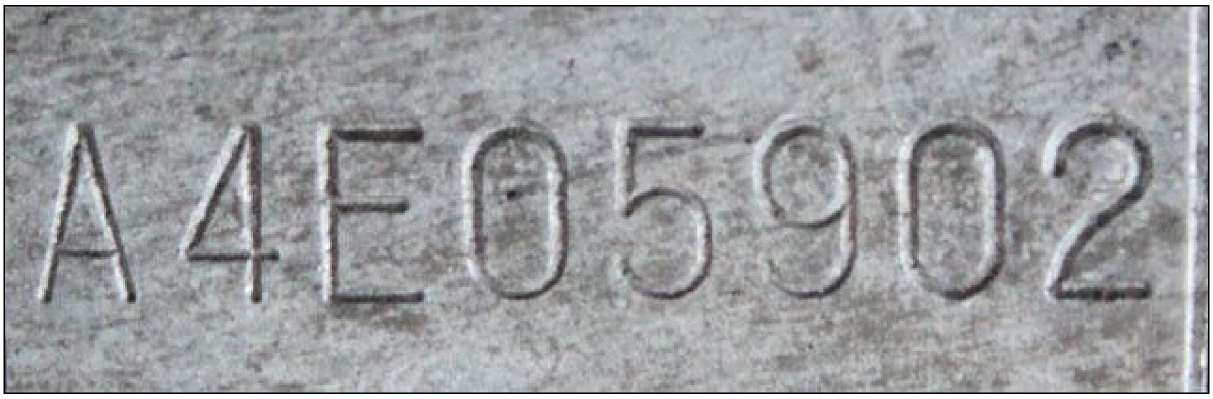
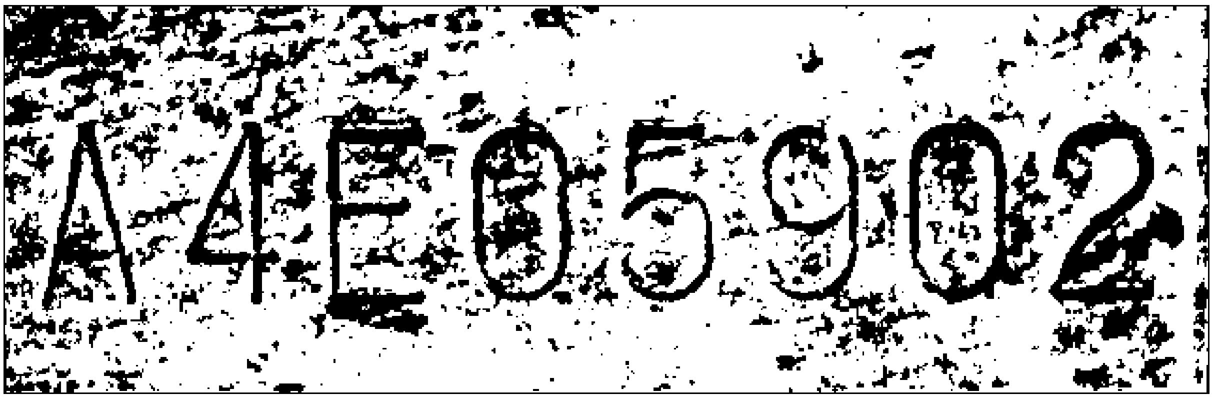
第 12 章　挑戰蝕刻與浮雕字元的辨識



12-1 如果目標與背景根本是同色怎麼辨識？

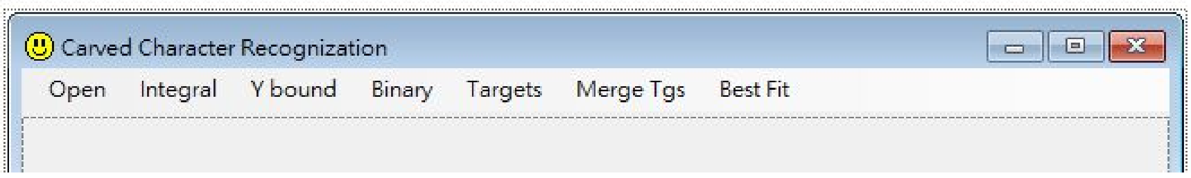
大多數的 OCR 辨識，目標字元似乎理所當然的不會跟背景完全同色，所以我們簡化影像凸顯目標的方式都是從目標與背景的**顏色**或**亮度**差異著手。但是如果目標如上圖是蝕刻或浮雕的字，也沒有著色時，我們就會頓失倚靠了！使用正常的二值化處理，不論如何調整門檻值，只能得到如下高雜訊且字元破碎到難以整合的結果。



在實務上，這種目標多半出現在汽車引擎或鋼材粗胚之類常會處於高熱狀態的金屬物體上，它們一樣有被辨識的需求，但無法以任何印刷或貼紙的方式標示，幾百度的高溫會將油漆或貼紙瞬間燒掉，所以只能用蝕刻或浮雕的方式打印出識別資訊。這也表示這類物體表面一定不會像不鏽鋼廚具一樣光滑平整，連背景也一定是很斑駁的高雜訊狀態，當然更增加了辨識的難度。

雖然這種字元影像一般人眼還是可以輕易識別，但是我開業之後已經有三次婉拒這種辨識專案了！原因當然是沒有把握可以作到如車牌辨識一般極高的辨識率，這讓我內心非常糾結。但這不表示我們會放棄這種目標的辨識研發！本章內容就是以此影像為例，介紹如何辨識這類目標，重點不只是如何凸顯字元目標？還包括一些可以抵抗高雜訊干擾辨識結果的特殊處理手段。

請先複製之前專案做修改的樣本，將主功能表改成這樣：Open, Integral, Y bound, Binary, Targets, Merge Tgs, Best Fit。



公用全域變數區的程式碼如下：

```
Dim B(,) As Byte '灰階陣列
Dim Z(,) As Byte '全圖二值化陣列
Dim C As ArrayList '目標物件集合
Dim brt As Integer = 128 '全圖平均亮度
Dim Ytop As Integer, Ybot As Integer '字元列上下切線之 Y 值
'目標物件結構
Public Structure TgInfo
    Dim np As Integer '目標點數
    Dim P As ArrayList '目標點的集合
    Dim xmn As Short, xmx As Short, ymn As Short, ymx As Short '四面座標極值
    Dim width As Integer, height As Integer '寬與高
End Structure
```

12-2 這是一個微分影像

首先我們來思考一下：「如果字元顏色其實與背景完全一樣，那人眼是如何可以輕易識別出字元的呢？」其實還是根據字元的亮度，只是這種字元目標本身有「地形起伏」會有比背景的平面更「向光」與更「背光」的部分，於是字元內部就會產生「比背景亮」與「比背景暗」的區塊。

對於這種影像來說，太亮或太暗都可能是字元的一部份，所以簡單的二值化概念就不能同時呈現它們當作同性質目標來處理了！我們必須設法將這些「**太亮**」與「**太暗**」的區域結合，才能組織出真正完整的字元目標。但是亮暗區之間還是有過渡地帶，它們的亮度與背景幾乎一樣，所以如果只用單點為基礎的處理方法，就永遠難以縫合亮與暗的區塊。

