第 13 章 動態影像辨識的基本動作

13-1 如何從動態串流影像中取得可辨識的靜態影像？

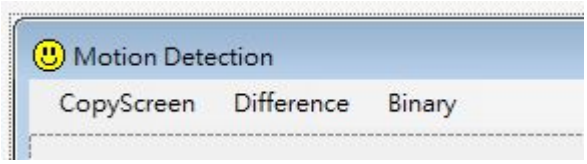
前面介紹的影像辨識程序處理的對象都是單張的靜態影像，但實際上現在多數時候我們都是從攝影機的動態畫面中取得影像作辨識的！譬如我們想從監視器上辨識一輛經過的車牌，就必須取得動態影像中的某些定格畫面作辨識！

現在的 IPCAM 攝影機架構上就是一個有 IP 的影像串流網站，理論上我們可以使用攝影機的 SDK 指令要求攝影機回傳單張即時影像，但是很多攝影機廠商會刻意隱藏這個指令，可能是要藉此強迫使用者購買他們搭配的軟體吧？

我們也可以使用可從網路免費取得的工具程式，將傳入電腦的串流資料進行解碼成為單張影像，但如果這麼作會消耗大量的 CPU 運算資源，讓也需要大量運算的影像辨識工作資源匱乏！所以要取得攝影機畫面的影像，最理想的方式就是拷貝螢幕了！原因是影像串流解碼的工作預設會由電腦的顯示卡執行，那樣就不會占用 CPU 了！

本章的主要內容就是要教大家如何製作程式介面，指定螢幕上的某個區塊，作連續擷取影像的動作。既然可以連續取像了，就順勢介紹動態偵測的概念，讓前後兩張影像作亮度差異的計算，沒變化的背景就會接近空白，移動中的物體則會有明顯的亮度擾動，就會被凸顯出來了！你不需要特殊的設備或軟體，也可以作出動態追蹤的程式！

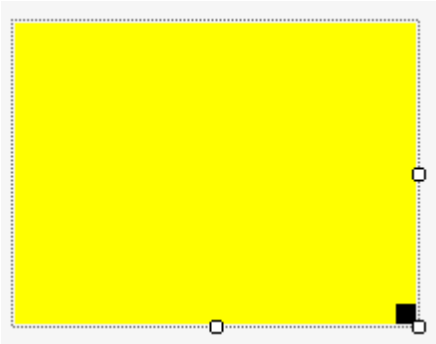
請模仿之前專案，建立 Form1 表單主功能表如下，包含：CopyScreen, GrayDifference 與 Binay 三個按鍵，並加入一個 PictueBox1，SizeMode 屬性設為 Zoom 讓擷取之影像不論大小都會完整顯示於此，最後加入一個 Timer1，Interval 屬性使用預設的 100 毫秒。



13-2 製作出一個半透明的辨識區域設定框

相信多數人都用過螢幕錄影程式，它們一定有一個可以自由調整錄影範圍的框框介面，在此我們也有一樣的需要。事實上它是一個特殊設計的表單，請先在專案中加入一個 Form2，將 FormBorderStyle 屬性設為 None，Opacity(不透明度)屬性從 100%改成 60%，然後將 BackColorg 設為鮮豔的黃色，這樣就可以做出一個無邊框且半透明的表單了！因為這個表單被使用時不能被其他視窗遮住，所以 TopMost 屬性要設定為 True！

接著請加入一個 Label1 物件，將 Text 內容刪除，AutoSize 屬性改成 False，寬高都改成 10， BackColor 設為黑色，將它放在表單的右下角，就可以作出類似小畫家用來調整影像大小的一個小黑方塊介面了！設計階段的外觀如下：



目前看起來表單還不是空心的，這無法在表單設計階段完成，必須在表單載入時使用特殊的程式去挖空它。請先在表單程式碼葉面的最上方，匯入需要使用的命名空間 System.Drawing.Drawing2D，然後建立 Form_Load 以及 ClipWindow 副程式，程式碼如下：

