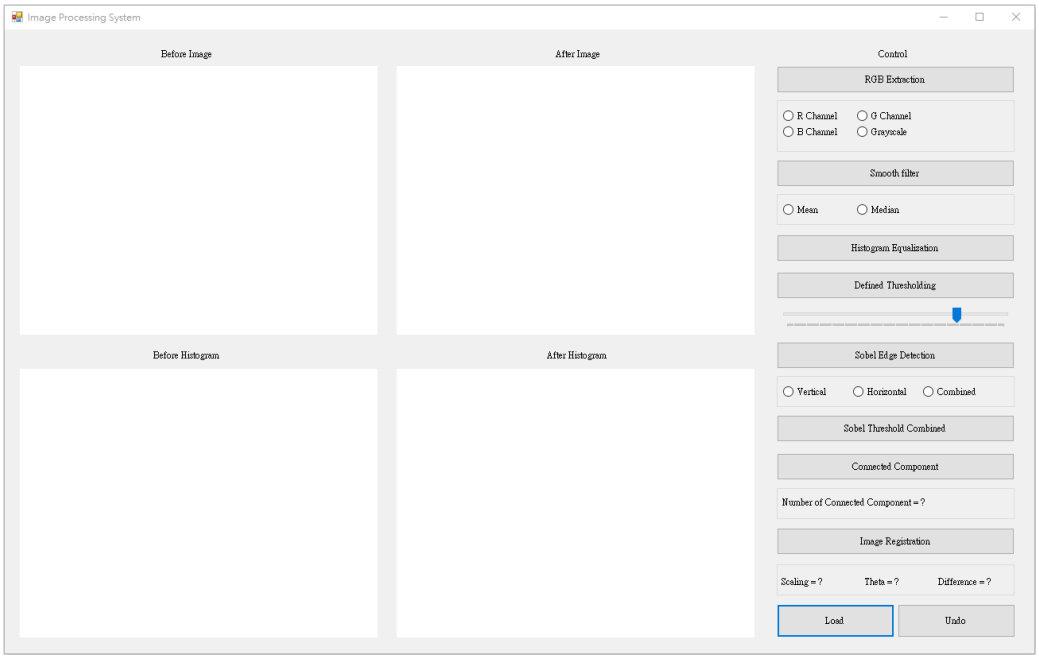
影像處理 Homework1

P76104370 鄭琮寶

2021/10/29

壹、Graphical User Interface



1. 介面說明

按鈕在 GUI 的右側，Control 的部分，其中包含各題可以選擇的 radio button 以及可以調整的 threshold 拉桿，以及各題執行時需使用的按鈕和執行後會顯示的數值(例如:scaling 等)。在右側最下方 load 可以載入所需的影像(第八題須一次選擇兩張)，Undo 可以回復上一張載入的影像或是影像執行的結果 (在關閉程式後才會清除所有 Undo 所需的影像)。

在 GUI 介面的左側危險是區域，會顯示 Load 之影像、Undo 之影像和按下右側按鈕後執行後的影像，其中按下滑鼠右**"左鍵"** Before Image 和 After Image 下方區域會標註紅點，按下滑鼠右**"右鍵"** Before Image 和 After Image 會進行交換 (第八題或是需要將執行後的結果接續執行可以使用)。

2. 操作說明

Step1: Push the "Load" button to load image

Step2: Choose the one or two image

Step3: Choose the radio button (如果不需要就跳過)

Step4: Push the button on the right side

Step5: Mouse clicked on image (如果不需要就跳過，例如: **"左鍵"或"右鍵"**)

Step5: See the result on the left side and right side

貳、Image Processing Method

一、RGB Extraction & Transformation

(一) Problems

將影像轉換成，R channel、G channel、B channel 和 Grayscale，並顯示原影像和轉換後的影像。

(二) Methods

1. R channel

將 RGB 三個 Channel 設定成與 R channel 相同的數值。

2. G channel

將 RGB 三個 Channel 設定成與 G channel 相同的數值。

3. B channel

將 RGB 三個 Channel 設定成與 B channel 相同的數值。

4. Grayscale

將 RGB 三個 Channel 設立用以下公式(1)設定數值

grayValue = R channel * 0.299 + G channel * 0.587 + B channel * 0.114 (1)