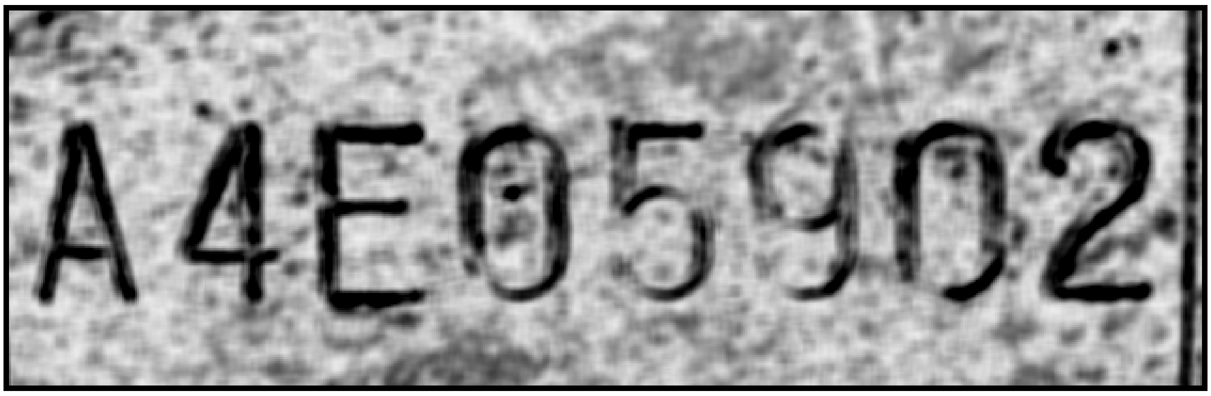
一個有趣的現象是：這種凹凸字形產生光影的強度，正好與數學上的地形變化的微分值相似，一座山峰從平地拔起時，面光的上坡微分值為正，背光的下坡微分值為負，平地的微分值則趨近於零，如果我們收集鄰近點的微分值作為中心點的新值，那就等於是在作小範圍的「**積分**」了！

因為微分值的正與負都代表地形有「變化」，也就可能是字元區，所以我們收集各點鄰近小區域內所有點的亮度與平均亮度(背景亮度)的差值(微分值)，就可以代表此點是不是字元的特徵強度了！如果是字元區，不管是偏亮或暗，這個值會偏高。反之，如

果是背景，則只有粗糙表面的較小亮度變化，這個值就會明顯低於被雕刻的字元區。這就是我們設計產生此辨識流程灰階圖的方式，按鍵 Integral 的程式碼如下：

```
'蝕刻或浮雕字影像的影像前處理→空間亮度偏差值積分
Private Sub IntegralToolStripMenuItem_Click(ByVal sender As Object, ByVal e As EventArgs) _
    Handles IntegralToolStripMenuItem.Click
    '計算全圖平均亮度
    brt = 0
    For i As Integer = 0 To nx - 1
        For j As Integer = 0 To ny - 1
            brt += Gv(i, j)
        Next
    Next
    brt /= nx * ny
    '計算空間亮度偏差值的局部積分
    Dim T(nx - 1, ny - 1) As Integer
    Dim mx As Integer = 0
    For i As Integer = 3 To nx - 4
        For j As Integer = 3 To ny - 4
            Dim d As Integer = 0
            For x As Integer = -3 To 3
                For y As Integer = -3 To 3
                    d += Math.Abs(Gv(i + x, j + y) - brt) '與平均亮度的偏差值
                Next
            Next
            T(i, j) = d '此點積分值
            If d > mx Then mx = d '最大積分值
        Next
    Next
    '積分陣列轉換為灰階
    ReDim B(nx - 1, ny - 1) '重設灰階陣列
    Dim f As Single = mx / 255 / 2 '積分值投射為灰階值之比例(X2 倍)
    For i As Integer = 3 To nx - 4
        For j As Integer = 3 To ny - 4
            Dim u As Integer = T(i, j) / f
            If u > 255 Then u = 255
            B(i, j) = 255 - u '灰階，目標為深色
        Next
    Next
    PictureBox1.Image = GrayImg(B) '灰階圖
End Sub
```

上面程式首先是計算全圖的平均亮度，再以 7x7 的矩陣作區域的亮度與背景亮度差值的積分，接著將此積分矩陣值正規化到 0-255 的區間，就可以作出能凸顯字元區的灰階圖如下：



12-3 鎖定字元列的上下邊界

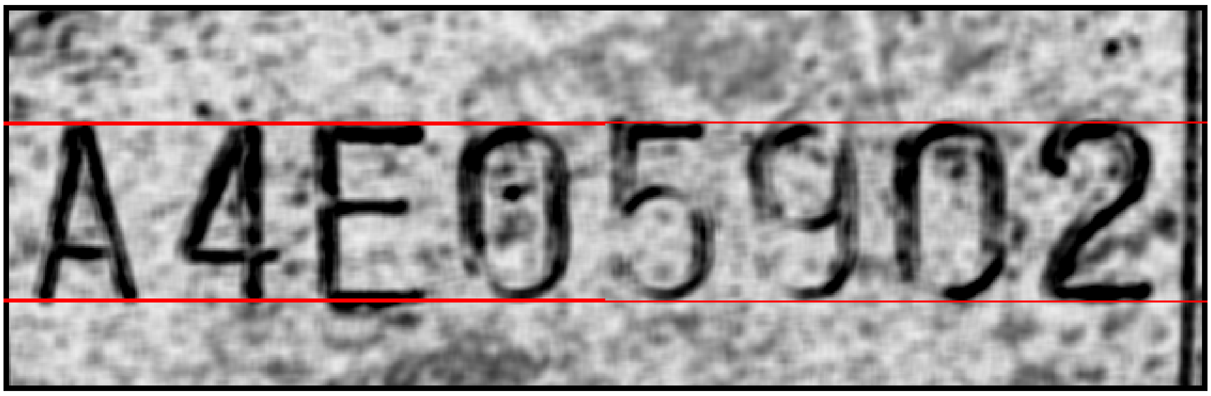
雖然我們已經作出了讓字元較好辨識的灰階圖，但背景的雜訊還是很高，字元內部也因為光線照射角度的關係，筆畫未必都很清晰連續，也因此接下來的辨識程序還是必須步步為營，要有特殊的設計。就是在每一階段都先盡量排除可以先行排除的雜訊，**讓後續辨識過程面對較少的雜訊**，降低因為雜訊誤判而迷航無法辨識成功的機率。這樣才能逐步聚焦，最終準確抓到最合理的字元位置。

首先是我們已經從灰階圖上看到了整排的字元，人類視覺上就一定會自動排除字元區以上及以下的背景區，只看有字元的區段。換言之，接下來我們要做二值化與建立可能目標時，應該根本不去處理字元區以外的區域，最好先排除這些區域的所有資訊，這樣也就等於先排除了一堆絕對不可能是字元的雜訊了！