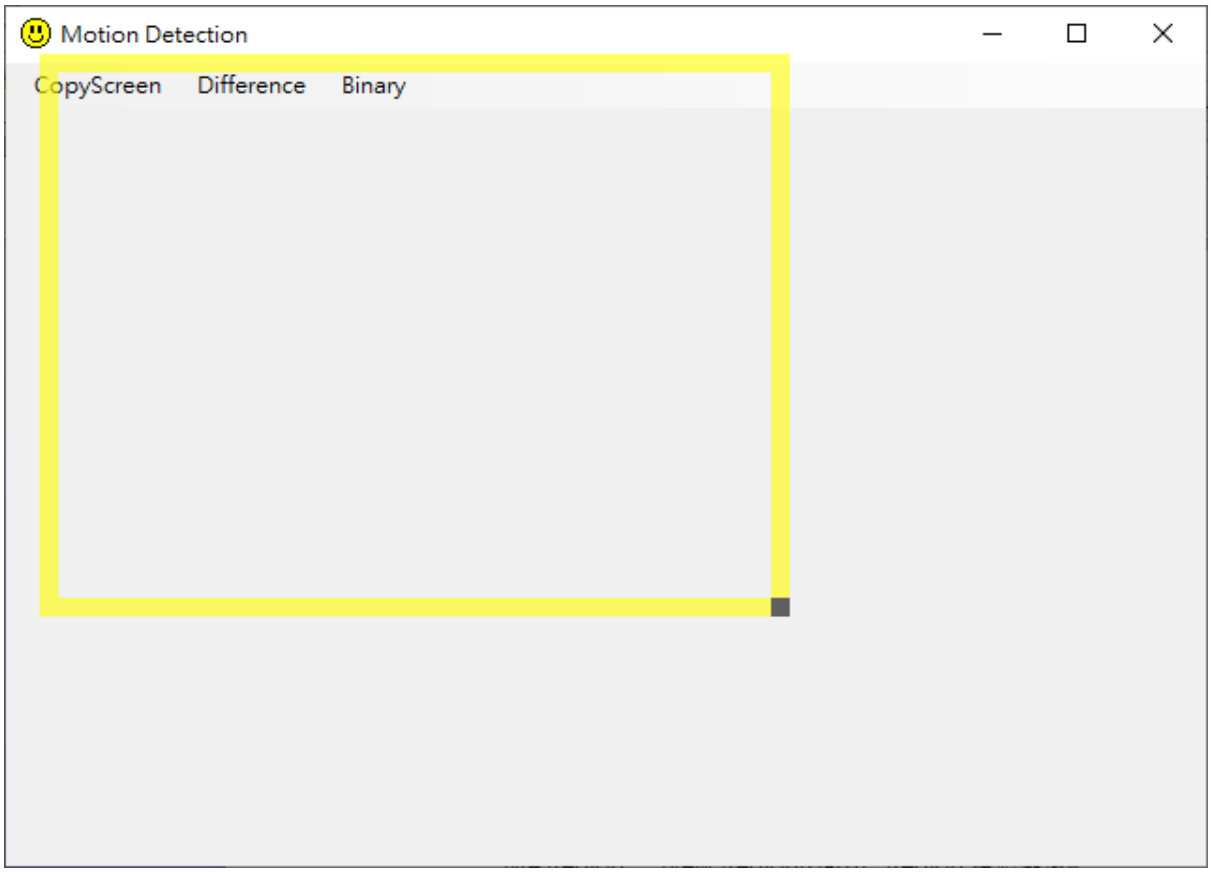
```
Imports System.Drawing.Drawing2D '匯入 2D 繪圖模組
Public Class Form2
    '開啟視窗，建立一個鏤空半透明外觀的視窗
    Private Sub Form2_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        ClipWindow() '鏤空視窗
    End Sub
    '鏤空視窗
    Private Sub ClipWindow()
        '影像辨識區透空處理
        Dim path As New GraphicsPath
        Dim pt(9) As Point
        pt(0).X = 0 : pt(0).Y = 0 '左上角
```

```
        pt(1).X = Me.Width : pt(1).Y = 0 '右上角
        pt(2).X = Me.Width : pt(2).Y = Me.Height '右下角
        pt(3).X = 0 : pt(3).Y = Me.Height '左下角
        pt(4).X = 0 : pt(4).Y = Me.Height - 10
        pt(5).X = Me.Width - 10 : pt(5).Y = Me.Height - 10
        pt(6).X = Me.Width - 10 : pt(6).Y = 10
        pt(7).X = 10 : pt(7).Y = 10
        pt(8).X = 10 : pt(8).Y = Me.Height - 10
        pt(9).X = 0 : pt(9).Y = Me.Height - 10
        path.AddPolygon(pt) '以多邊形的方式加入 path
        Me.Region = New Region(path) 'Region 視窗區域
    End Sub
```

