الجلسة الثالثة

Spatial Filtering

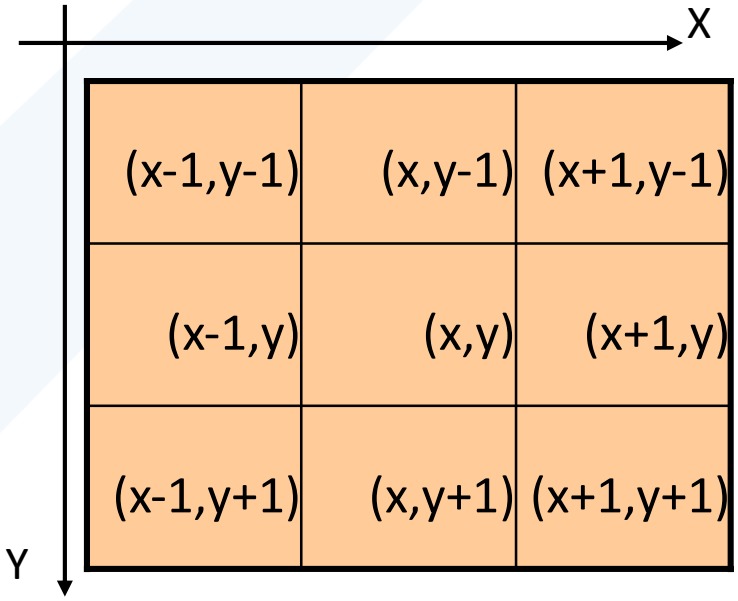
الترشيح في المجال الفراغي

Noise, Convolution, Smoothing, Edge detection, Edge enhancement

1.1 مفهوم الترشيح Filtering

تتألف عملية الترشيح من جوار وعملية محددة

- يهدف الترشيح إلى عدة أمور منها:
- ترشيح الضجيج
- توضيح (تعزيز) الحواف
- كشف مناطق محددة من الصورة



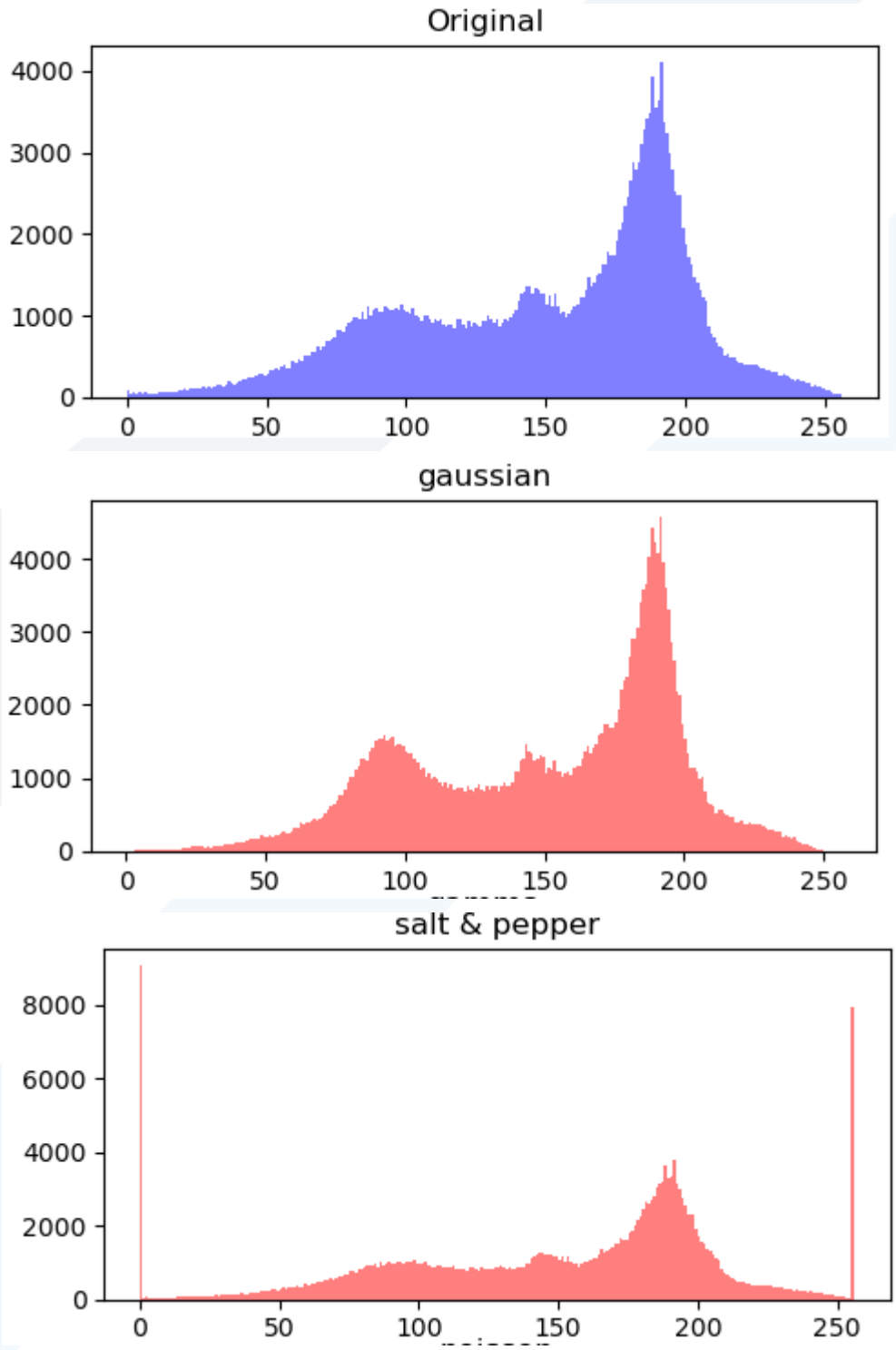
1.2 أنواع الضجيج Noise Types:

هناك أنواع كثيرة للضجيج لكن يمكن تصنيفها ضمن نوعين أساسيين:

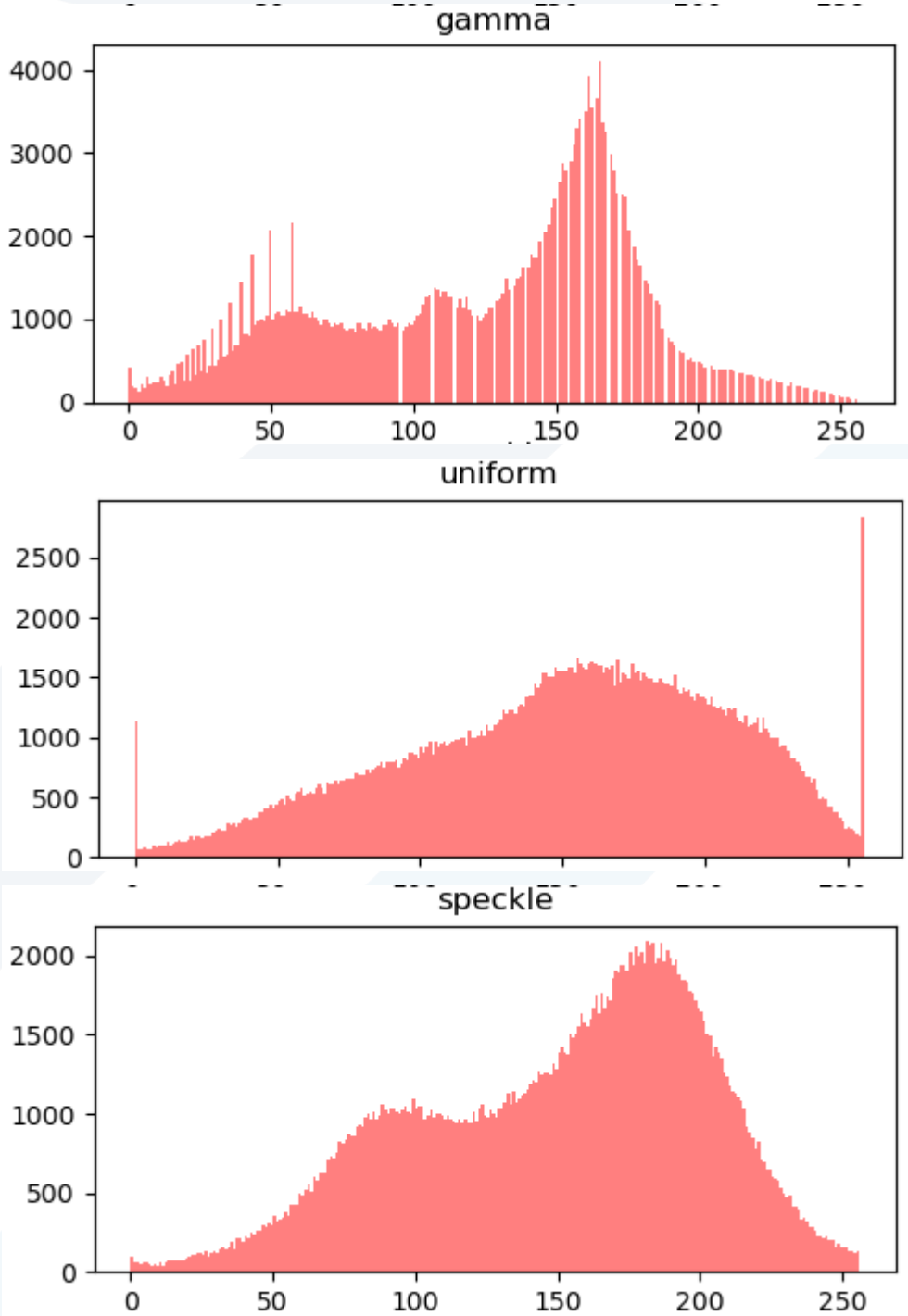
ضجيج خطي linear noise مثل Gaussian Noise (additive noise)

ضجيج لا خطي Non-linear noise مثل Salt & Pepper noise

1.3 كيفية معرفة الضجيج من الهيستوغرام:



يحدث ضجيج Salt & Pepper بسبب أخطاء البت أثناء نقل الصور (تبديل البت الأكثر أهمية من 0 إلى 255 أو العكس)



يشاهد ضجيج Speckle noise في صور Ultrasound مثل الصور الناتجة من أجهزة الإيكو وهو ضجيج ضربي يشوه الصورة.

1.4 إضافة الضجيج برمجياً:

نفذ الكود التالي الذي يقوم بقراءة صورة بشكلها الرمادي ثم إضافة عدة أنواع مختلفة من الضجيج وعرض الصورة الأصلية مع هيستوغرامها ثم الصور الضجيجية مع هيستوغرام كل منها من أجل مقارنة شكل الهيستوغرام بعد إضافة الضجيج.