一個很簡單的想法是字元區比較黑，背景區比較白，如果我們統計每一個 Y 值橫列的總亮度值，字元區的上邊界應該會從背景亮度驟降到字元區的較低亮度，因此會有個很大的亮度差值，應該就是全圖上半部最大的亮度差值位置。反之，下邊界也會有類似的現象，我們只要用程式找出這兩個亮度驟變的位置就可以找到字元區的上下邊界了！Y bound 按鍵就是執行上述功能的程式，程式碼與執行結果如下：

```
'搜尋鎖定字元列的上下切線
Private Sub YBoundToolStripMenuItem_Click(ByVal sender As Object, ByVal e As EventArgs) _
    Handles YBoundToolStripMenuItem.Click
    Dim Yb(ny - 1) As Integer '各 Y 值橫列的亮度總和
    For j As Integer = 0 To ny - 1
        For i As Integer = 0 To nx - 1
            Yb(j) += B(i, j)
        Next
    Next
    '用相鄰 Y 值的亮度差最大位置鎖定字元列上下邊界
    Dim tmx As Integer = 0, bmx As Integer = 0 '最大亮度差值
    Ytop = 0 : Ybot = ny - 1 '預設字元列上下邊界 Y 值
    For j As Integer = 0 To ny - 2
        If Yb(j) = 0 Or Yb(j + 1) = 0 Then Continue For '無資料略過
```

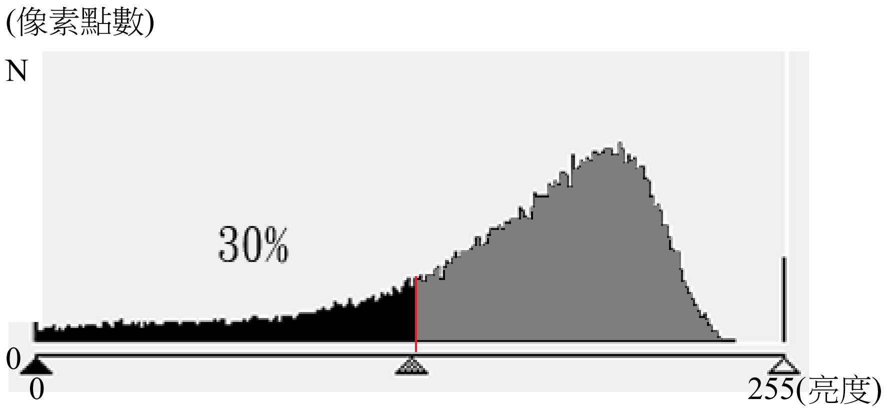
```
            If j < ny / 2 Then '上邊界搜尋
                Dim dy As Integer = Yb(j) - Yb(j + 1)
                If dy > tmx Then
                    tmx = dy : Ytop = j
                End If
            Else '下邊界搜尋
                Dim dy As Integer = Yb(j + 1) - Yb(j)
                If dy > bmx Then
                    bmx = dy : Ybot = j + 1
                End If
            End If
        Next
        '繪製上下邊界
        Dim bmp As Bitmap = GrayImg(B)
        Dim G As Graphics = Graphics.FromImage(bmp)
        G.DrawLine(Pens.Red, 0, Ytop, nx - 1, Ytop)
        G.DrawLine(Pens.Red, 0, Ybot, nx - 1, Ybot)
        PictureBox1.Image = bmp
    End Sub
```



12-4 二值化

如前所述，我們已經定義了字元區的 Y 值範圍，所以只在這個區域內做二值化，本書中之前我們作二值化的概念是以平均亮度作為黑白兩色的門檻，但這是比較粗糙，通常會讓門檻偏高的作法，在這種高雜訊的影像上其實會造成災難，結果大概會變成這樣：

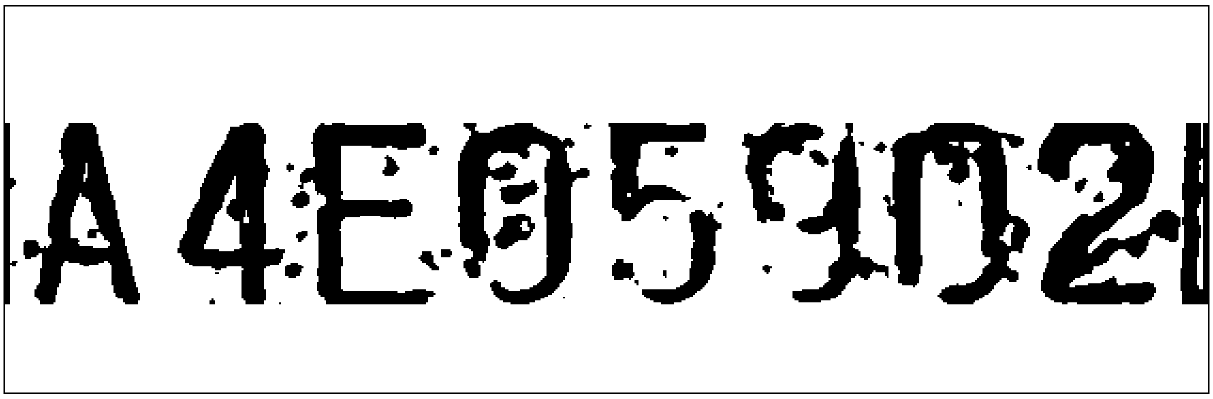
在此我們介紹一種使用灰階直方圖分佈為基礎，根據預期的黑點覆蓋率來決定門檻值的較精準方式，就是先找出字元區的灰階亮度直方圖分佈，因為我們知道每個亮度的畫素點數，當然也知道所有畫素的總點數，所以可以計算到哪個亮度的累積畫素點數會佔全圖的幾分之幾？示意圖如下：



依據我們的經驗，一般字元區的覆蓋率依筆畫疏密不同，大約落在 30-50%之間，所以就暫定 40%的覆蓋率來推算門檻值，Binary 按鍵的程式碼與執行結果如下：

```
'二值化
Private Sub BinaryToolStripMenuItem_Click(ByVal sender As Object, ByVal e As System.EventArgs) _
    Handles BinaryToolStripMenuItem.Click
    Dim his(255) As Integer, n As Integer = 0 '字元區的亮度分佈直方圖
    For j As Integer = Ytop To Ybot
        For i As Integer = 0 To nx - 1
            his(B(i, j)) += 1 : n += 1
        Next
    Next
    '計算二值化門檻值
    brt = 0
    Dim ac As Integer = his(brt)
    Do While ac < n * 0.4 '假設點強度前 25%為目標
        brt += 1
        ac += his(brt)
    Loop
```

```
        ReDim Z(nx - 1, ny - 1) '建立二值化陣列
        For i As Integer = 0 To nx - 1
            For j As Integer = Ytop To Ybot
                If B(i, j) < brt Then Z(i, j) = 1 '低於亮度門檻設為目標點
            Next
        Next
        Dim bmp As Bitmap = BWImg(Z) '建立二值化圖
        PictureBox1.Image = bmp '顯示
    End Sub
```



