20115621-buidinhhanhdu-lap

October 7, 2023

[115]:
```
#khái báo thư viện
import matplotlib.pyplot as plt
import cv2
import numpy as np
```

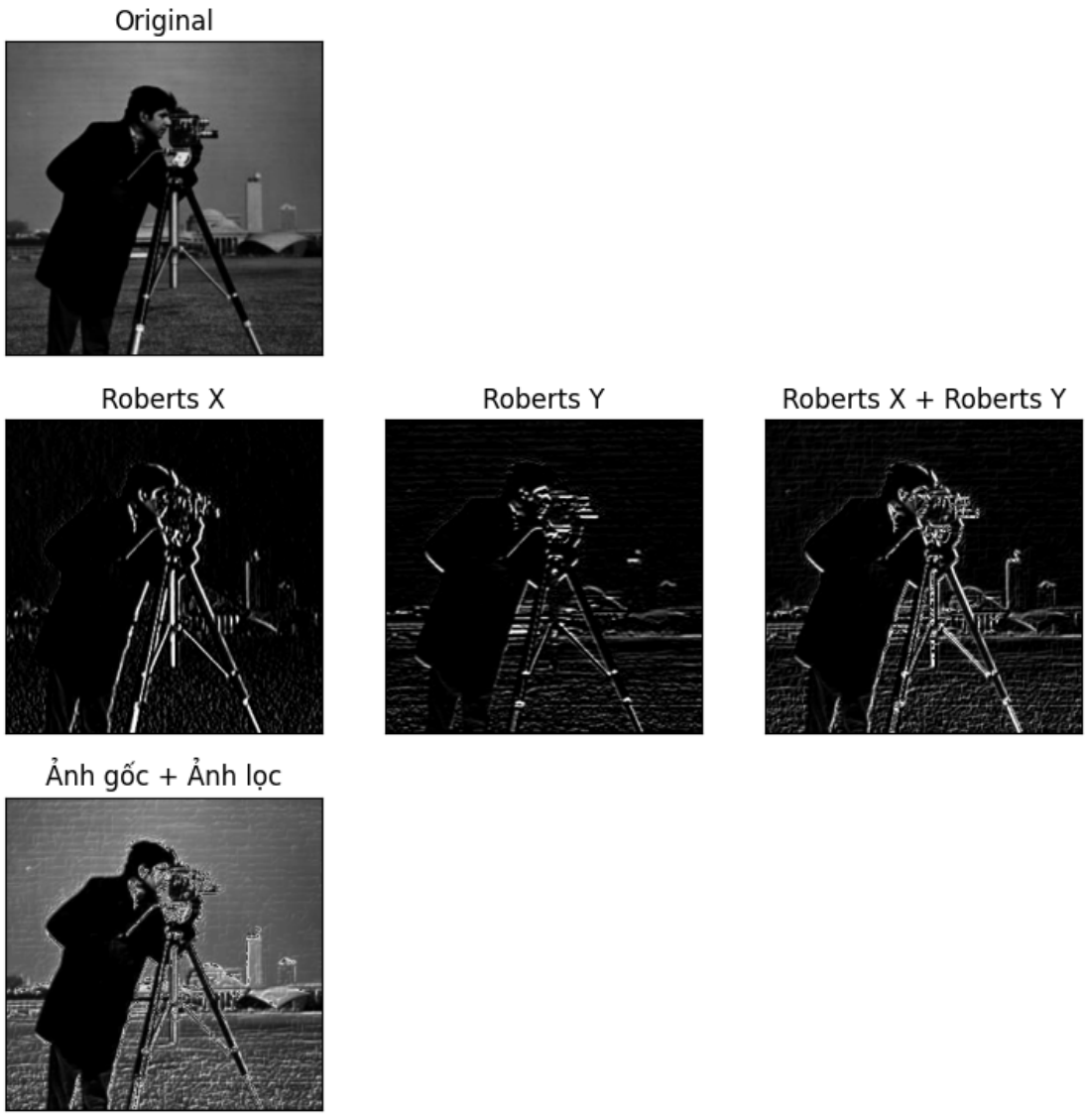
[116]:
```
#Đọc ảnh và chuyển ảnh sang màu xám
img = cv2.imread('image1.png', cv2.IMREAD_GRAYSCALE)
```

