Computer vision

المحاضرة التاسعة

Image Enhancement

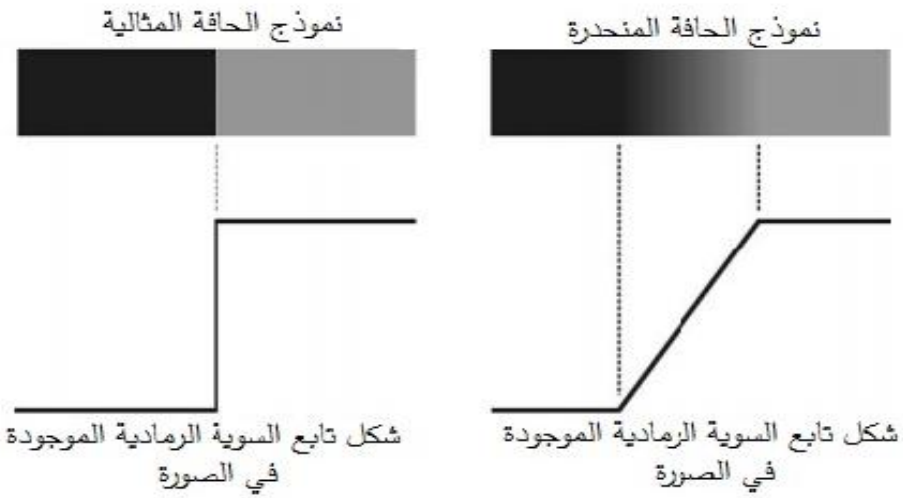
Spatial Domain Methods

العمليات على مستوى القناع أو جيران البكسل باستخدام المرشحات المكانية

د. عيسى الغنام د. إياد حاتم

الحواف

- الحدود الفاصلة بين منطقتين تتميّزان بخواص مختلفة في الصورة

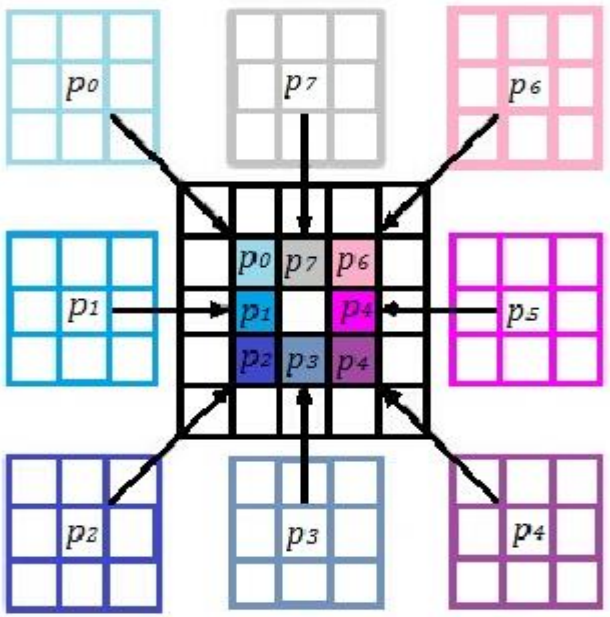


- مرشح القيمة المتوسطة "هو تابع التكامل في حالته المتقطعة" يؤدي إلى تنعيم الصورة وبالآتي فقدان بعض التفاصيل

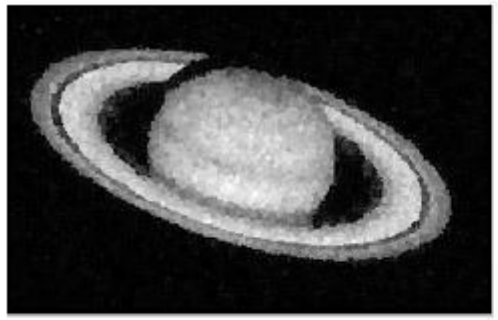
- لإظهار وتوضيح التفاصيل يتمّ اللجوء إلى المرشحات التفاضلية

مرشحات التنعيم مع الحفاظ على الحواف
مرشح ناغامود
Nagamod filter

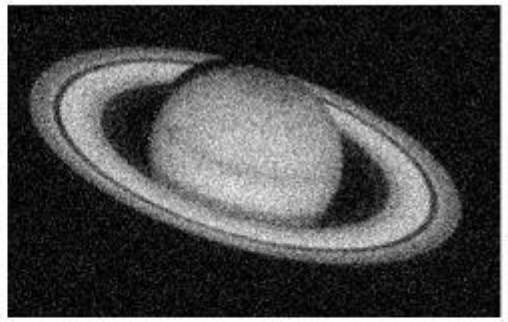
- يطبق وفق الخطوات:



- ✓ يعرف قناع المرشح H(m,n) لجيران البكسل المركزي بحيث لا يقلّ حجم هذا المرشح عن 5x5
- ✓ تحسب القيمة المتوسطة والتباين من أجل كلّ بكسل
- ✓ تكون القيمة الجديدة للبكسل المركزي هي القيمة المتوسطة للبكسل الذي يكون التباين عنده أقل

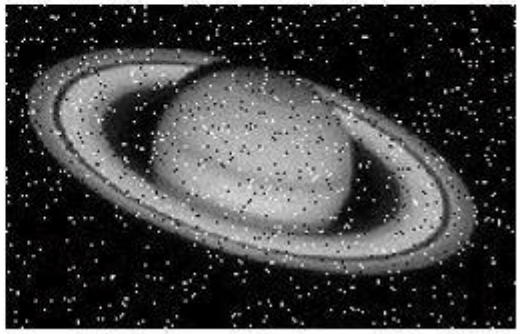


الصورة بعد تطبيق مرشح ناغامود



صورة تحتوي على تشويش غاوص

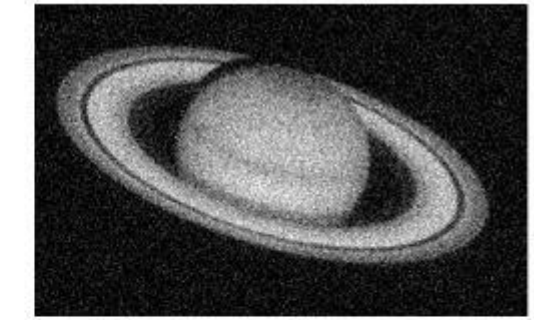
الصورة بعد تطبيق مرشح ناغامود



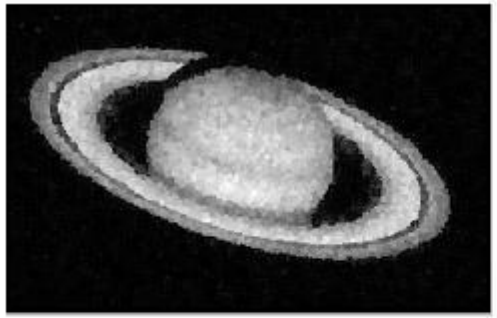
صورة تحتوي على تشويش الملح والفلفل

مرشح ناغامود
مثال

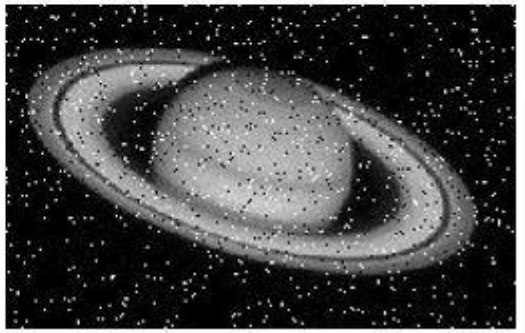
```
a=imread('nspsaturn.jpg');
figure, imshow(a)
[r c]=size(a);
af=zeros(size(a));
for i=3:r-3
for j=3:c-3
b=a(i-2:i+2,j-2:j+2);
b1=colfilt(b,[3 3],'sliding',@mean);
b2=colfilt(b,[3 3],'sliding',@var);
d=b1(find(b2==min(min(b2))));
af(i,j)=d;
end
end
```



صورة تحتوي على تشويش غاوص



الصورة بعد تطبيق مرشح ناغامود



صورة تحتوي على تشويش الملح والفلفل

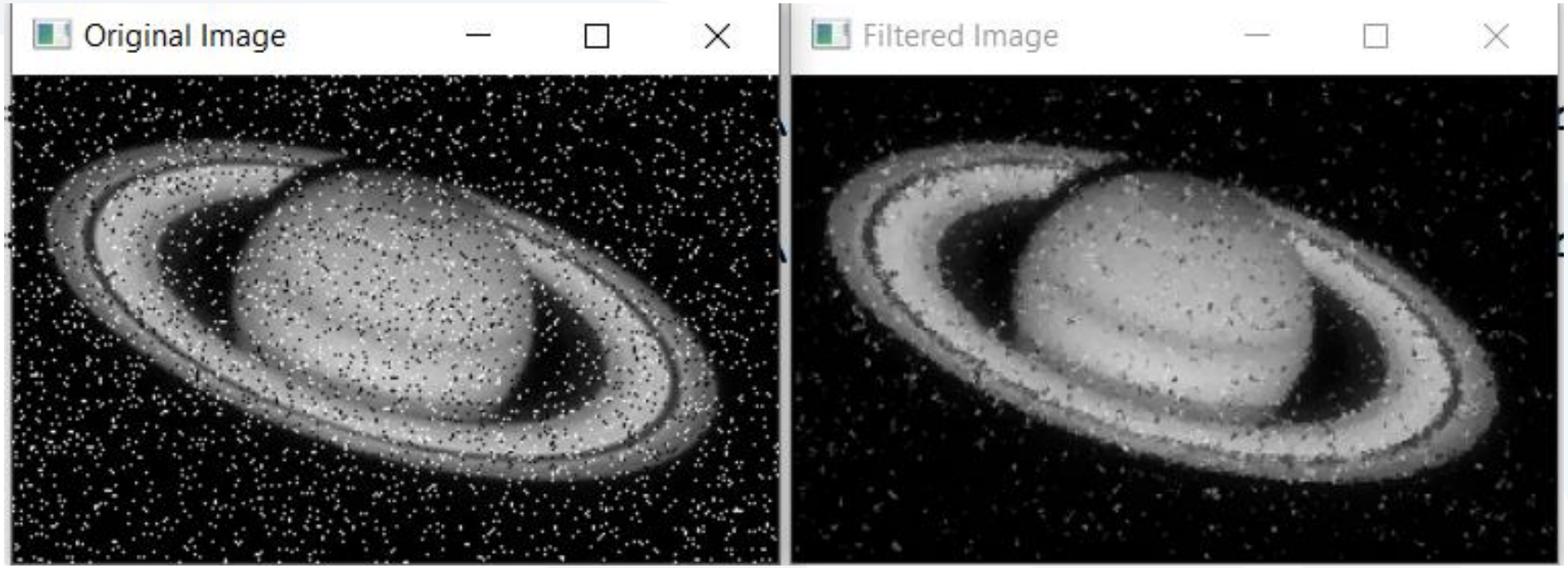
الصورة بعد تطبيق مرشح ناغامود

`B = colfilt(A,[m n],block_type,fun)`

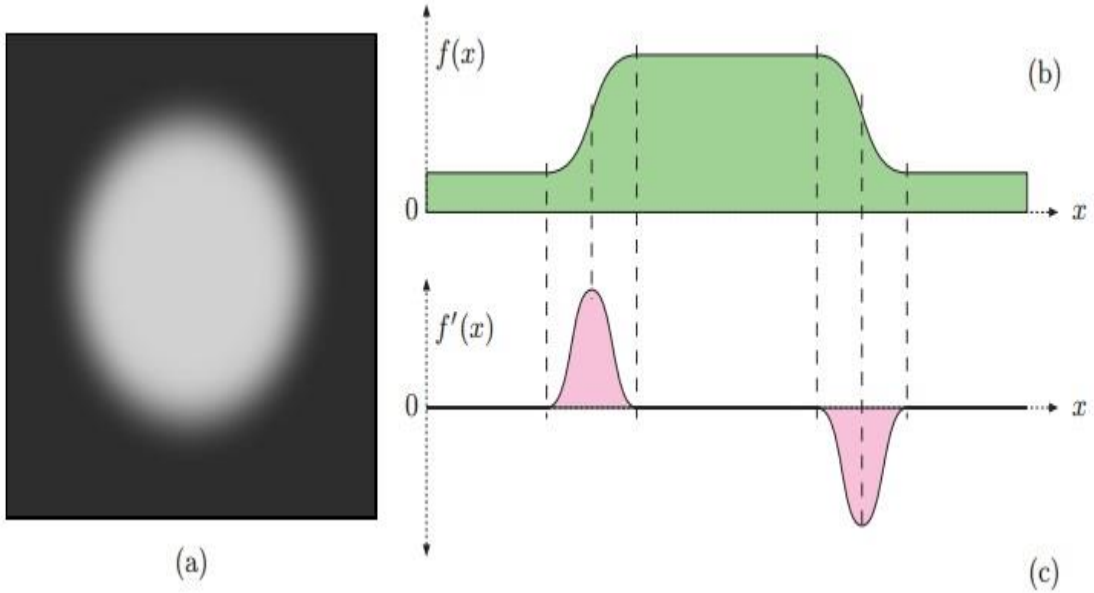
processes the image A by rearranging each m-by-n block of A into a column of a temporary matrix, and then applying the function fun to this matrix.

```
import cv2
import numpy as np

# Load the image (replace 'nspsaturn.jpg' with your actual image path)
a = cv2.imread('nspsaturn.jpg')
if a is None:
    print("Error: Could not load image.")
    exit()

# Convert to grayscale if it's not already
a = cv2.cvtColor(a, cv2.COLOR_BGR2GRAY)

# Display the image
cv2.imshow('Original Image', a)
cv2.waitKey(0)

# Get image dimensions
r, c = a.shape

# Initialize the output array
af = np.zeros_like(a, dtype=np.float64)
```

```
# Perform the local minimum variance filtering
# for i in range(2, r - 2):
    for j in range(2, c - 2):
        # Extract the 3x3 neighborhood
        b = a[i - 2:i + 2, j - 2:j + 2]

        # Calculate the mean and variance using OpenCV's boxFilter
        b1 = cv2.boxFilter(b, -1, (3, 3)) # Mean
        #`cv2.boxFilter` is a valuable tool for simple image smoothing
        #`-1`: Same data type as the input image.
        b2 = cv2.boxFilter((b - b1)**2, -1, (3, 3)) # Variance - manual calc for consistency

        # Find the index of the minimum variance and assign it to the output
        min_variance_index = np.unravel_index(np.argmin(b2), b2.shape)
        af[i, j] = b1[min_variance_index]
# Convert back to uint8 if needed for display
af = af.astype(np.uint8)
# Display the filtered image
cv2.imshow('Filtered Image', af)
cv2.waitKey(0)
cv2.destroyAllWindows()
#Save the image
cv2.imwrite('filtered_nspsaturn.jpg', af)
```