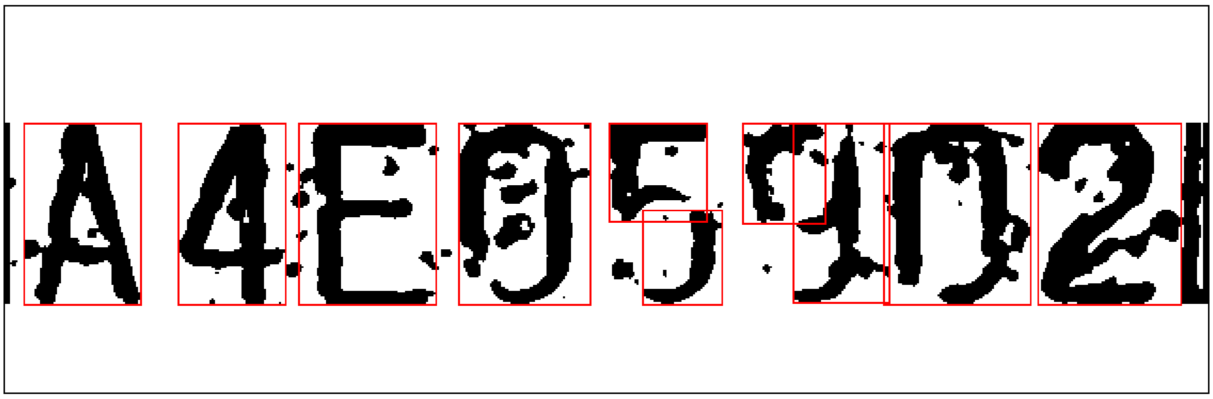
對於高雜訊的影像來說，二值化門檻的決定很困難，但也非常關鍵！太高會讓字元間沾連嚴重無法正確切割成單一字元，太低則會過於破碎，一個字會變成好多碎片。對於高雜訊影像我們不能總是期望有單一的理想門檻值可以讓所有字元粒粒分明，每個字都是一個獨立目標。即使偶爾運氣好，真的一字一目標，也會因為雜訊讓字元變形，如上圖的 9 字就多了一個尾巴，將它當作正常字元目標比對字模是一定無法辨識成功的！

所以我們必須引進很多統計平均或取中值的概念，或是外部的已知邊界條件來介入影像處理，不能完全信任影像辨識的過程資訊，譬如字元忽大忽小是一定不對的！我們必須用目標群的統計值或外部資訊預先假設字元的長寬比，讓每個目標都變成一樣大，這就是我們接下來會做的事情。在此二值化的階段，我們希望至少做到「**大部分**」的字元目標都能獨立出來，不要過度沾連或破碎，讓後續修補目標的措施難以施展即可。

12-5 建立目標物件

這部分程式與前面幾章高度類似，沒甚麼需要特別解釋，只是將目標寬與高的下限設定在 20 畫素，太破碎的小目標加以忽略而已，Target 按鍵的程式碼與執行結果如下：

```
    '建立目標物件
    Private Sub TargetsToolStripMenuItem_Click(ByVal sender As Object, ByVal e As System.EventArgs) _
        Handles TargetsToolStripMenuItem.Click
        C = getTargets(Z) '建立目標物件集合
        '繪製目標輪廓點
        Dim bmp As Bitmap = BWImg(Z)
        Dim G As Graphics = Graphics.FromImage(bmp)
        For k As Integer = 0 To C.Count - 1
            Dim T As TgInfo = C(k)
            G.DrawRectangle(Pens.Red, T.xmn, T.ymn, T.width, T.height)
        Next
        PictureBox1.Image = bmp '顯示目標輪廓
    End Sub
    '建立目標
    Function getTargets(ByVal q(,) As Byte) As ArrayList
        Dim minwidth As Integer = 20, minHeight As Integer = 20   '最小有效目標寬度寬高度
        Dim A As New ArrayList
        Dim b(,) As Byte = q.Clone '建立輪廓點陣列副本
        For i As Integer = 1 To nx - 2
            For j As Integer = 1 To ny - 2
                If b(i, j) = 0 Then Continue For
                Dim G As New TgInfo
                G.xmn = i : G.xmx = i : G.ymn = j : G.ymx = j : G.np = 1 : G.P = New ArrayList
                Dim nc As New ArrayList '每一輪搜尋的起點集合
                nc.Add(New Point(i, j)) '輸入之搜尋起點
                G.P.Add(New Point(i, j))
                b(i, j) = 0 '清除此起點之輪廓點標記
                Do
                    Dim nb As ArrayList = nc.Clone '複製此輪之搜尋起點集合
                    nc = New ArrayList '清除準備蒐集下一輪搜尋起點之集合
                    For m As Integer = 0 To nb.Count - 1
                        Dim p As Point = nb(m) '搜尋起點
                        '在此點周邊 3X3 區域內找輪廓點
                        For ii As Integer = p.X - 1 To p.X + 1
                            If ii < 0 Or ii > nx - 1 Then Continue For
                            For jj As Integer = p.Y - 1 To p.Y + 1
                                If jj < 0 Or jj > ny - 1 Then Continue For
                                If b(ii, jj) = 0 Then Continue For '非輪廓點忽略
                                Dim k As New Point(ii, jj) '建立點物件
                                nc.Add(k) '本輪搜尋新增的輪廓點
                                G.P.Add(k)
                                G.np += 1 '點數累計
                                If ii < G.xmn Then G.xmn = ii
                                If ii > G.xmx Then G.xmx = ii
                                If jj < G.ymn Then G.ymn = jj
                                If jj > G.ymx Then G.ymx = jj
                                b(ii, jj) = 0 '清除輪廓點點標記
                            Next
                        Next
                    Next
                Loop While nc.Count > 0 '此輪搜尋有新發現輪廓點時繼續搜尋
                If Z(i - 1, j) = 1 Then Continue For '排除白色區塊的負目標，起點左邊是黑點
                G.width = G.xmx - G.xmn + 1 '寬度計算
                If G.width < minwidth Then Continue For
                G.height = G.ymx - G.ymn + 1 '高度計算
                If G.height < minHeight Then Continue For
                A.Add(G) '加入有效目標集合
            Next
        Next
        Return A '回傳目標物件集合
    End Function
```