(三) Results

origin

grayscale

R channel

G channel

B channel

(四) Discussion

可以看到透過上述的方法可以將一張彩色的 RGB 影像切換成 R channel、G channel、 B channel，來凸顯 RGB 三個色調，以 G channel 為例可以看到原圖隻綠色位置部分，在 R channel image 的影像上非常的白，同時也可以透過灰階的轉換公式，將影像轉成灰階的影像，其中 RGB 的比例可以在豁接圖中凸顯綠色的部分。

(五) Conclusion

各種 channel 的轉換可以凸顯紅色、綠色和藍色在原本同型上的重要性，而灰階可以利用單一 channel 來代表三個 channel 節省空間，並利用 RGB 不同的 weight 讓影像更接近我們所觀察的原影像。

二、Smooth Filter (Mean and Median)

(一) Problems

將影像透過 mean filter 和 median filter 轉換成新的影像。

(二) Methods

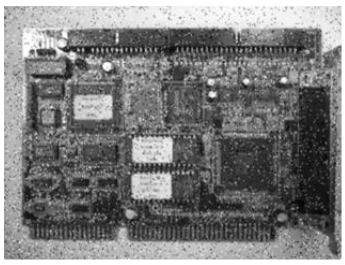
1. Mean Filter

每個灰階點為中心展開 3x3 的 kernel 取正方形述職的平均值代表該點的灰階值。

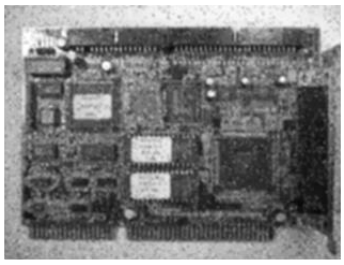
2. Median Filter

每個灰階點為中心展開 3x3 的 kernel 取正方形中的中位數代表該點的灰階值。

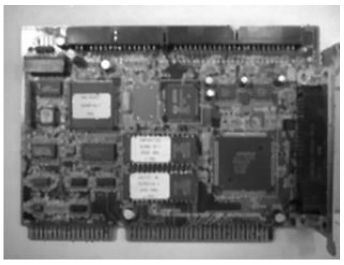
(三) Results



origin image



Mean Filter



Median Filter

(四) Discussion

透過 mean filter 和 median filter 的方式可以有效地較低影像的躁點，但因為此圖形的躁點較為嚴重(可以想像成經常會是 kernel 中的極端值)，因此透過 mean filter 只能讓躁點變的不明顯，而 median filter 可以去極端值來有效的去除影像中所有的躁點。

(五) Conclusion

透過 mean filter 和 median filter 的方式都能有效地去除躁點，不過需要看使用場景，若躁點較明顯(如本題)，利用 median filter 會比較好，反之則利用 mean filter 較佳。

三、Histogram Equalization

(一) Problems

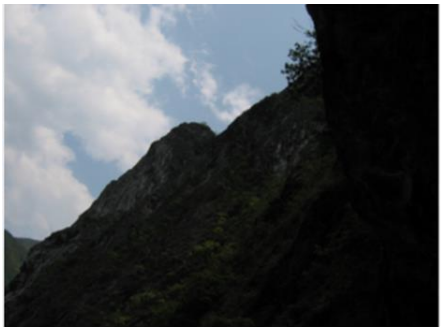
將影像透過 Histogram Equalization 轉換成新的影像。

(二) Methods

$$h(v) = round\left(\frac{cdf(v)-cdf_{min}}{cdf_{max}-cdf_{min}} \times (L-1)\right) \qquad (2)$$

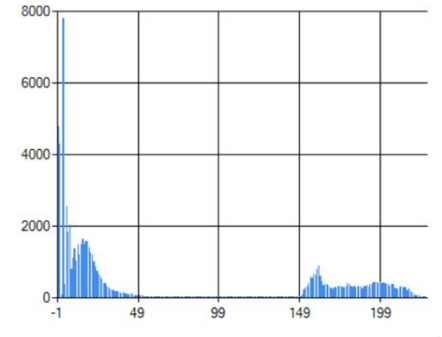
透過上述公式(2)，找到 origin image 每個 pixel 的灰階值所對應之灰階值，一個一個對應過去，並畫出 origin image 和 transfer image 的灰階直方圖。

