20115621-buidinhhanhdu-lap4

September 30, 2023

[67]:
```
#khai báo thư viện sử dụng trong bài
import cv2
import matplotlib.pyplot as plt
import numpy as np
from scipy import ndimage
```

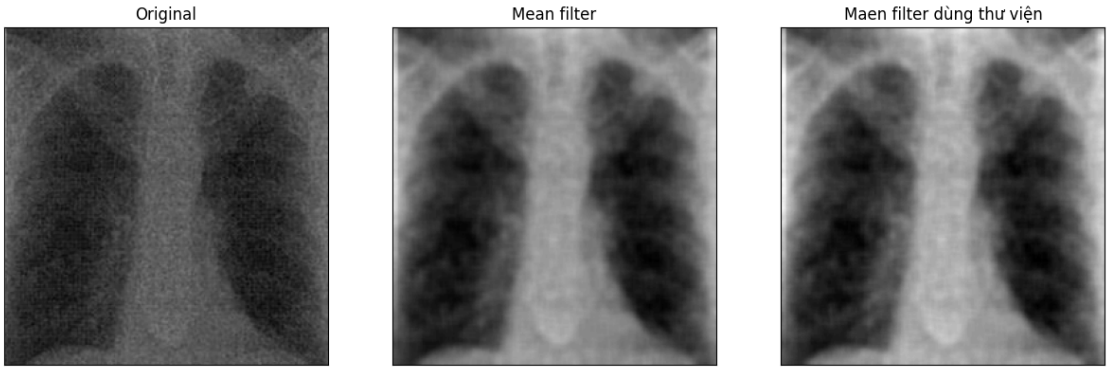
[68]:
```
#Đọc hình ảnh
img = cv2.imread('img.png', cv2.IMREAD_GRAYSCALE)
```

[69]:
```
#Hàm mean filter
def mean_filter(image, kernel_size):

    #Lấy kích thước của ảnh
    h, w = image.shape

    #Tạo ma trận mới có kích thước bằng ảnh
    filtered_image = np.zeros((h, w),dtype=np.uint8)

    #Duyệt qua từng pixel trong ảnh
    for i in range (h):
        for j in range (w):
            start_i = max(0, i - kernel_size // 2)
            end_i = min(h, i + kernel_size // 2 + 1)
            start_j = max(0, j - kernel_size // 2)
            end_j = min(w , j +kernel_size //2 + 1)

            #Lấy giá trị mean
            mean_value = np.mean(image[start_i: end_i, start_j: end_j])

            #Gán giá trị mean
            filtered_image[i, j] = mean_value

    return filtered_image
```

[70]:
```
mean_image = mean_filter(img, 7)

kernel_size = (7 ,7)
```

```
mean_image_li = cv2.blur(img, kernel_size)
```

[71]:
```
#Hiển thị kết quả
plt.figure(figsize= (15, 5))

plt.subplot(131),plt.imshow(img, cmap= 'gray'),plt.title('Original')
plt.xticks([]), plt.yticks([])

plt.subplot(132),plt.imshow(mean_image, cmap= 'gray'), plt.title('Mean filter')
plt.xticks([]), plt.yticks([])

plt.subplot(133),plt.imshow(mean_image_li, cmap= 'gray'), plt.title('Maen␣
 ↪filter dùng thư viện')
plt.xticks([]), plt.yticks([])

plt.show()
```

