Digital Image Processing

HOMEWORK ASSIGNMENT #4

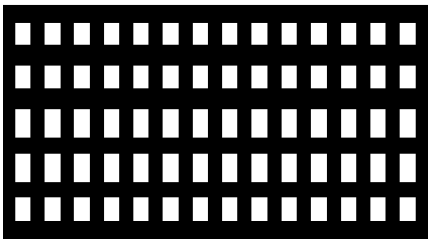
R06922097 資工碩一 鄭雅文

PROBLEM 1 Optical Character Recognition (OCR)

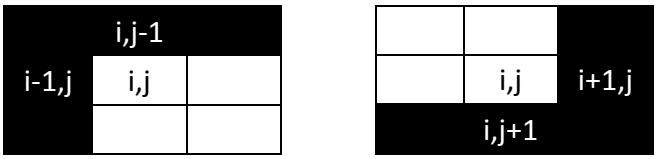
Algorithm:

TrainingSet Preprocessing:

1. 將原圖轉為 binary image，若 pixel 值小於 170，則設為 0，否則設為 1。
2. 用兩個一維陣列儲存垂直及水平投影，若有 pixel 值為 1，則陣列設為 1。可得以下 segmentation。

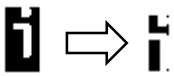


3. 用一個二維陣列儲存 Position 資訊(上下左右邊界)。由 segmentation 可得：



若在垂直及水平方向同時有變化，則中間的 pixel 的位置(i,j)為切割的物件的最左上角或最右下角。

4. 對每個 Training Character，若有一整行或一整列為空白，則從四邊開始縮減，若只從上面開始縮減，則 i 的兩部分會被拆開，所以必須從四邊開始縮減，如下圖，左邊為縮減前，右邊為縮減後(字母為 i)。

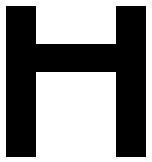


Sample1 Preprocessing:

1. 因為背景有雜訊，所以需要進行去背。因為背景顏色偏淺色，所以設定若 pixel 值大於 127，則設為 1，反之為 0。

2. 因為最後一個字 8 筆畫太細，所以多做了一步 Dilation，將每個字筆畫變粗。

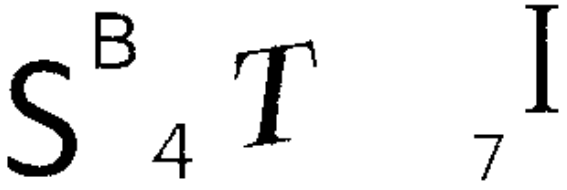
3. 做字母切割，因為只有一列，所以只需要投影到一維的陣列判斷該行有沒有 pixel 為 1。
4. 從此一維向量判斷圖中有幾個字，若有 array(i-1)=0 且 array(i)=1 的情形，則字數加一。用一個二維陣列儲存 Position 資訊，array(i-1)=0 且 array(i)=1 的地方為字母左界，array(i-1)=1 且 array(i)=0 的地方為字母右界。
5. 縮減時從第一列及最後一列(上下)開始縮減，最後得到如下圖：



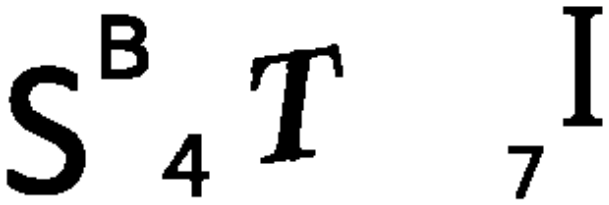
Sample2 Preprocessing:

1. 原本想用去除 Salt and pepper noise 的方式去除 noise，但效果不好。發現因為字的 pixel 值都是 65，所以若 pixel 值為 65，該位置值設為 1，反之為 0。

2. 因為字母中還是有一些白色的點，因此用周圍的 neighbors 決定白點要不要塗黑，若周圍 8 個點有超過 5 個點是黑色，則該點設為黑，結果如下：



3. 因為 B、4、7 筆畫都太細，所以也做 Dilation。



4. 做字母切割，因為只有一列，所以只需要投影到一維的陣列判斷該行有沒有 pixel 為 1。
5. 從此一維向量判斷圖中有幾個字，若有 array(i-1)=0 且 array(i)=1 的情形，則字數加一。用一個二維陣列儲存 Position 資訊，array(i-1)=0 且 array(i)=1 的地方為字母左界，array(i-1)=1 且 array(i)=0 的地方為字母右界。
6. 縮減時從第一列及最後一列(上下)開始縮減，最後得到如下圖：

Recognition:

用三個 score 算出目標字母和 training data 中的字母的差異。

1. 因為已經把字母的框格縮減到最小，所以框格的長寬比就是字母的長寬比。第一個 score 即是算長寬比的相差的絕對值。
2. 第二個 score 是算 pixel 值為 1 的面積占框格的面積的比例，該面積比例的相差取絕對值。
3. 第三個 score 是算 training data 中的字母的每個 pixel 值和目標的字母的每個 pixel 值的差，若 pixel 值不相等，diff 加一，最後再除以面積。因為目標字母和 training data 中的字母大小不同，無法一一對應，因此若放大 N 倍，則取

$$\mathrm{j} + \mathrm{floor}((\mathrm{x} - \mathrm{i}) \times \mathrm{N})$$

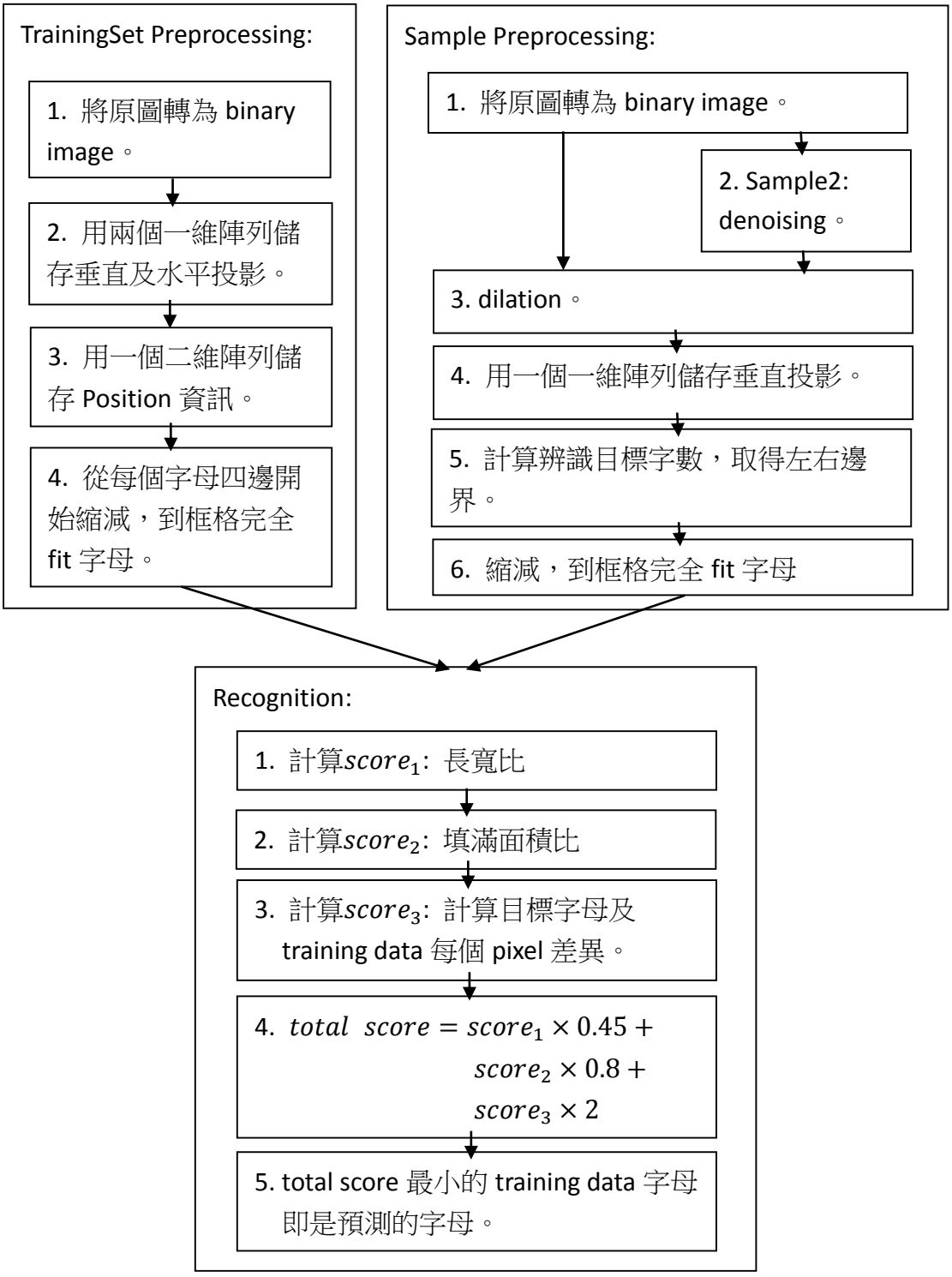
其中 j 為目標字母在 sample1 圖中的起始位置，$(\mathrm{x} - \mathrm{i})$為欲取的 pixel 在 TrainingSet 的 offset。