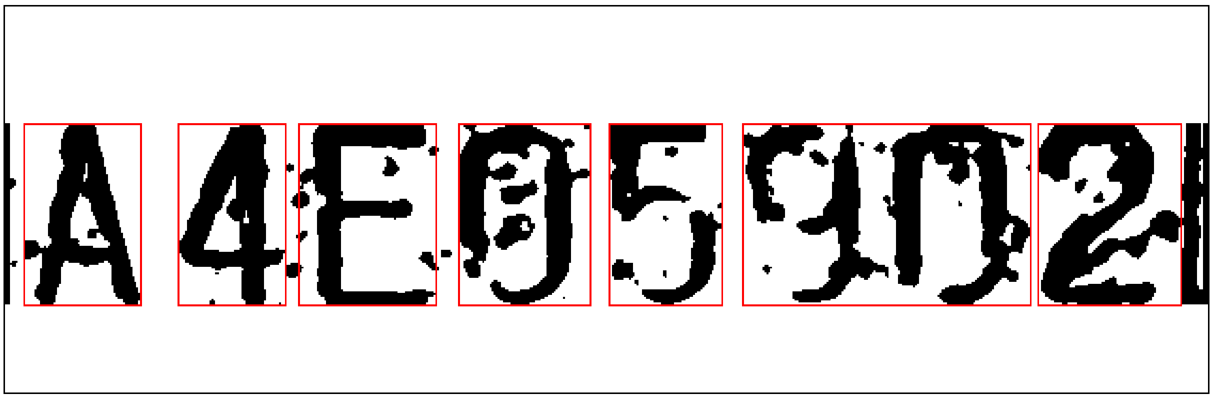


12-6 融合與分割目標

可以看到上圖目標擷取的結果中的一些異常現象是：某些字元因為突出正常字元外的雜訊變形了，最嚴重的是 5 與 9 兩個字破碎了！用只包含字元**局部**資訊的目標去比對標準字模，當然不可能認出正確的字，所以首要的處理項目是：要**保證每個字元的全部資訊必須包含在單一目標之內**！這就是 Merge 按鍵的功能，程式碼與執行結果如下：

```
'融合交疊目標
Private Sub MergeTgsToolStripMenuItem_Click(ByVal sender As Object, ByVal e As EventArgs) _
    Handles MergeTgsToolStripMenuItem.Click
    Dim merged(C.Count - 1) As Boolean '是否已被她目標融合之標記陣列
    For k As Integer = 0 To C.Count - 1
        If merged(k) Then Continue For '已被融合目標略過
        Dim T As TgInfo = C(k)
        For m As Integer = k + 1 To C.Count - 1
            If merged(k) Then Continue For
            Dim G As TgInfo = C(m)
            If T.xmx > G.xmn Then '目標交疊進行融合
                T.xmn = Math.Min(T.xmn, G.xmn)
                T.xmx = Math.Max(T.xmx, G.xmx)
                T.ymn = Math.Min(T.ymn, G.ymn)
                T.ymx = Math.Max(T.ymx, G.ymx)
                T.width = T.xmx - T.xmn + 1
                T.height = T.ymx - T.ymn + 1
                For i As Integer = 0 To G.P.Count - 1
```

```
                    T.P.Add(G.P(i))
                Next
                merged(m) = True '標記已被融合目標
                C(k) = T '回置目標物件
            End If
        Next
    Next
    '排除已被融合目標，建立新目標物件集合
    Dim A As New ArrayList
    For k As Integer = 0 To C.Count - 1
        If merged(k) Then Continue For
        A.Add(C(k))
    Next
    C = A '回置目標集合
    '繪製已融合處理之目標範圍框架
    Dim bmp As Bitmap = BWImg(Z)
    Dim Gr As Graphics = Graphics.FromImage(bmp)
    For k As Integer = 0 To C.Count - 1
        Dim T As TgInfo = C(k)
        Gr.DrawRectangle(Pens.Red, T.xmn, T.ymn, T.width, T.height)
    Next
    PictureBox1.Image = bmp
End Sub
```

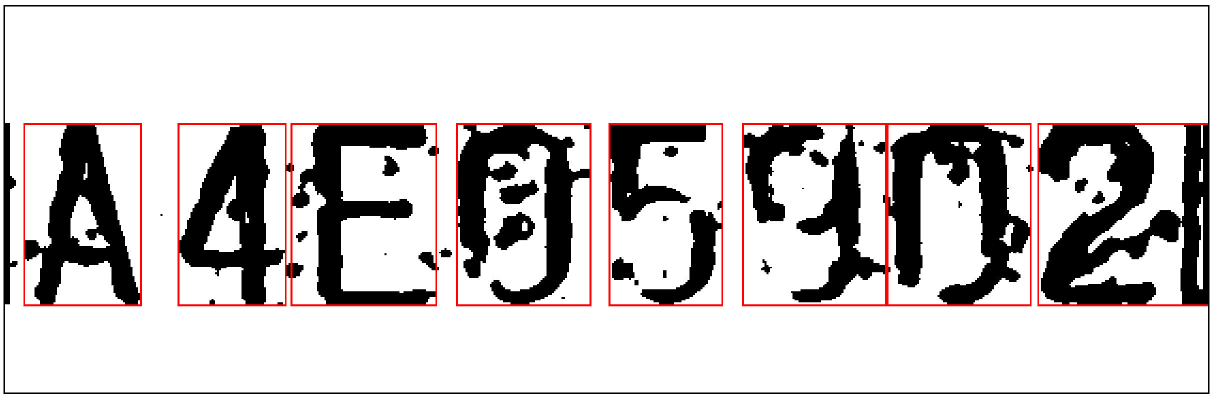


我們使用的需融合目標條件是：兩個目標在 X 方向上有重複交疊的部分，因為我們已經將辨識區縮減到單排字元，所以包含有同樣 X 座標的目標「應該」是同一個字！所謂的「融合」就是取兩個目標的四方座標極值，重新定義給第一個目標，被吸收的目標則先加上標記，避免被重複使用，隨後被排除掉。

比對上一節的圖，果然該融合的目標都融合了，5 和 9 的合理內容都被包覆到單一目標內了！但是 9 與 0 兩個不應該融合的目標也被融合了！原因是 9 字類似尾巴的雜訊延伸超過了原來 0 字目標的 X 起點，也符合了上述融合的條件，所以就被融合了！這當然也是一個問題，在此我們假設最多就是兩個目標沾連，只做簡單的處理，就是讓過胖到明顯是兩個目標的目標直接從中切開為兩個目標，將以下程式片段插入即可：

```
'分割過大的沾連目標
Dim bh As Integer = Ybot - Ytop + 1 '理想字元高度
Dim bw As Integer = bh * 0.6 '理想字元寬度
A = New ArrayList
For k As Integer = 0 To C.Count - 1
    Dim T As TgInfo = C(k)
    If T.width > bw * 2 Then '兩字沾連進行等分切割
        Dim G As TgInfo = T
        Dim midx As Integer = (T.xmn + T.xmx) / 2
        T.xmx = midx : T.width = T.xmx - T.xmn + 1
        G.xmn = midx : G.width = G.xmx - G.xmn + 1
        A.Add(T) : A.Add(G)
    Else '無沾連
        A.Add(T)
    End If
Next
C = A '回置標物件集合
```