(三) Results

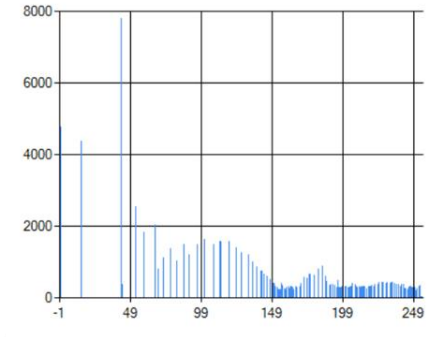


Origin Image

Transfer Image



Origin Histogram



Transfer Histogram

(四) Discussion

從 origin histogram 和 transfer histogram 的灰階直方圖中，發現灰階值的確更平均的被分佈到 0 到 255 之間了，讓影像整體的灰階分布到所有的灰階區域中，有助於調整影像整體之對比度，在 origin image 和 transfer image 之間也可以看到在 origin image 原本較暗之區塊，之對比度提升在 transfer image 可以更清晰地看到細節。

(五) Conclusion

透過 Histogram Equalization 的轉換方式，可以有效的提升影像整體之對比度，讓原本難以分清楚之區域變得更加清晰。

四、A user-defined thresholding

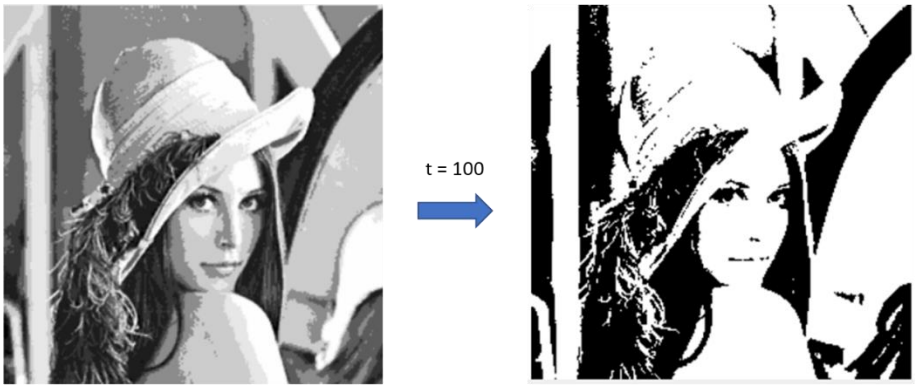
(一) Problems

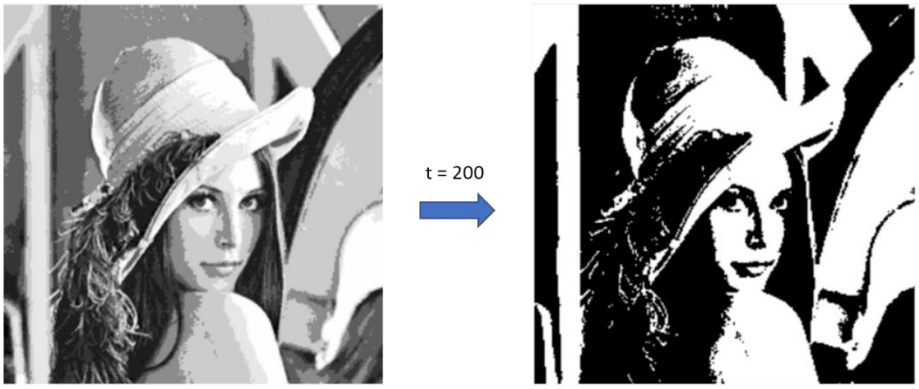
利用 thresholding 的方式進行影像處理，並要在 user interface 中可以調整 threshold 值 t。

(二) Methods

給定一個 threshold 值 t (給一個拉桿，讀取拉桿的數值)，每個 pixel 的數值在 t 值以上位置將其灰階值設定為 255，其他數值設定為 0。

(三) Results





(四) Discussion

可以看到在二值化後的影像中，雖然提高了各 threshold 值上下 pixel 的對比，但因為只有白和黑兩種資訊，因此大大降低了圖形的資訊量，所以在使用此方時要注意後續是否有要做其他的影像處理。

(五) Conclusion

使用 thresholding 的影像處理方式要慎選閥值 t 的大小，因為會加大 t 值上下 pixel 的差異，且在處理後會喪失大量的影像資訊，需要注意後續的影像處理，像是進行 thresholding 再進行 sobel 就沒有太大的意義因為無法精準地提供出影像之梯度大小。

五、Sobel edge detection

(一) Problems

利用 Sobel 運算子計算 Horizontal、Vertical 和 Combined 的梯度影像。

(二) Methods

$$S_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} \quad S_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{3}$$