最後用

$$total\ \ score = score_1 \times 0.45 + score_2 \times 0.8 + score_3 \times 2$$

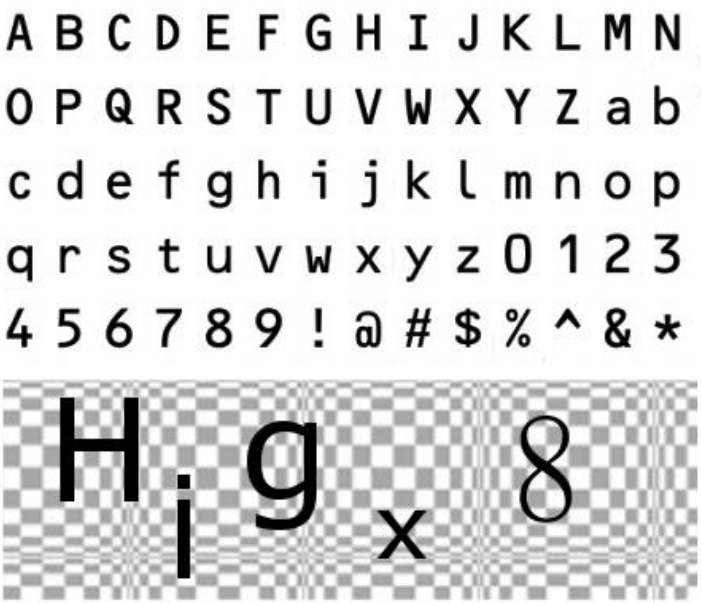
算出最小的 total score 的字母(差異最小的字母)，即為辨識的字母。

Flowchart:

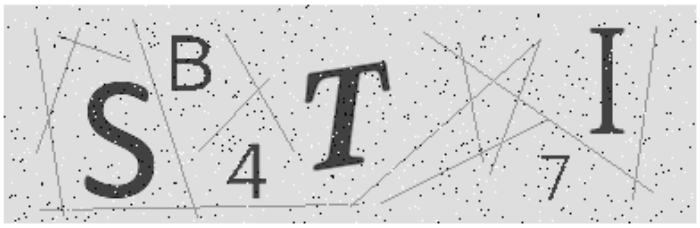

TrainingSet Preprocessing:
1. 將原圖轉為 binary image。
2. 用兩個一維陣列儲存垂直及水平投影。
3. 用一個二維陣列儲存 Position 資訊。
4. 從每個字母四邊開始縮減，到框格完全 fit 字母。
Sample Preprocessing:
1. 將原圖轉為 binary image。
2. Sample2: denoising。
3. dilation。
4. 用一個一維陣列儲存垂直投影。
5. 計算辨識目標字數，取得左右邊界。
6. 縮減，到框格完全 fit 字母
Recognition:
1. 計算$score_1$: 長寬比
2. 計算$score_2$: 填滿面積比
3. 計算$score_3$: 計算目標字母及 training data 每個 pixel 差異。
4. $total\ \ score = score_1 \times 0.45 + score_2 \times 0.8 + score_3 \times 2$
5. total score 最小的 training data 字母即是預測的字母。

Discussion:

TrainingSet：



Sample1 的預測結果是：H!gx8



Sample2 的預測結果是：SBAT7I

用以上 3 個 scores 即可在 Sample1 及 Sample2 得到不錯的辨識結果，兩者都只辨識錯一個字。在 Sample1 中將 i 辨識為!，所以必須將標點符號去除後才能將 i 辨識為 i。我認為給的 Training dataset 的 i 和圖片中的 i 確實是不太像，Training dataset 的 i 多了往左的一橫，導致長寬比會跟!差很多，所以如果加入標點符號很難將兩者分出來。

在 Sample2 中 4 被辨識為 A，我認為也是 Training dataset 中所給的 4 在影像上跟 Sample2 中的 4 也有差，因為少了那段將垂直線將 4 的頂點連起來，所以也無法用有沒有 holes，判斷 Training data 是不是目標字母。

如果針對上面問題做參數的 tuning，則可以做到單張 100%的預測結果，但是在另一張圖片辨識就會更差，上述演算法是同一參數對兩張都最好的結果。

以下是針對單張可以做到 100%辨識率的調整：

1. Sample1: 只辨識英文和數字，預測結果：Higx8
2. Sample2: 調整$score_2$參數，預測結果：SB4T7I

$$total\ \ score = score_1 \times 0.45 + score_2 \times 1.15 + score_3 \times 2$$