Original Image
Filtered Image

الكشف عن الحواف بالاعتماد على التدرج



- لتابع احادي البعد

$$f'(x) = \frac{df}{dx}(x)$$

$$\frac{df}{du}(u) \approx \frac{f(u+1) - f(u-1)}{(u+1) - (u-1)} = \frac{f(u+1) - f(u-1)}{2}$$

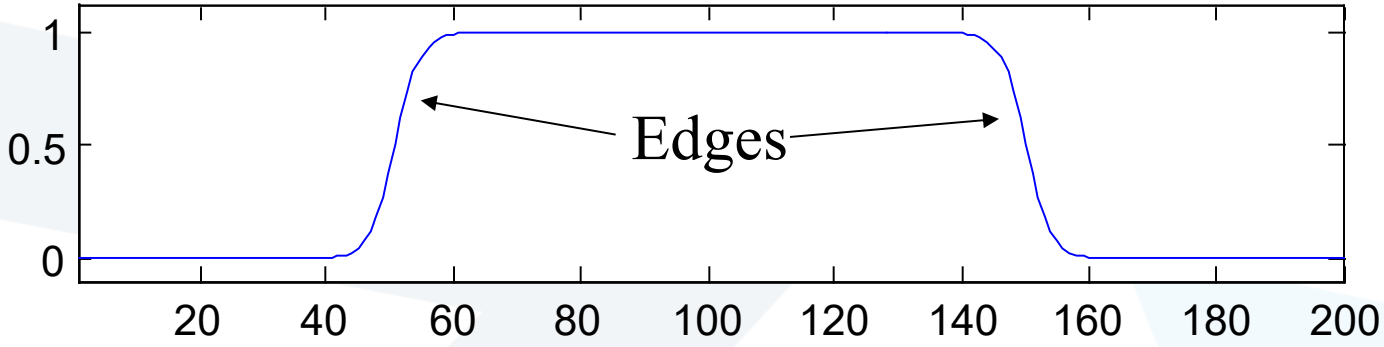
- لتابع ثنائي البعد يحسب التدرج وطويلة التدرج

$$\nabla I(u,v) = \begin{bmatrix} \frac{\partial I}{\partial u}(u,v) \\ \frac{\partial I}{\partial v}(u,v) \end{bmatrix} \qquad |\nabla I|(u,v) = \sqrt{\left(\frac{\partial I}{\partial u}(u,v)\right)^2 + \left(\frac{\partial I}{\partial v}(u,v)\right)^2}$$

First Order Derivative

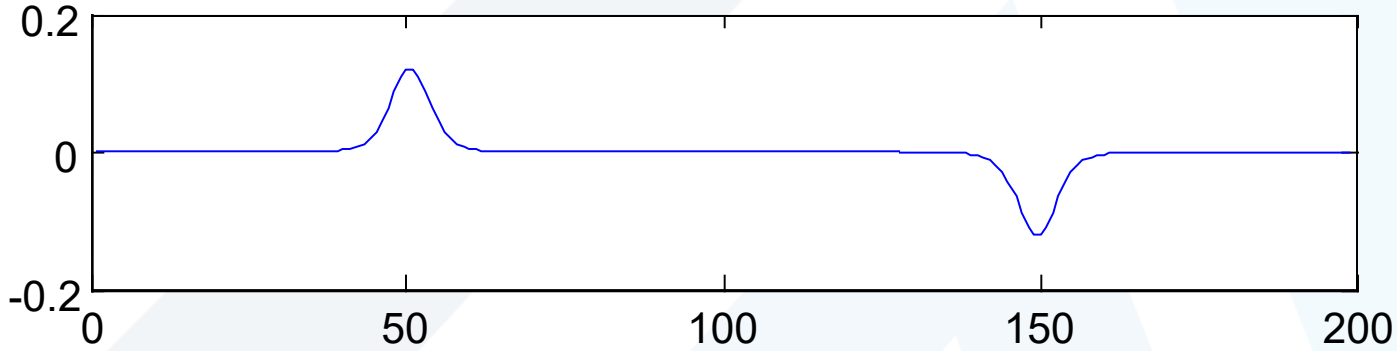
Intensity profile

$p(x)$

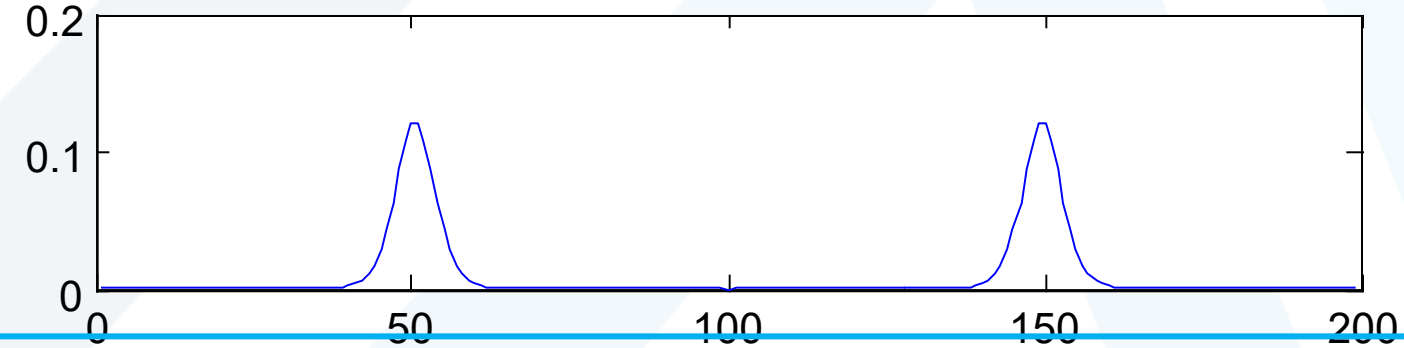


1st derivative

$\frac{dp}{dx}$



$\left|\frac{dp}{dx}\right|$



مرشحات الكشف عن الحواف
مرشحات المشتق الجزئي الأول
First Order Partial Derivative:

يعدّ حساب التدرج المحلي لتابع الصورة هو الأساس في معظم مرشحات كشف الحواف.

- مرشح روبرتس Roberts Filter
- مرشح برويت Prewitt Filters
- مرشح سوبل Sobel Filters
- مرشح روبنسون
- مرشح الاتجاه

مرشحات الكشف عن الحواف

مرشح برويت Prewitt Filter

- يعمل هذا المرشح على القيام بعملية تنعيم قبل حساب التدرج

$$H_x^p = \begin{bmatrix}1\\1\\1\end{bmatrix} * [-1\ 0\ 1] \qquad H_y^p = [1\ 1\ 1] * \begin{bmatrix}-1\\0\\1\end{bmatrix}$$

- يستخدم مرشح برويت القناع:

$$H_x^p = \begin{bmatrix}-1 & 0 & 1\\-1 & 0 & 1\\-1 & 0 & 1\end{bmatrix} \qquad H_y^p = \begin{bmatrix}-1 & -1 & -1\\0 & 0 & 0\\1 & 1 & 1\end{bmatrix}$$

- قيمة التدرج:

$$\nabla I(u,v) \approx \frac{1}{6}\begin{bmatrix}(I * H_x^p)(u,v)\\(I * H_y^p)(u,v)\end{bmatrix}$$

مرشحات الكشف عن الحواف

مرشح سوبل Sobel Filter

Sobel operators – combine smoothing with derivative

- يعمل أيضا هذا المرشح على القيام بعملية تنعيم قبل حساب التدرج
- تكاد أقنعته تطابق مرشح برويت إلا أن سوبل يعطي وزناً أكبر للخط أو العمود المركزي في قسم التنعيم

$$H_x^S = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

dx filter

to compute $\frac{\partial P}{\partial x}$

$$H_y^S = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

dy filter

to compute $\frac{\partial P}{\partial y}$

- قيمة التدرج

$$\nabla I(u,v) \approx \frac{1}{8}\begin{bmatrix} (I * H_x^S)(u,v) \\ (I * H_y^S)(u,v) \end{bmatrix}$$

مرشحات الكشف عن الحواف
مرشح سوبل Sobel Filters

$f(x,y)$

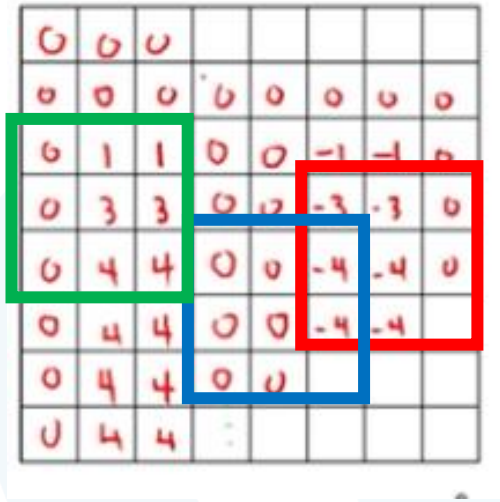
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	1	1	1	1	0	0
0	0	1	1	1	1	0	0
0	0	1	1	1	1	0	0
0	0	1	1	1	1	0	0
0	0	1	1	1	1	0	0

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

$$H_x^s = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

dx filter

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



طويلة وزاوية التدرج

- بفرض نتائج تطبيق المرشحات (سواء سوبل أو برويت) بعد تنسيبها:

$$D_x = H_x * I \qquad D_y = H_y * I$$

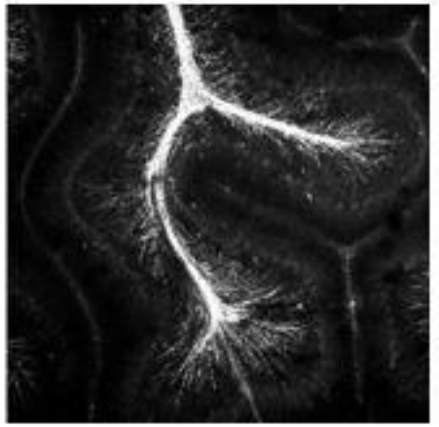
- تعطى طويلة التدرج (قوة الحافة المحلية) بالعلاقة:

$$E(u,v) = \sqrt{(D_x(u,v)^2 + D_y(u,v)^2}$$

- تعطى زاوية التدرج (منحى الحافة المحلية) بالعلاقة:

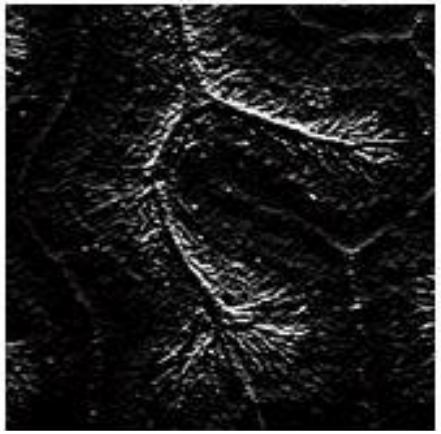
$$\Phi(u,v) = \tan^{-1}\left(\frac{D_y(u,v)}{D_x(u,v)}\right) \quad = Arctan\left(D_y(u,v), D_x(u,v)\right)$$