1. Roberts

[117]:
```
#Ma trận bộ lọc Roberts
#Chiều x
Gx = np.array([[-1, 0, 1], [-2, 0, 2], [-1, 0, 1]], dtype=np.float32)
#Chiều y
Gy = np.array([[-1, -2, -1], [0, 0, 0], [1, 2, 1]], dtype=np.float32)

# Áp dụng lọc Roberts
#Ảnh lọc X
dx = cv2.filter2D(img, -1, Gx)
#Ảnh lọc Y
dy = cv2.filter2D(img, -1, Gy)

# Ảnh lọc X + ảnh lọc  Y
gradient = dx + dy

#Ảnh gốc + ảnh lọc
edg = img + gradient
```

[118]:
```
#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(dx, cmap= 'gray'), plt.title('Roberts X')
plt.xticks([]), plt.yticks([])
```

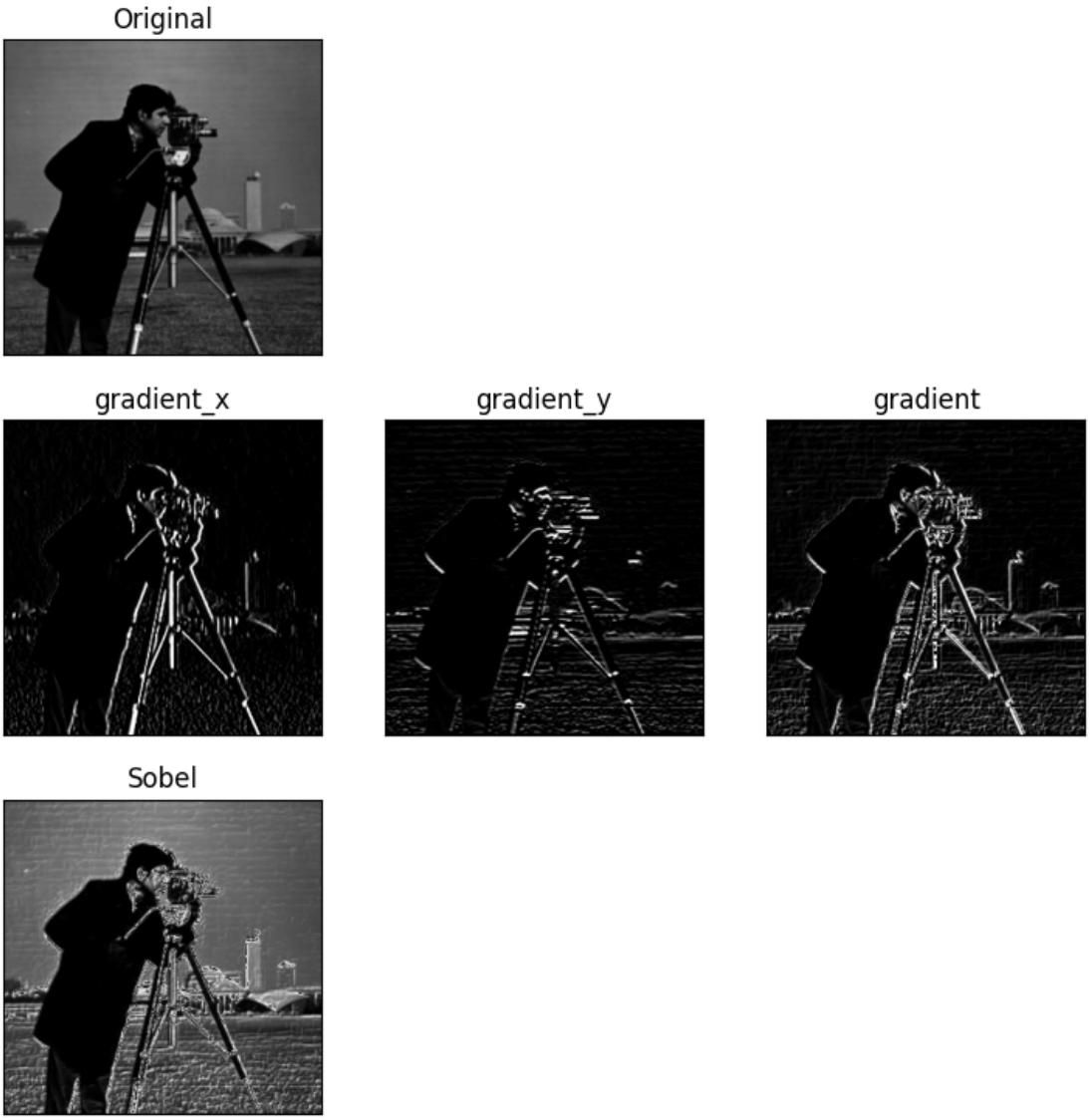
```
plt.subplot(335),plt.imshow(dy, cmap= 'gray'), plt.title('Roberts Y')
plt.xticks([]), plt.yticks([])

plt.subplot(336),plt.imshow(gradient, cmap= 'gray'), plt.title('Roberts X +␣
 ↪Roberts Y')
plt.xticks([]), plt.yticks([])

plt.subplot(337),plt.imshow(edg, cmap= 'gray'), plt.title('Ảnh gốc + Ảnh lọc')
plt.xticks([]), plt.yticks([])

plt.show()
```



