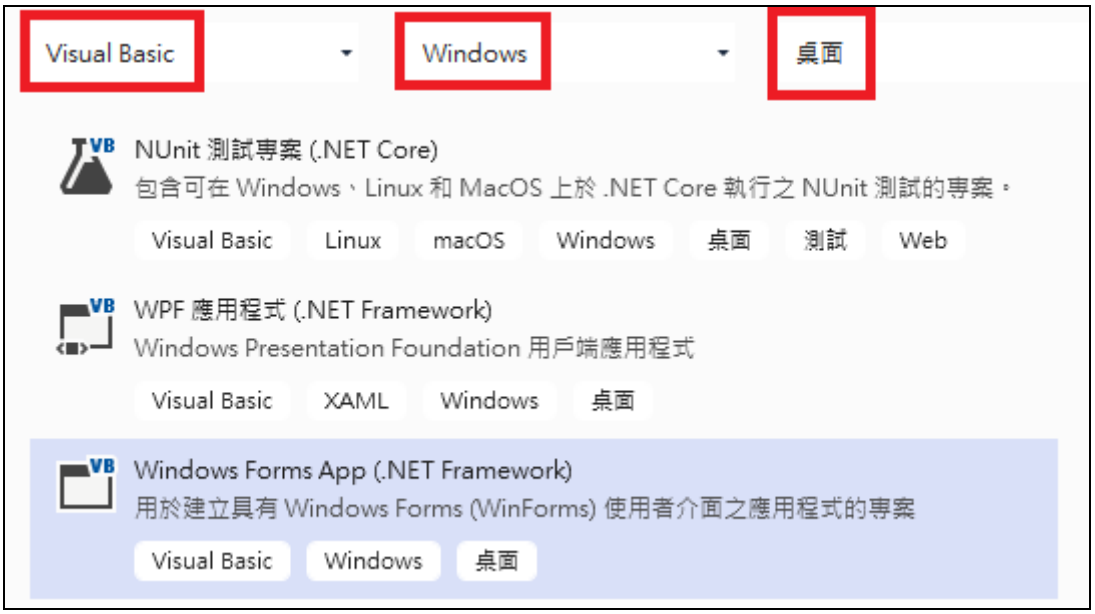
第 2 章　數位影像的拆解與顯示

2-0 本書程式使用 Visual Studio 2019 軟體製作

本書的每一章都會有完整的程式範例，以淺顯易懂的 VB 程式實作。選擇 VB 的主要原因是本公司各種影像辨識產品，都是以 VB 程式開發，這與許多影像辨識專業人士認為必須使用 C 語言的認知有些落差。但影像辨識的核心技術在於**演算法**，任何數學其實都可以用任何程式語言實作，加上在 Windows 作業系統下，不論是 VB，C#或 C++語言皆奠基於同一.NET 函式庫，所以執行效能是完全一樣的！

基於 VB 程式寫作環境的友善性，我們認為使用 VB 可以讓更多初學者輕鬆深入影像辨識的技術層面，尤其是實驗用的影像操作介面，可以更快在專案內直接製作完成，無須在不同程式語言、專案與軟體之間切換，可以讓讀者更專注於核心演算法的理解。建立專案類型時的選擇應該如下圖所示：



事實上因為本書程式範例都只包含基本的程式物件與演算法內容，使用的程式設計軟體版本影響極微，基本上 Visual Studio 2010 之後的版本皆可通用，如果您已有舊版軟體不必刻意改用較新的版本。

2-1 拆解數位影像的 RGB 資訊

現在多數的數位影像是由 RGB，也就是紅綠藍，三原色的三個陣列組合而成，各種影像格式的差異只是壓縮的方式不同，附加資訊會以不同的欄位寫在檔頭。檔案格式的差異不是本書的重點，在此我們是用.NET 函式庫的指令將它們讀入程式，成為 Bitmap 物件，也就是解碼成為可用程式操作的影像資料物件。

雖然 Bitmap 的本質就是數值陣列，但影像物件為了配合電腦顯示方便，封裝了很

多額外的功能與資料，反而讓我們的影像處理操作不便，用預設的程式指令讀出(GetPixel)或寫入(SetPixel)畫素資料時還特別的慢！所以要有專業級的影像存取速度，就必須使用記憶體直接存取的技巧，我們將幾個為此目的設計的副程式彙整到一個名為 FastPixel 的 VB 模組之中。

這個模組的概念是暫時在記憶體中鎖定 Bitmap 物件，將其畫素資料用記憶體直接讀寫的方式操作，讀出時就是將原始的一維記憶體資料陣列讀到 RGB 三個二維陣列，寫入時則是反向將 RGB 矩陣資料按照記憶體指標推算位置寫回記憶體，完成操作解除 Bitmap 物件的記憶體鎖定之後，就再度可以用一般的程式操作整個 Bitmap 物件了！