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
img = cv2.imread('Panda.jpg', cv2.IMREAD_GRAYSCALE)
noise_types = ['gaussian', 'salt & pepper', 'gamma', 'speckle']
noisy_imgs = []
for noise_type in noise_types:
  if noise_type == 'gaussian':
    # Add true Gaussian noise
    # Add true Gaussian noise
    noise = np.random.normal(0,0.5,img.shape).astype('uint8')
    noisy_img=cv2.add(img,noise)
    noisy_img = np.clip(noisy_img, 0, 255)  # Ensure pixel values are within [0, 255]
  elif noise_type == 'salt & pepper':
    noisy_img = img.copy()
    noise = np.zeros_like(img)
    cv2.randu(noise, 0, 255)
    noisy_img[noise < 10] = 0
    noisy_img[noise > 245] = 255
  elif noise_type == 'gamma':
    noisy_img = np.power(img /np.max(img), 1.5) * 255
    noisy_img = noisy_img.astype(np.uint8)
  elif noise_type == 'uniform':
    noisy_img = img.copy()
```

Additive noise (Gaussian)
الضجيج الجمعي الغوصي

Salt & Pepper replace noise
ضجيج الاستبدال

Gamma Noise
ضجيج غاما الأسي

```
        noise = np.random.uniform(-50, 50, size=img.shape)
        noisy_img = noisy_img + noise
        noisy_img[noisy_img < 0] = 0
        noisy_img[noisy_img > 255] = 255
    elif noise_type == 'speckle':
        noisy_img = img + img * np.random.randn(*img.shape) * 0.1
    noisy_imgs.append(noisy_img)
# Create a single subplot for all images
fig, axs = plt.subplots(2, len(noise_types) + 1, figsize=(15, 6))
axs[0, 0].imshow(img, cmap='gray')
axs[0, 0].set_title('Original Image')
axs[0, 0].axis('off')
for i, noise_type in enumerate(noise_types):
    axs[0, i + 1].imshow(noisy_imgs[i], cmap='gray')
    axs[0, i + 1].set_title(noise_type)
    axs[0, i + 1].axis('off')
axs[1, 0].hist(img.ravel(), 256, [0, 256], alpha=0.5, color='red')
axs[1, 0].set_title(' Original Histogram')
# Plot histograms
for i, noise_type in enumerate(noise_types):
    axs[1, i + 1].hist(noisy_imgs[i].ravel(), 256, [0, 256], alpha=0.5, color='red')
    axs[1, i + 1].set_title(f'{noise_type} Histogram')
plt.tight_layout()
plt.show()
```

Uniform Random Noise

ضجيج منتظم من خلال إضافة قيم عشوائية في المجال -50 وحتى +50 ثم ضبط المجالات

Speckle Multiplicative noise

الضجيج الضربي (تم استخدام قيم عشوائية بانحراف معياري 1 وقيمة متوسطة 0 وجداء هذه القيم في بكسلات الصورة)

عرض الصورة الأصلية وهيستوغرامها

وعرض الصور بعد إضافة الضجيج مع هيستوغرام كل منها

1.5 مرشحات التنعيم Smoothing Filters:

نفذ الكود التالي الذي يقوم بترشيح نوعين من الضجيج:

- الضجيج الغوصي (الجمعي) باستخدام مرشح التنعيم المتوسط Mean Filter 5*5
- ضجيج الاستبدال اللاخطي باستخدام المرشح اللاخطي Median Filter