再次執行此功能結果如下：



12-7 正規化目標大小

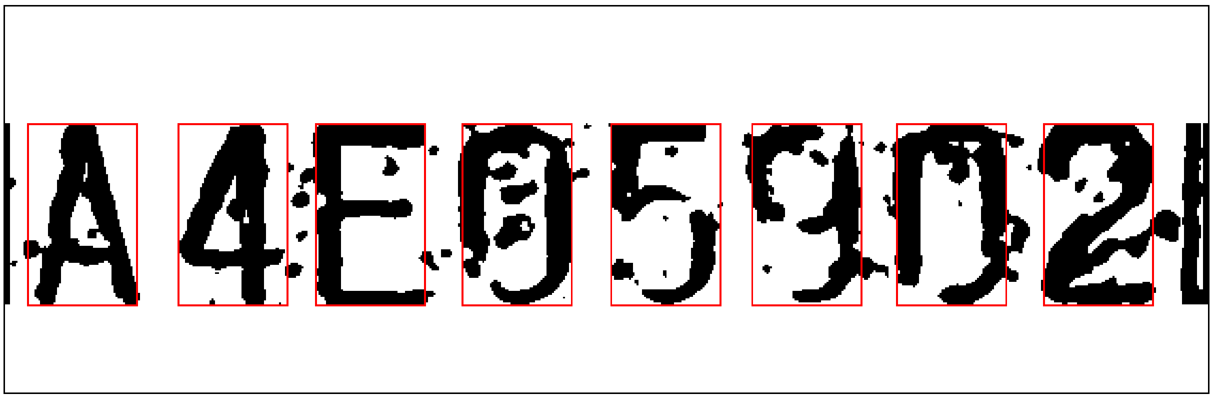
上節的處理結果還有甚麼可以挑剔的呢？每個字元都已經合理的變成獨立目標了！但是大家可以想像一下，其中的 A、4 與 5 字或許可以直接辨識成功，但是其他幾個字都因為字元邊緣突出的雜訊而明顯變形了，辨識失敗的可能性遠高於僥倖成功！如果沒有進一步的處理，整體辨識率還是會很低的！

我們還有甚麼資訊，或說招數可用呢？就是「**圖上字元都應該完全一樣大**」，而且「字元寬高比(形狀)也應該一樣」的合理假設！通常這些打印字體都是固定的，所以我們可以從外部資訊知道字元的寬高比，在此例中是大約 3 比 5！當我們從影像辨識中得到字元在影像中的實際高度是 bh = Tbot - Ttop 時，合理寬度 bw 就是 bh * 0.6 了！這個資訊其實在前面分割過胖字元時已經參考使用過了！

所以我們將每個目標寬度都強制定義成 bw，以此為前提，左右移動測試，找到哪個偏移位置可以包含最多的黑點，目標點密度最高也就是最理想的目標位置了！Best Fit 的按鍵功能程式碼與執行結果如下：

```
'鎖定最佳目標區塊
Private Sub BestFitToolStripMenuItem_Click(ByVal sender As Object, ByVal e As EventArgs) _
    Handles BestFitToolStripMenuItem.Click
    Dim bh As Integer = Ybot - Ytop + 1 '理想字元高度
    Dim bw As Integer = bh * 0.6 '理想字元寬度
    '建立 X 方向的每行強度總和變化陣列
    Dim Xb(nx - 1) As Integer
    For i As Integer = 0 To nx - 1
        For j As Integer = Ytop To Ybot
            Xb(i) += Z(i, j)
        Next
    Next
    '假設目標為理想寬度，搜尋目標點密度最高之 X 位置
    For k As Integer = 0 To C.Count - 1
        Dim T As TgInfo = C(k)
        Dim dw As Integer = Math.Abs(bw - T.width) '目標寬度與理想寬度之差
        Dim x1 As Integer = T.xmn, x2 As Integer = T.xmn '左右搜尋之 X 範圍
        If T.width < bw Then x1 -= dw '目標窄於理想寬度時，搜尋起點左移
        If T.width > bw Then x2 += dw '目標寬於理想寬度時，搜尋終點右移
        Dim mx As Integer = 0, mi As Integer = T.xmn '假設目標內之最大目標點總量與 X 位置
        For i As Integer = x1 To x2
            Dim v As Integer = 0 '此估計目標內之目標點累計值
            For x As Integer = 0 To bw - 1
                If i + x > nx - 1 Then Continue For
                v += Xb(i + x)
            Next
            If v > mx Then '找到目前最高目標點總量位置
                mx = v : mi = i
            End If
        Next
        T.width = bw '修改目標點寬度為理想寬度
        T.xmn = mi : T.xmx = mi + bw - 1 '修改目標左右邊界
        '重建目標物件內之目標點集合內容
        T.P = New ArrayList
        For i As Integer = T.xmn To T.xmx
            For j As Integer = T.ymn To T.ymx
                If Z(i, j) = 1 Then T.P.Add(New Point(i, j))
            Next
        Next
        C(k) = T
    Next
```

```
        '繪製最佳目標位置與範圍
        Dim bmp As Bitmap = BWImg(Z)
        Dim Gr As Graphics = Graphics.FromImage(bmp)
        For k As Integer = 0 To C.Count - 1
            Dim T As TgInfo = C(k)
            Gr.DrawRectangle(Pens.Red, T.xmn, T.ymn, T.width, T.height)
        Next
        PictureBox1.Image = bmp
    End Sub
```



看到了嗎？現在每一個目標都變成完全一樣大，形狀也完全一樣了！最重要的是：我們看到程式選擇的位置確實都很理想！都能正確框住我們視覺上認為最合理的字元區域，成功的忽略了從字元意外延伸出來的雜訊。可以預期這種目標要拿去字元比對的成功率就一定非常高了！