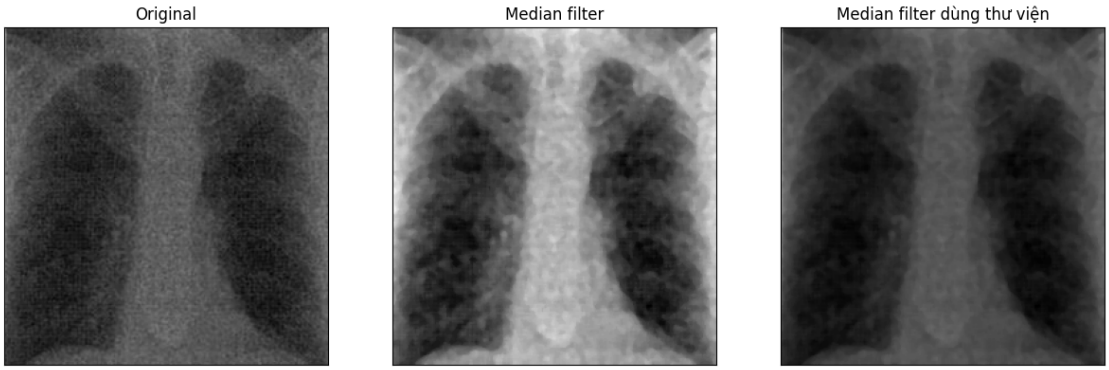


[72]:
```
def median_filter(image, kernel_size):
    #Lấy kích thước của ảnh
    h, w = image.shape

    #Tạo ma trận mới có kích thước bằng ảnh
    filtered_image = np.zeros((h, w),dtype=np.uint8)

    #Duyệt qua từng pixel trong ảnh
    for i in range (h):
        for j in range (w):
            start_i = max(0, i - kernel_size // 2)
            end_i = min(h, i + kernel_size // 2 + 1)
            start_j = max(0, j - kernel_size // 2)
            end_j = min(w , j +kernel_size //2 + 1)

            #Lấy giá trị median
```

```
            median_value = np.median(image[start_i: end_i, start_j: end_j])

            #Gắn giá trị trung bình
            filtered_image[i, j] = median_value

    return filtered_image
```

[73]:
```
median_image = median_filter(img, 5)

median_image_li = cv2.medianBlur(img, 5)
```

[74]:
```
#Hiển thị kết quả
plt.figure(figsize= (15, 5))

plt.subplot(131),plt.imshow(img, cmap= 'gray'),plt.title('Original')
plt.xticks([]), plt.yticks([])

plt.subplot(132),plt.imshow(median_image, cmap= 'gray'), plt.title('Median␣
 ↪filter')
plt.xticks([]), plt.yticks([])

plt.subplot(133),plt.imshow(median_image_li, cmap= 'gray'), plt.title('Median␣
 ↪filter dùng thư viện')
plt.xticks([]), plt.yticks([])

plt.show()
```



[75]:
```
def min_filter(image, kernel_size):
    #Lấy kích thước ảnh
    h, w = image.shape

    #Tạo ma trận bằng kích thước ảnh
    filtered_image = np.zeros((h, w),dtype=np.uint8)
```

```
    #Duyệt qua từng pixel trong ảnh
    for i in range (h):
        for j in range (w):
            start_i = max(0, i - kernel_size // 2)
            end_i = min(h, i + kernel_size // 2 + 1)
            start_j = max(0, j - kernel_size // 2)
            end_j = min(w , j +kernel_size //2 + 1)

            #Lấy giá trị min
            min_value = np.min(image[start_i: end_i, start_j: end_j])

            #Gắn giá trị min
            filtered_image[i, j] = min_value

    return filtered_image
```

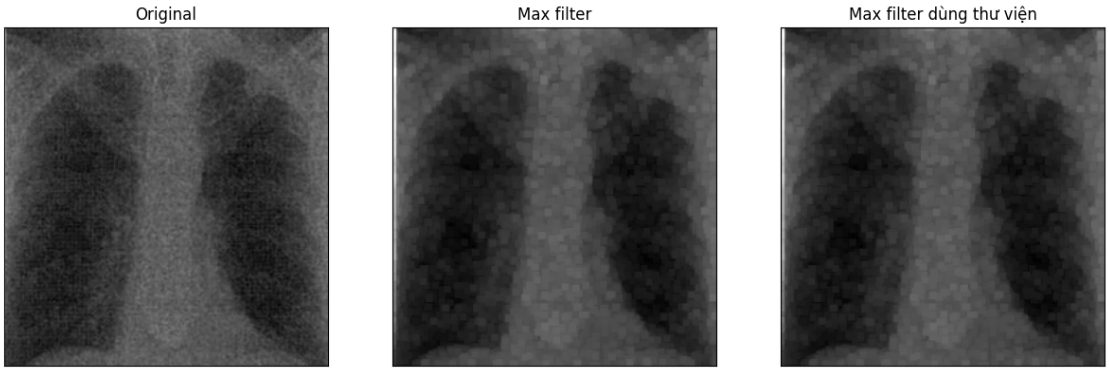
[76]:
```
min_image = min_filter(img, 5)

kernel = np.ones((5, 5), np.uint8)
min_image_li = cv2.erode(img, kernel)
```

