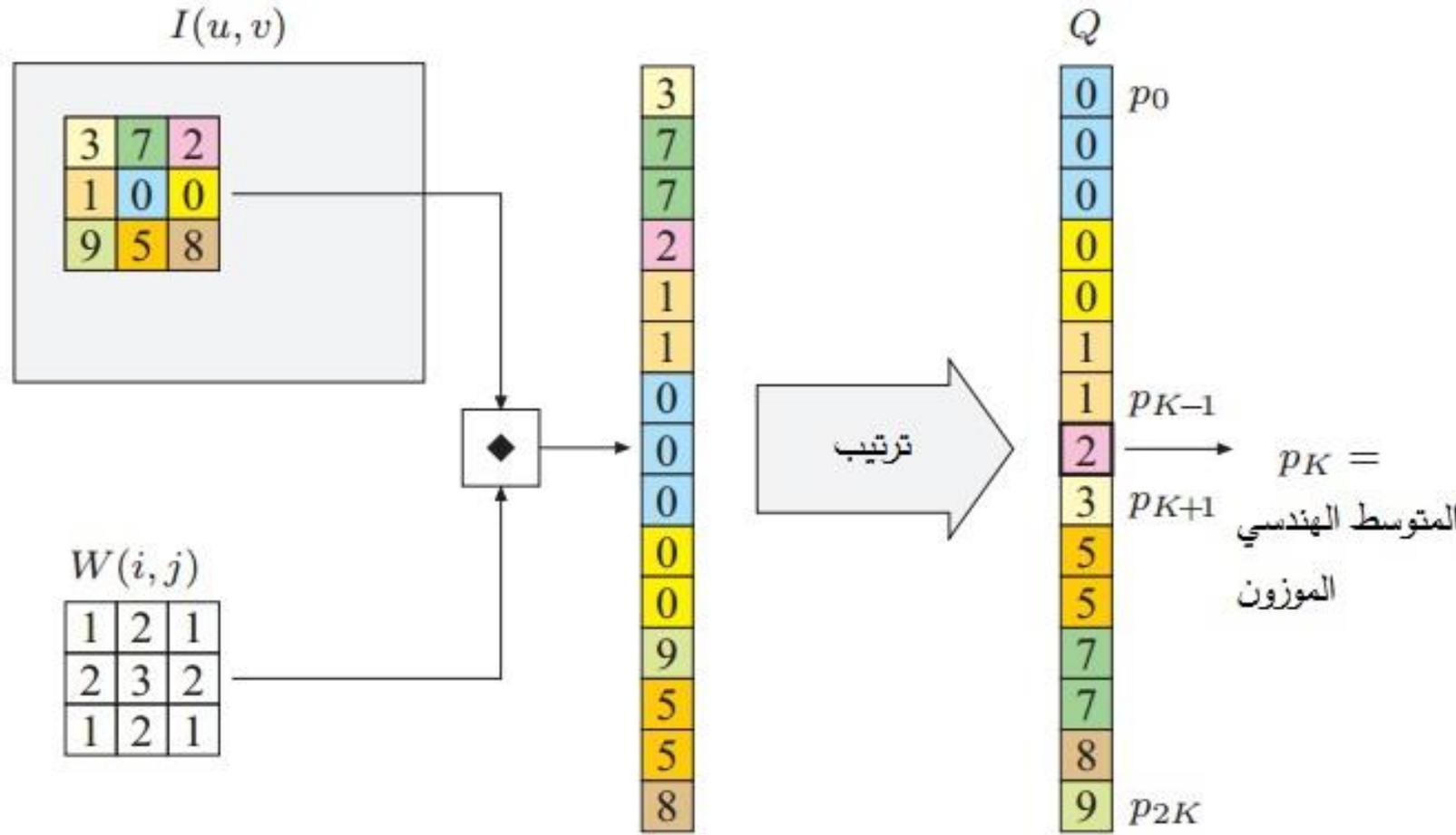
```
# Wait for a key press and close the image
```

```
windows
```

```
cv2.waitKey(0)
```

```
cv2.destroyAllWindows()
```



