影像處理作業2

5113056047 曾炳憲

## 程式邏輯

1. 讀取影像。
2. 定義 convolution 方法，以 filter size 都是 3\*3 的前提，把大於 255 或小於 0 的結果做處理後才輸出。

def conv(src: np.ndarray, kernel: np.ndarray) -> np.ndarray:

k = kernel.shape[0]

l = kernel.shape[1]

new\_image = np.zeros\_like(src)

src = cv2.copyMakeBorder(

src, 1, 1, 1, 1, cv2.BORDER\_REPLICATE,

)

for i in range(src.shape[0] - 2):

for j in range(0, src.shape[1] - 2):

product = np.sum(

np.multiply(

src[i : i + k, j : j + l],

kernel,

)

)

if product < 0:

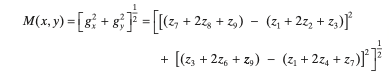
product = 0

elif product > 255:

product = 255

new\_image[i, j] = product

return new\_image

1. 用 l\_kernal = np.array([[1, 1, 1], [1, -8, 1], [1, 1, 1]]) 作為 laplacian 的 kernel 進行 convolution 取得 laplacian 結果。
2. 用 g\_x\_kernal = np.array([[-1, -2, -1], [0, 0, 0], [1, 2, 1]]) 以及 g\_y\_kernal = np.array([[-1, 0, 1], [-2, 0, 2], [-1, 0, 1]]) 作為 kernel 分別取垂直和水平的 gradiant，並用

公式計算最終的 image gradiant 結果。

1. 將 gradient 結果通過 avarage filter 去除雜訊，並 normalize 成 [0,1] 作為 mask 與 laplacian 結果相乘，得到只針對細節做增強的 laplacia。
2. 用原始影像 - laplacian 得到最後輸出。

* 有設定 scale 參數讓銳化結果不要太顯著導致影像不自然

result = cv2.addWeighted(image, 1, enhanced\_detail, -scale, 0)

## 範例

* 輸入



* 輸出