[77]:
```
#Hiển thị kết quả
plt.figure(figsize= (15, 5))
plt.subplot(131),plt.imshow(img, cmap= 'gray'),plt.title('Original')
plt.xticks([]), plt.yticks([])

plt.subplot(132),plt.imshow(min_image, cmap= 'gray'), plt.title('Min filter')
plt.xticks([]), plt.yticks([])

plt.subplot(133),plt.imshow(min_image_li, cmap= 'gray'), plt.title('Min filter␣
 ↪dùng thư viện')
plt.xticks([]), plt.yticks([])

plt.show()
```

[78]:
```
def max_filter(image, kernel_size):
    #Lấy kích thước ảnh
    h, w = image.shape

    #Tạo ma trận bằng kích thước ảnh
    filtered_image = np.zeros((h, w),dtype=np.uint8)

    #duyệt qua các pixel
    for i in range (h):
        for j in range (w):
            start_i = max(0, i - kernel_size // 2)
            end_i = min(h, i + kernel_size // 2 + 1)
            start_j = max(0, j - kernel_size // 2)
            end_j = min(w , j +kernel_size //2 + 1)

            #Lấy giá trị max
            max_value = np.max(image[start_i: end_i, start_j: end_j])

            #Gắn giá trị max
            filtered_image[i, j] = max_value

    return filtered_image
```

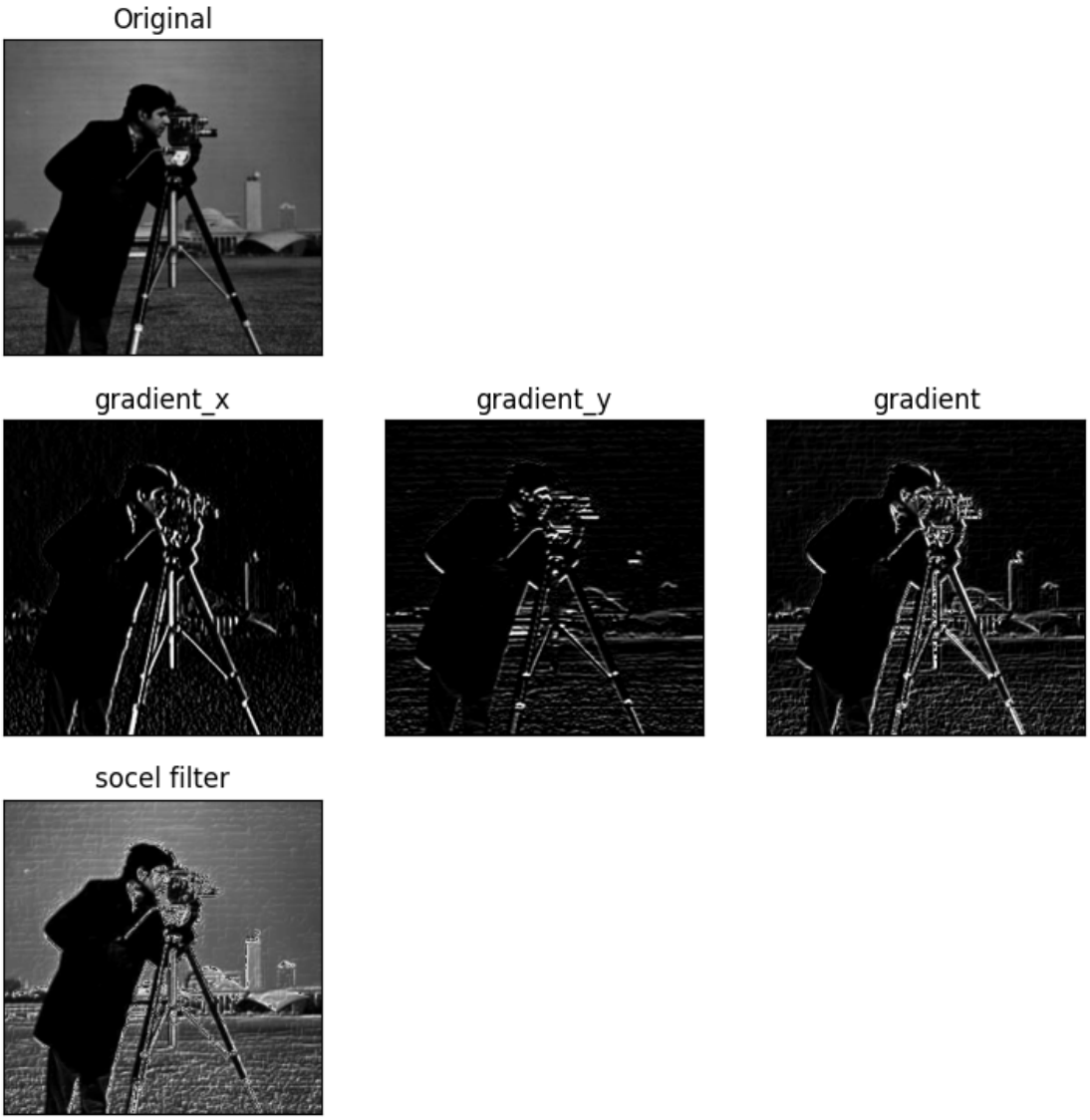
[79]:
```
max_image = max_filter(img, 5)

kernel = np.ones((5 ,5 ), np.uint8)
max_image_li = cv2.dilate(img, kernel)
```