```
import cv2
from matplotlib import pyplot as plt
import numpy as np
image=cv2.imread("Panda.jpg")
gray_image=cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
gauss=np.random.normal(0,0.5,gray_image.shape).astype('uint8')
gauss_image=cv2.add(gray_image,gauss)
mean_filter_image=cv2.blur(gray_image,(5,5))
# Create a single subplot for all images
fig, axs = plt.subplots(1, 3)
axs[0].imshow(gray_image, cmap='gray')
axs[0].set_title('Original Image')
axs[1].imshow(gauss_image, cmap='gray')
axs[1].set_title('noisy Image')
axs[2].imshow(mean_filter_image, cmap='gray')
axs[2].set_title('Smoothed Image')
plt.show()
```

إضافة ضجيج غوصي للصورة

ترشيح الصورة باستخدام مرشح متوسط بحجم 5*5

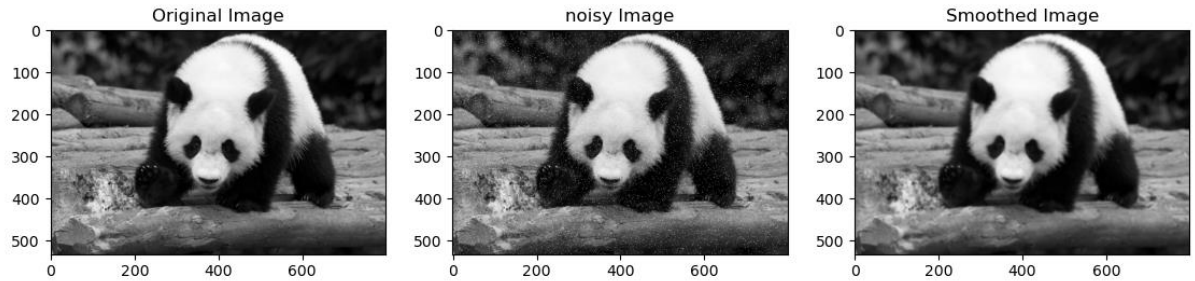
عرض الصورة الأصلية والضجيجية والمنعمة Smoothed

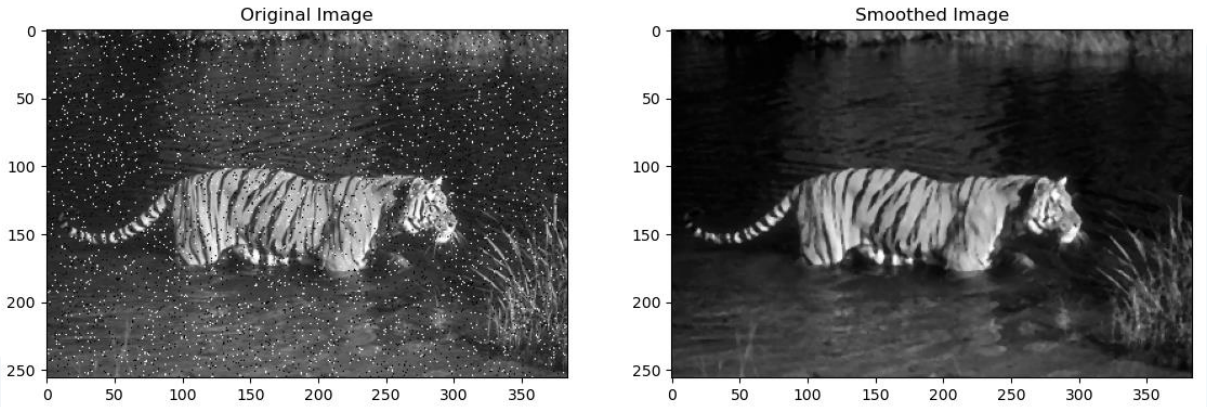
```
gray_image=cv2.imread('salt and pepper noise.png',0)
Median_filter_image=cv2.medianBlur(gray_image,3)
# Create a single subplot for all images
fig, axs = plt.subplots(1, 2)
axs[0].imshow(gray_image, cmap='gray')
axs[0].set_title('Original Image')
axs[1].imshow(Median_filter_image, cmap='gray')
axs[1].set_title('Smoothed Image')
plt.show()
```

قراءة صورة تتضمن ضجيج استبدال S&P

تنعيم الضجيج باستخدام مرشح Median 3*3

والنتيجة:





تقوم مرشحات التنعيم بتشويش الصورة رغم أنها تخفف الضجيج والسبب أنها تستهدف الحواف بنفس طريقة استهدافها للضجيج.

1.6 التجزيء (كشف الأجسام) باستخدام مرشحات التنعيم:

يمكن تنفيذ عملية تجزيء بسيطة يتم فيها إلغاء المناطق غير المهمة والاحتفاظ بالمناطق المهمة في الصورة من خلال:

1- تطبيق مرشح تنعيم على الصورة يقوم بتشويه المناطق التي تمتلك أبعاداً أقل من أبعاد المرشح ويحافظ على المناطق المهمة التي تمتلك أبعاداً أكبر من أبعاد المرشح (على الرغم من أنه سينعم حوافها).

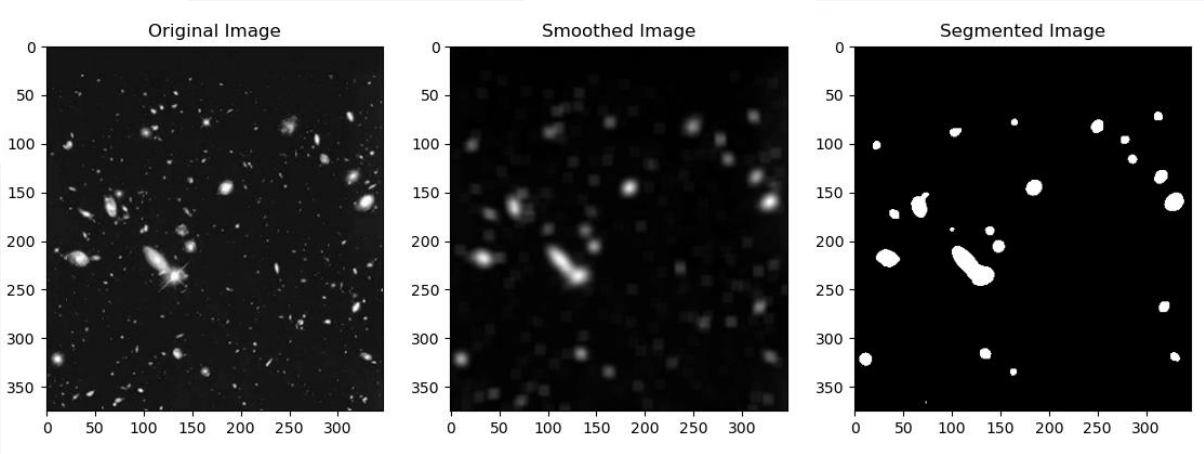
2- تطبيق عملية تعتيب Threshold على الصورة الناتجة من الطلب 1 بحيث يتم حذف السويات الرمادية الضعيفة التي تمثل المناطق التي تم تشويشها من قبل مرشح التنعيم والاحتفاظ بالمناطق المهمة.

نفذ الكود:

```
gray_image=cv2.imread('space.png',0)

#h=np.matrix([[1/9,1/9,1/9],[1/9,1/9,1/9],[1/9,1/9,1/9]])

k = (1/(11*11))*np.ones((11,11))

Median_filter_image=cv2.filter2D(src=gray_image, kernel=k, ddepth=-1)

_,bw_image=cv2.threshold(Median_filter_image, 127, 255, cv2.THRESH_BINARY)

# Create a single subplot for all images
```

```
fig, axs = plt.subplots(1, 3)
axs[0].imshow(gray_image, cmap='gray')
axs[0].set_title('Original Image')
axs[1].imshow(Median_filter_image, cmap='gray')
axs[1].set_title('Smoothed Image')
axs[2].imshow(bw_image, cmap='gray')
axs[2].set_title('Segmented Image')
plt.show()
```

لتحصل على النتيجة:



1.7 مرشحات الحد Sharpening Filters

تقوم بأحد مهمتين:

- كشف الحواف Edge Detection (من خلال تطبيق عملية اشتقاق درجة أولى أو ثانية ثم عملية تعتيب Threshold)
- تعزيز حواف Edge Enhancement من خلال طرح الصورة المشتقة من الصورة الأصلية.

لها نوعان مشتقات درجة أولى (Sobel, prewitt) ومشتقات درجة ثانية مثل (Laplacian)

1.7.1 Sobel Filter

مرشح سوبل لكشف الحواف الأفقية

يتم استخدام المصفوفة التالية في عملية الترشيح:

$$\begin{matrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{matrix}$$

مرشح سوبل لكشف الحواف العمودية

يتم استخدام المصفوفة التالية في عملية الترشيح:

$$\begin{matrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{matrix}$$

نفذ الكود التالي:

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
image=cv2.imread("edges.jpg",0)
Hy=np.array([[1,2,1],[0,0,0],[-1,-2,-1]])
horizontal_filtering= cv2.filter2D(src=image, kernel=Hy, ddepth=-1)
```

قراءة صورة رمادية ثم بناء مرشح سوبل لكشف التغيرات العمودية (الحواف الأفقية) ثم تطبيقه على الصورة

```
Hx= np.array( [[1,0,-1],[2,0,-2],[1,0,-1]])
vertical_filtering=cv2.filter2D(src=image,kernel=horizantal_sobel,
ddepth=-1)
```

بناء مرشح سوبل لكشف التغيرات الأفقية (الحواف العمودية) ثم تطبيقه على الصورة

```
final_image=np.sqrt(pow(horizontal_filtering,2.0)+
pow(vertical_filtering, 2.0))
```

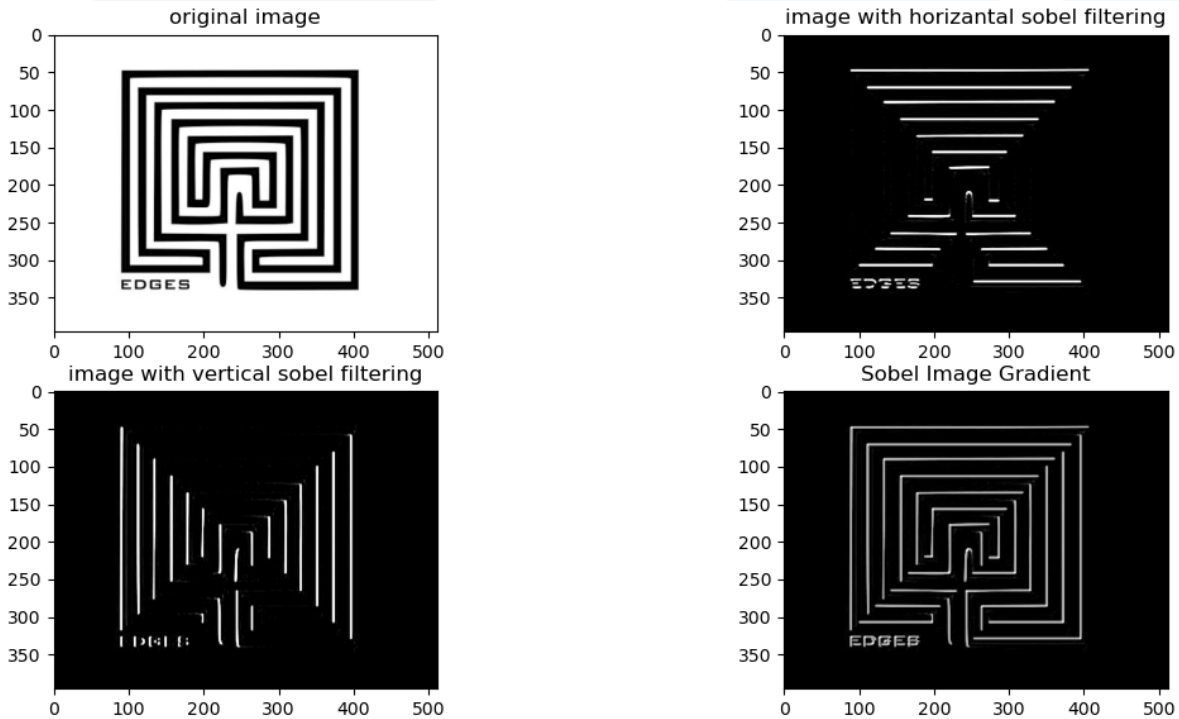
حساب طويلة التدرج Gradient من خلال حساب الجذر التربيعي لمجموع مربعات قيم ناتجي الترشيح باستخدام سوبل

```
plt.figure(figsize=(30,15))
plt.subplot(221)
plt.imshow(image, cmap='gray')
plt.title("original image")
plt.subplot(222)
```

