Image Processing HW #4

**Edge Detection**

611410122 許恒誠

Data due:Jan. 6, 2023

Data handed in: Jan. 5, 2023

題目：Given three images, perform edge detection on the three images using Sobel and Laplacian of a Gaussian (LoG) operators.

Technical Description

**Package requirements:**

1. glob # 取得圖像路徑。
2. opencv # 讀取圖片。
3. math、 numpy # 計算用途。
4. os # 建立資料夾用途。

**Github Link:**

<https://github.com/AUDOSt0ck1ng/ImageProcessing_hw4.git>

using Sobel operator for edge detection

先用原圖與Sx Sy 分別convolution算出Gx與Gy再算出Gradient Magnitude=square root((Gx)^2+(Gy)^2)，再mapping至[0,255]。

*def* convolution(*image*, *filter*):

    image\_height = *image*.shape[0]

    image\_width = *image*.shape[1]

    H = (*filter*.shape[0] -1)//2

    W = (*filter*.shape[1] -1)//2

    result = np.zeros((image\_height, image\_width))

    # iterate over all the pixel of image X

    for i in np.arange(H, image\_height-H):

        for j in np.arange(W, image\_width-W):

            sum = 0

            # iterate over the filter

            for k in np.arange(-H, H+1):

                for l in np.arange(-W, W+1):

                    # get the corresponding value from image and filter

                    a = *image*[i+k, j+l]

                    w = *filter*[H+k, W+l]

                    sum += (w \* a)

            result[i, j] = sum

    # return convolution

    return result

#sobel operator

*def* sobel\_operator(*image*):

    #sobel\_filter

    Sx = np.array([[-1, 0, 1], [-2, 0, 2], [-1, 0, 1]])

    Sy = np.array([[1, 2, 1], [0, 0, 0], [-1, -2, -1]])

    Gx = convolution(*image*, Sx)

    Gy = convolution(*image*, Sy)

    result = np.sqrt(np.power(Gx, 2) + np.power(Gy, 2))

    result = (result/np.max(result)) \* 255

    return result

using Laplacian of a Gaussian (LoG) operator for edge detection

先用Gaussian filter降噪再使用Laplacian operator 做edge detection，得到dst，

再對dst做影像增強，但我覺得增強後沒有很清晰，就把增強前後都印出來比較看看。

*def* laplacian\_of\_gaussian(*image*):

    #default輸入為gray image

    #Gaussian降噪

    blur = cv2.GaussianBlur(*image*, (3,3), 0)

    #laplacain做edge detection

    dst = cv2.Laplacian(blur, cv2.CV\_16S, *ksize* = 3)