مرشح سوبل Sobel Filter



الصورة الأصلية

تطبيق مرشح سوبل الأفقي



تطبيق مرشح سوبل الشاقولي

قوة الحواف «طويلة التدرج»

منحى الحواف «زاوية التدرج»

مرشح سوبل Sobel Filter

First Order Partial Derivative

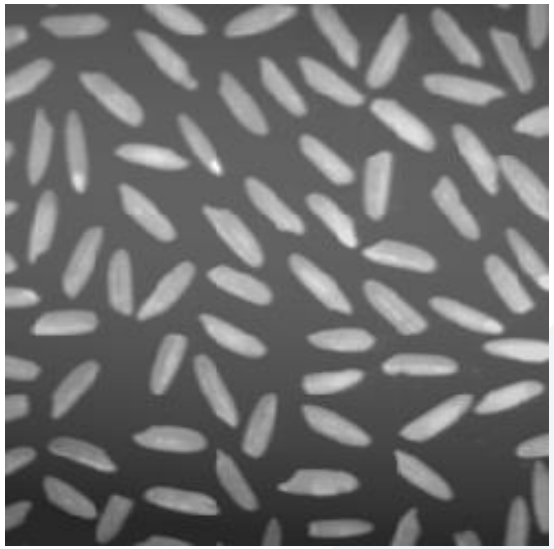
المشتقات الجزئية الأول

-1	0	1
-2	0	2
-1	0	1

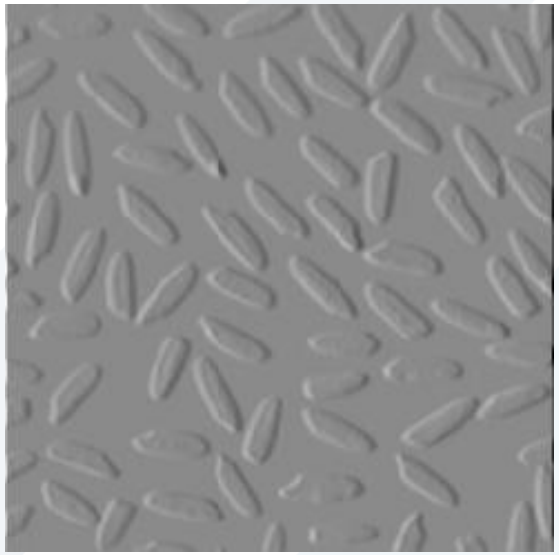
to compute $\frac{\partial P}{\partial x}$

-1	-2	-1
0	0	0
1	2	1

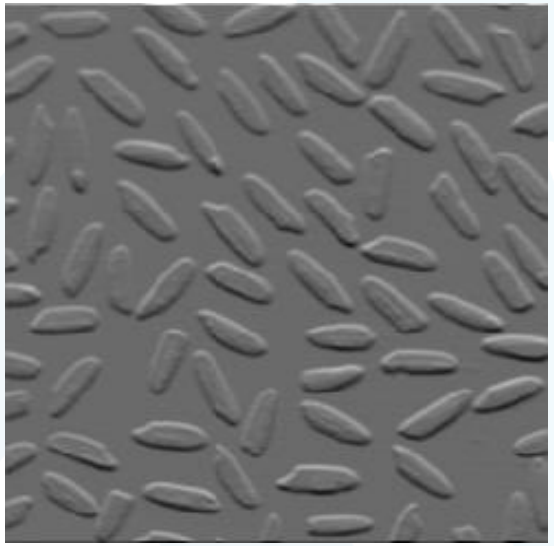
to compute $\frac{\partial P}{\partial y}$



P



$\frac{\partial P}{\partial x}$



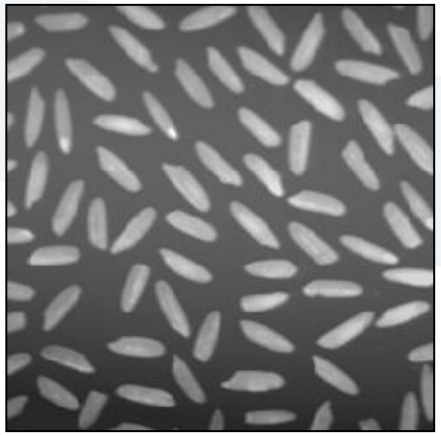
$\frac{\partial P}{\partial y}$

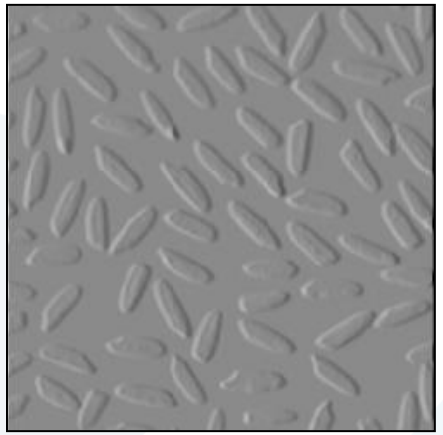
Sobel Filter مرشح سوبل

First Order Partial Derivative

Image Gradient

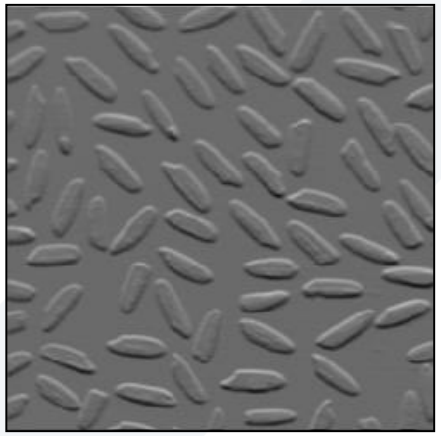
P

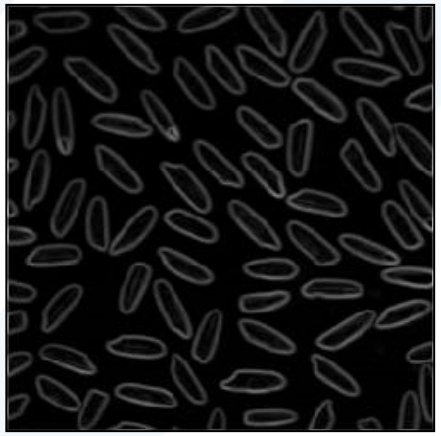




$\frac{\partial P}{\partial y}$

$\frac{\partial P}{\partial x}$



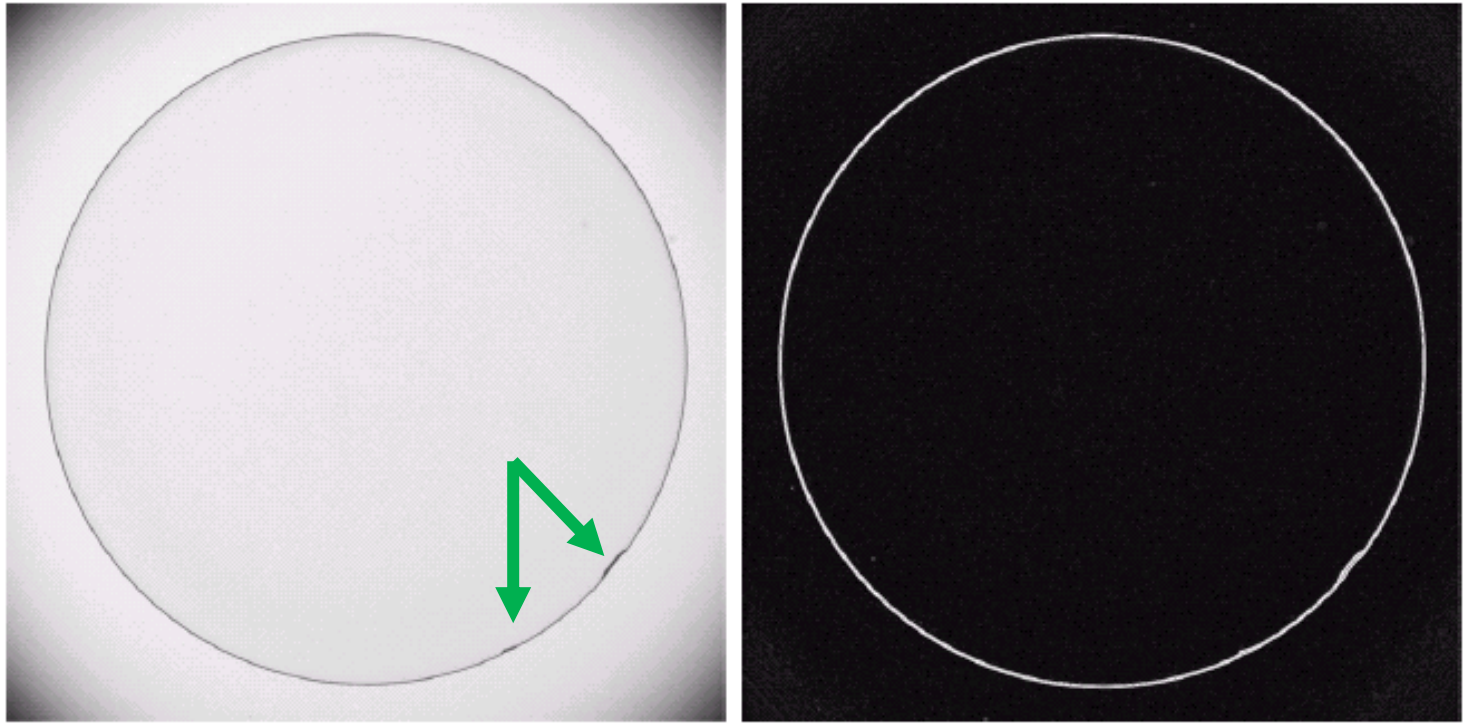


$|\nabla P|$

Sobel Filter مرشح سوبل

Image Gradient

Gradient magnitude $$|\nabla P| = \sqrt{\left(\frac{\partial P}{\partial x}\right)^2 + \left(\frac{\partial P}{\partial y}\right)^2}$$



a b

FIGURE 3.45 Optical image of contact lens (note defects on the boundary at 4 and 5 o'clock). (b) Sobel gradient. (Original image courtesy of Mr. Pete Sites, Perceptics Corporation.)

A gradient image emphasizes edges

(Images from Rafael C. Gonzalez and Richard E. Wood, Digital Image Processing, 2nd Edition.

مناقشة

لايمكن لمرشحات الكشف عن الحواف البسيطة -التي تمّت مناقشتها سابقاً- إظهار كلّ الحواف التي تعدّ مهمة للعين الناظرة وذلك لأسباب عدّة هي:

- تعتمد مرشحات كشف الحواف السابقة فقط على الاختلاف في الشدّة الضوئية " قوة الحافة" في حين يستطيع نظام الرؤية لدى الإنسان كشف الحواف التي يكون اختلاف الشدة الضوئية فيها صغيرا جدا أو حتى معدوم
- تعدّ مرشحات كشف الحواف هذه شديدة الحساسية للتشويش؛ إذ يؤدي وجود تشويش في الصورة إلى استجابة المرشح لهذا التشويش مع عدم وجود حافة أصلا
- إن أغلب الحواف هي مستمرة في الحقيقة وليست متقطعة؛ إذ إنها عادة ما تكون إنسيابية وتتغيّر على نحو تدريجي تصاعدا أو تنازلا ضمن مساحات محلية صغيرة. في حين أن أقنعة *المرشحات السابقة غالبا ما تكون صغيرة الحجم 3x3 وهذا يجعل من الصعوبة بمكان* /كتشاف الحافة ضمن هذ الظروف. ولاكتشاف الحواف في هذه الحالة يتمّ اللجوء إلى مرشحات ذات حجم قناع أكبر. أو استخدام حجم القناع السابق نفسه بعد تصغير حجم الصورة، وتعدّ هذه هي الفكرة الأساس لما يسمى بالتقنيات متعددة الدقة (وتدعى أيضا بالتقنيات الهرمية) والتي جرت العادة على استخدامها في العديد من تطبيقات معالجة الصورة.

```
clear all
close all
clc
a=imread('cameraman.tif');
a=im2double(a);
sx=[-1 0 1; -2 0 2; -1 0 1];
sy=[-1 -2 -1; 0 0 0; 1 2 1];
bx=imfilter(a,sx);
by=imfilter(a,sy);
b=sqrt(double(bx).^2+double(by).^2);
bt=im2bw(b,0.4);
e=edge(a,'sobel');
```

```
figure;
subplot (3,4,5) ; imshow(a); title('Original Image')
subplot (3,4,2) ; imshow(bx); title('Vertical Edges')
subplot (3,4,10) ; imshow(by); title('Horizontal Edges')
subplot (3,4,7) ; imshow(b); title('All Edges')
subplot (3,4,4) ; imshow(bt); title('Thresholded Edges')
subplot (3,4,12) ; imshow(e); title('sobel function')
```