عرض النتائج

```
plt.imshow(horizontal_filtering, cmap='gray')
plt.title("image with horizantal sobel filtering")
plt.subplot(223)
plt.imshow(vertical_filtering, cmap='gray')
plt.title("image with vertical sobel filtering")
plt.subplot(224)
plt.imshow(final_image, cmap='gray')
plt.title("Sobel Image Gradient")
plt.show()
```

والنتيجة:



1.7.2 كشف الحواف باستخدام مرشحات الدرجة الثانية Laplacian

يستخدم قناع واحد فقط لكشف الحواف الأفقية والعمودية.

$$\begin{matrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{matrix} \qquad \begin{matrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{matrix} \qquad \begin{matrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{matrix} \qquad \begin{matrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{matrix}$$

أي أن استجابته مضاعفة ومستقلة عن الاتجاه.

يمكنه كشف نقطة معزولة داكنة أو فاتحة.

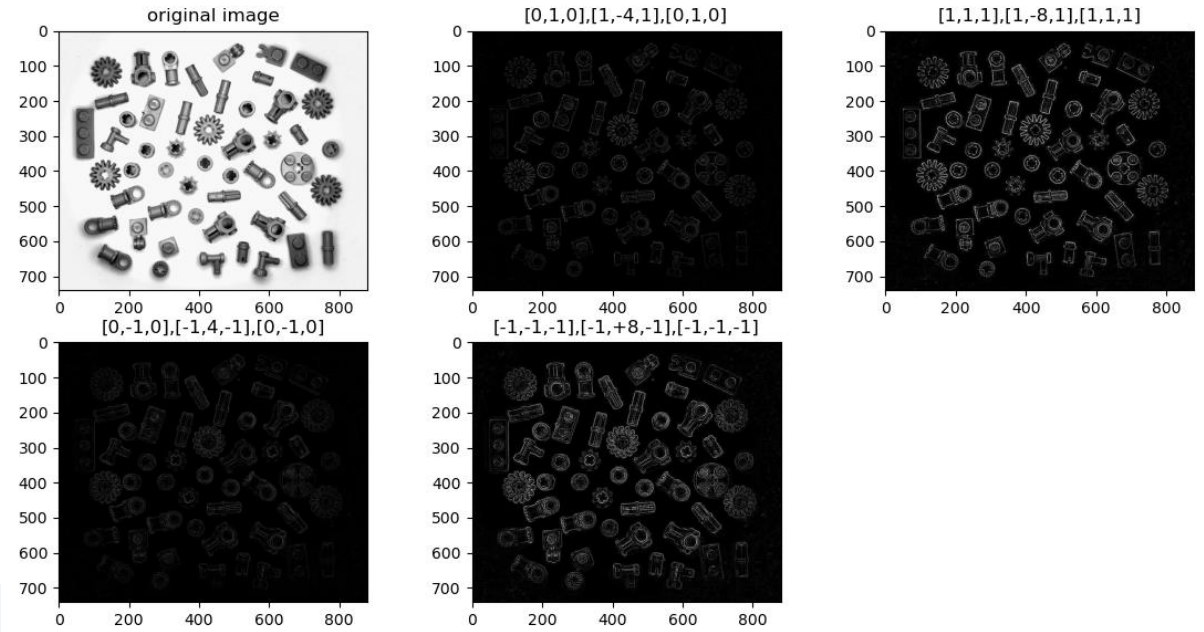
ويمكن استخدامه لكشف الحواف بشكل عام حيث يطبق مرة واحدة فقط.

نفذ الكود التالي الذي يقوم بتطبيق أقنعة اللابلاسيان الأربعة على الصورة باستخدام تابع cv2.filter2D:

```
image=cv2.imread('edge.jpg',0)
laplacian1=np.array([[0,1,0],[1,-4,1],[0,1,0]])
laplacian2=np.array([[1,1,1],[1,-8,1],[1,1,1]])
laplacian3=np.array([[0,-1,0],[-1,4,-1],[0,-1,0]])
laplacian4=np.array([[-1,-1,-1],[-1,+8,-1],[-1,-1,-1]])
image_laplacian1=cv2.filter2D(src=image, kernel=laplacian1, ddepth=-1)
image_laplacian2=cv2.filter2D(src=image, kernel=laplacian2, ddepth=-1)
image_laplacian3=cv2.filter2D(src=image, kernel=laplacian3, ddepth=-1)
image_laplacian4=cv2.filter2D(src=image, kernel=laplacian4, ddepth=-1)
plt.figure(figsize=(30,15))
plt.subplot(231)
plt.imshow(image, cmap='gray')
plt.title("original image")
plt.subplot(232)
plt.imshow(image_laplacian1, cmap='gray')
plt.title("[0,1,0],[1,-4,1],[0,1,0]")
plt.subplot(233)
plt.imshow(image_laplacian2, cmap='gray')
plt.title("[1,1,1],[1,-8,1],[1,1,1]")
plt.subplot(234)
plt.imshow(image_laplacian3, cmap='gray')
plt.title("[0,-1,0],[-1,4,-1],[0,-1,0]")
plt.subplot(235)
plt.imshow(image_laplacian4, cmap='gray')
plt.title("[-1,-1,-1],[-1,+8,-1],[-1,-1,-1]")
```

```
plt.show()
```