請建立一個 VB 視窗程式專案，並在專案中新增一個命名為 FastPixel 的模組，開始建立此模組的內容，首先是鎖定與解除鎖定 Bitmap 物件的程式碼如下：

```
Public nx As Integer, ny As Integer, Rv(,) As Byte, Gv(,) As Byte, Bv(,) As Byte
Dim rgb() As Byte '影像的可存取副本資料陣列
Dim D As System.Drawing.Imaging.BitmapData '影像資料
Dim ptr As IntPtr '影像資料所在的記憶體指標(位置)
Dim n As Integer '影像總共佔據的位元組數
Dim L As Integer '一個影像列的記憶體位元組數
Dim nB As Integer '每一像素點是幾個位元組？通常為 3(24bits)或 4(32bits)
'鎖定點陣圖(Bitmap)物件的記憶體位置，建立一個可操作的為元組陣列副本
Sub LockBMP(ByVal bmp As Bitmap)
    Dim rect As New Rectangle(0, 0, bmp.Width, bmp.Height) '矩形物件，定義影像範圍
    '鎖定影像區記憶體(暫時不接受作業系統的移動)
    D = bmp.LockBits(rect, Drawing.Imaging.ImageLockMode.ReadWrite, bmp.PixelFormat)
    ptr = D.Scan0 '影像區塊的記憶體指標
    L = D.Stride '每一影像列的長度(bytes)
    nB = L \ bmp.Width '每一像素的位元組數(3 或 4)
    n = L * bmp.Height '影像總位元組數
    ReDim rgb(n - 1) '宣告影像副本資料陣列
    System.Runtime.InteropServices.Marshal.Copy(ptr, rgb, 0, n) '拷貝點陣圖資料到副本陣列
End Sub
'複製位元組陣列副本的處理結果到 Bitmap 物件，並解除其記憶體鎖定
Sub UnLockBMP(ByVal bmp As Bitmap)
    System.Runtime.InteropServices.Marshal.Copy(rgb, 0, ptr, n) '拷貝副本陣列資料到點陣圖
    bmp.UnlockBits(D) '解除鎖定
End Sub
```

當我們要將某影像的 RGB 資料取出時，公開變數 nx 與 ny 代表該影像的寬與高，Rv 代表其紅光陣列，Gv 代表綠光，Bv 代表藍光，這是本書建立的習慣，不是系統的限制。取出 RGB 時的 Sub 副程式如下：

```
'取得 RGB 陣列
Public Sub Bmp2RGB(ByVal bmp As Bitmap)
    nx = bmp.Width : ny = bmp.Height
```

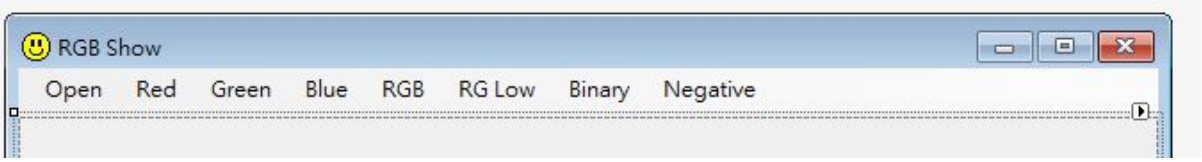
```
    ReDim Rv(nx - 1, ny - 1), Gv(nx - 1, ny - 1), Bv(nx - 1, ny - 1) '輸出用灰階陣列
    LockBMP(bmp)
    For j As Integer = 0 To ny - 1
        Dim Lj As Integer = j * D.Stride
        For i As Integer = 0 To nx - 1
            Dim k As Integer = Lj + i * nB
            Rv(i, j) = rgb(k + 2) '紅光
            Gv(i, j) = rgb(k + 1) '綠光
            Bv(i, j) = rgb(k) '藍光
        Next
    Next
    bmp.UnlockBits(D) '解除鎖定
End Sub
```

在資料處理實驗過程中，我們常常需要依據陣列資料繪製出灰階圖與二值化的黑白圖，因此以下兩個 Function 也是常常用到的。繪製灰階時就是饋入一個值域分布為 0-255 的二維陣列，0 是最黑，255 最白。繪製二值化圖時，則饋入值域為 0 或 1 的陣列，0 代表白色背景，1 代表黑色目標。繪製完畢就回傳一個可以顯示或儲存的 Bitmap 物件！

```
'灰階圖
Function GrayImg(ByVal b(,) As Byte) As Bitmap
    Dim bmp As New Bitmap(b.GetLength(0), b.GetLength(1))
    LockBMP(bmp)
    For i As Integer = 0 To b.GetLength(0) - 1
        For j As Integer = 0 To b.GetLength(1) - 1
            Dim k As Integer = j * L + i * nB '索引位置計算
            Dim c As Byte = b(i, j)
            rgb(k) = c : rgb(k + 1) = c : rgb(k + 2) = c 'RGB 一致的灰色
            rgb(k + 3) = 255 '實心不透明
        Next
    Next
    UnLockBMP(bmp)
    Return bmp
End Function
'黑白圖
Function BWImg(ByVal b(,) As Byte) As Bitmap
    Dim bmp As New Bitmap(b.GetLength(0), b.GetLength(1))
    LockBMP(bmp)
    For i As Integer = 0 To b.GetLength(0) - 1
        For j As Integer = 0 To b.GetLength(1) - 1
            Dim k As Integer = j * L + i * nB '索引位置計算
            If b(i, j) = 1 Then
                rgb(k) = 0 : rgb(k + 1) = 0 : rgb(k + 2) = 0 '黑色
            Else
                rgb(k) = 255 : rgb(k + 1) = 255 : rgb(k + 2) = 255 '白色
            End If
```