Vertical Edges

Thresholded Edges

Original Image

All Edges

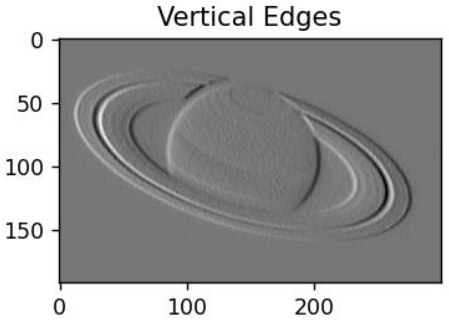
Horizontal Edges

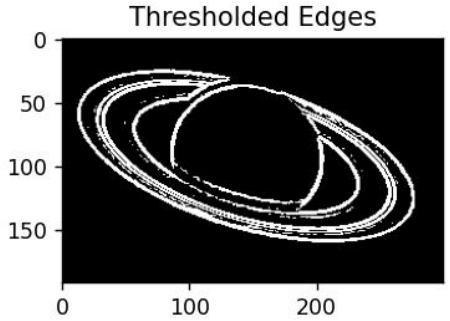
sobel function

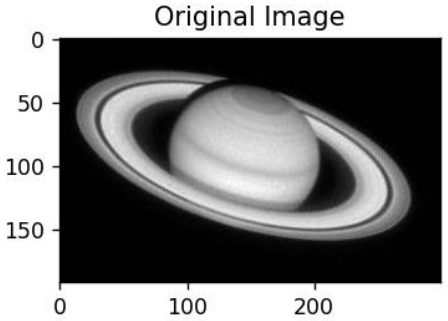
```
import cv2
import numpy as np

# Load the image
img = cv2.imread('Saturn.jpg', cv2.IMREAD_GRAYSCALE)
if img is None:
    print("Error: Could not load image.")
    exit()

img = img.astype(np.float64) / 255.0 # Normalize to double

# Sobel operator kernels
sx = np.array([[-1, 0, 1], [-2, 0, 2], [-1, 0, 1]], dtype=np.float64)
sy = np.array([[-1, -2, -1], [0, 0, 0], [1, 2, 1]], dtype=np.float64)

# Apply Sobel filters
bx = cv2.filter2D(img, -1, sx)
by = cv2.filter2D(img, -1, sy)

# Calculate magnitude of gradient
b = np.sqrt(bx**2 + by**2)

# Thresholding
_, bt = cv2.threshold(b, 0.4, 1, cv2.THRESH_BINARY)
```

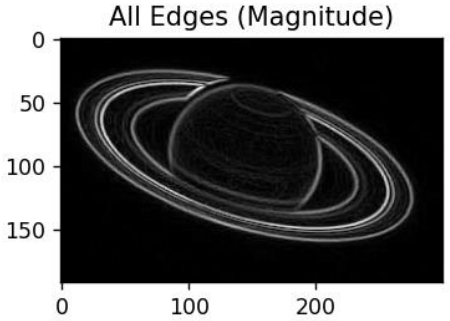
```
# OpenCV's Sobel function
e_x = cv2.Sobel(img, cv2.CV_64F, 1, 0, ksize=3) # x-direction
e_x = np.abs(e_x) #Take the absolute value of the gradient
e_y = cv2.Sobel(img, cv2.CV_64F, 0, 1, ksize=3) #y-direction
e_y = np.abs(e_y)
e = np.sqrt(e_x**2 + e_y**2)
_, e = cv2.threshold(e,0.4,1, cv2.THRESH_BINARY)
```

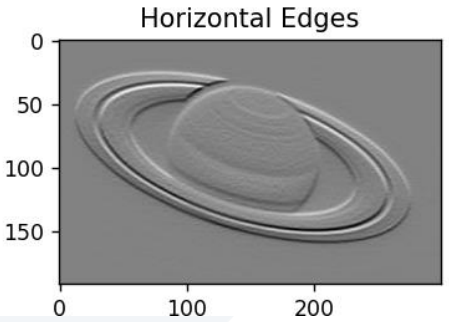
```
# Display the images (using matplotlib for subplot functionality)
import matplotlib.pyplot as plt

plt.figure(figsize=(12, 8))

plt.subplot(3, 4, 5)
plt.imshow(img, cmap='gray')
plt.title('Original Image')

plt.subplot(3, 4, 2)
plt.imshow(bx, cmap='gray')
plt.title('Vertical Edges')

plt.subplot(3, 4, 10)
plt.imshow(by, cmap='gray')
plt.title('Horizontal Edges')

plt.subplot(3, 4, 7)
plt.imshow(b, cmap='gray')
plt.title('All Edges (Magnitude)')

plt.subplot(3, 4, 4)
plt.imshow(bt, cmap='gray')
plt.title('Thresholded Edges')

plt.subplot(3, 4, 12)
plt.imshow(e, cmap='gray')
plt.title('OpenCV Sobel')

plt.tight_layout()
plt.show()
```

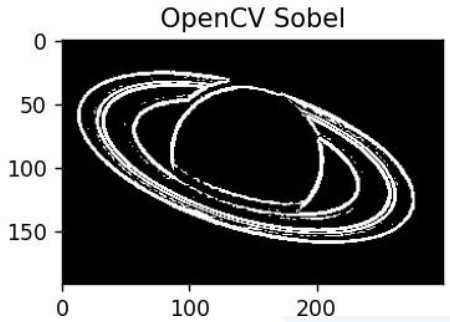

Vertical Edges
0
50
100
150
0
100
200


Thresholded Edges
0
50
100
150
0
100
200


Original Image
0
50
100
150
0
100
200

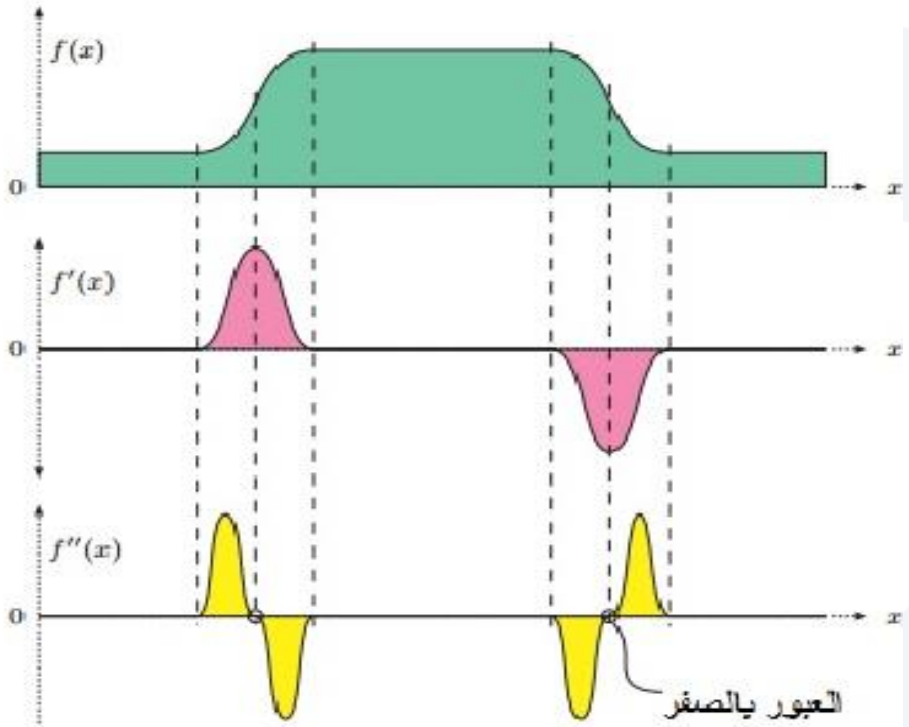

All Edges (Magnitude)
0
50
100
150
0
100
200


Horizontal Edges
0
50
100
150
0
100
200


OpenCV Sobel
0
50
100
150
0
100
200

كشف الحواف بالاعتماد على المشتق الثاني

- يقوم المشتق الثاني لتابع ما في العادة بقياس انحنائه المحلي
- عبور المشتق الثاني بالصفر يدلّ على أقصى تدرج محلي "الحافة"
- يميل هذا النوع من المرشحات غالبا إلى تضخيم التشويش في الصورة



$$\frac{\partial^2 f(x,y)}{\partial x^2} = \frac{\partial}{\partial x}\left(\frac{\partial f(x,y)}{\partial x}\right) = f(x+1,y) - 2f(x,y) + f(x-1,y)$$

As a filter

+1	-2	+1

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

– As a filter

– the Laplacian

0	+1	0
+1	-4	+1
0	+1	0

=

$\frac{\partial^2}{\partial x^2}$

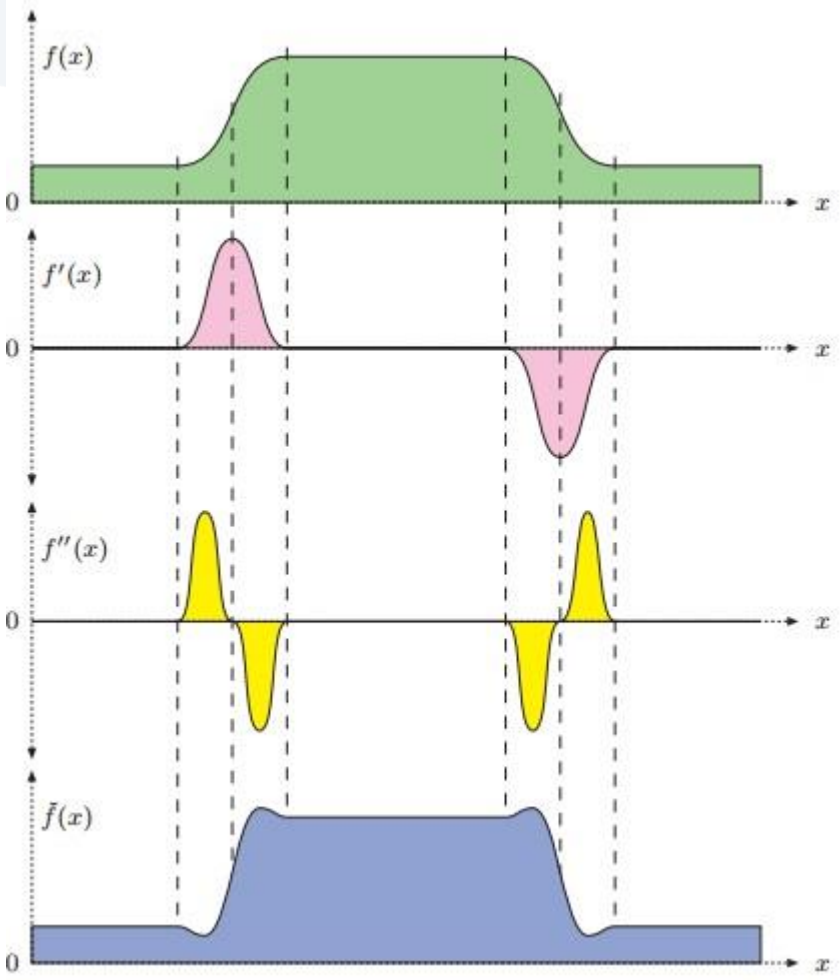
0	0	0
+1	-2	+1
0	0	0

+

$\frac{\partial^2}{\partial y^2}$

0	+1	0
0	-2	0
0	+1	0

زيادة وضوح حواف الصورة اعتماداً على مرشح لابلاس



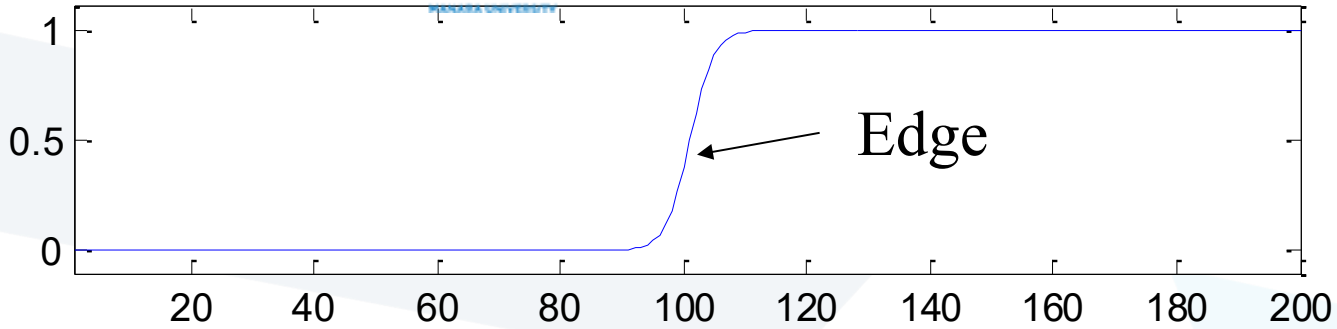
- من أجل تابع أحادي البعد يتم زيادة وضوح الحواف عن طريق طرح المشتق الثاني للتابع $f(x)$ بعد تثقيله بوزن ما w من التابع الأصلي $f''(x)$