完整專案

```
Public Class Form1
    Dim B(,) As Byte '灰階陣列
    Dim Z(,) As Byte '全圖二值化陣列
    Dim C As ArrayList '目標物件集合
    Dim brt As Integer = 128 '全圖平均亮度
    Dim Ytop As Integer, Ybot As Integer '字元列上下切線之 Y 值
    '目標物件結構
    Public Structure TgInfo
        Dim np As Integer '目標點數
        Dim P As ArrayList '目標點的集合
        Dim xmn As Short, xmx As Short, ymn As Short, ymx As Short '四面座標極值
        Dim width As Integer, height As Integer '寬與高
    End Structure
    '開啟檔案
    Private Sub OpenToolStripMenuItem_Click(ByVal sender As Object, ByVal e As System.EventArgs) _
        Handles OpenToolStripMenuItem.Click
        If OpenFileDialog1.ShowDialog = Windows.Forms.DialogResult.OK Then
            Dim bmp As New Bitmap(OpenFileDialog1.FileName)
            Bmp2RGB(bmp) '擷取影像資訊
            PictureBox1.Image = bmp '顯示
        End If
    End Sub
    '蝕刻或浮雕字影像的影像前處理→空間亮度偏差值積分
    Private Sub IntegralToolStripMenuItem_Click(ByVal sender As Object, ByVal e As EventArgs) _
        Handles IntegralToolStripMenuItem.Click
        '計算全圖平均亮度
        brt = 0
        For i As Integer = 0 To nx - 1
            For j As Integer = 0 To ny - 1
                brt += Gv(i, j)
            Next
        Next
        brt /= nx * ny
        '計算空間亮度偏差值的局部積分
        Dim T(nx - 1, ny - 1) As Integer
        Dim mx As Integer = 0
        For i As Integer = 3 To nx - 4
            For j As Integer = 3 To ny - 4
                Dim d As Integer = 0
                For x As Integer = -3 To 3
                    For y As Integer = -3 To 3
                        d += Math.Abs(Gv(i + x, j + y) - brt) '與平均亮度的偏差值
                    Next
                Next
                T(i, j) = d '此點積分值
                If d > mx Then mx = d '最大積分值
            Next
        Next
        '積分陣列轉換為灰階
        ReDim B(nx - 1, ny - 1) '重設灰階陣列
        Dim f As Single = mx / 255 / 2 '積分值投射為灰階值之比例(X2 倍)
        For i As Integer = 3 To nx - 4
            For j As Integer = 3 To ny - 4
                Dim u As Integer = T(i, j) / f
                If u > 255 Then u = 255
                B(i, j) = 255 - u '灰階，目標為深色
            Next
        Next
        PictureBox1.Image = GrayImg(B) '灰階圖
    End Sub
    '搜尋鎖定字元列的上下切線
    Private Sub YBoundToolStripMenuItem_Click(ByVal sender As Object, ByVal e As EventArgs) _
        Handles YBoundToolStripMenuItem.Click
        Dim Yb(ny - 1) As Integer '各 Y 值橫列的亮度總和
        For j As Integer = 0 To ny - 1
            For i As Integer = 0 To nx - 1
                Yb(j) += B(i, j)
            Next
        Next
        '用相鄰 Y 值的亮度差最大位置鎖定字元列上下邊界
        Dim tmx As Integer = 0, bmx As Integer = 0 '最大亮度差值
        Ytop = 0 : Ybot = ny - 1 '預設字元列上下邊界 Y 值
        For j As Integer = 0 To ny - 2
            If Yb(j) = 0 Or Yb(j + 1) = 0 Then Continue For '無資料略過
            If j < ny / 2 Then '上邊界搜尋
                Dim dy As Integer = Yb(j) - Yb(j + 1)
                If dy > tmx Then
                    tmx = dy : Ytop = j
                End If
            Else '下邊界搜尋
                Dim dy As Integer = Yb(j + 1) - Yb(j)
                If dy > bmx Then
                    bmx = dy : Ybot = j + 1
                End If
            End If
        Next
        '繪製上下邊界
        Dim bmp As Bitmap = GrayImg(B)
        Dim G As Graphics = Graphics.FromImage(bmp)
        G.DrawLine(Pens.Red, 0, Ytop, nx - 1, Ytop)
```

```
        G.DrawLine(Pens.Red, 0, Ytop + 1, nx - 1, Ytop)
        G.DrawLine(Pens.Red, 0, Ybot, nx - 1, Ybot)
        G.DrawLine(Pens.Red, 0, Ybot - 1, nx - 1, Ybot)
        PictureBox1.Image = bmp
    End Sub
    '二值化
    Private Sub BinaryToolStripMenuItem_Click(ByVal sender As Object, ByVal e As System.EventArgs) _
        Handles BinaryToolStripMenuItem.Click
        Dim his(255) As Integer, n As Integer = 0 '字元區的亮度分佈直方圖
        For j As Integer = Ytop To Ybot
            For i As Integer = 0 To nx - 1
                his(B(i, j)) += 1 : n += 1
            Next
        Next
        '計算二值化門檻值
        brt = 0
        Dim ac As Integer = his(brt)
        Do While ac < n * 0.4 '假設點強度前 25%為目標
            brt += 1
            ac += his(brt)
        Loop
        ReDim Z(nx - 1, ny - 1) '建立二值化陣列
        For i As Integer = 0 To nx - 1
            For j As Integer = Ytop To Ybot
                If B(i, j) < brt Then Z(i, j) = 1 '低於亮度門檻設為目標點
            Next
        Next
        PictureBox1.Image = BWImg(Z) '建立二值化圖
    End Sub
    '建立目標物件
    Private Sub TargetsToolStripMenuItem_Click(ByVal sender As Object, ByVal e As System.EventArgs) _
        Handles TargetsToolStripMenuItem.Click
        C = getTargets(Z) '建立目標物件集合
        '繪製目標框線
        Dim bmp As Bitmap = BWImg(Z)
        Dim G As Graphics = Graphics.FromImage(bmp)
        For k As Integer = 0 To C.Count - 1
            Dim T As TgInfo = C(k)
            G.DrawRectangle(Pens.Red, T.xmn, T.ymn, T.width, T.height)
        Next
        PictureBox1.Image = bmp '顯示目標輪廓
    End Sub
    '建立目標
    Function getTargets(ByVal q(,) As Byte) As ArrayList
        Dim minwidth As Integer = 20, minHeight As Integer = 20    '最小有效目標寬度寬高度
        Dim A As New ArrayList
        Dim b(,) As Byte = q.Clone '建立輪廓點陣列副本
        For i As Integer = 1 To nx - 2
            For j As Integer = 1 To ny - 2
                If b(i, j) = 0 Then Continue For
                Dim G As New TgInfo
                G.xmn = i : G.xmx = i : G.ymn = j : G.ymx = j : G.np = 1 : G.P = New ArrayList
                Dim nc As New ArrayList '每一輪搜尋的起點集合
                nc.Add(New Point(i, j)) '搜尋起點
                G.P.Add(New Point(i, j))
                b(i, j) = 0 '清除此起點之輪廓點標記
                Do
                    Dim nb As ArrayList = nc.Clone '複製此輪之搜尋起點集合
                    nc = New ArrayList '清除準備蒐集下一輪搜尋起點之集合
                    For m As Integer = 0 To nb.Count - 1
                        Dim p As Point = nb(m) '搜尋起點
                        '在此點周邊 3X3 區域內找目標點
                        For ii As Integer = p.X - 1 To p.X + 1
                            If ii < 0 Or ii > nx - 1 Then Continue For '避免搜尋越界
                            For jj As Integer = p.Y - 1 To p.Y + 1
                                If jj < 0 Or jj > ny - 1 Then Continue For '避免搜尋越界
                                If b(ii, jj) = 0 Then Continue For '非目標點忽略
                                Dim k As New Point(ii, jj) '建立點物件
                                nc.Add(k) '本輪搜尋新增的目標點
                                G.P.Add(k)
                                G.np += 1 '點數累計
                                If ii < G.xmn Then G.xmn = ii