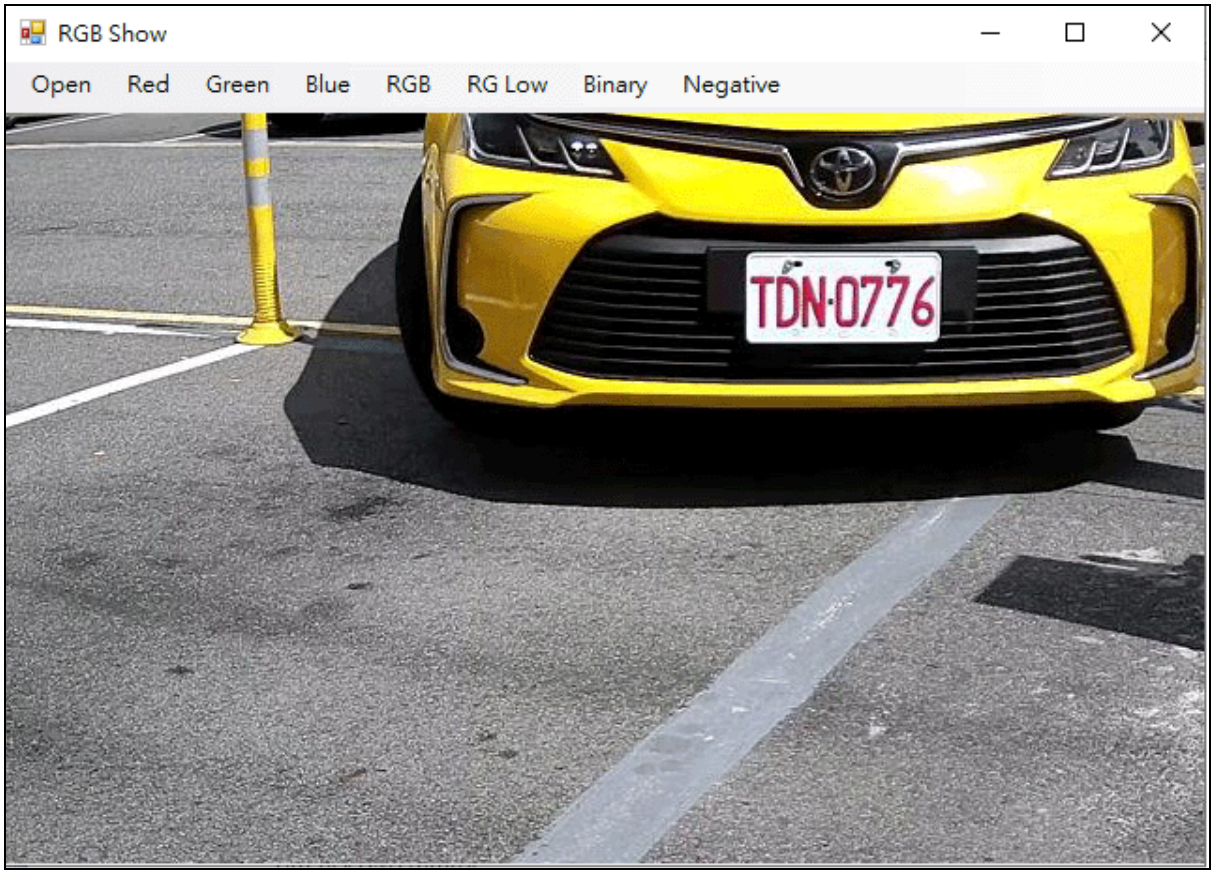
```
                rgb(k + 3) = 255 '實心不透明
            Next
        Next
        UnLockBMP(bmp)
        Return bmp
    End Function
```

2-2 繪製 RGB 代表的灰階資訊



接下來請在主表單上建立一個 MenuStrip1 主功能表，分別製作 Open, Red, Green, Blue, RGB, RG Low, Binary 與 Negative 等幾個按鍵。新增一個 PictureBox1 物件準備顯示影像，建議 SizeMode 屬性設定為 AutoSize，讓影像框可以自行隨影像實際大小縮放，再加入一個 OpenFileDialog1 物件用於選取輸入影像檔案。請在 Open 按鍵中寫程式如下，即可讀取原始影像顯示並將 RGB 資訊分別載入 Rv, Gv 與 Bv 陣列。

```
'開啟檔案
Private Sub OpenToolStripMenuItem_Click(ByVal sender As Object, ByVal e As EventArgs) _
    Handles OpenToolStripMenuItem.Click
    If OpenFileDialog1.ShowDialog = Windows.Forms.DialogResult.OK Then
        Dim bmp As New Bitmap(OpenFileDialog1.FileName)
        Bmp2RGB(bmp) '讀取 RGB 亮度陣列
        PictureBox1.Image = bmp '顯示
    End If
End Sub
```