والنتيجة



1.7.3 تعزيز الحواف Edge Enhancement

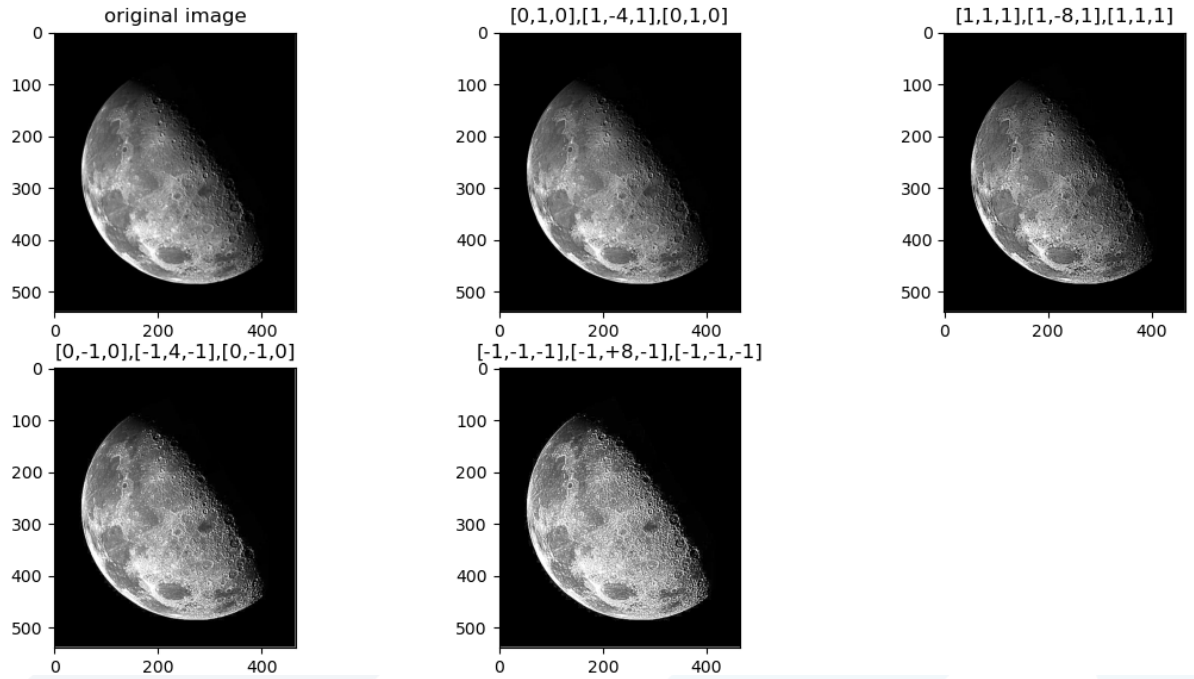
يتم من خلال تطبيق مرشح Laplacian على الصورة ثم ضرب ناتج الترشيح بمعامل توضيح الحواف وهو قيمة سالبة في حال كان مركز مرشح Laplacian سالب وموجب في حال كان مركز مرشح Laplacian موجب. يمكن اختيار قيم صغيرة لهذا المعامل لأن اختيار قيم عالية تؤدي لنتائج غير جيدة.

يتم أيضاً إجراء عملية قص Clip للسويات الرمادية لتكون ضمن المجال 0-255 حتى لا نحصل على استجابة خاطئة خارج المجال المطلوب أو يمكن التحويل لنمط Double ثم تطبيق عملية الترشيح.

```
import cv2

import numpy as np

import matplotlib.pyplot as plt

image=cv2.imread('north pole.png',0)

laplacian1=np.array([[0,1,0],[1,-4,1],[0,1,0]])

laplacian2=np.array([[1,1,1],[1,-8,1],[1,1,1]])

laplacian3=np.array([[0,-1,0],[-1,4,-1],[0,-1,0]])

laplacian4=np.array([[-1,-1,-1],[-1,+8,-1],[-1,-1,-1]])

image_laplacian1=cv2.filter2D(src=image, kernel=laplacian1, ddepth=-1)
```

```
image_laplacian2=cv2.filter2D(src=image, kernel=laplacian2, ddepth=-1)
image_laplacian3=cv2.filter2D(src=image, kernel=laplacian3, ddepth=-1)
image_laplacian4=cv2.filter2D(src=image, kernel=laplacian4, ddepth=-1)
image_sharping_1=np.clip(image-0.5*image_laplacian1, 0, 255)
image_sharping_2=np.clip(image-0.5*image_laplacian2, 0, 255)
image_sharping_3=np.clip(image+1.2*image_laplacian3, 0, 255)
image_sharping_4=np.clip(image+1.2*image_laplacian4, 0, 255)
plt.figure(figsize=(30,15))
plt.subplot(231)
plt.imshow(image, cmap='gray')
plt.title("original image")
plt.subplot(232)
plt.imshow(image_sharping_1, cmap='gray')
plt.title("[0,1,0],[1,-4,1],[0,1,0]")
plt.subplot(233)
plt.imshow(image_sharping_2, cmap='gray')
plt.title("[1,1,1],[1,-8,1],[1,1,1]")
plt.subplot(234)
plt.imshow(image_sharping_3, cmap='gray')
plt.title("[0,-1,0],[-1,4,-1],[0,-1,0]")
plt.subplot(235)
plt.imshow(image_sharping_4, cmap='gray')
plt.title("[-1,-1,-1],[-1,+8,-1],[-1,-1,-1]")
plt.show()
```

والنتيجة:



يمكنك التحكم بدرجة التوضيح من خلال زيادة معامل توضيح الحواف أو إنقاصه.