[80]:
```
#Hiển thị kết quả
plt.figure(figsize= (15, 5))

plt.subplot(131),plt.imshow(img, cmap= 'gray'),plt.title('Original')
plt.xticks([]), plt.yticks([])

plt.subplot(132),plt.imshow(max_image, cmap= 'gray'), plt.title('Max filter')
plt.xticks([]), plt.yticks([])

plt.subplot(133),plt.imshow(max_image_li, cmap= 'gray'), plt.title('Max filter␣
 ↪dùng thư viện')
plt.xticks([]), plt.yticks([])

plt.show()
```



[81]:
```
img = cv2.imread('image1.png', cv2.IMREAD_GRAYSCALE)

sobel_x = np.array([[-1, 0, 1], [-2, 0, 2], [-1, 0, 1]], dtype=np.float32)
sobel_y = np.array([[-1, -2, -1], [0, 0, 0], [1, 2, 1]], dtype=np.float32)

# Apply the Sobel filters using cv2.filter2d
gradient_x = cv2.filter2D(img, -1, sobel_x)
gradient_y = cv2.filter2D(img, -1, sobel_y)
gradient = gradient_x + gradient_y
# Optionally, you can also combine the filtered outputs to get the final edge␣
 ↪map
edge = img + gradient
```

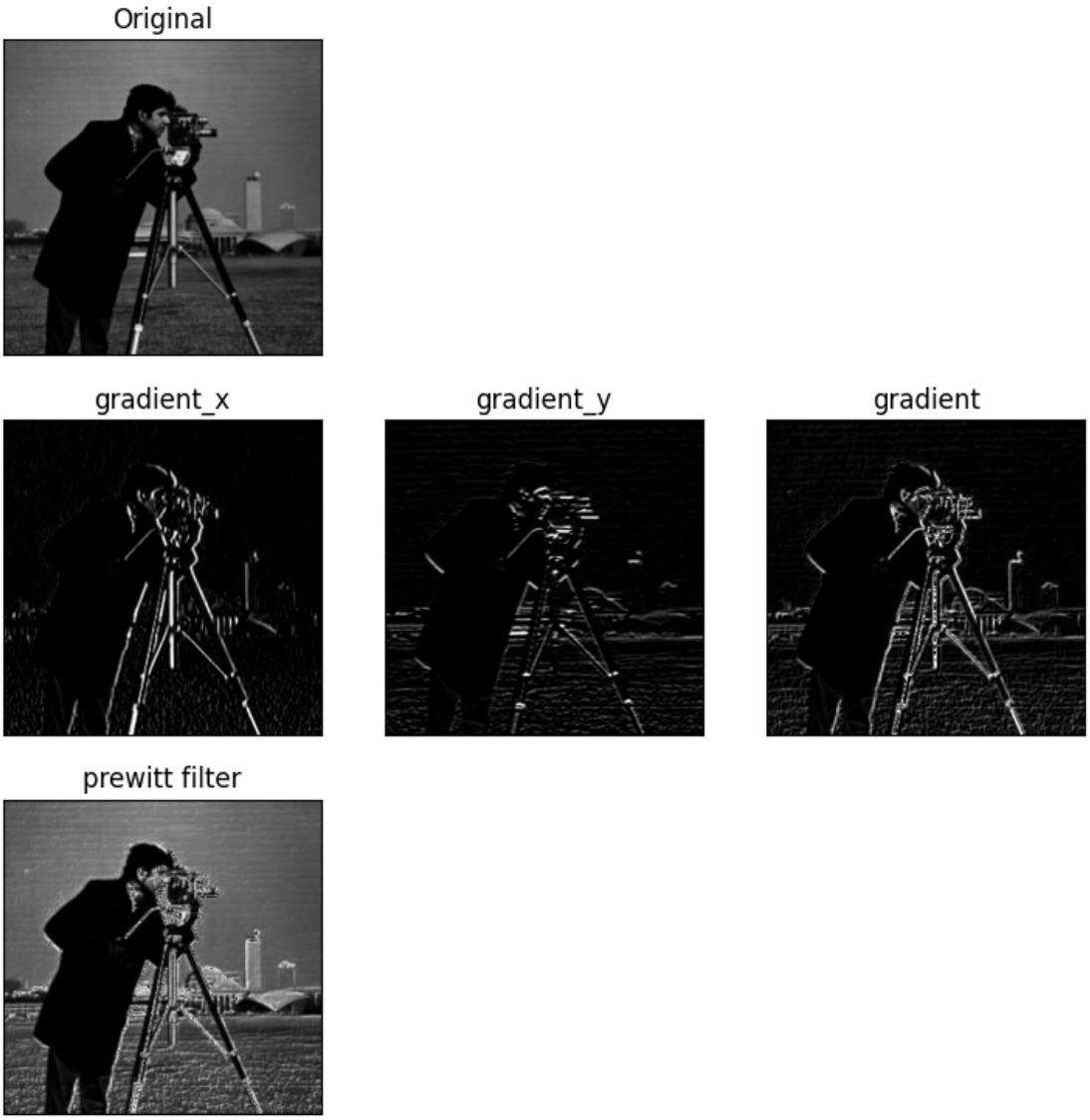
[82]:
```
#Hiển thị kết quả
plt.figure(figsize= (9, 9))

plt.subplot(331),plt.imshow(img, cmap= 'gray'),plt.title('Original')
plt.xticks([]), plt.yticks([])


plt.subplot(334),plt.imshow(gradient_x, cmap= 'gray'), plt.title('gradient_x')
plt.xticks([]), plt.yticks([])

plt.subplot(335),plt.imshow(gradient_y, cmap= 'gray'), plt.title('gradient_y')
plt.xticks([]), plt.yticks([])

plt.subplot(336),plt.imshow(gradient, cmap= 'gray'), plt.title('gradient')
plt.xticks([]), plt.yticks([])

plt.subplot(337),plt.imshow(edge, cmap= 'gray'), plt.title('socel filter')
plt.xticks([]), plt.yticks([])

plt.show()
```