接著寫入 Red, Green 與 Blue 三個按鍵的程式如下：

```
'以紅光為灰階
Private Sub RedToolStripMenuItem_Click(ByVal sender As Object, ByVal e As EventArgs) _
        Handles RedToolStripMenuItem.Click
        PictureBox1.Image = GrayImg(Rv) '建立灰階圖
End Sub
'以綠光為灰階
Private Sub GreenToolStripMenuItem_Click(ByVal sender As Object, ByVal e As EventArgs) _
        Handles GreenToolStripMenuItem.Click
        PictureBox1.Image = GrayImg(Gv) '建立灰階圖
End Sub
'以藍光為灰階
Private Sub BlueToolStripMenuItem_Click(ByVal sender As Object, ByVal e As EventArgs) _
        Handles BlueToolStripMenuItem.Click
        PictureBox1.Image = GrayImg(Bv) '建立灰階圖
End Sub
```

分別按下這三個按鍵就可以看到以這三種顏色亮度為基礎產生的灰階圖，因為這是一輛計程車的紅字車牌，使用 RGB 三種顏色作為灰階會有很不一樣的結果，依序分別如下：


TDN·0776


TDN·0776

可以明顯看出：以紅光為基礎的灰階會讓字元很不清楚！原因是紅字本身的紅光亮度已經很強，車牌背景的白色其實也是極亮的 RGB 三色光所組成，所以字元與其背景的亮度差異很小，字元就看不清楚了！相對的，取綠光時，紅字目標的綠光極弱就會變成黑色了，藍光狀況相似，都可以讓字元比較清晰。那是不是應該選用綠光或藍光為辨識車牌的基礎呢？請看碰到下面綠字車牌工程車的狀況再說。


455·VR


455·VR

很明顯的，此例中使用紅光的灰階對比會比綠光好很多！即使車牌狀況很糟，至少目視還是可以辨識出車牌，使用綠或藍光則最後一字的 R 幾乎不可能辨識成功！這兩個計程車與工程車的例子可以充分讓我們體會到：如何從 RGB 全彩將顏色資訊簡化到灰階的最佳方式，針對不同的目標，作法可能完全不同！

但是台灣的車牌就是有紅綠這兩種極端的顏色，如果我們不想辨識兩次，能否有兩全其美的灰階選擇方式呢？較簡單的想法是將三原色平均(或組合)作成灰階，這可以避免較極端的狀況，但紅或綠字的整體對比度其實是下降的！但 RGB 的組合演算法不是只有一種，我們也可以**在不同的點選擇使用不同的顏色作成灰階圖**。譬如顏色偏紅的點就刻意選用該點較弱的綠光，顏色偏綠的點就選用該點較暗的紅光，那**不管紅字或綠字都可以變得更黑**了！

在此同時，字元背景的白色，因為紅綠光本來都很強，即使選擇較暗的色光還是會形成很亮的灰階，不會影響字元與背景的對比度太多。這是我們製作台灣車牌辨識軟體獲得的經驗之一，重點是要讓大家了解：我們必須好好研究，如何製作出最佳目標對比度的灰階圖，只要灰階圖針對辨識目標的對比度高，後續的辨識成功率自然會提高。

下面是 RGB 按鍵與 RG Low 按鍵的程式碼，分別代表上述兩種混和式的灰階製作過程：

```
'以 RGB 整合亮度為灰階
Private Sub RGBToolStripMenuItem_Click(ByVal sender As Object, ByVal e As EventArgs) _
    Handles RGBToolStripMenuItem.Click
    Dim A(nx - 1, ny - 1) As Byte
    For i As Integer = 0 To nx - 1
```

```
            For j As Integer = 0 To ny - 1
                Dim Gray As Integer = Rv(i, j) * 0.299 + Gv(i, j) * 0.587 + Bv(i, j) * 0.114
                A(i, j) = Gray
            Next
        Next
        PictureBox1.Image = GrayImg(A) '建立灰階圖
    End Sub
    '選擇紅綠光之較暗亮度為灰階
    Private Sub RGLowToolStripMenuItem_Click(ByVal sender As Object, ByVal e As EventArgs) _
        Handles RGLowToolStripMenuItem.Click
        Dim A(nx - 1, ny - 1) As Byte
        For i As Integer = 0 To nx - 1
            For j As Integer = 0 To ny - 1
                If Rv(i, j) > Gv(i, j) Then
                    A(i, j) = Gv(i, j)
                Else
                    A(i, j) = Rv(i, j)
                End If
            Next
        Next
        PictureBox1.Image = GrayImg(A) '建立灰階圖
    End Sub
```