它的邏輯是用一個複雜的多邊形定義出一個「**需要顯示出來**」的範圍！就是一個空心的框框。在此範圍以外的表單在繪製表單時就變成完全透明的了！要看到它執行的效果，請先返回 Form1 表單，在 Form_Load 事件中開啟一個 Form2，程式碼如下：

```
Dim F2 As New Form2 '監看範圍設定框
'啟動程式顯示監看範圍設定框
Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
    F2.Show()
End Sub
```

執行後可以看到如下畫面：



此時這個框框和那個黑方塊都還無法被拖曳改變識窗位置與大小，請在 Form2 程式碼中依序加入 Form2 與 Label1 的 MouseDown 與 MouseMove 事件程式碼如下：

```
'拖曳視窗功能，定義視窗位置
Dim mdp As Point
Private Sub Form2_MouseDown(sender As Object, e As MouseEventArgs) _
    Handles Me.MouseDown
    mdp = e.Location
End Sub
Private Sub Form2_MouseMove(sender As Object, e As MouseEventArgs) _
    Handles Me.MouseMove
    If e.Button = Windows.Forms.MouseButtons.Left Then
        Me.Location += e.Location - mdp
    End If
End Sub
'拖曳右下角功能，定義視窗寬高大小
Private Sub Label1_MouseDown(sender As Object, e As MouseEventArgs) _
    Handles Label1.MouseDown
    mdp = e.Location
End Sub
Private Sub Label1_MouseMove(sender As Object, e As MouseEventArgs) _
    Handles Label1.MouseMove
    If e.Button = MouseButtons.Left Then
        Label1.Location += e.Location - mdp
        Me.Width = Label1.Right : Me.Height = Label1.Bottom
        Me.Refresh()
        ClipWindow()
    End If
End Sub
```

這樣再度執行程式時，用滑鼠按住黃框框的任何部分都可以拖曳整個視窗了！拖曳黑方塊時則只有右邊框與下邊框會跟著移動，這樣就可以達到讓視窗大小改變的目的了！

13-3 拷貝螢幕的程式

回到 Form1 請先將 CopyScreen 按鍵與 Timer1 的預設 Tick 事件內容寫成這樣：

```
'啟動監看
Private Sub CopyScreenToolStripMenuItem_Click(sender As Object, e As EventArgs) _
    Handles CopyScreentToolStripMenuItem.Click
    F2.Hide() '隱藏設定框
    Timer1.Start() '啟動螢幕拷貝
End Sub
'動態監看迴圈
Private Sub Timer1_Tick(sender As Object, e As EventArgs) Handles Timer1.Tick
    '拷貝螢幕指定範圍
```

```
    Dim bmp As New Bitmap(F2.Width, F2.Height)
    Dim G As Graphics = Graphics.FromImage(bmp)
    G.CopyFromScreen(F2.Location, New Point(0, 0), F2.Size)
    PictureBox1.Image = bmp '原圖顯示
End Sub
```

這段程式的意義是：先建立一個與 Form2(F2)一樣大的影像物件 bmp，再建立此影像的繪圖物件 G，然後用 G 執行拷貝螢幕畫面到 bmp 的動作。因為是反覆執行的 Timer，這樣就可以用每 100 毫秒(0.1 秒)的週期持續複製黃框框指定範圍的影像縮放到 PictueBox1 了！各位可以用任何會動的影片，譬如 YouTube 的視訊作測試。

一個有趣的事實是：其實我們拷貝的資料並不在「**螢幕**」裡面，而是在**顯示卡**的某個記憶體區塊。所以如果你將螢幕電源關掉，甚至訊號線也拔除，程式還是會繼續拷貝「**螢幕影像**」的！不相信你可以寫存檔程式檢驗我的說法！但是如果讓螢幕進入「休眠」，作業系統就會暫停顯示卡的作用，你就拷貝不到畫面了！

13-4 動態偵測的程式

「**動態**」偵測就是偵測影像亮度何處有變化？也就表示有物體在動，這類功能在安全監控行業當然是很重要的！譬如基本的門窗是否被開關等等。我們如果持續追蹤擾動區的位置，也可以變成目標追蹤的程式！在此我們先將最簡單偵測擾動的基本動作教給大家，概念就是連續兩個影像的灰階亮度差異，所以我們不但要解析當下影像的內容，還要保留前一張影像的亮度資訊作為判斷有無擾動的參考。請先在公用變數區域加上這兩行宣告：

```
Dim B1(,) As Byte, B2(,) As Byte '連續影像的灰階陣列
Dim Type As Integer = 0 '顯示模式：0→原圖，1→灰階，2→二值化
```