2. Sobel

[119]:
```
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
```

[120]:
```
#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('Sobel')
plt.xticks([]), plt.yticks([])

plt.show()
```



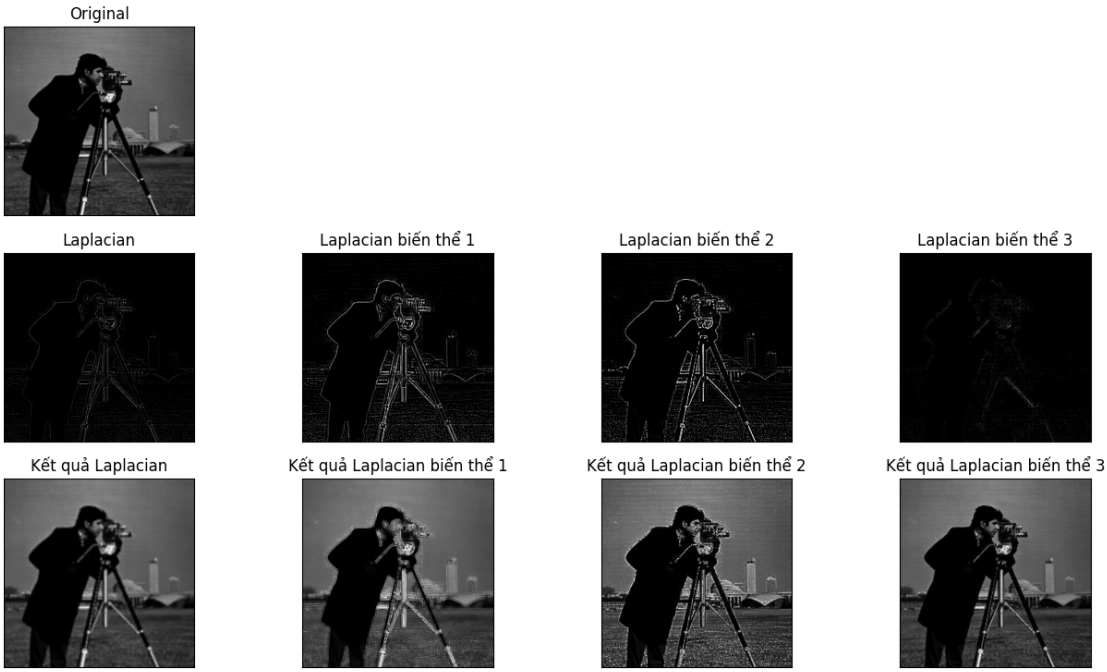
3. Laplacian

[121]:
```
#Ma trận bộ lọc Laplacian
#Chuẩn
Laplacian_chuan = np.array([[0, 1, 0], [1, -4, 1], [0, 1, 0]], dtype=np.float32)
#Biến thể 1
Laplacian_1 = np.array([[1, 1, 1], [1, -8, 1], [1, 1, 1]], dtype=np.float32)
#Biến thể 2
Laplacian_2 = np.array([[-1, -1, -1], [-1, 8, -1], [-1, -1, -1]], dtype=np.
↪float32)
#Biến thể 3
Laplacian_3 = np.array([[1, -2, 1], [-2, 4, -2], [1, -2, 1]], dtype=np.float32)
```

```
#Lọc ảnh
lc = cv2.filter2D(img, -1, Laplacian_chuan)
imc = img + lc

l1 = cv2.filter2D(img, -1, Laplacian_1)
im1 = img + l1

l2 = cv2.filter2D(img, -1, Laplacian_2)
im2 = img + l2

l3 = cv2.filter2D(img, -1, Laplacian_3)
im3 = img + l3
```

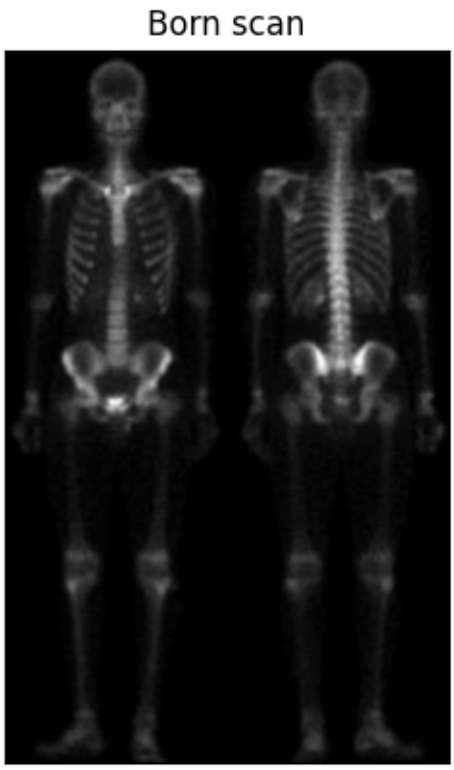
[122]:
```
#Hiển thị kết quả
plt.figure(figsize= (16, 9))