1. Horizontal

利用S_x運算子與原影像進行平面卷積，找出 x 方向的梯度大小G_x。

2. Vertical

利用S_y運算子與原影像進行平面卷積，找出 y 方向的梯度大小G_y。

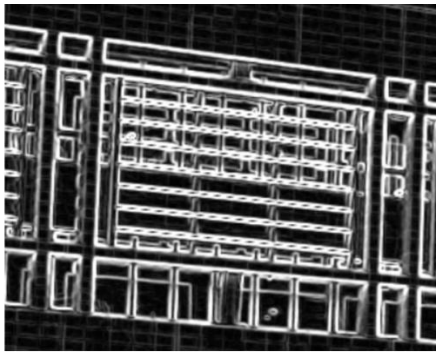
3. Combined

$$\text{梯度大小：G} = \sqrt{G_x + G_y} \tag{4}$$

利用先前算好的G_x和G_y帶入上述公式(4)計算出該項速點之梯度值 G。

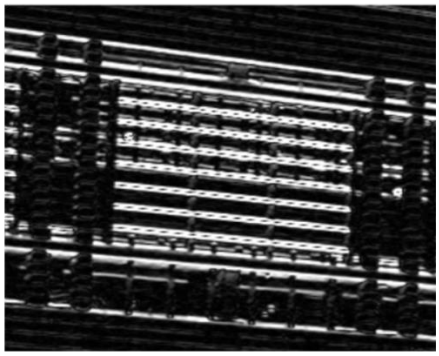
(三) Results

Origin Image



Sobel Image

Vertical Image



Horizontal Image

(四) Discussion

$$\text{梯度方向：}\ \Theta = \tan^{-1}\left(\frac{G_y}{G_x}\right) \qquad (5)$$

可以看到利用設計好的利用S_x和S_y，透過平面卷積的方式，計算出梯度G_x和G_y，我們就可以知道每一個 pixel 的梯度方向，除了 combined 成梯度大小外，也可以透過上述公式(5)計算出其梯度方向進一步了解影像的走勢，而非單單的知道該點為影像的邊緣的機率大小。

(五) Conclusion

透過 Sobel 運算子可以找出影像之邊緣，以及該邊緣之法向量的大小與方向，且在數據離散而非連續的影像數據中，可以透過這種方式來取代連續函數中的微分行為，是一個值得學習的知識點。