為何在此將藍光忽略？只針對紅綠光作處理呢？我們可以看到上面程式中的 RGB 轉灰階的運算式中，藍光的比重只有 0.114，對於整體的亮度代表性最低，這是科學實驗出來的結果。在實務上，我們也發現，任何時候用藍光與綠光作處理效果都很接近，當然就不必太在意藍光了，台灣車牌甚至完全沒有用到藍色！我們可以看看這兩種灰階演算法對於計程車與工程車車牌的對比度差異，都是先用平均亮度，後用選色方式：


TDN·0776

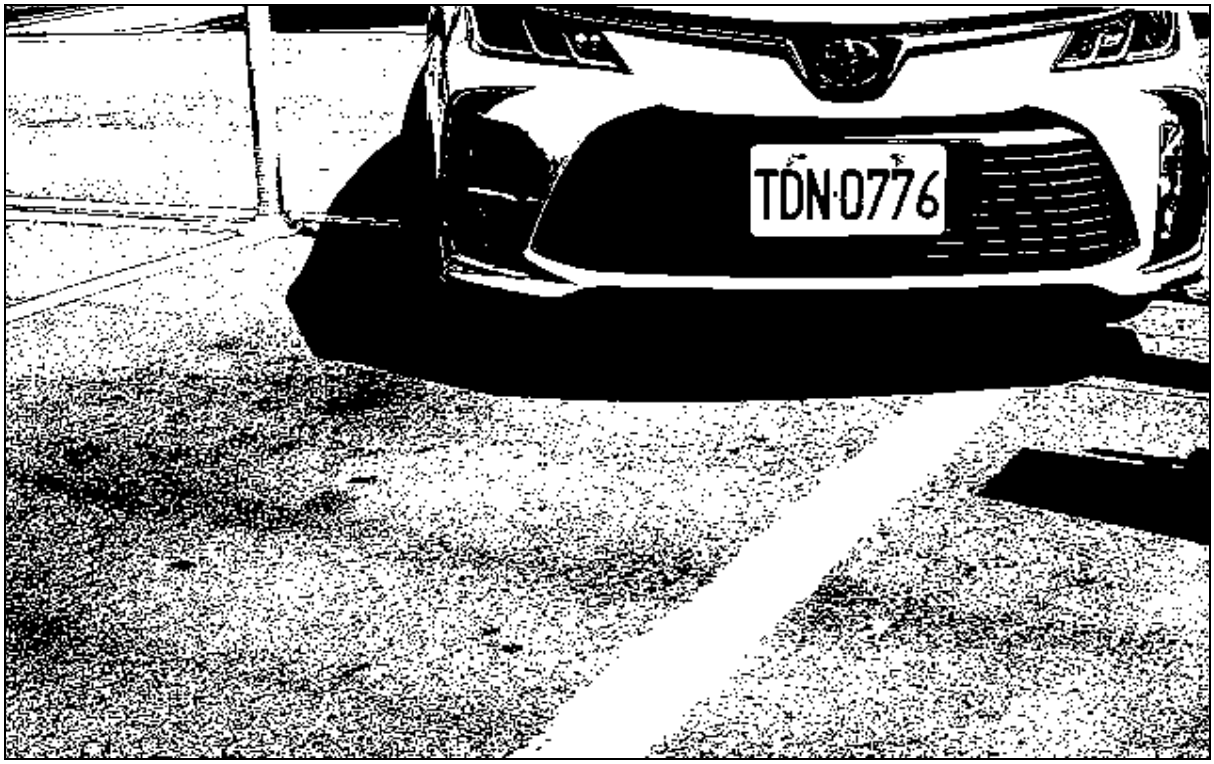

TDN·0776

可以明顯看出，後者增強車牌字元對比的效果好多了！在此也可以知道，當我們希望能增強待辨識目標的對比度時，不是一定要用甚麼銳利化之類的**數位濾波**演算法。事實上，所有影像增強的數位濾波式演算法都會有空間扭曲的副作用，常常導致字元意外變得破碎或沾連，如果可以在灰階產生的方式上著手，應該是更好的**影像增強**選擇。

2-3 繪製二值化圖

下面是 Binary 按鍵的程式碼，所謂二值化就是依據灰階的各點亮度決定他們是「**目標**」或「**背景**」，目標通常畫成純黑色，背景則保持空白。如何決定黑白？是影像辨識過程中的關鍵工作！尤其是目標不清楚的時候！我們會在下一章詳細介紹如何作出較聰明的二值化過程，在此則只是直接將亮度區間在 0-255 之間的綠光以 128 的亮度一分為二而已，多數時候如果車牌黑白分明非常清晰，這也是行得通的！

```
'二值化
Private Sub BinaryToolStripMenuItem_Click(ByVal sender As Object, ByVal e As EventArgs) _
    Handles BinaryToolStripMenuItem.Click
    Dim A(nx - 1, ny - 1) As Byte
    For i As Integer = 1 To nx - 2
        For j As Integer = 1 To ny - 2
            If Gv(i, j) < 128 Then A(i, j) = 1
        Next
    Next
    PictureBox1.Image = BWImg(A) '建立二值化圖
End Sub
```