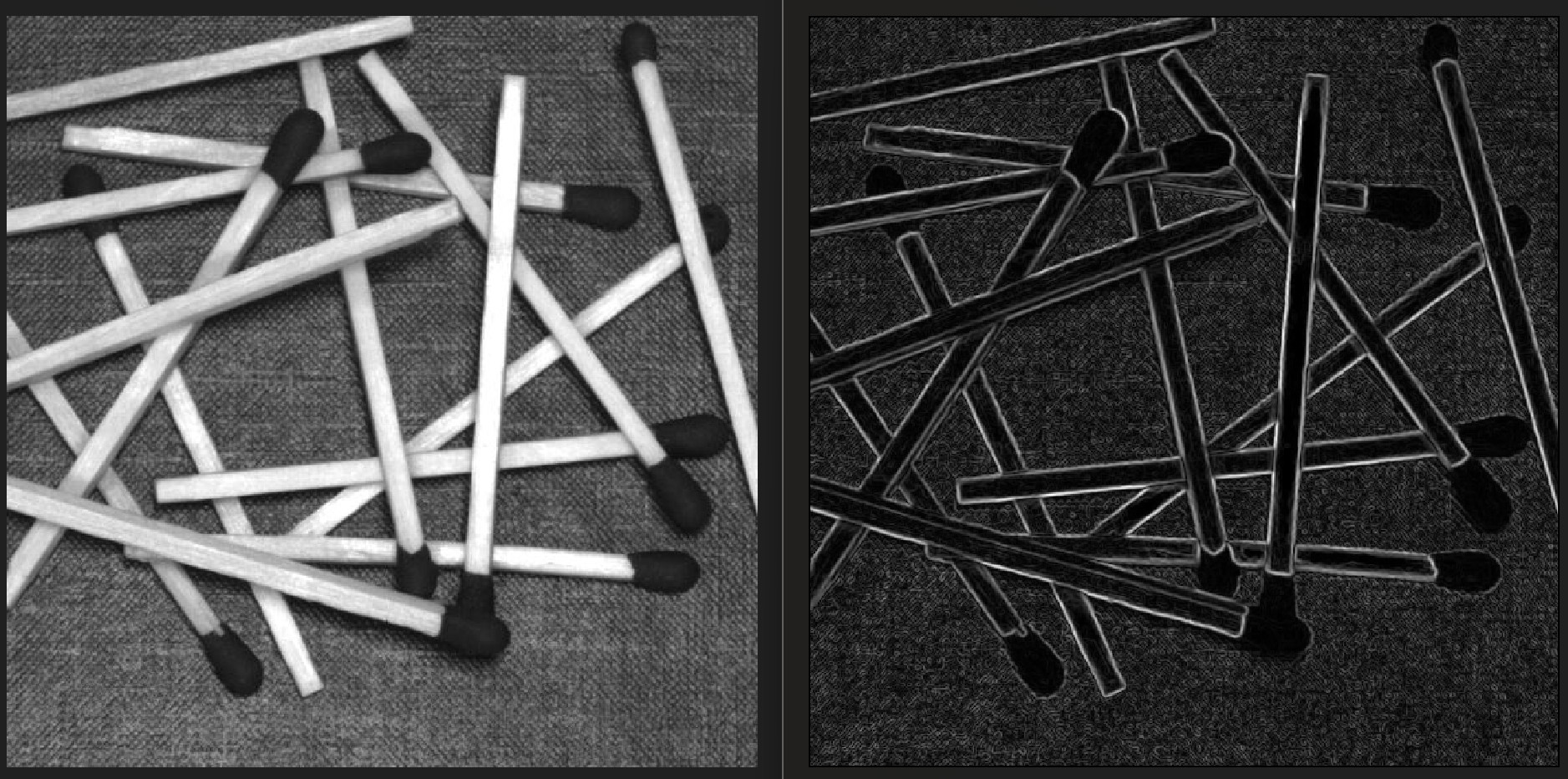
    result = cv2.convertScaleAbs(dst)