                                If ii > G.xmx Then G.xmx = ii
                                If jj < G.ymn Then G.ymn = jj
                                If jj > G.ymx Then G.ymx = jj
                                b(ii, jj) = 0 '清除輪廓點點標記
                            Next
                        Next
                    Next
                Loop While nc.Count > 0 '此輪搜尋有新發現輪廓點時繼續搜尋
                If Z(i - 1, j) = 1 Then Continue For '排除白色區塊的負目標，起點左邊是黑點
                G.width = G.xmx - G.xmn + 1 '寬度計算
                If G.width < minwidth Then Continue For
                G.height = G.ymx - G.ymn + 1 '高度計算
                If G.height < minHeight Then Continue For
                A.Add(G) '加入有效目標集合
            Next
        Next
        Return A '回傳目標物件集合
    End Function
    '儲存目前影像
    Private Sub SaveImageToolStripMenuItem_Click(ByVal sender As Object, ByVal e As System.EventArgs)
```

```
_
        Handles SaveImageToolStripMenuItem.Click
        If SaveFileDialog1.ShowDialog = Windows.Forms.DialogResult.OK Then
            PictureBox1.Image.Save(SaveFileDialog1.FileName)
        End If
    End Sub
    '融合交疊目標
    Private Sub MergeTgsToolStripMenuItem_Click(ByVal sender As Object, ByVal e As EventArgs) _
        Handles MergeTgsToolStripMenuItem.Click
        Dim merged(C.Count - 1) As Boolean '是否已被她目標融合之標記陣列
        For k As Integer = 0 To C.Count - 2
            If merged(k) Then Continue For '已被融合目標略過
            Dim T As TgInfo = C(k)
            For m As Integer = k + 1 To C.Count - 1
                If merged(k) Then Continue For
                Dim G As TgInfo = C(m)
                If T.xmx > G.xmn Then '目標交疊進行融合
                    T.xmn = Math.Min(T.xmn, G.xmn)
                    T.xmx = Math.Max(T.xmx, G.xmx)
                    T.ymn = Math.Min(T.ymn, G.ymn)
                    T.ymx = Math.Max(T.ymx, G.ymx)
                    T.width = T.xmx - T.xmn + 1
                    T.height = T.ymx - T.ymn + 1
                    For i As Integer = 0 To G.P.Count - 1
                        T.P.Add(G.P(i))
                    Next
                    merged(m) = True '標記已被融合目標
                    C(k) = T '回置目標物件
                End If
            Next
        Next
        '排除已被融合目標，建立新目標物件集合
        Dim A As New ArrayList
        For k As Integer = 0 To C.Count - 1
            If merged(k) Then Continue For
            A.Add(C(k))
        Next
        C = A '回置目標集合
        '分割過大的沾連目標
        Dim bh As Integer = Ybot - Ytop + 1 '理想字元高度
        Dim bw As Integer = bh * 0.6 '理想字元寬度
        A = New ArrayList
        For k As Integer = 0 To C.Count - 1
            Dim T As TgInfo = C(k)
            If T.width > bw * 2 Then '兩字沾連進行等分切割
                Dim G As TgInfo = T
                Dim midx As Integer = (T.xmn + T.xmx) / 2
                T.xmx = midx : T.width = T.xmx - T.xmn + 1
                G.xmn = midx : G.width = G.xmx - G.xmn + 1
                A.Add(T) : A.Add(G)
            Else '無沾連
                A.Add(T)
            End If
        Next
        C = A '回置標物件集合
        '繪製已融合處理之目標範圍框架
        Dim bmp As Bitmap = BWImg(Z)
        Dim Gr As Graphics = Graphics.FromImage(bmp)
        For k As Integer = 0 To C.Count - 1
            Dim T As TgInfo = C(k)
            Gr.DrawRectangle(Pens.Red, T.xmn, T.ymn, T.width, T.height)
        Next
        PictureBox1.Image = bmp
    End Sub
    '點選目標顯示位置與屬性
    Private Sub PictureBox1_MouseDown(ByVal sender As Object, ByVal e As
Windows.Forms.MouseEventArgs) _
        Handles PictureBox1.MouseDown
        If e.Button = MouseButtons.Left Then
            Dim m As Integer = -1
            For k As Integer = 0 To C.Count - 1
                Dim T As TgInfo = C(k)
                If e.X < T.xmn Then Continue For
                If e.X > T.xmx Then Continue For
                If e.Y < T.ymn Then Continue For
                If e.Y > T.ymx Then Continue For
                m = k '被點選目標
                Exit For
            Next
            If m >= 0 Then '有被選目標時
                Dim bmp As Bitmap = BWImg(Z)
                Dim T As TgInfo = C(m)
                For k As Integer = 0 To T.P.Count - 1
                    Dim p As Point = T.P(k)
                    bmp.SetPixel(p.X, p.Y, Color.Red)
                Next
                Dim G As Graphics = Graphics.FromImage(bmp)
                G.DrawRectangle(Pens.Lime, T.xmn, T.ymn, T.width, T.height)
                PictureBox1.Image = bmp
                '指定目標的資訊
                Dim S As String = "Width=" + T.width.ToString
                S += vbNewLine + "Height=" + T.height.ToString
            S += vbNewLine + "Points=" + T.np.ToString
            MsgBox(S)
        End If
    End If
End Sub
'鎖定最佳目標區塊
Private Sub BestFitToolStripMenuItem_Click(ByVal sender As Object, ByVal e As EventArgs) _
    Handles BestFitToolStripMenuItem.Click
    Dim bh As Integer = Ybot - Ytop + 1 '理想字元高度
    Dim bw As Integer = bh * 0.6 '理想字元寬度
    '建立 X 方向的每行強度總和變化陣列
    Dim Xb(nx - 1) As Integer
    For i As Integer = 0 To nx - 1
        For j As Integer = Ytop To Ybot
            Xb(i) += Z(i, j)
        Next
    Next
    '假設目標為理想寬度，搜尋目標點密度最高之 X 位置
    For k As Integer = 0 To C.Count - 1
        Dim T As TgInfo = C(k)
        Dim dw As Integer = Math.Abs(bw - T.width) '目標寬度與理想寬度之差
        Dim x1 As Integer = T.xmn, x2 As Integer = T.xmn '左右搜尋之 X 範圍
        If T.width < bw Then x1 -= dw '目標窄於理想寬度時，搜尋起點左移
        If T.width > bw Then x2 += dw '目標寬於理想寬度時，搜尋終點右移
        Dim mx As Integer = 0, mi As Integer = T.xmn '假設目標內之最大目標點總量與 X 位置
        For i As Integer = x1 To x2
            Dim v As Integer = 0 '此估計目標內之目標點累計值
            For x As Integer = 0 To bw - 1
                If i + x > nx - 1 Then Continue For
                v += Xb(i + x)
            Next
            If v > mx Then '找到目前最高目標點總量位置
                mx = v : mi = i
            End If
        Next
        T.width = bw '修改目標點寬度為理想寬度
        T.xmn = mi : T.xmx = mi + bw - 1 '修改目標左右邊界
        '重建目標物件內之目標點集合內容
        T.P = New ArrayList
        For i As Integer = T.xmn To T.xmx
            For j As Integer = T.ymn To T.ymx
                If Z(i, j) = 1 Then T.P.Add(New Point(i, j))
            Next
        Next
        C(k) = T
    Next
```

```
        '繪製最佳目標位置與範圍
        Dim bmp As Bitmap = BWImg(Z)
        Dim Gr As Graphics = Graphics.FromImage(bmp)
        For k As Integer = 0 To C.Count - 1
            Dim T As TgInfo = C(k)
            Gr.DrawRectangle(Pens.Red, T.xmn, T.ymn, T.width, T.height)
        Next
        PictureBox1.Image = bmp
    End Sub
End Class
```