[83]:
```
# Apply the Prewitt filter
prewitt_x = np.array([[-1, 0, 1],
                      [-1, 0, 1],
                      [-1, 0, 1]])
prewitt_y = np.array([[-1, -1, -1],
                      [0, 0, 0],
                      [1, 1, 1]])

gradient_x = cv2.filter2D(img, -1, prewitt_x)
gradient_y = cv2.filter2D(img, -1, prewitt_y)

gradient = gradient_x + gradient_y
```

```
edge = img + gradient
```

[84]:
```
#Hiển thị kết quả
plt.figure(figsize= (9, 9))

plt.subplot(331),plt.imshow(img, cmap= 'gray'),plt.title('Original')
plt.xticks([]), plt.yticks([])


plt.subplot(334),plt.imshow(gradient_x, cmap= 'gray'), plt.title('gradient_x')
plt.xticks([]), plt.yticks([])

plt.subplot(335),plt.imshow(gradient_y, cmap= 'gray'), plt.title('gradient_y')
plt.xticks([]), plt.yticks([])

plt.subplot(336),plt.imshow(gradient, cmap= 'gray'), plt.title('gradient')
plt.xticks([]), plt.yticks([])

plt.subplot(337),plt.imshow(edge, cmap= 'gray'), plt.title('prewitt filter')
plt.xticks([]), plt.yticks([])

plt.show()
```



[85]:
```
t_lower = 100 # Lower Threshold
t_upper = 100 # Upper threshold

# Applying the Canny Edge filter
gradient = cv2.Canny(img, t_lower, t_upper)

edge = img + gradient
```

[86]:
```
plt.figure(figsize= (10, 5))

plt.subplot(131),plt.imshow(img, cmap= 'gray'),plt.title('Original')
plt.xticks([]), plt.yticks([])
```

```
plt.subplot(132),plt.imshow(gradient, cmap= 'gray'), plt.title('gradient')
plt.xticks([]), plt.yticks([])

plt.subplot(133),plt.imshow(edge, cmap= 'gray'), plt.title('Candy filter')
plt.xticks([]), plt.yticks([])

plt.show()
```