B2 代表目前影像的灰階，B1 則是前一張的灰階。我們準備用灰階及二值化的兩種方式顯示擾動狀態，所以定義了 Type 這個參數，0 是直接顯示原圖，1 是顯示灰階差異，2 是顯示差異的二值化圖。這個參數是在 Timer1 顯示輸出影像時產生作用，由主功能表的按鍵，搭配一個副程式加以定義，灰階與二值化的程式碼如下：

```
'灰階差異
Private Sub DifferenceToolStripMenuItem_Click(sender As Object, e As EventArgs) _
    Handles DifferenceToolStripMenuItem.Click
    Type = 1
End Sub
'灰階動態圖
Private Function GrayDiff() As Bitmap
    Dim D(nx - 1, ny - 1) As Byte
    For i As Integer = 0 To nx - 1
        For j As Integer = 0 To ny - 1
            D(i, j) = 255 - Math.Abs(CInt(B1(i, j)) - B2(i, j))   '負片顯示
        Next
```

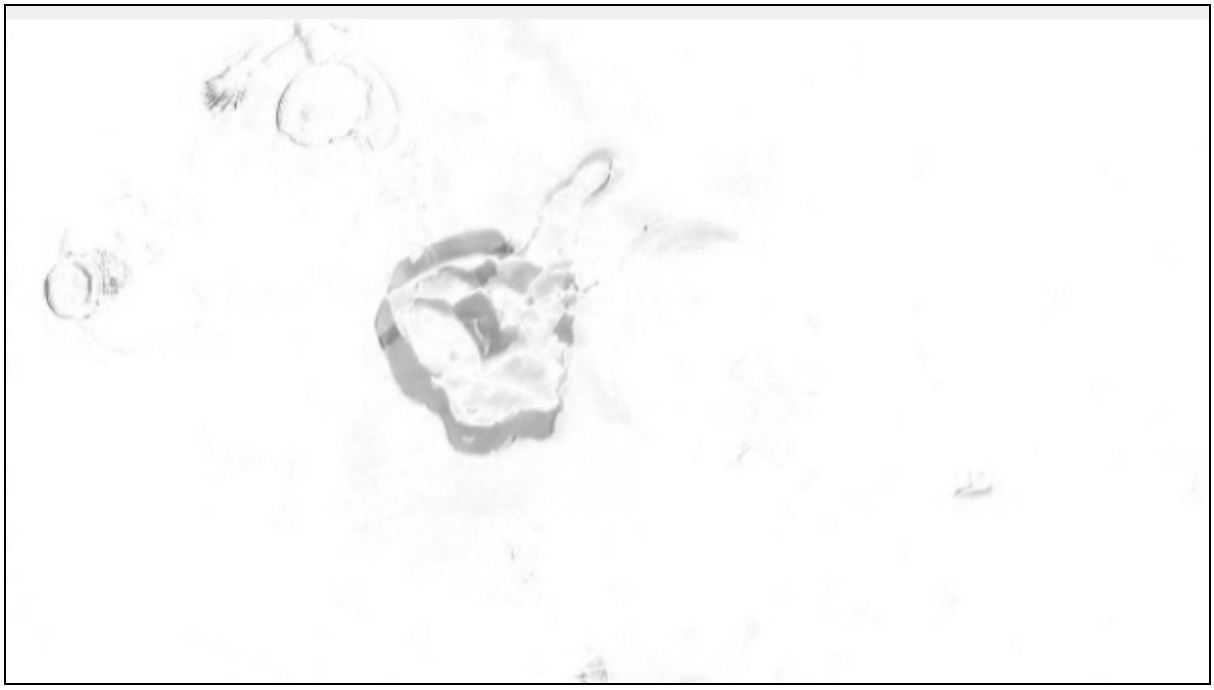
```
        Next
        Return GrayImg(D)
    End Function

    '高差異點二值化
    Private Sub BinaryToolStripMenuItem_Click(sender As Object, e As EventArgs) _
        Handles BinaryToolStripMenuItem.Click
        Type = 2
    End Sub
    '二值化動態圖
    Private Function BinDiff() As Bitmap
        Dim Th As Integer = 10
        Dim D(nx - 1, ny - 1) As Byte
        For i As Integer = 0 To nx - 1
            For j As Integer = 0 To ny - 1
                Dim dd As Integer = Math.Abs(CInt(B1(i, j)) - B2(i, j))
                If dd > Th Then D(i, j) = 1
            Next
        Next
        Return BWImg(D)
    End Function
```