plt.subplot(341),plt.imshow(img, cmap= 'gray'),plt.title('Original')
plt.xticks([]), plt.yticks([])


plt.subplot(345),plt.imshow(lc, cmap= 'gray'), plt.title('Laplacian')
plt.xticks([]), plt.yticks([])

plt.subplot(346),plt.imshow(l1, cmap= 'gray'), plt.title('Laplacian biến thể 1')
plt.xticks([]), plt.yticks([])

plt.subplot(347),plt.imshow(l2, cmap= 'gray'), plt.title('Laplacian biến thể 2')
plt.xticks([]), plt.yticks([])

plt.subplot(348),plt.imshow(l3, cmap= 'gray'), plt.title('Laplacian biến thể 3')
plt.xticks([]), plt.yticks([])

plt.subplot(349),plt.imshow(imc, cmap= 'gray'), plt.title('Kết quả Laplacian')
plt.xticks([]), plt.yticks([])

plt.subplot(3, 4, 10),plt.imshow(im1, cmap= 'gray'), plt.title('Kết quả␣
 ↪Laplacian biến thể 1')
plt.xticks([]), plt.yticks([])

plt.subplot(3, 4, 11),plt.imshow(im2, cmap= 'gray'), plt.title('Kết quả␣
 ↪Laplacian biến thể 2')
plt.xticks([]), plt.yticks([])

plt.subplot(3, 4, 12),plt.imshow(im3, cmap= 'gray'), plt.title('Kết quả␣
 ↪Laplacian biến thể 3')
plt.xticks([]), plt.yticks([])
```

```
plt.show()
```



4.