六、Threshold and overlap on the original image

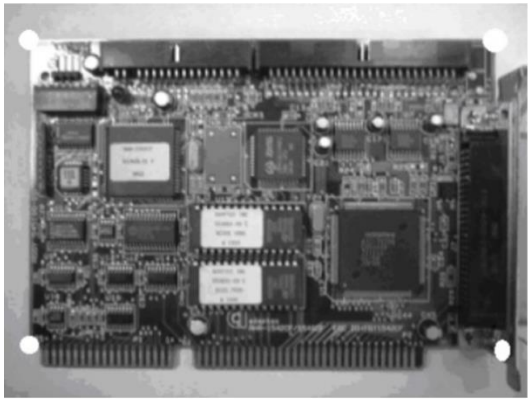
(一) Problems

將前一題 Sobel 後的影像，透過 thresholding 的影像處理，將其較亮的部分設定成綠色，並與原影像合併。

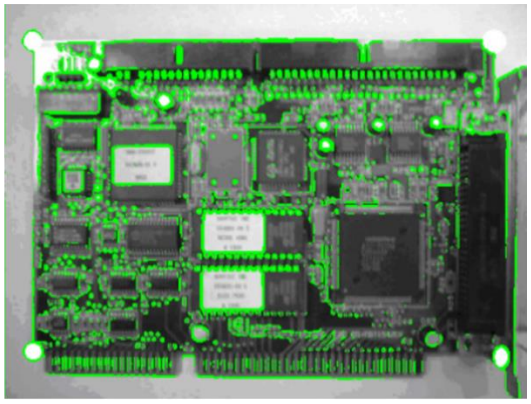
(二) Methods

將 Sobel 後的影像，進行 thresholding，白色的 pixel 改成綠色，取代原圖中的該 pixel。

(三) Results



Origin Image



Sobel Threshold Combined

(四) Discussion

可以看到將 Sobel 後的影像進行 thresholding，可以將其二值化，進一步凸顯影像的邊緣區域，並疊回原影像可以更清楚的觀察所抓取到的邊於區域。

(五) Conclusion

透過 Sobel 與 thresholding 的方式層層過濾可以找到影像梯度較大之區域，也就是影像的邊緣，並與原影像結合，方便使用者觀察並調整相關參數。

七、Connected Component

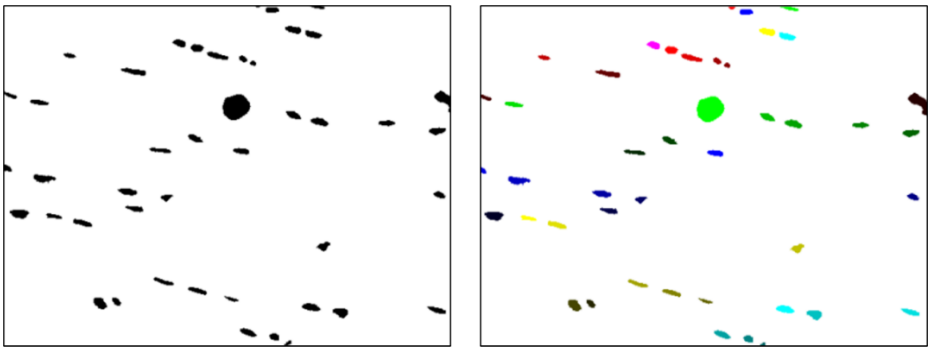
(一) Problems

利用 8-adjacency 的方式，找出影像中的連通區域，並將其塗上相同之顏色。

(二) Methods

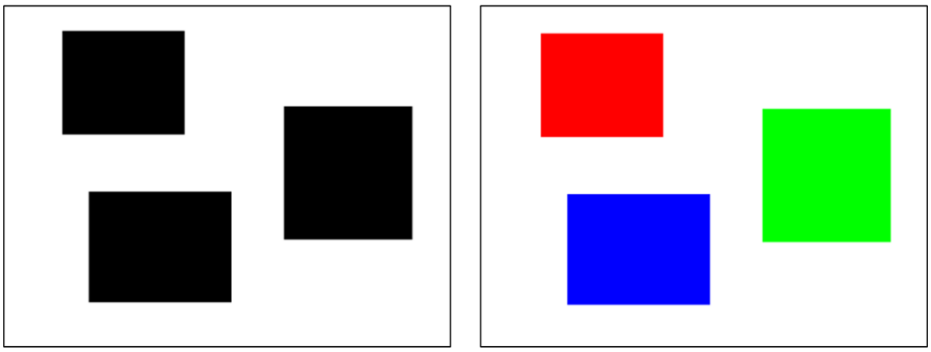
利用 Breadth-First Search (BFS 廣度優先搜尋)或是 Depth-First Search (DFS 廣度優先搜尋)的方法找到所有與該點連通知 pixel 並塗上顏色 (程式中兩種方法都有實現，但選擇 BFS)。

(三) Results



Origin Image

Connected Component Image
(Number of Connected Component = 44)



Origin Image

Connected Component Image
(Number of Connected Component = 3)

(四) Discussion

除了 Breadth-First Search 的方法外，我們也可以透過 Depth-First Search 或 Disjoint-Set 的方式找尋連通圖，其中因為 Depth-First Search 和 Breadth-First Search 的方法，其時間複雜度為 O(N)，但 Disjoint-Set 之時間複雜度為 O(NlogN)，所以利用 Depth-First Search 和 Breadth-First Search 的方式較好，不過由於系統對於 Recursion 的次數有所限制 (下方 Example 為 788x672 的範例，若用 DFS 會直接超出 Recursion 的限制，程式會直接 crash)，因此利用 Breadth-First Search 會是最好的選擇。

(五) Conclusion

透過 BFS 的方式可以有效地找尋到所有的連通區域並計算出有幾個連通區域，並將這幾個區域塗上顏色，凸顯出各區域間的差異。

八、Image registration

(一) Problems

選取兩張影像的白色點，並計算出兩張影響之間的 scale 和 rotation degree 的數值，透過這些數值將原影響 origin image 旋轉後得到 transfer image，並計算出 origin image 和 transfer image 的 Intensity difference。