2-4 繪製負片

大部分的車牌字元背景都是白色，相對來說字元都是較深的顏色，我們的演算法也習慣以黑色當作目標，白色當作背景。但是車牌也有白色字元的狀況，譬如工程車多半是綠底白字，遊覽車是紅底白字，550cc 以上的重機也是紅底白字。碰到這種狀況最簡單的處理方式是做**負片**的辨識，只要翻轉灰階之後就可以繼續使用一般的演算流程做辨識了！下面是 Negative 按鍵的程式碼，直接將綠光以負片方式顯示：

```
'負片
Private Sub NegativeToolStripMenuItem_Click(ByVal sender As Object, ByVal e As EventArgs) _
        Handles NegativeToolStripMenuItem.Click
        Dim A(nx - 1, ny - 1) As Byte
        For i As Integer = 0 To nx - 1
            For j As Integer = 0 To ny - 1
                A(i, j) = 255 - Gv(i, j)
            Next
        Next
        PictureBox1.Image = GrayImg(A) '建立灰階圖
End Sub
```

使用這個功能就可以將如下的工程車車牌做成可以後續處理的灰階影像，車牌字元變成黑色了！

2-5 儲存處理過程的影像

相信作到這裡，你可能會有點想將某些過程圖放大來研究，或者將不同的過程圖並排放在一起做比較，我們的範例程式當然無法也不必作到像影像處理軟體一樣完備，只要處理過程的圖可以輸出為檔案就好了！你可以替專案加入一個 ContextMenuStrip1 功能表，將他附加於 PictureBox1 物件，寫入一個 Save Image 的功能，程式碼如下：

```
'儲存影像
Private Sub SaveImageToolStripMenuItem_Click(ByVal sender As Object, ByVal e As EventArgs) _
        Handles SaveImageToolStripMenuItem.Click
        If SaveFileDialog1.ShowDialog = Windows.Forms.DialogResult.OK Then
            PictureBox1.Image.Save(SaveFileDialog1.FileName)
        End If
End Sub
```

以後要任何時候顯示的畫面，只要按滑鼠右鍵即可，執行畫面如下：

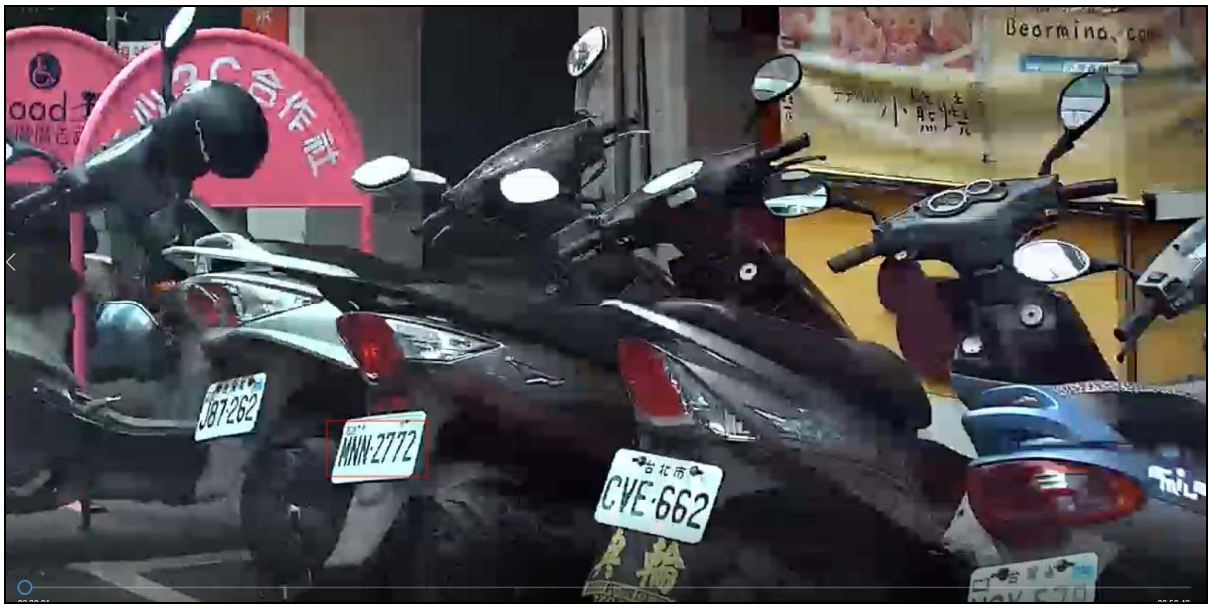
2-6 車牌辨識不是你想的那麼簡單

一個商業化的車牌辨識核心，必須能辨識所有種類的車牌，所以不可避免的，我們必須對正負片都做辨識，才能不遺漏白字的車牌，這是一個潛在的沉重負擔，會讓辨識時間拉長為幾乎兩倍！一般的作法如果只是單一車牌的辨識，就先作正片辨識，如果已有可靠的車牌答案，就不做負片辨識了！但如果是一張影像上可能有多個車牌，且要求全部都要辨識出來，那就不能如此了！