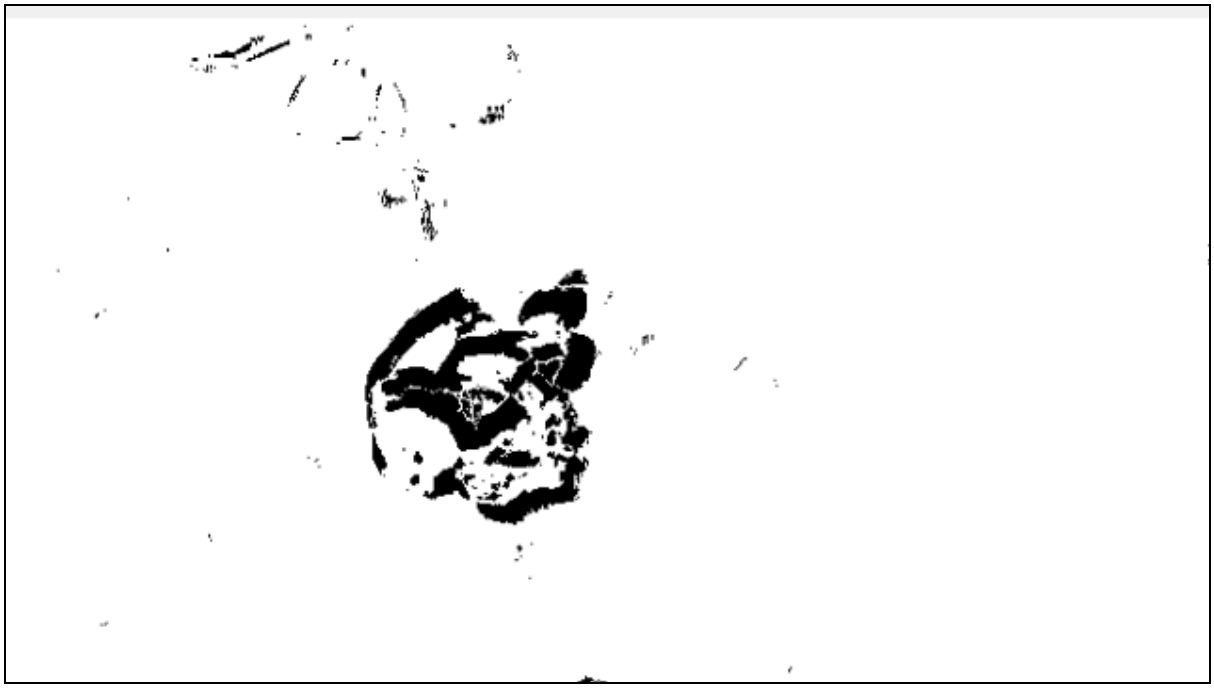
程式的意義很簡單，在灰階顯示時是將 B1 與 B2 的亮度差異絕對值當新的資料繪製灰階圖，預期亮度不變的背景會趨近於零，我們將它用負片顯示，0 值就會呈現白色，擾動區則顯示深淺不一的灰色。二值化只是定義一個亮度差異的門檻值，低於門檻表示擾動幅度小加以忽略，大於門檻則是確定為移動的目標。接下來我們還必須修改 Timer1 的程式如下：

```
    Private Sub Timer1_Tick(sender As Object, e As EventArgs) Handles Timer1.Tick
        '拷貝螢幕指定範圍
        Dim bmp As New Bitmap(F2.Width, F2.Height)
        Dim G As Graphics = Graphics.FromImage(bmp)
        G.CopyFromScreen(F2.Location, New Point(0, 0), F2.Size)
        Bmp2RGB(bmp) '取得影像陣列
        If IsNothing(B1) Then
            B1 = Gv : B2 = Gv '首張影像
        Else
            B1 = B2.Clone : B2 = Gv.Clone    '舊影像推移
        End If
        Select Case Type
            Case 0
                PictureBox1.Image = bmp '原圖顯示
            Case 1
                PictureBox1.Image = GrayDiff() '差異灰階顯示
            Case 2
                PictureBox1.Image = BinDiff() '差異二值化顯示
```

```
        End Select
    End Sub
```