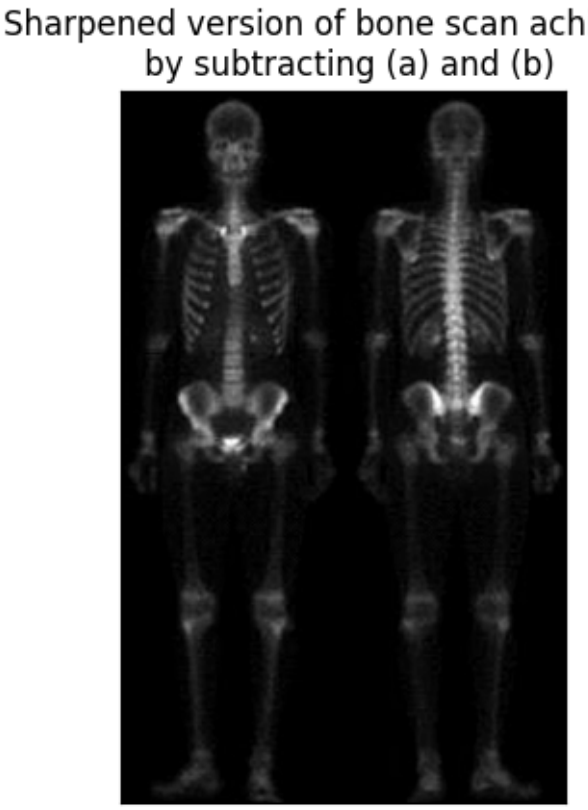
[123]:
```
img = cv2.imread('img.jpg', cv2.IMREAD_GRAYSCALE)
plt.imshow(img, cmap = 'gray')
plt.title('Born scan')
plt.xlabel ('(a)')
plt.xticks([]), plt.yticks([])

plt.show()
```



(a)

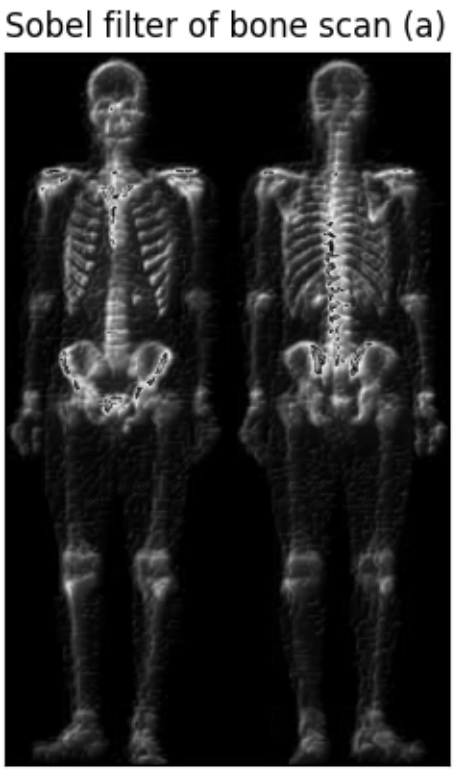
[124]:
```
blurred = cv2.GaussianBlur(img, (3, 3), 0)
laplacian = cv2.Laplacian(blurred, cv2.CV_64F)


plt.imshow(laplacian, cmap = 'gray')
plt.title('Laplacian filter of bone scan (a)')
plt.xlabel ('(b)')
plt.xticks([]), plt.yticks([])

plt.show()
```

[125]:
```
sharpened = cv2.subtract(img, laplacian, dtype=cv2.CV_8U)


plt.imshow(sharpened, cmap = 'gray')
plt.title('Sharpened version of bone scan achieved \n by subtracting (a) and␣
 ↪(b)')
plt.xlabel ('(c)')
plt.xticks([]), plt.yticks([])

plt.show()
```



