Hw3 Color Image Enhancement

資工四 陳昱瑋 409410118

Date due: 2024/6/14

Date handed in: 2024/6/2

Technical description

- **Gradient Operators**

  假設 image f(x, y)的 gradient 定義為：

$$\nabla \boldsymbol{f} = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}.$$

  上述之 gradient vector 指向 f 在座標 (x, y) 之像素值變化最大的方向。其中很重要的一個式子：

$$\nabla f = mag(\nabla \boldsymbol{f}) = \left[G_x^2 + G_y^2\right]^{1/2}$$

  代表在 $\nabla \boldsymbol{f}$ 方向之下像素值每單位的最大變化值。一般也稱 ∇f 為 gradient。

  定義 $\alpha(x, y)$ 為 gradient 之方向：

$$\alpha(x, y) = tan^{-1}\left(\frac{G_y}{G_x}\right)$$

  此角度以 x 軸為基礎；同時此方向跟 edge 為 perpendicular (垂直)。

  假設 $\begin{bmatrix} z_1 & z_2 & z_3 \\ z_4 & z_5 & z_6 \\ z_7 & z_8 & z_9 \end{bmatrix}$ 為 gray-level style 的 image。對此 z_5 做微分最簡單的方法就是 Roberts cross-gradient operators：

$$G_x = (z_9 - z_5),$$
$$G_y = (z_8 - z_6).$$

  另一個方法是對 Prewitt operators：

$$G_x = (z_7 + z_8 + z_9) - (z_1 + z_2 + z_3),$$
$$G_y = (z_3 + z_6 + z_9) - (z_1 + z_4 + z_7).$$

  而針對上述兩者之應用與變形即為 **Sobel operators**：

$$G_x = (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3),$$
$$G_y = (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7),$$

$$filter_{sobel_x} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix},$$

$$filter_{sobel_y} = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}.$$

Execution process

- **Functions**
  1) sobel(img)
     ⇨ 進行 Sobel operation
     ⇨ img: 輸入處理的圖片
  2) plot(list:[], len: len(list[]), list:[titles])
     ⇨ 呈現 list 中的 images 與其對應的 titles

- **Sobel operation process**
  Loading images => Do Sobel operation for R, G, B spaces => Show edge detection images.

- **Execution**
  - Sobel operation
    `python main.py`

Experimental result

- **Baboon**

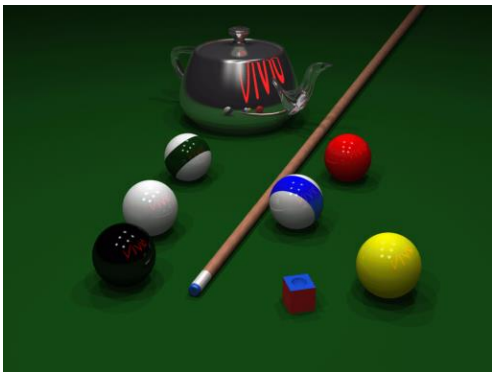
- **Peppers**

- **Pool**

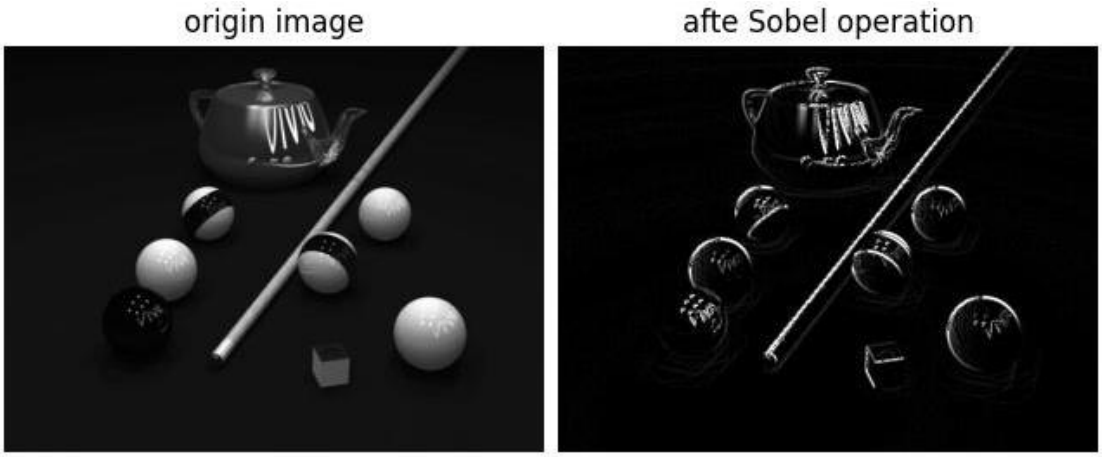
Discussions

- **Observe the strength of edge detection with Sobel operation in R, G, B space, respectively**

  以 pool.png 為例，觀察做完 Sobel operation 後的 RGB 結果。在這裡可以想成對三原色進行分離後轉成灰階，針對每個原色分別進行 Sobel Operation。