$$fn(x) = f(x) - w\ f''(x)$$

Laplacian Sharpening : How it works

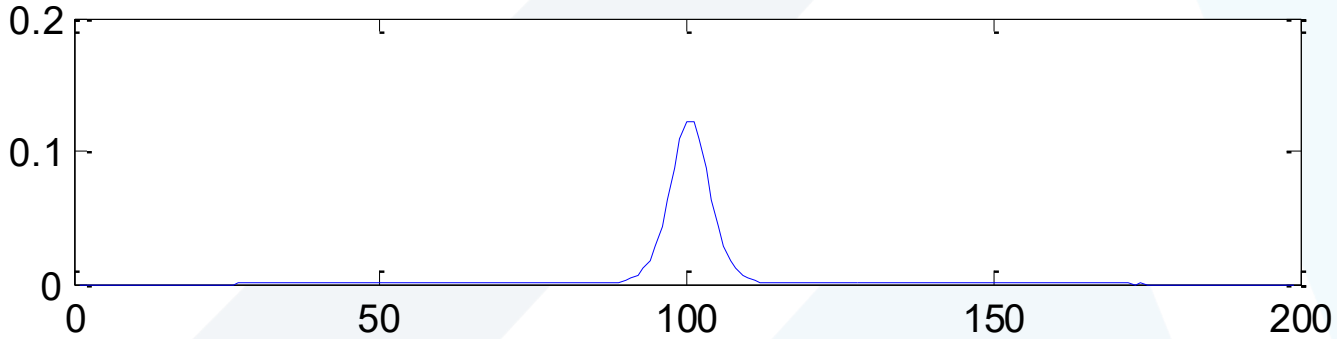
Intensity profile

$p(x)$



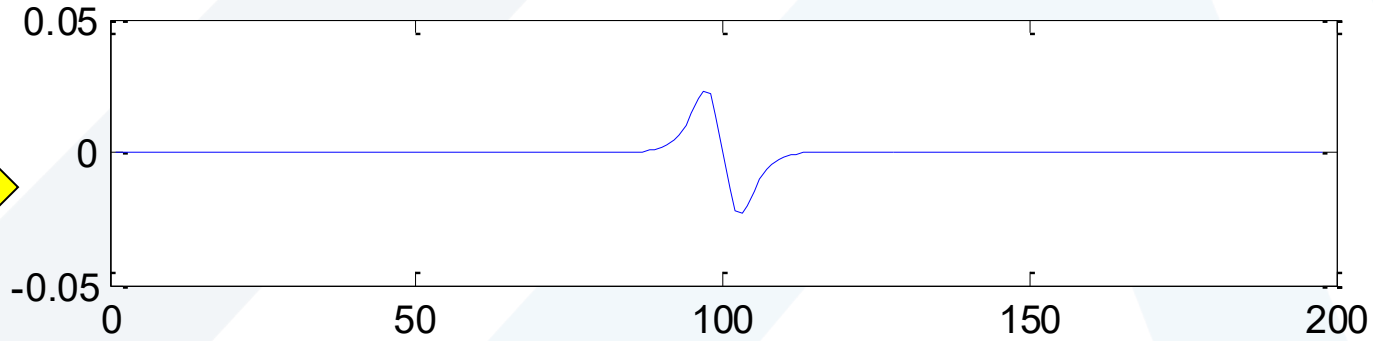
1st derivative

$\frac{dp}{dx}$



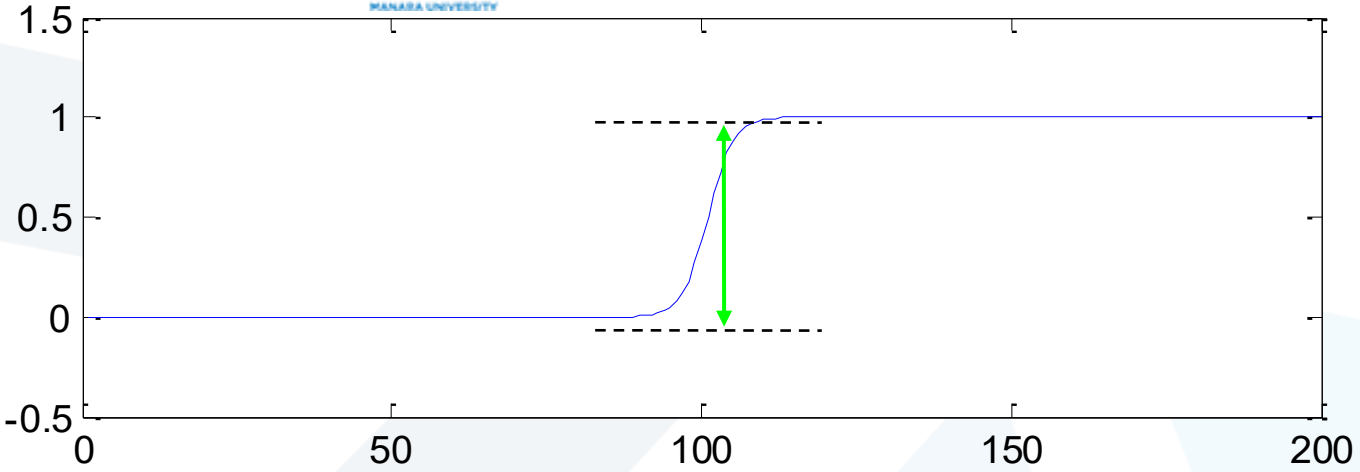
2nd derivative

$\frac{d^2p}{dx^2}$

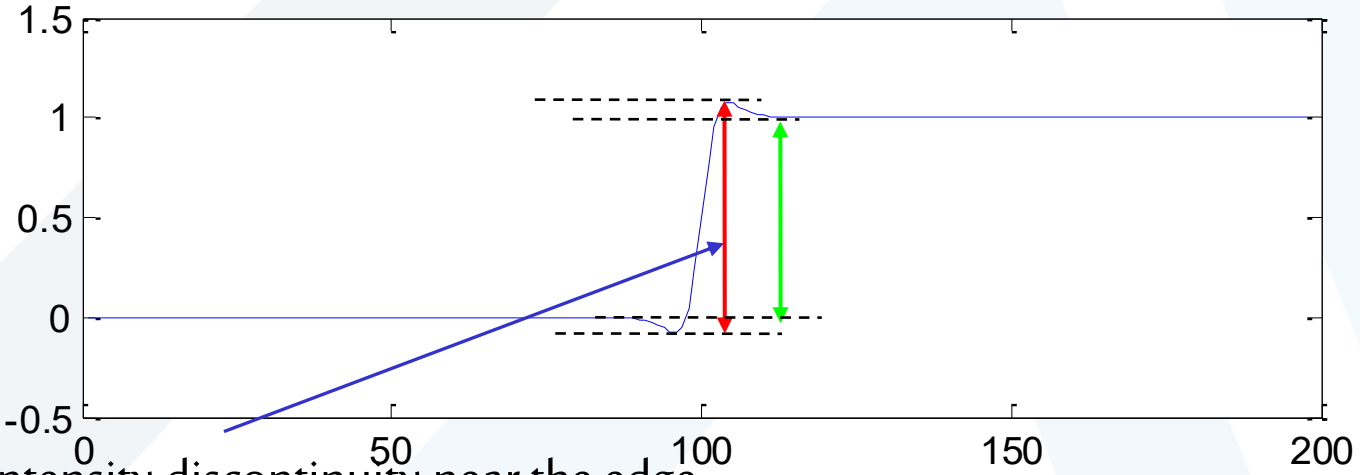


Laplacian Sharpening : How it works

$p(x)$

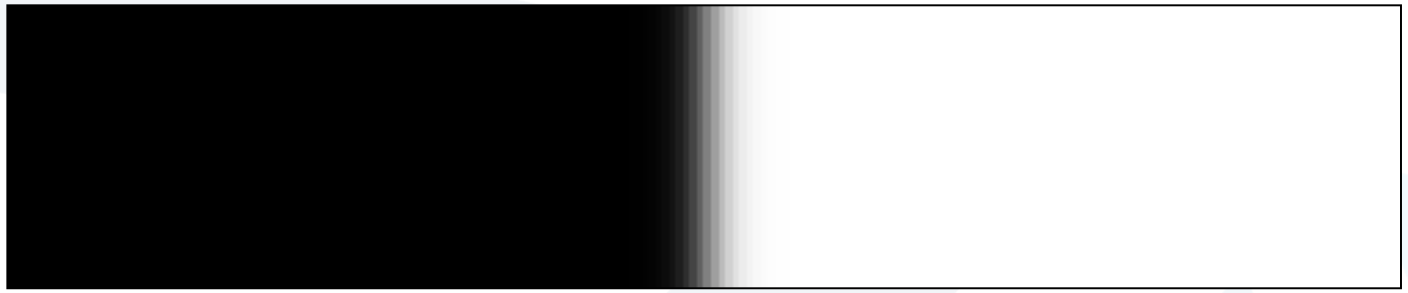


$$p(x) - 10\frac{d^2 p}{dx^2}$$

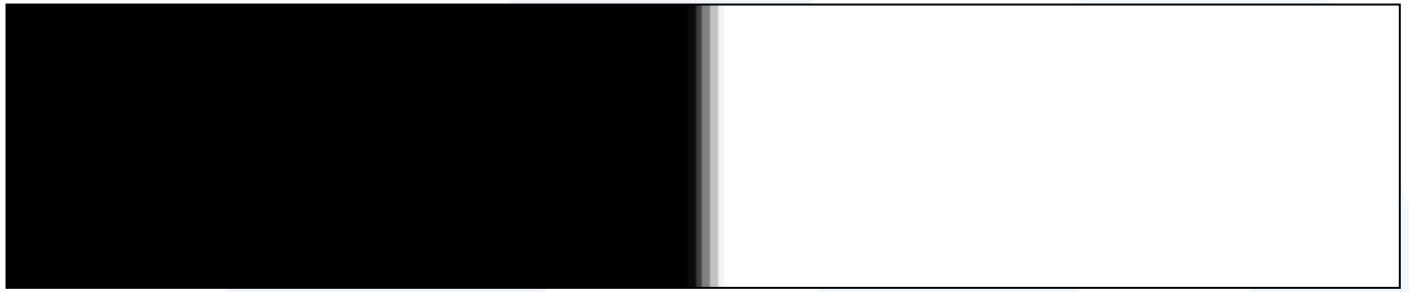


Laplacian sharpening results in larger intensity discontinuity near the edge.

Laplacian Sharpening : How it works



Before sharpening $p(x)$



After sharpening $p(x) - 10\frac{d^2p}{dx^2}$

Laplacian Masks

من أجل التوابع ثنائية البعد يتم زيادة وضوح الحواف عن طريق تطبيق المشتق الثاني في الاتجاهين الأفقي والشاقولي (جمعهما في مرشح لابلاس) ومن ثم طرح الناتج بعد تثقيله من الصورة الأصلية

Used for estimating image Laplacian

$$\nabla^2 P = \frac{\partial^2 P}{\partial x^2} + \frac{\partial^2 P}{\partial y^2}$$

أشكالاً أخرى شائعة لقناع مرشح لابلاس

مرشح لابلاس ∇^2 لتابع ثنائي البعد $f(x,y)$

$$(\nabla^2 f)(x,y) = \frac{\partial^2 f}{\partial^2 x}(x,y) + \frac{\partial^2 f}{\partial^2 y}(x,y)$$