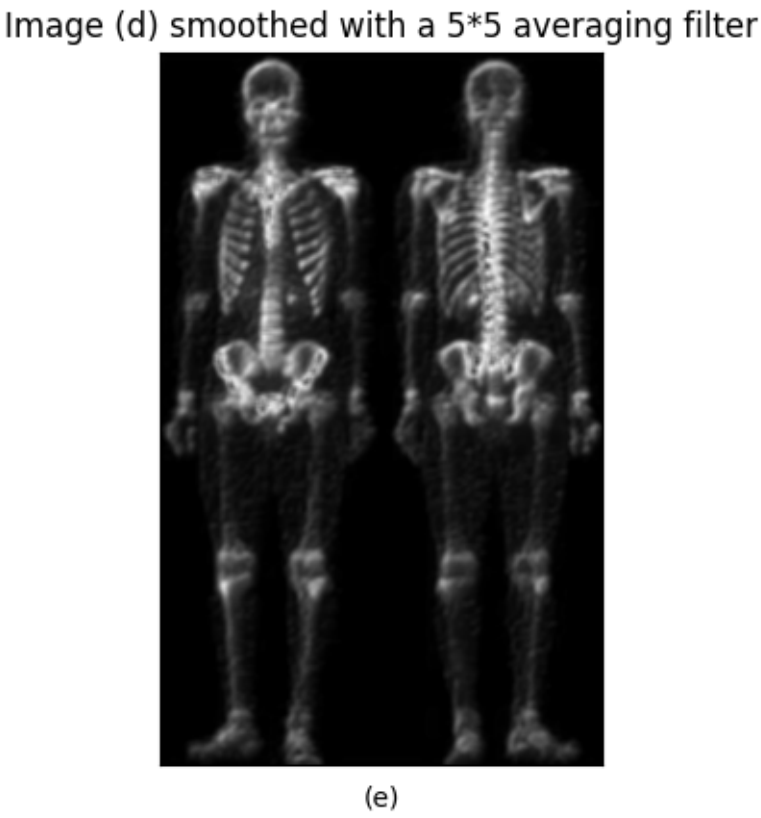
(c)

[126]:
```
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
sobel_filter = img + gradient

plt.imshow(sobel_filter, cmap = 'gray')
plt.title('Sobel filter of bone scan (a)')
plt.xlabel ('(d)')
plt.xticks([]), plt.yticks([])

plt.show()
```



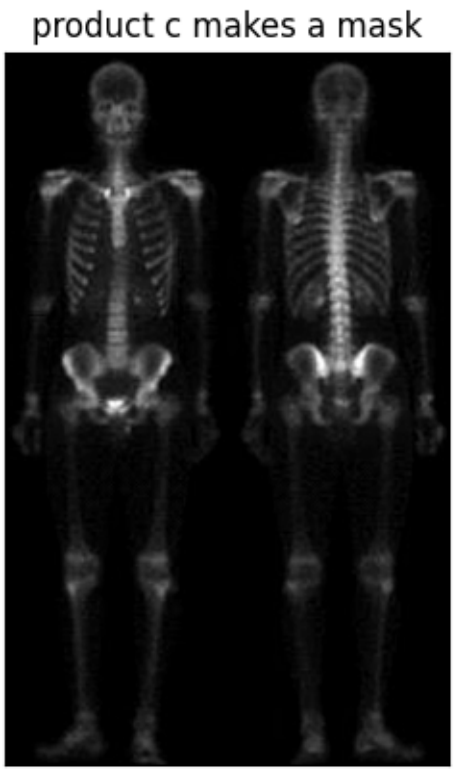
(d)

[127]:
```
# Define the kernel for smoothing
kernel = np.ones((5, 5), np.float32) / 25

# Apply the filter using OpenCV's filter2D function
smoothed_image = cv2.filter2D(sobel_filter, -1, kernel)

plt.imshow(smoothed_image, cmap = 'gray')
plt.title('Image (d) smoothed with a 5*5 averaging filter')
plt.xlabel ('(e)')
plt.xticks([]), plt.yticks([])

plt.show()
```