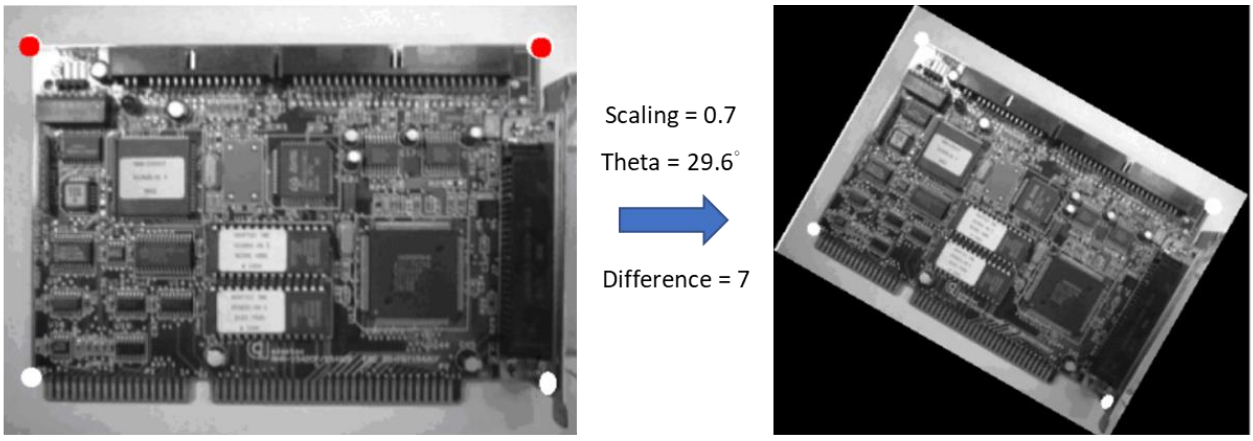
(二) Methods

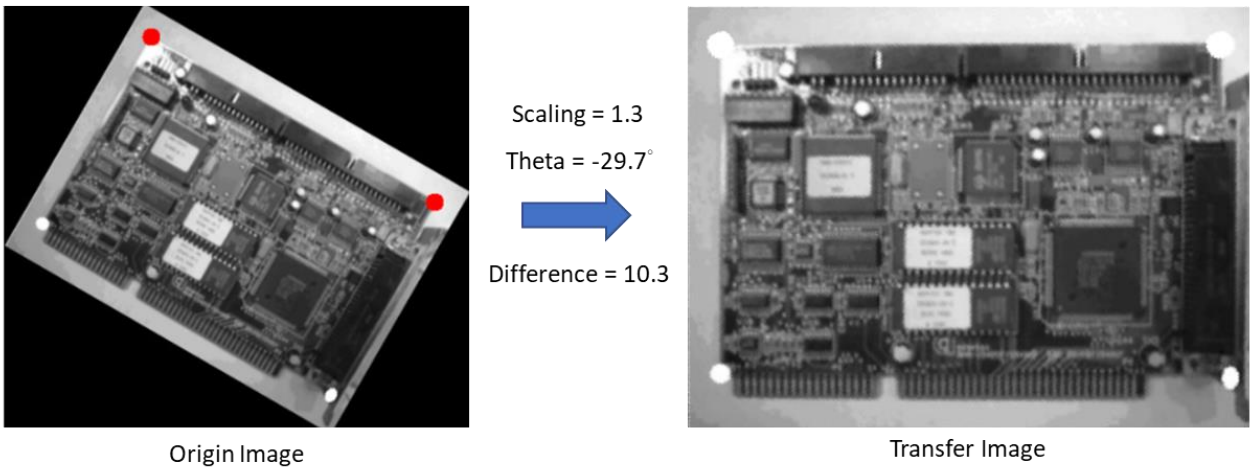
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = s \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x - w \\ y - h \end{bmatrix} + \begin{bmatrix} x + w' \\ y + h' \end{bmatrix} \quad (6)$$

$$D_{pixel} = \frac{1}{|I|} \sum_{p \in I} |\hat{i_p} - i_p| \quad (7)$$

透過點取 origin image 和 transfer image 的各兩個點，計算出兩影像的點分別產生的向量，再利用這兩個向量計算出 scale 和 rotation degree 數值，再將 origin image 座標點透過上述公式(6)轉成 transfer image (其中 w 和 h 分別為影像的 width 和 height)，並利用公式(7)計算出 origin image 和 transfer image 的 Intensity difference。

(三) Results





(四) Discussion

透過 scale、rotation 和 translation 的方式，計算出旋轉矩陣，來對應 origin image 和 transfer image 之間的關係，成功的將原影像旋轉到正確的位置，並計算 Intensity difference 確保旋轉的精準度，是個很好的方法。

(五) Conclusion

透過 scale、rotation 和 translation 的方式，可以將原影像以影像中央為中心進行旋轉，並且伸縮，並計算出其 Intensity difference 來觀察影像旋轉後的準確度是否如原本預期。

參、Reference

[1] Histogram equalization。檢自 https://en.wikipedia.org/wiki/Histogram_equalization