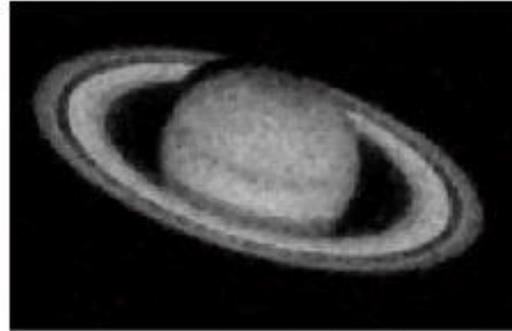


- تستدعي هذه المصفوفة توسعة شعاع البكسل ذي الطول 9 عناصر ليصبح طوله 15 عنصر والذي يساوي مجموع عناصر مصفوفة الأوزان W .
- يحدد مرشح الوسيط الهندسي المثقل أوزاناً منفصلة لكل موقع في نافذة المرشح والتي تفسر على أنها عدد الأصوات المعطاة لهذا الموقع.



```
a=imread('ngsaturn.jpg');  
figure, imshow(a)  
af=ordfilt2(a,5,ones(5));
```

$B = \text{ordfilt2}(A, \text{order}, \text{domain})$ replaces each element in A by the orderth element in the sorted set of neighbors specified by the nonzero elements in domain.



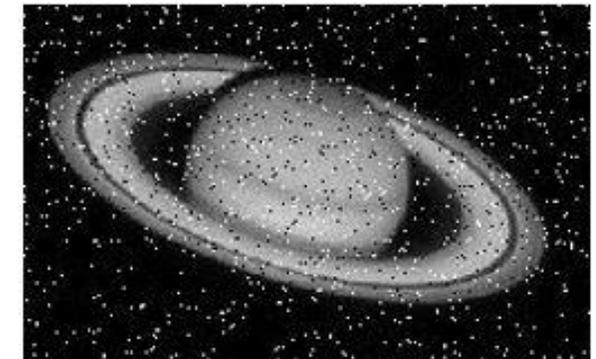
تطبيق مرشح الوسيط الهندسي



صورة تحتوي على تشويش غاوص



تطبيق مرشح الوسيط الهندسي



صورة تحتوي على تشويش الملح والفلفل



Type of Filtering Operation	MATLAB code	Neighborhood	Sample Image Data, Indica																		
Median filter	<code>B = ordfilt2(A,5,ones(3,3))</code>	<table border="1"> <tr><td>1</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>1</td></tr> </table>	1	1	1	1	1	1	1	1	1	<table border="1"> <tr><td>88</td><td>16</td><td>56</td></tr> <tr><td>5</td><td>3</td><td>30</td></tr> <tr><td>21</td><td>63</td><td>42</td></tr> </table>	88	16	56	5	3	30	21	63	42
1	1	1																			
1	1	1																			
1	1	1																			
88	16	56																			
5	3	30																			
21	63	42																			
Minimum filter	<code>B = ordfilt2(A,1,ones(3,3))</code>	<table border="1"> <tr><td>1</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>1</td></tr> </table>	1	1	1	1	1	1	1	1	1	<table border="1"> <tr><td>88</td><td>16</td><td>56</td></tr> <tr><td>5</td><td>3</td><td>30</td></tr> <tr><td>21</td><td>63</td><td>42</td></tr> </table>	88	16	56	5	3	30	21	63	42
1	1	1																			
1	1	1																			
1	1	1																			
88	16	56																			
5	3	30																			
21	63	42																			
Maximum filter	<code>B = ordfilt2(A,9,ones(3,3))</code>	<table border="1"> <tr><td>1</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>1</td></tr> </table>	1	1	1	1	1	1	1	1	1	<table border="1"> <tr><td>88</td><td>16</td><td>56</td></tr> <tr><td>5</td><td>3</td><td>30</td></tr> <tr><td>21</td><td>63</td><td>42</td></tr> </table>	88	16	56	5	3	30	21	63	42
1	1	1																			
1	1	1																			
1	1	1																			
88	16	56																			
5	3	30																			
21	63	42																			
Minimum of north, east, south, and west neighbors	<code>B = ordfilt2(A,1,[0 1 0; 1 0 1; 0 1 0])</code>	<table border="1"> <tr><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>0</td></tr> </table>	0	1	0	1	0	1	0	1	0	<table border="1"> <tr><td>88</td><td>16</td><td>56</td></tr> <tr><td>5</td><td>3</td><td>30</td></tr> <tr><td>21</td><td>63</td><td>42</td></tr> </table>	88	16	56	5	3	30	21	63	42
0	1	0																			
1	0	1																			
0	1	0																			
88	16	56																			
5	3	30																			
21	63	42																			



Purpose of the Order Statistic Filter

- **Noise Reduction:** Order statistic filters, like the one implemented here, effectively reduce noise in images, particularly impulse noise (often referred to as salt-and-pepper noise). By selecting the 5th smallest value, the filter can effectively suppress outlier pixel values while maintaining the overall structure of the image.
- **Edge Preservation:** Unlike linear filters (like the mean filter), order statistic filters tend to preserve edges better because they do not average out the pixel values, which can blur edges.



نهاية المحاضرة