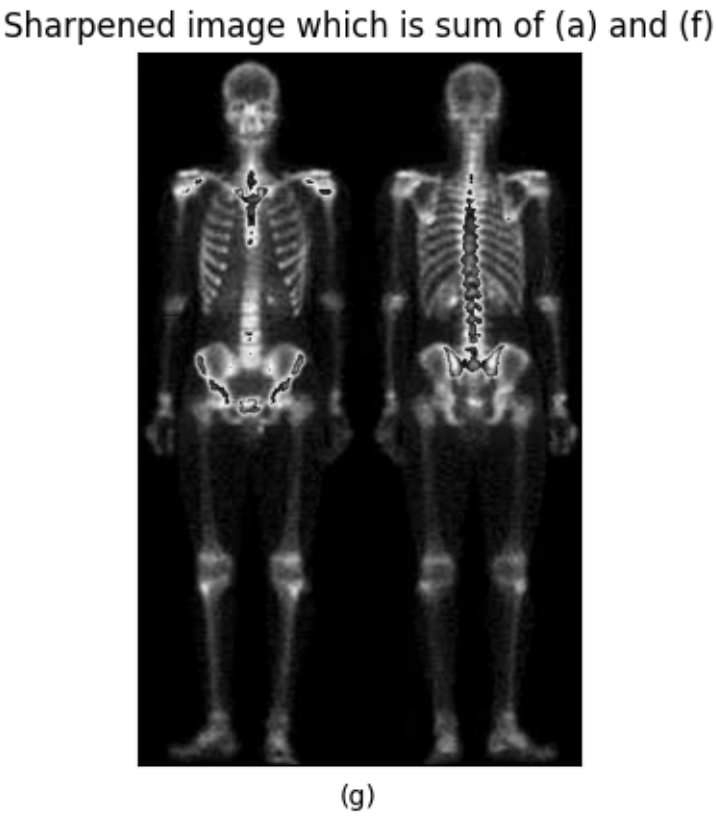
(e)

[128]:
```
plt.imshow(sharpened, cmap = 'gray')
plt.title('product c makes a mask')
plt.xlabel ('(f)')
plt.xticks([]), plt.yticks([])

plt.show()
```



(f)

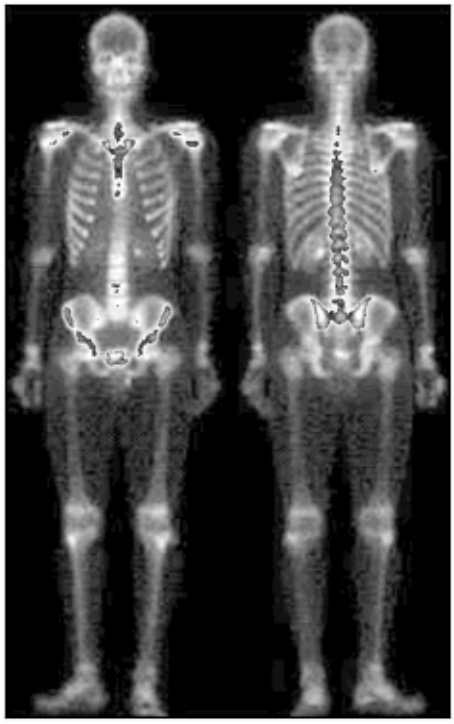
[129]:
```
sharpenedf = img + sharpened

plt.imshow(sharpenedf, cmap = 'gray')
plt.title('Sharpened image which is sum of (a) and (f)')
plt.xlabel ('(g)')
plt.xticks([]), plt.yticks([])

plt.show()
```



(g)

[130]:
```
c = 1.0  # Constant
gamma = 0.5  # Power exponent
output_image = np.power(c * sharpenedf, gamma).astype(np.uint8)

plt.imshow(output_image, cmap = 'gray')
plt.title('Result of applying a power-law trans. to (g)')
plt.xlabel ('(h)')
plt.xticks([]), plt.yticks([])

plt.show()
```

Result of applying a power-law trans. to (g)



(h)