車牌辨識的技術層次差異是很大的！一般停車場內只需要在近距離做理想角度的單車牌辨識，但是在道路情境的街景中，要辨識到所有可見的車牌，難度就很高了！不僅車牌遠近、傾斜與側視角度不一，還包括可能有正負片的車牌。可以想見這種全景辨識需要的運算量與設計難度一定遠高於停車場情境的單車牌辨識。如果還是移動式的全景辨識，譬如裝在車上掃描路邊停車的車載車牌辨識，還需要極高的辨識速度，那就更困難了！

因此，大家應該跳脫傳統上對於車牌辨識的簡單認知，目前車牌辨識研究的挑戰已經進入如下圖的這些困難狀況，這也是本公司目前核心產品的研究重點。在本書中我們無法深入介紹太多這類進階的技術，但書中介紹的每一步驟，確實都是本公司所有車牌辨識產品的基本動作，值得大家深入閱讀參考。

路邊燈桿角度拍攝的影像



車內攝影機掃描路邊停車的影像

加油站監視器影像

完整專案：

```
'Fast RGB Processing Module
Module FastPixel
    Public nx As Integer, ny As Integer, Rv(,) As Byte, Gv(,) As Byte, Bv(,) As Byte
    Dim rgb() As Byte '影像的可存取副本資料陣列
    Dim D As System.Drawing.Imaging.BitmapData '影像資料
    Dim ptr As IntPtr '影像資料所在的記憶體指標(位置)
    Dim n As Integer '影像總共佔據的位元組數
    Dim L As Integer '一個影像列的記憶體位元組數
    Dim nB As Integer '每一像素點是幾個位元組？通常為 3(24bits)或 4(32bits)
    '鎖定點陣圖(Bitmap)物件的記憶體位置，建立一個可操作的為元組陣列副本
    Sub LockBMP(ByVal bmp As Bitmap)
        Dim rect As New Rectangle(0, 0, bmp.Width, bmp.Height) '矩形物件，定義影像範圍
        '鎖定影像區記憶體(暫時不接受作業系統的移動)
        D = bmp.LockBits(rect, Drawing.Imaging.ImageLockMode.ReadWrite, bmp.PixelFormat)
        ptr = D.Scan0 '影像區塊的記憶體指標
        L = D.Stride '每一影像列的長度(bytes)
        nB = L \ bmp.Width '每一像素的位元組數(3 或 4)
        n = L * bmp.Height '影像總位元組數
        ReDim rgb(n - 1) '宣告影像副本資料陣列
        System.Runtime.InteropServices.Marshal.Copy(ptr, rgb, 0, n) '拷貝點陣圖資料到副本陣列
    End Sub
    '複製位元組陣列副本的處理結果到 Bitmap 物件，並解除其記憶體鎖定
    Sub UnLockBMP(ByVal bmp As Bitmap)
        System.Runtime.InteropServices.Marshal.Copy(rgb, 0, ptr, n) '拷貝副本陣列資料到點陣圖
        bmp.UnlockBits(D) '解除鎖定
    End Sub
    '取得 RGB 陣列
    Public Sub Bmp2RGB(ByVal bmp As Bitmap)
        nx = bmp.Width : ny = bmp.Height
        ReDim Rv(nx - 1, ny - 1), Gv(nx - 1, ny - 1), Bv(nx - 1, ny - 1) '輸出用灰階陣列
        LockBMP(bmp)
        For j As Integer = 0 To ny - 1
            Dim Lj As Integer = j * D.Stride
            For i As Integer = 0 To nx - 1
                Dim k As Integer = Lj + i * nB
                Rv(i, j) = rgb(k + 2) '紅光
                Gv(i, j) = rgb(k + 1) '綠光
                Bv(i, j) = rgb(k) '藍光
            Next
        Next
        bmp.UnlockBits(D) '解除鎖定
    End Sub
    '灰階圖
    Function GrayImg(ByVal b(,) As Byte) As Bitmap
        Dim bmp As New Bitmap(b.GetLength(0), b.GetLength(1))
        LockBMP(bmp)
        For i As Integer = 0 To b.GetLength(0) - 1
            For j As Integer = 0 To b.GetLength(1) - 1
                Dim k As Integer = j * L + i * nB '索引位置計算
                Dim c As Byte = b(i, j)
                rgb(k) = c : rgb(k + 1) = c : rgb(k + 2) = c 'RGB 一致的灰色
                rgb(k + 3) = 255 '實心不透明
            Next
        Next
        UnLockBMP(bmp)
```

```
        Return bmp
    End Function
    '黑白圖
    Function BWImg(ByVal b(,) As Byte) As Bitmap
        Dim bmp As New Bitmap(b.GetLength(0), b.GetLength(1))
        LockBMP(bmp)
        For i As Integer = 0 To b.GetLength(0) - 1
            For j As Integer = 0 To b.GetLength(1) - 1
                Dim k As Integer = j * L + i * nB '索引位置計算
                If b(i, j) = 1 Then
                    rgb(k) = 0 : rgb(k + 1) = 0 : rgb(k + 2) = 0 '黑色
                Else
                    rgb(k) = 255 : rgb(k + 1) = 255 : rgb(k + 2) = 255 '白色
                End If
                rgb(k + 3) = 255 '實心不透明
            Next
        Next