1.7.4 مرشح Unsahrp

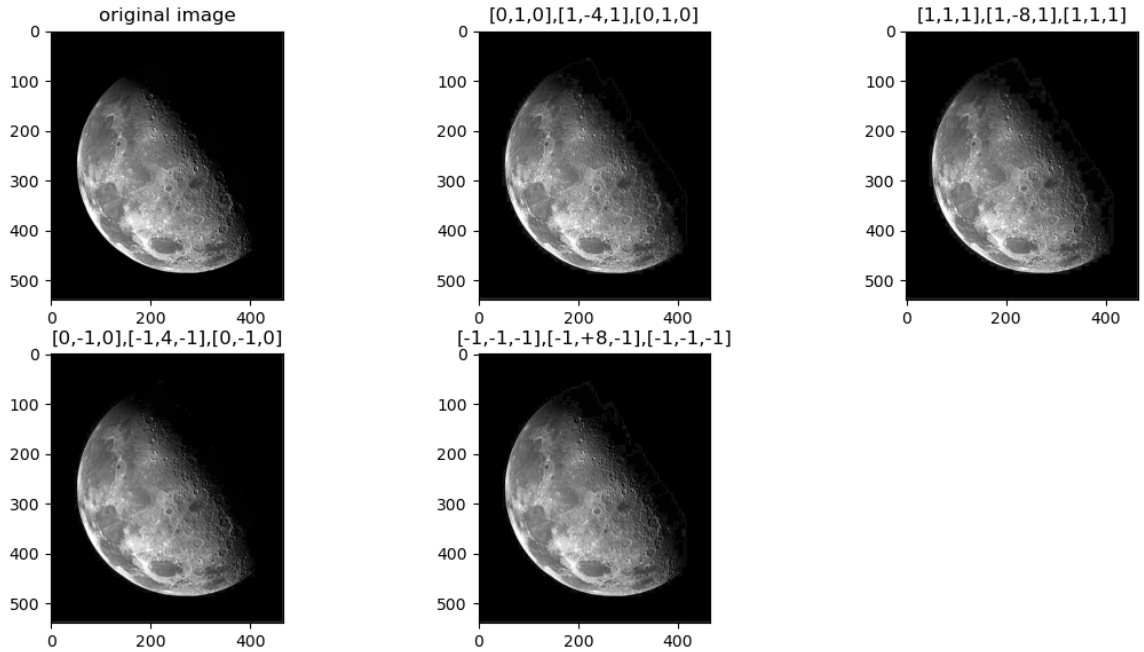
المقصود بهذه العملية طرح نسخة منعمة للصورة من الصورة الأصلية نفسها بعد ضرب النسخة المنعمة بمعامل توضيح الحواف.

نفذ الكود

```
k = (1/(3*3))*np.ones((3,3))
smooth_filter_image=cv2.filter2D(src=image, kernel=k, ddepth=-1)
result_image1=image-smooth_filter_image
image_unsharping_1=np.clip(image+0.1*result_image1,0,255)
image_unsharping_2=np.clip(image+0.2*result_image1,0,255)
image_unsharping_3=np.clip(image+0.3*result_image1,0,255)
image_unsharping_4=np.clip(image+0.4*result_image1,0,255)
```

ترشيح الصورة أولاً باستخدام مرشح تنعيم (مثلا المتوسط 3*3)
ثم تطرح من الصورة الأصلية
ثم يضرب الناتج بمعامل توضيح الحواف K

والنتيجة



1.8 التنعيم مع المحافظة على الحواف Bilateral Filtering

يتضمن هذا المرشح جداء مرشحين غوصيين معاً، الأول يسمى Brightness Gaussian وهو مسؤول عن عملية تنعيم السويات الرمادية والثاني spatial Gaussian وهو يعطي وزن عالي للبكسل في حال كان متشابه مع مجاوراته ووزن منخفض في حال كان حافة (مختلف عن مجاوراته) بحيث تخف عملية الترشيح عند بكسلات الحواف وتؤثر بشكل أكبر على البكسلات الضجيجية.

نفذ الكود:

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
image=cv2.imread('woman.png')
image =cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
blur1 = cv2.bilateralFilter(image,10,10, 10)
blur2 = cv2.bilateralFilter(image,10,25, 25)
plt.figure(figsize=(30,15))
plt.subplot(131)
plt.imshow(image, cmap='gray')
plt.title("original image")
```

تطبيق مرشح Bilateral باستخدام نافذة بقطر 10 وانحرافات معيارية متشابهة 25 لكل من σ_b, σ_s

```
plt.subplot(132)
plt.imshow(blur1, cmap='gray')
plt.subplot(133)
plt.imshow(blur2, cmap='gray')
plt.show()
```

والنتيجة باستخدام قيم انحرافات معيارية 25 ثم 75.