    return result,dst

Experience Results



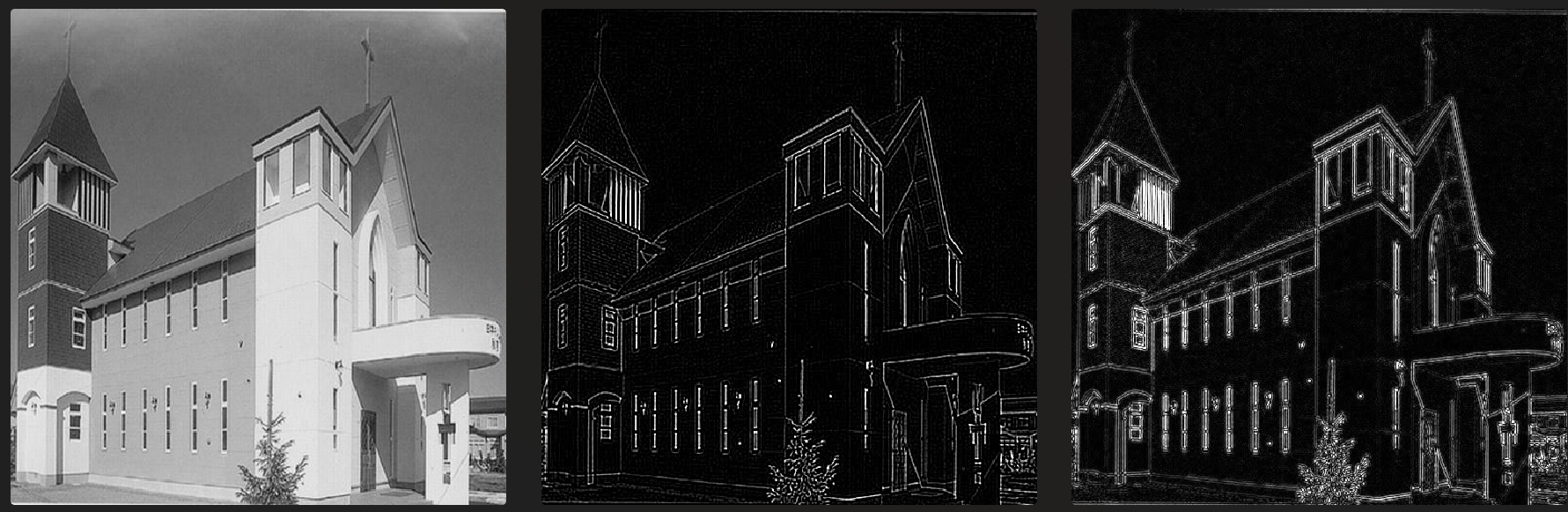
左:原圖 右:Sobel operator



左:原圖 右:Sobel operator



左:原圖 右:Sobel operator



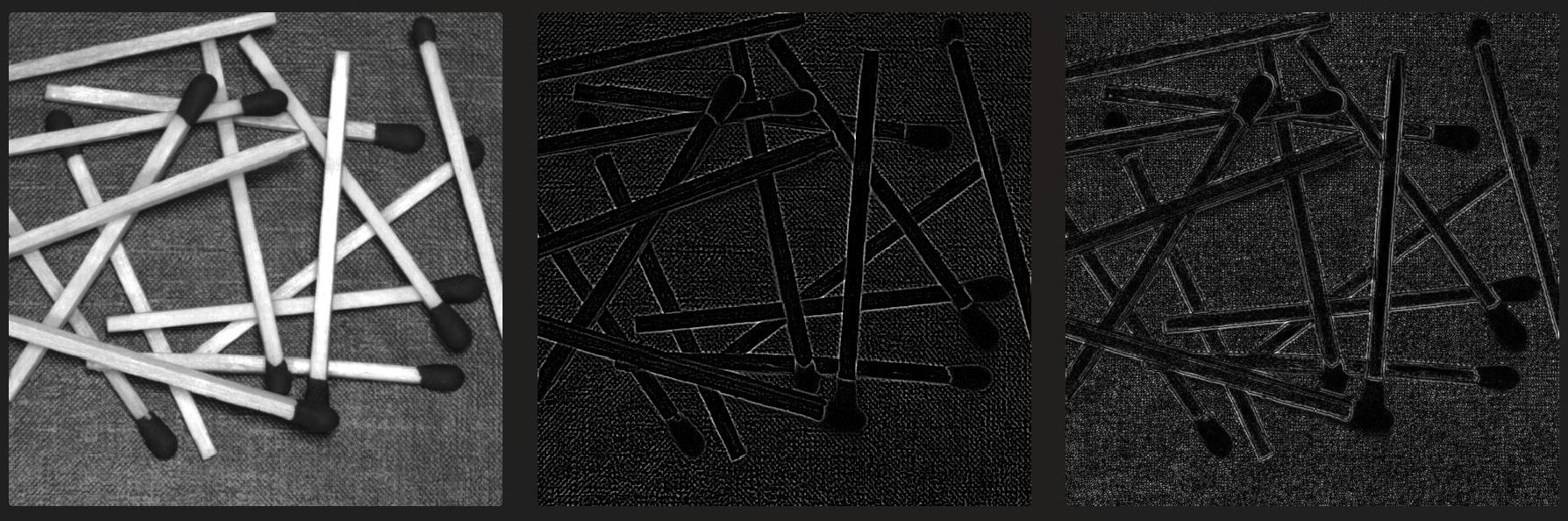
左:原圖

右:Laplacian of a Gaussian operator

with convertScaleAbs()

中:Laplacian of a Gaussian operator

without convertScaleAbs()



左:原圖

右:Laplacian of a Gaussian operator

with convertScaleAbs()

中:Laplacian of a Gaussian operator

without convertScaleAbs()



左:原圖

右:Laplacian of a Gaussian operator

with convertScaleAbs()

中:Laplacian of a Gaussian operator

without convertScaleAbs()

Discussions

我覺得sobel operator還有Laplacian of a Gaussian operator without convertScaleAbs()做出來的圖比較簡潔、看起來也比較清楚，當然有些細節就被犧牲掉。

Laplacian of a Gaussian operator with convertScaleAbs()看起來，想要呈現的細節比較多，但是反而比較模糊。

如果要強調edge detection去取輪廓線的話，我會比較喜歡前面兩種。

References and Appendix

<https://medium.com/@nikatsanka/comparing-edge-detection-methods-638a2919476e>

lab的學長姐