        UnLockBMP(bmp)
        Return bmp
    End Function
End Module

Public Class Form1
    '開啟檔案
    Private Sub OpenToolStripMenuItem_Click(ByVal sender As Object, ByVal e As EventArgs) _
        Handles OpenToolStripMenuItem.Click
        If OpenFileDialog1.ShowDialog = Windows.Forms.DialogResult.OK Then
            Dim bmp As New Bitmap(OpenFileDialog1.FileName)
            Bmp2RGB(bmp) '讀取 RGB 亮度陣列
            PictureBox1.Image = bmp '顯示
        End If
    End Sub
    '以紅光為灰階
    Private Sub RedToolStripMenuItem_Click(ByVal sender As Object, ByVal e As EventArgs) _
        Handles RedToolStripMenuItem.Click
        PictureBox1.Image = GrayImg(Rv) '建立灰階圖
    End Sub
    '以綠光為灰階
    Private Sub GreenToolStripMenuItem_Click(ByVal sender As Object, ByVal e As EventArgs) _
        Handles GreenToolStripMenuItem.Click
        PictureBox1.Image = GrayImg(Gv) '建立灰階圖
    End Sub
    '以藍光為灰階
    Private Sub BlueToolStripMenuItem_Click(ByVal sender As Object, ByVal e As EventArgs) _
        Handles BlueToolStripMenuItem.Click
        PictureBox1.Image = GrayImg(Bv) '建立灰階圖
    End Sub
    '以 RGB 整合亮度為灰階
    Private Sub RGBToolStripMenuItem_Click(ByVal sender As Object, ByVal e As EventArgs) _
        Handles RGBToolStripMenuItem.Click
        Dim A(nx - 1, ny - 1) As Byte
        For i As Integer = 0 To nx - 1
            For j As Integer = 0 To ny - 1
                Dim Gray As Integer = Rv(i, j) * 0.299 + Gv(i, j) * 0.587 + Bv(i, j) * 0.114
                A(i, j) = Gray
            Next
        Next
        PictureBox1.Image = GrayImg(A) '建立灰階圖
```

```
    End Sub
    '選擇紅綠光之較暗亮度為灰階
    Private Sub RGLowToolStripMenuItem_Click(ByVal sender As Object, ByVal e As EventArgs) _
        Handles RGLowToolStripMenuItem.Click
        Dim A(nx - 1, ny - 1) As Byte
        For i As Integer = 0 To nx - 1
            For j As Integer = 0 To ny - 1
                If Rv(i, j) > Gv(i, j) Then
                    A(i, j) = Gv(i, j)
                Else
                    A(i, j) = Rv(i, j)
                End If
            Next
        Next
        PictureBox1.Image = GrayImg(A) '建立灰階圖
    End Sub
    '二值化
    Private Sub BinaryToolStripMenuItem_Click(ByVal sender As Object, ByVal e As EventArgs) _
        Handles BinaryToolStripMenuItem.Click
        Dim A(nx - 1, ny - 1) As Byte
        For i As Integer = 1 To nx - 2
            For j As Integer = 1 To ny - 2
                If Gv(i, j) < 128 Then A(i, j) = 1
            Next
        Next
        PictureBox1.Image = BWImg(A) '建立二值化圖
    End Sub
    '負片
    Private Sub NegativeToolStripMenuItem_Click(ByVal sender As Object, ByVal e As EventArgs) _
        Handles NegativeToolStripMenuItem.Click
        Dim A(nx - 1, ny - 1) As Byte
        For i As Integer = 0 To nx - 1
            For j As Integer = 0 To ny - 1
                A(i, j) = 255 - Gv(i, j)
            Next
        Next
        PictureBox1.Image = GrayImg(A) '建立灰階圖
    End Sub
    '儲存影像
    Private Sub SaveImageToolStripMenuItem_Click(ByVal sender As Object, ByVal e As EventArgs) _
        Handles SaveImageToolStripMenuItem.Click
        If SaveFileDialog1.ShowDialog = Windows.Forms.DialogResult.OK Then
            PictureBox1.Image.Save(SaveFileDialog1.FileName)
        End If
    End Sub
End Class
```