$$H^L = H_x^L + H_y^L = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

مرشح لابلاس لحساب المشتق الثاني للصورة

$H_8^L =$

-1	-1	-1
-1	8	-1
-1	-1	-1

or

0	-1	0
-1	4	-1
0	-1	0

The center of the mask is positive

1	1	1
1	-8	1
1	1	1

or

0	1	0
1	-4	1
0	1	0

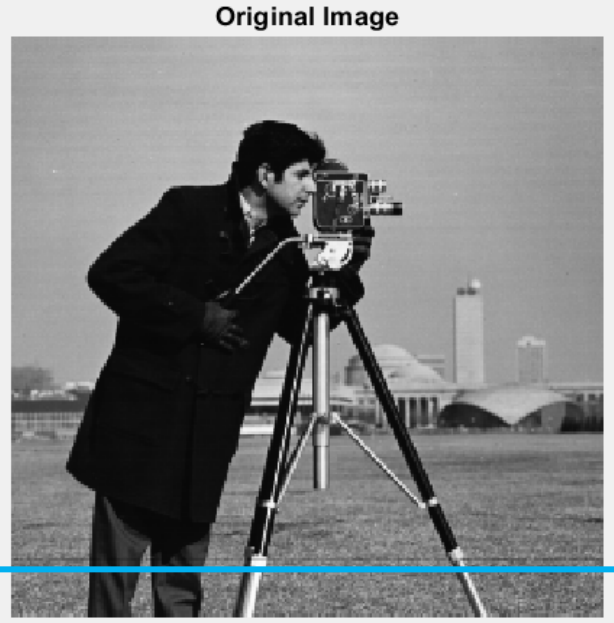
The center of the mask is negative

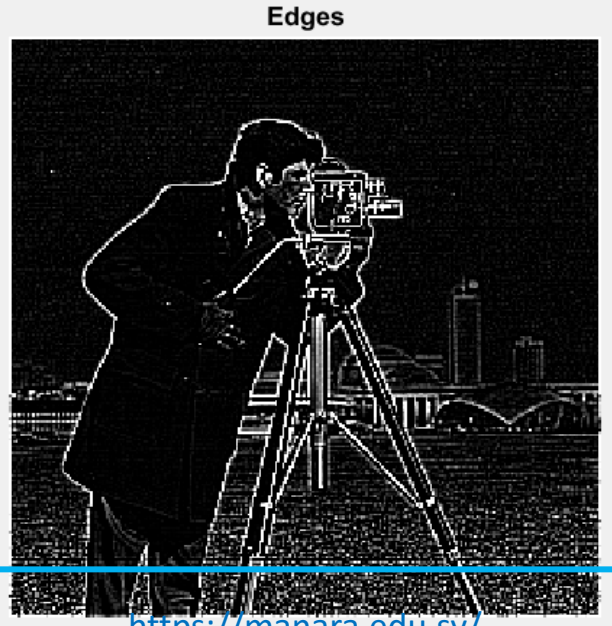
Application: Enhance edge, line, point

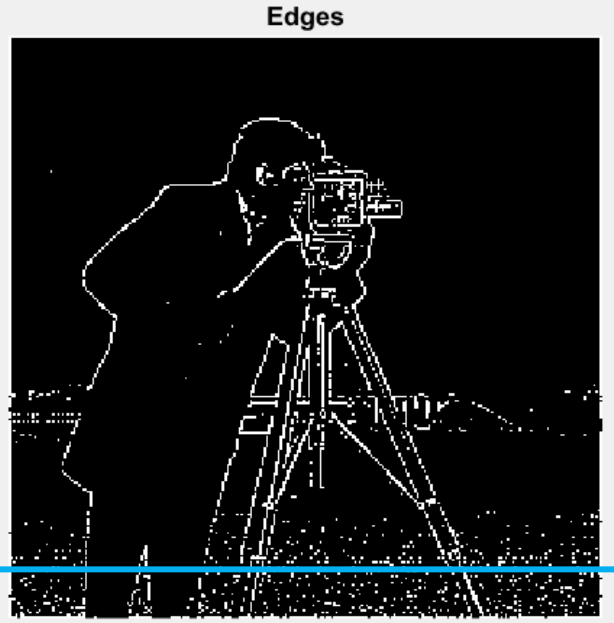
Disadvantage: Enhance noise

$$H_{12}^L = \begin{bmatrix} 1 & 2 & 1 \\ 2 & -12 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

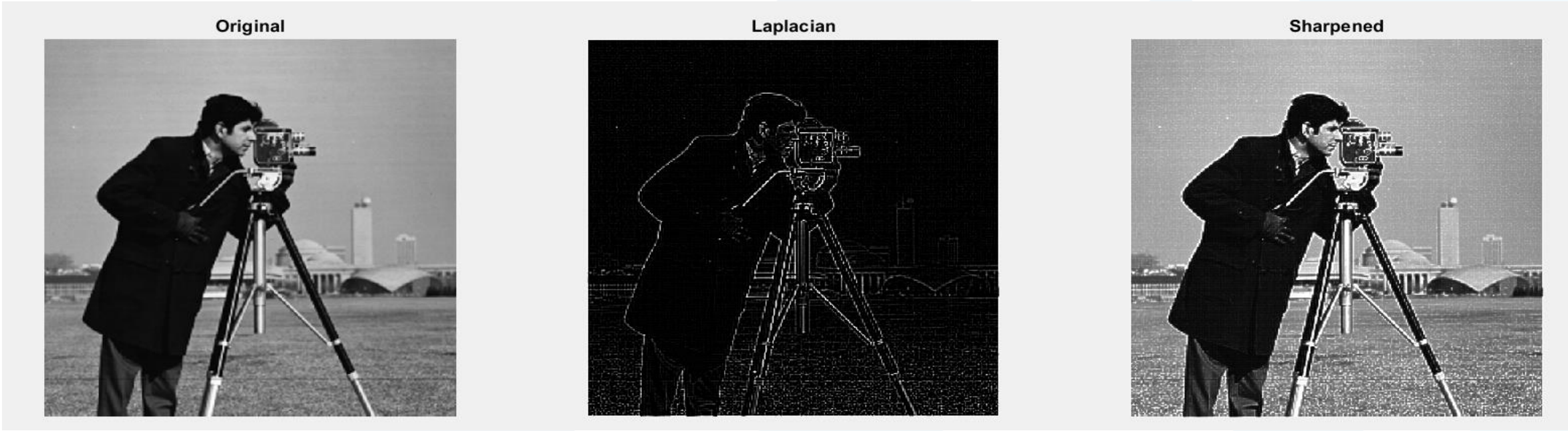
```
close all
clc
a=imread('cameraman.tif');
a=im2double(a);
L=[-1 -1 -1; -1 8 -1; -1 -1 -1];
b=imfilter(a,L);
bt=im2bw(b,0.7);
figure;
subplot (1,3,1) ; imshow(a); title ('Original Image')
subplot (1,3,2) ; imshow(b); title ('Edges')
subplot (1,3,3) ; imshow(bt); title ('Edges')
```



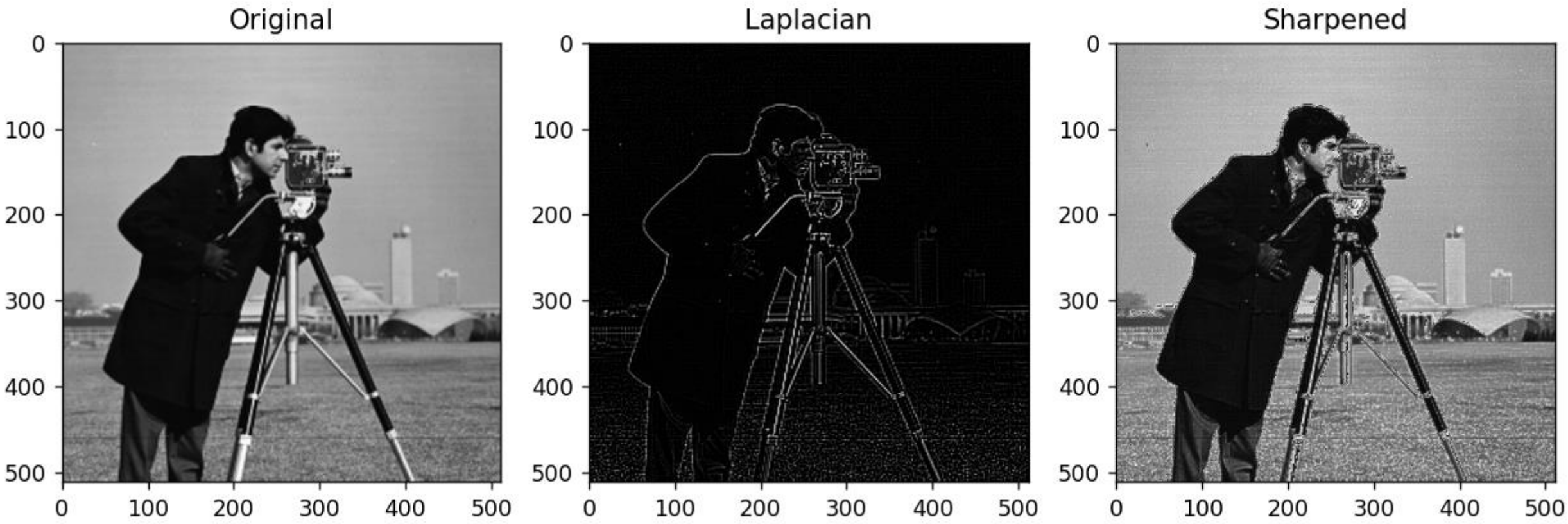




```
% Load the image
img = imread('cameraman.tif');

% Convert to grayscale if necessary
if size(img,3) > 1
img = rgb2gray(img);
end

% Apply Laplacian filter
laplacian = imfilter(img, [-1 -1 -1; -1 8 -1; -1 -1 -1]);

% Sharpened image (unsharp masking)
sharpened = img + laplacian;

%Display Images
figure;
subplot(1,3,1); imshow(img); title('Original');
subplot(1,3,2); imshow(laplacian); title('Laplacian');
subplot(1,3,3); imshow(sharpened); title('Sharpened');
```



```
import cv2
import numpy as np
import matplotlib.pyplot as plt

# Load the image in grayscale
img = cv2.imread('cameraman.tif', cv2.IMREAD_GRAYSCALE)
if img is None:
    print("Error: Could not load image.")
    exit()

# Apply Laplacian filter (using filter2D)
laplacian = cv2.filter2D(img, -1, np.array([[-1, -1, -1], [-1, 8, -1], [-1, -1, -1]]))


# Sharpened image (unsharp masking)
sharpened = np.clip(img + laplacian, 0, 255).astype(np.uint8) #Clip to 0-255 to avoid overflow

#Display the Images using matplotlib
plt.figure(figsize=(12, 4))
plt.subplot(1,3,1); plt.imshow(img, cmap='gray'); plt.title('Original');
plt.subplot(1,3,2); plt.imshow(laplacian, cmap='gray'); plt.title('Laplacian');
plt.subplot(1,3,3); plt.imshow(sharpened, cmap='gray'); plt.title('Sharpened');
plt.show()
```


Original
Laplacian
Sharpened
0
100
200
300
400
500

Laplacian Sharpening

Mask for

$\nabla^2 P$

1	1	1
1	-8	1
1	1	1

or

0	1	0
1	-4	1
0	1	0

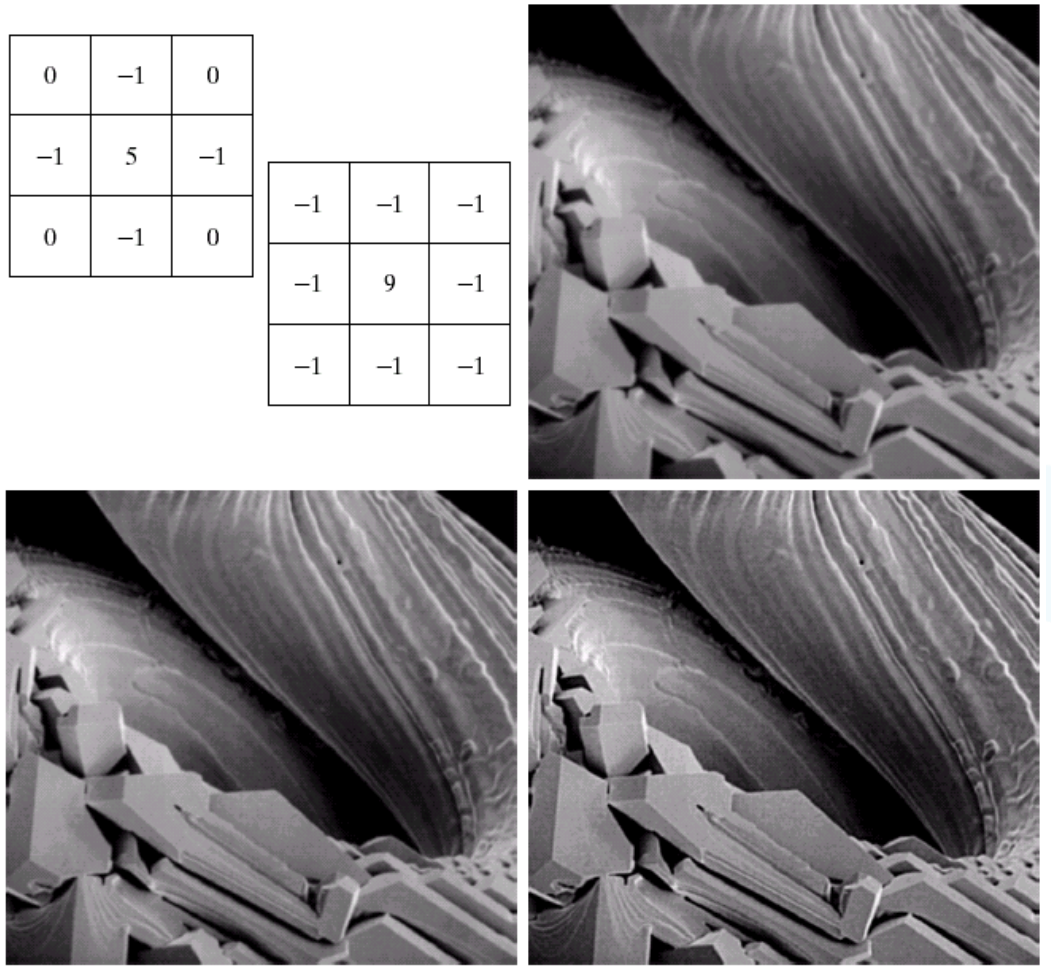
Mask for

$P - \nabla^2 P$

0	-1	0
-1	5	-1
0	-1	0

or

-1	-1	-1
-1	9	-1
-1	-1	-1



a b c
d e

FIGURE 3.41 (a) Composite Laplacian mask. (b) A second composite mask. (c) Scanning electron microscope image. (d) and (e) Results of filtering with the masks in (a) and (b), respectively. Note how much sharper (e) is than (d). (Original image courtesy of Mr. Michael Shaffer, Department of Geological Sciences, University of Oregon, Eugene.)