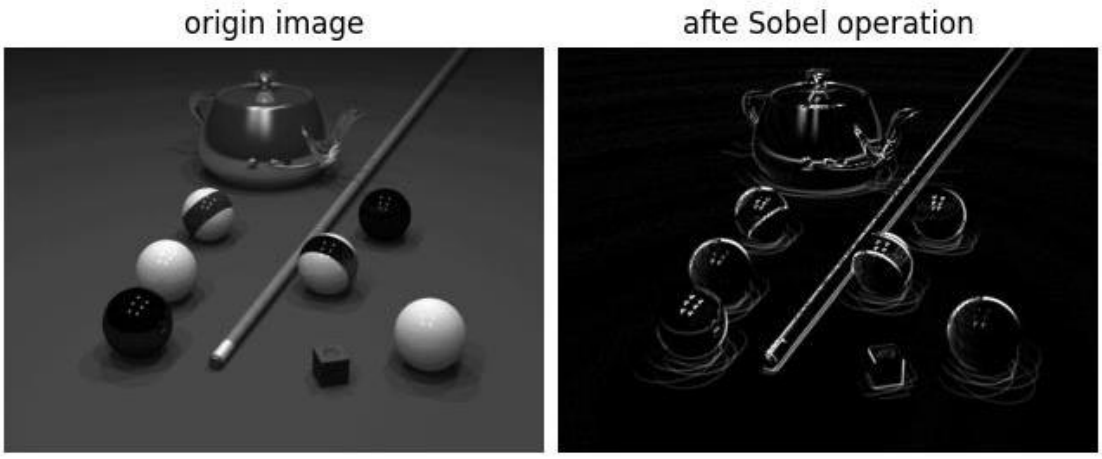
  

  - **Red**

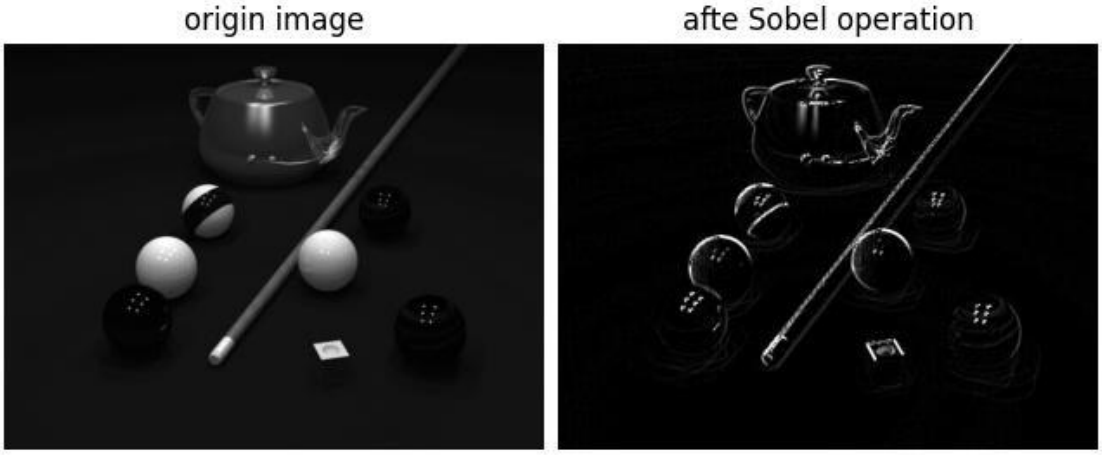
    可以發現在鐵壺上紅色的線條可以明顯的被偵測出來。另外在原圖白色、黑色與其他顏色的地方也可以看到明顯的分割，這是因為 white = (R, G, B) = (255, 255, 255) 所組成，black = (R, G, B) = (0, 0, 0) 所組成，舉例黑條紋白底的撞球就可以明顯地分割出黑色條紋；藍色條紋白底的撞球以此類推。

  - **Green**



可以發現底色相較於另外 Red 與 Blue 的圖更亮。由於底色為綠色，因此跟物品之間值的差距就會拉開，因此可發現在物件 (撞球、鐵壺、撞球桿) 的邊緣都可以有效的檢測出來，尤其可以發現撞球感的邊緣檢測的最明顯 (因為撞球是咖啡色，與綠色相差更大)。

- **Blue**



可以發現藍色條紋白底的撞球的藍條紋完全消失，在灰階的呈現下與白底合成一體，與 Red space 的鐵圖上的紅線不同，由於在 Red space 中鐵壺為鐵灰色，紅線依舊與底色有所值的差距；藍條紋則無法與白底區分，因為都位在峰值 255 附近。Blue space 在這張圖可說是檢測效能較差的原色，沒有 Green space 可以將物件邊緣明顯呈現，也沒有 Red space 可以將一些細節邊緣呈現出來。

- Summary

  我們可以針對圖中想檢測的物件顏色去選擇想要進行 Sobel Operation 的 space，如 Peppers 這張圖就可以選擇綠色與紅色進行邊緣檢測，更能完整框出物件的邊緣。如可以只針對物件整體顏色分布去做分析，也就可以選擇相對應的 space 去做 edge detection，進而提升效能、速度與結果呈現。

References and Appendix

https://medium.com/@allen73/%E5%BF%83%E5%BE%97-edge-detection-%E8%88%87-sobel-operator-%E7%B0%A1%E4%BB%8B-bd69204e1352