(Images from Rafael C. Gonzalez and Richard E. Wood, Digital Image Processing, 2nd Edition.

مرشح الحواف كاني

❑ طريقة كاني لازالت تعدّ أفضل خوارزمية لكشف الحواف من جميع النواحي فهي تقدم:

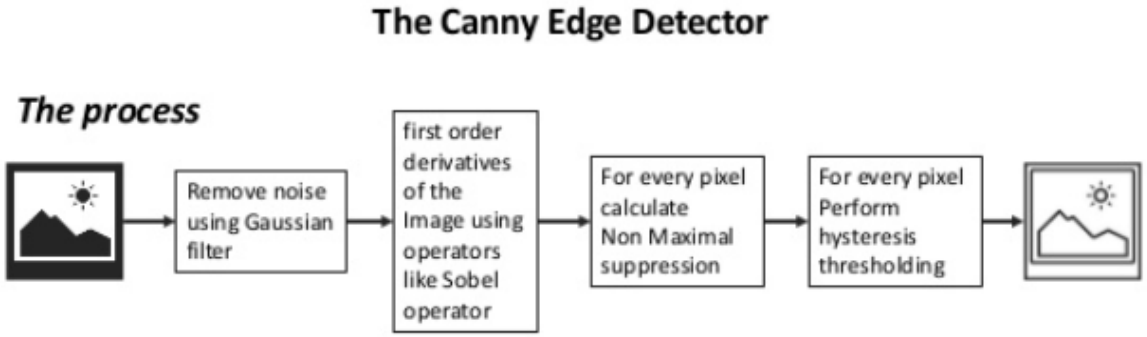
✓ **معدل خطأ منخفض**
✓ **التوضع الصحيح للحواف**
✓ **الاستجابة الوحيدة للحافة**

Canny is Optimal because:
- Less sensitive to noise
- It removes streaking by using two thresholds t_{high} t_{low}
- Offers good localization of edges and utilizes gradient of the edge to generate thin, one-pixel wide edges

❑ يعتمد كاشف الحواف كاني على سلسلة من العمليات الرياضية وعلى **تنعيم الصورة باستخدام مرشح غاوص σ**

❑ يستخدم كاشف الحواف كاني عتبتين مختلفتين قيمة إحداهما *T2* أكبر من الأخرى T1

خطوات تطبيق مرشح الحواف كاني



The Canny edge detection algorithm is composed of 5 steps:
1.Noise reduction;
2.Gradient calculation;
3.Non-maximum suppression;
4.Double threshold;
5.Edge Tracking by Hysteresis.

- تنعيم الصورة باستخدام مرشح غاوص
- حساب قوة الحافة ومنحاها
- الإخماد اللاأعظمي
- تعتيب مزدوج
- التخلفية لتتبع الحواف

خطوات تطبيق مرشح الحواف كاني

Noise Reduction → **Gradient Calculation** → **Non-Maximum Suppression** → **Double threshold** → **Edge Tracking by Hysteresis**

get rid of the noise. image convolution technique is applied with a Gaussian Kernel (3x3, 5x5, 7x7 etc…) to smooth it.
Basically, the smallest the kernel, the less visible is the blur.

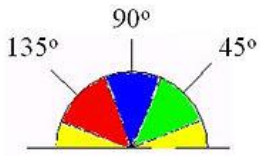
First order derivative of an image. It can be implemented by convolving I with Sobel kernels Kx and Ky, respectively:

$$K_x = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}, K_y = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}.$$

Sobel filters for both direction (horizontal and vertical)

$$|G| = \sqrt{I_x^2 + I_y^2},$$

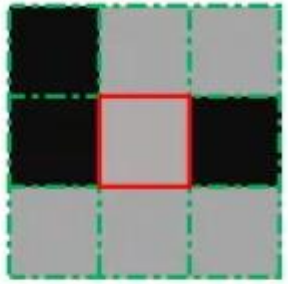
$$\theta(x,y) = arctan\left(\frac{I_y}{I_x}\right)$$



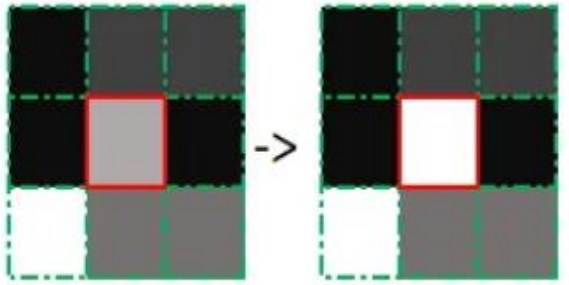
Gradient intensity and Edge direction

The edge direction angle is rounded to one of four angles representing vertical, horizontal, and the two diagonals (0°, 45°, 90°, and 135°).

The double threshold step aims **at identifying 3 kinds of pixels: strong, weak, and non-relevant**:
Strong pixels are pixels that have an intensity so high that we are sure they contribute to the final edge.
Weak pixels are pixels that have an intensity value that is not enough to be considered as strong ones, but yet not small enough to be considered as non-relevant for the edge detection.
Other pixels are considered as non-relevant for the edge.



No strong pixels around



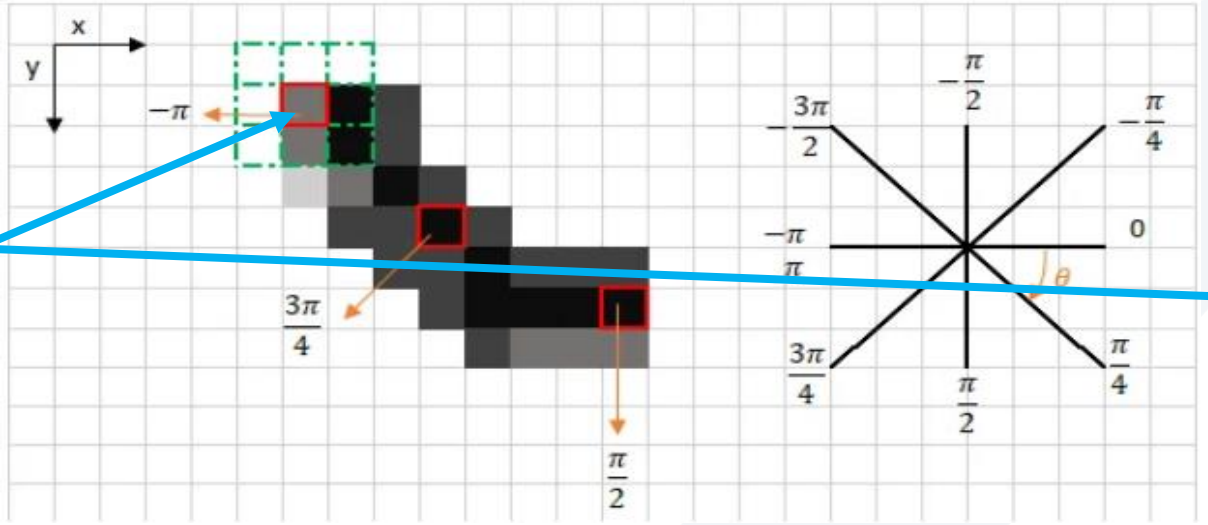
One strong pixel around

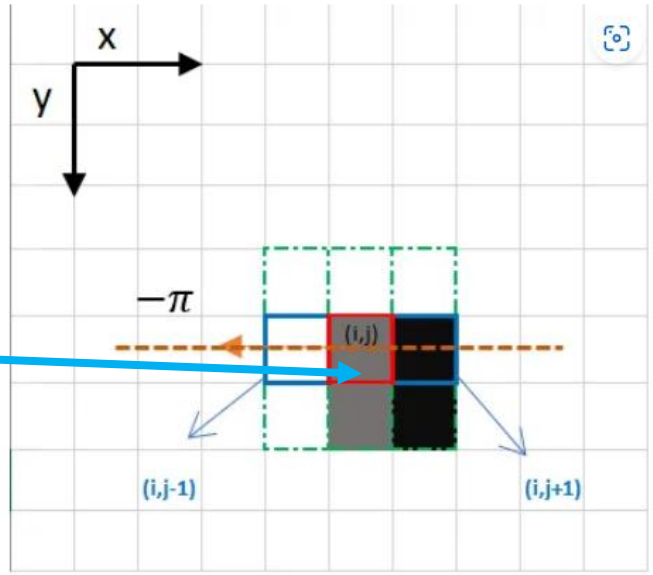
Based on the threshold results, the hysteresis consists of transforming weak pixels into strong ones, if and only if at least one of the pixels around the one being processed is a strong one.

Non-Maximum Suppression

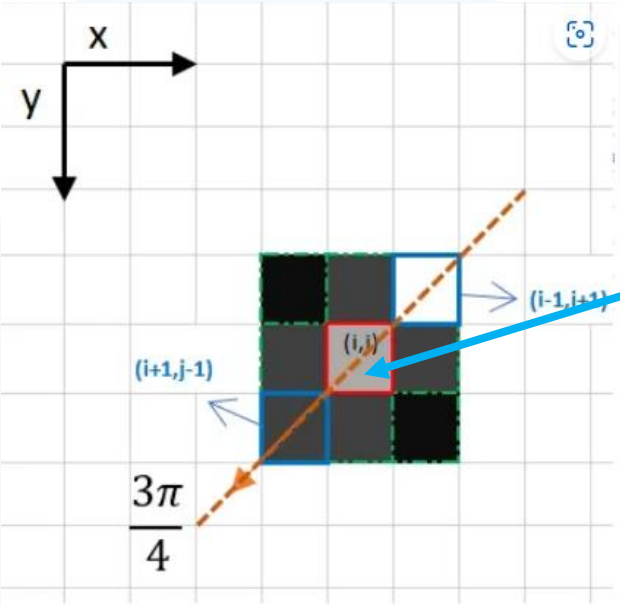
perform non-maximum suppression to thin out the edges.

The principle is simple: the algorithm goes through all the points on the gradient intensity matrix and finds the pixels with the maximum value in the edge directions. المبدأ بسيط: تمر الخوارزمية عبر جميع النقاط الموجودة في مصفوفة كثافة التدرج وتجد وحدات البكسل ذات القيمة القصوى في اتجاهات الحافة.





Focus on the upper left corner red box pixel



If there are no pixels in the edge direction having more intense values, then the value of the current pixel is kept.

```
% Load and convert to grayscale (as in Method 1)
img = imread('cameraman.tif');
if size(img, 3) == 3
img = rgb2gray(img);
end
```

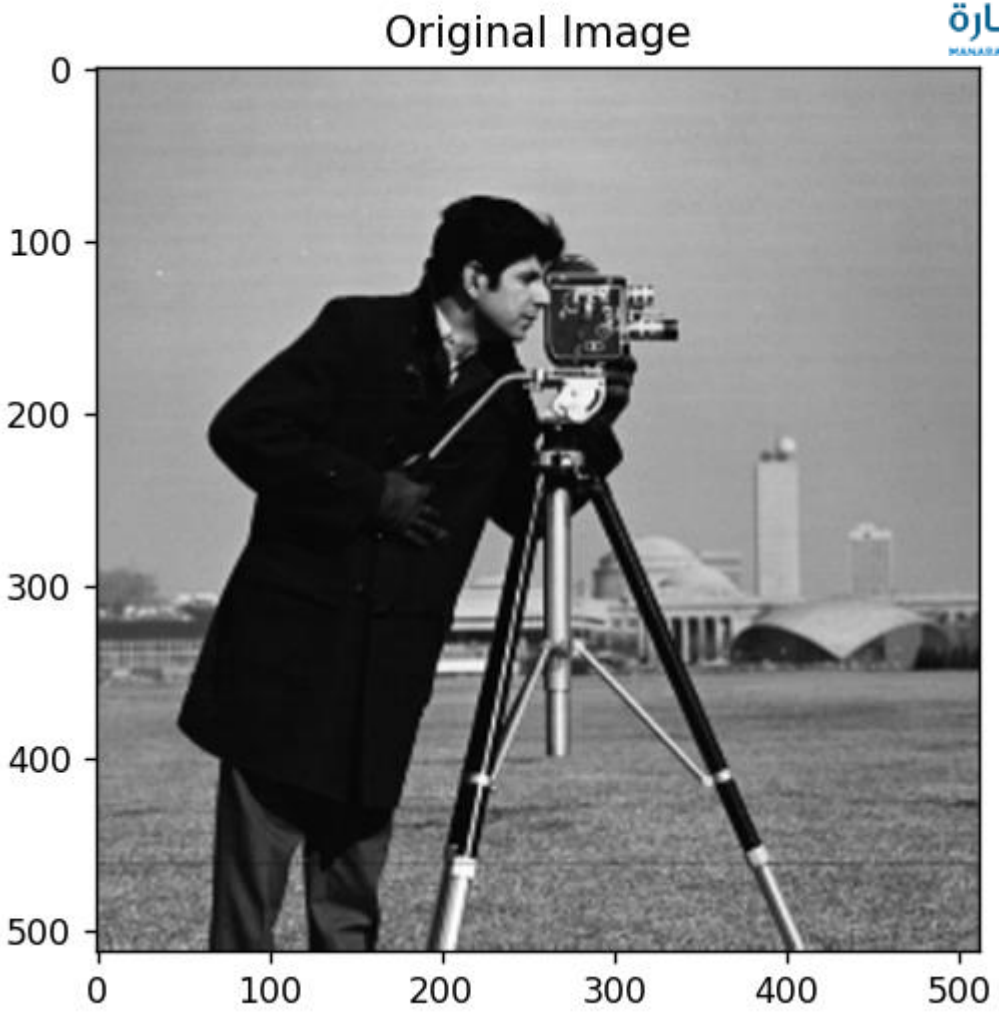
```
% Set sigma value for Gaussian smoothing
sigma = 2; % Adjust this value
```

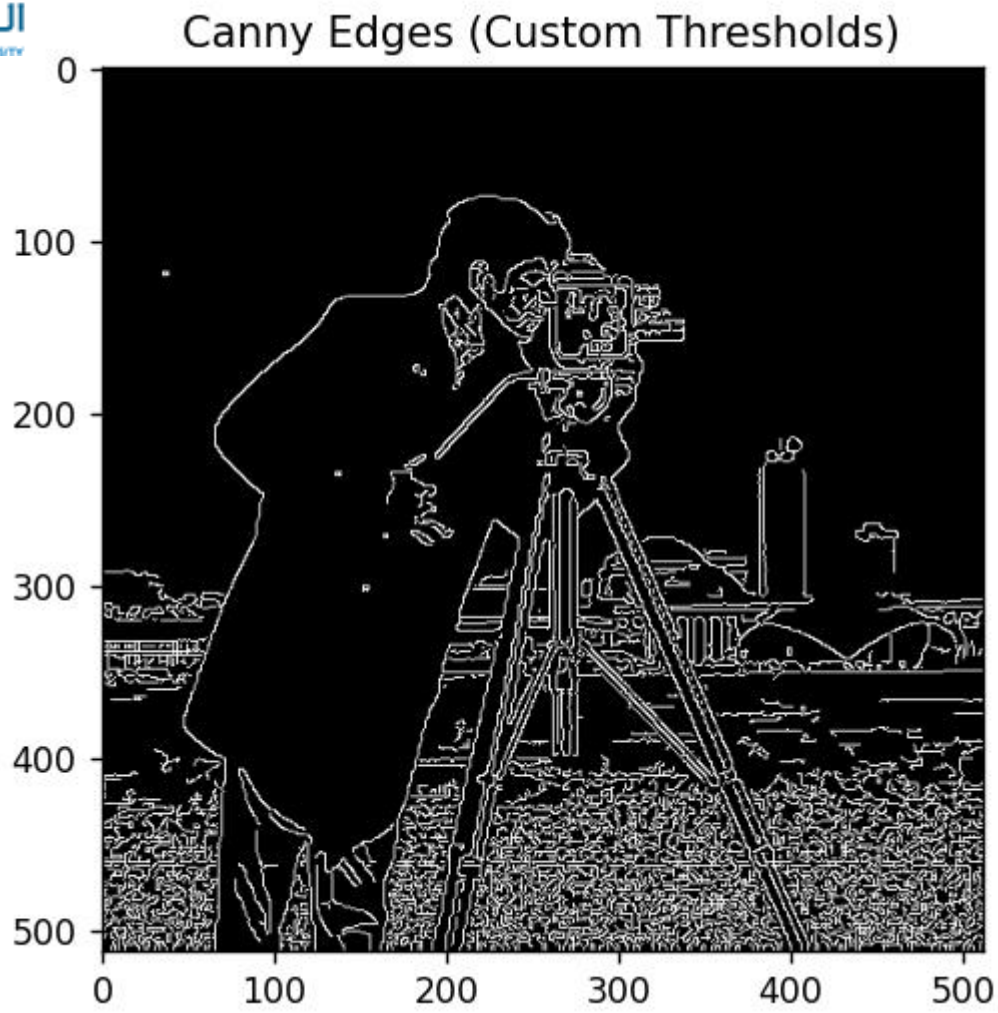
```
% Apply Canny edge detection with sigma specification
edges = edge(img, 'canny', [], sigma);
```

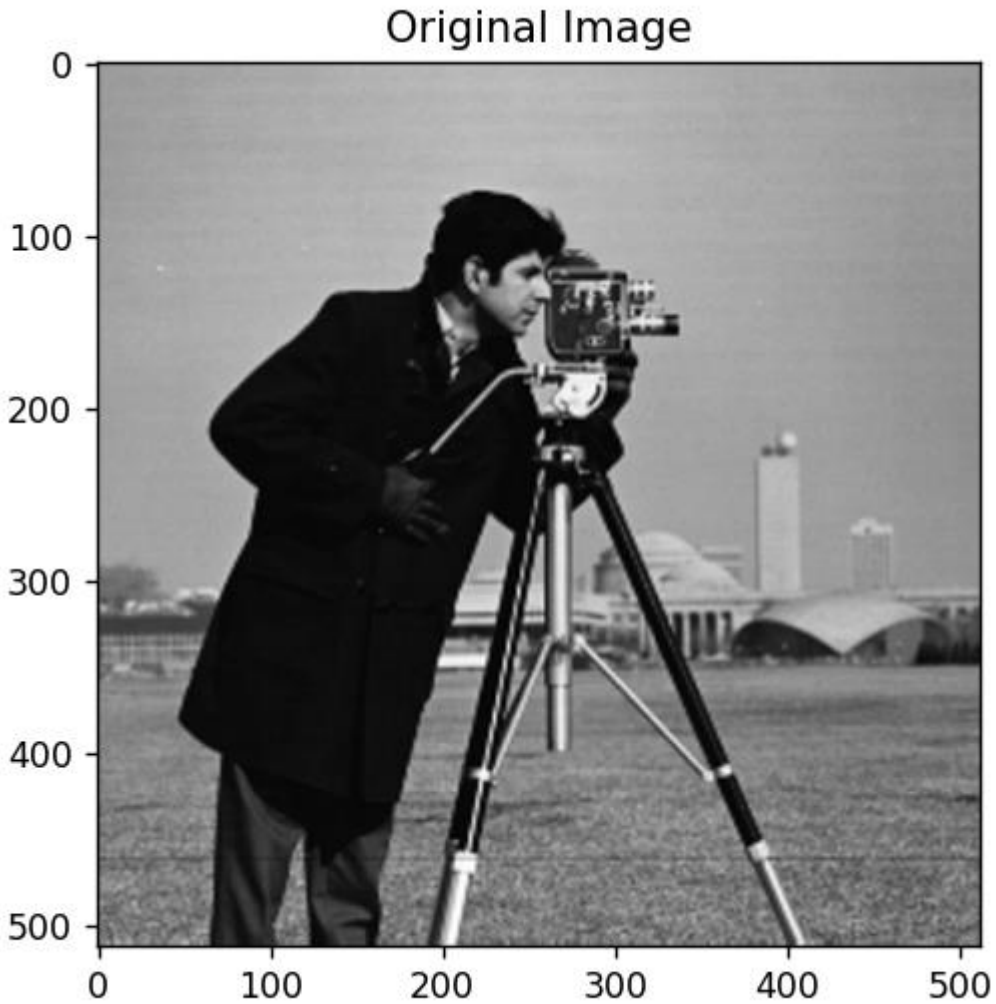
```
% Display the results (as in Method 1)
figure;
subplot(1, 2, 1);
imshow(img);
title('Original Image');
subplot(1, 2, 2);
imshow(edges);
title('Canny Edges (Custom Sigma)');
```

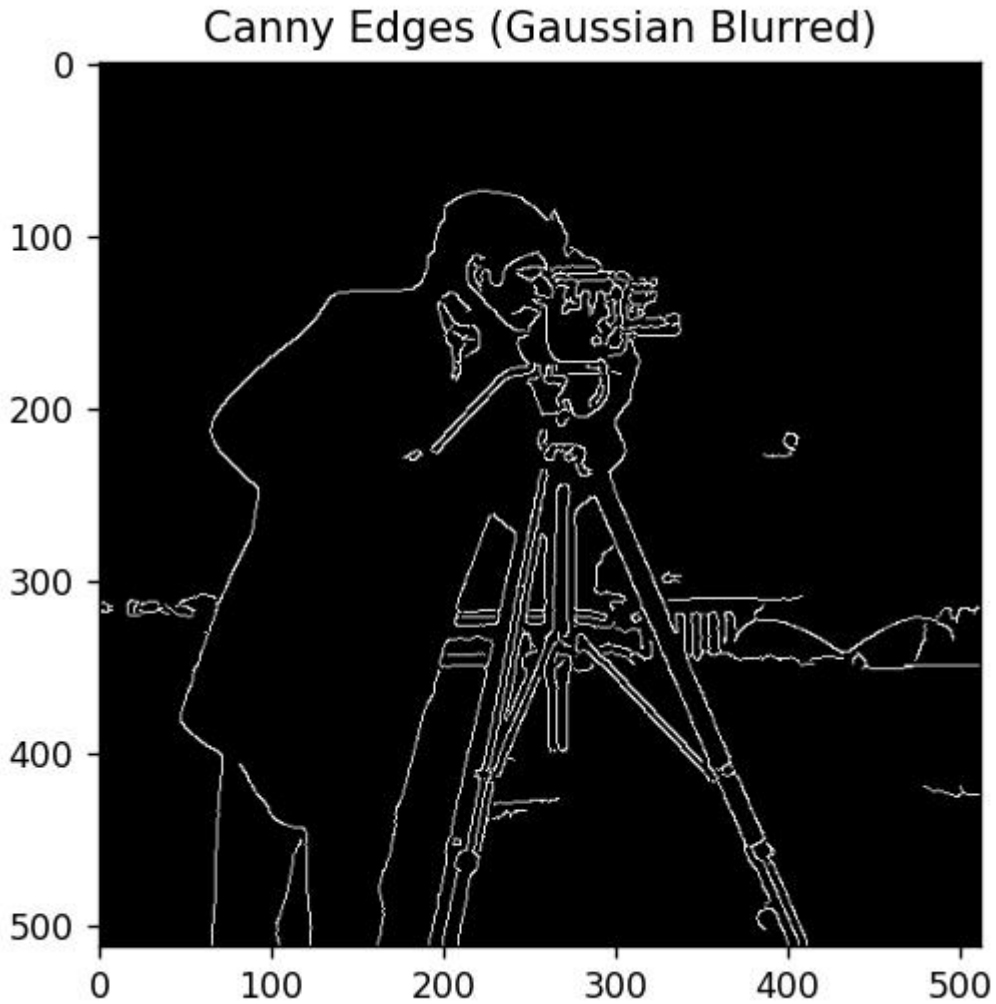
```
import cv2
import numpy as np
import matplotlib.pyplot as plt

# Helper function to display images side-by-side
def display_images(img1, img2, title1, title2):
    plt.figure(figsize=(10, 5))
    plt.subplot(1, 2, 1)
    plt.imshow(img1, cmap='gray')
    plt.title(title1)
    plt.subplot(1, 2, 2)
    plt.imshow(img2, cmap='gray')
    plt.title(title2)
    plt.show()

# Load the image (replace 'cameraman.tif' with your image)
img = cv2.imread('cameraman.tif', cv2.IMREAD_GRAYSCALE)
if img is None:
    print("Error: Could not load image.")
    exit()
```

```
# Canny with custom thresholds
high_thresh = 100 # Adjust these values
low_thresh = 50   # Adjust these values
edges2 = cv2.Canny(img, low_thresh, high_thresh)
display_images(img, edges2, "Original Image", "Canny Edges (Custom Thresholds)")


# Method 2: Canny with custom sigma (Gaussian smoothing before Canny)

#Note: OpenCV's Canny doesn't directly take a sigma value for pre-smoothing.
#You'll need to smooth the image separately using a Gaussian blur.

blurred = cv2.GaussianBlur(img, (5, 5), 2) # sigma = 2 (adjust as needed)
edges3 = cv2.Canny(blurred, 50, 150) #You can use custom thresholds here too.
display_images(img, edges3, "Original Image", "Canny Edges (Gaussian Blurred)")
```


Original Image
0
100
200
300
400
500
0
100
200
300
400
500


Canny Edges (Custom Thresholds)
0
100
200
300
400
500
0
100
200
300
400
500


Original Image
0
100
200
300
400
500
0
100
200
300
400
500


Canny Edges (Gaussian Blurred)
0
100
200
300
400
500
0
100
200
300
400
500

جَامعة
المَنارة
MANARA UNIVERSITY

نهاية المحاضرة