重點在於每張新的影像要抽取 RGB 的資訊，以 Gv 綠光為灰階，B2 資料還要先複製到 B1 陣列，再用 B2 承接新的資料。以本章開頭的影片畫面來測試，有人走過的同一事件發生時，原圖、灰階與二值化圖依序大概會是這個樣子：





13-5 後記

本章最重要的意義是讓讀者知道如何自由取得螢幕畫面上的影像，所有的「監視」程式都是以此為始。但是老實說，個人經營公司承接各種影像辨識專案，最戒慎恐懼的就是動態目標追蹤的程式！原因是差異影像的本質是非常不穩定的，譬如監看有無小偷翻牆的程式，如果碰到旁邊有樹木隨風搖擺就會是很難克服的雜訊。

就算是室內固定攝影機的影像，也常會因為攝影機本身內部自動調整焦距或亮度時產生的變化讓差異影像出現高雜訊，這些因素來自硬體，我們在軟體層面及難掌握，安全監控的責任也很重大，不能漏接真實闖入事件，又不能誤報太頻繁，所以看似「簡單」的電子圍籬軟體其實很不簡單！

在此又要再次提醒過度相信機器學習或深度學習會有神效的讀者，這些影響影像辨識的誤差雜訊都是來自不同的環境因素，包括攝影機本身的特性在內，如果你依賴機率統計降低誤差，提高辨識率是很危險的作法！因為一換環境，甚至季節的更迭，攝影機焦距改變都會讓之前的「學習成果」大幅偏離現狀。最合理的研發方向，還是要分析觀察每一種誤差雜訊的成因與特性，作好針對性的處理，影像辨識軟體要穩定可靠，絕對沒有簡單的公式或 SOP 可以套用。

完整專案

```
Public Class Form1
    Dim B1(,) As Byte, B2(,) As Byte '連續影像的灰階陣列
    Dim Type As Integer = 0 '顯示模式：0→原圖，1→灰階，2→二值化
    Dim F2 As New Form2 '監看範圍設定框
    '啟動程式顯示監看範圍設定框
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        F2.Show()
    End Sub
    '啟動監看
    Private Sub CopyScreenToolStripMenuItem_Click(sender As Object, e As EventArgs) _
        Handles CopyScreentToolStripMenuItem.Click
        F2.Hide() '隱藏設定框
        Timer1.Start() '啟動螢幕拷貝
    End Sub
    '動態監看迴圈
    Private Sub Timer1_Tick(sender As Object, e As EventArgs) Handles Timer1.Tick
        '拷貝螢幕指定範圍
        Dim bmp As New Bitmap(F2.Width, F2.Height)
        Dim G As Graphics = Graphics.FromImage(bmp)
        G.CopyFromScreen(F2.Location, New Point(0, 0), F2.Size)
        Bmp2RGB(bmp) '取得影像陣列
        If IsNothing(B1) Then
            B1 = Gv : B2 = Gv '首張影像
        Else
            B1 = B2.Clone : B2 = Gv.Clone   '舊影像推移
        End If
        Select Case Type
            Case 0
                PictureBox1.Image = bmp'原圖顯示
            Case 1
                PictureBox1.Image = GrayDiff()'差異灰階顯示
            Case 2
                PictureBox1.Image = BinDiff() '差異二值化顯示
        End Select
    End Sub
    '灰階差異
    Private Sub DifferenceToolStripMenuItem_Click(sender As Object, e As EventArgs) _
        Handles DifferenceToolStripMenuItem.Click
        Type = 1
    End Sub
    '灰階動態圖
    Private Function GrayDiff() As Bitmap
        Dim D(nx - 1, ny - 1) As Byte
        For i As Integer = 0 To nx - 1
```

```
                For j As Integer = 0 To ny - 1
                    D(i, j) = 255 - Math.Abs(CInt(B1(i, j)) - B2(i, j))
                Next
            Next
            Return GrayImg(D)
        End Function
        '高差異點二值化
        Private Sub BinaryToolStripMenuItem_Click(sender As Object, e As EventArgs) _
            Handles BinaryToolStripMenuItem.Click
            Type = 2
        End Sub
        '二值化動態圖
        Private Function BinDiff() As Bitmap
            Dim Th As Integer = 10
            Dim D(nx - 1, ny - 1) As Byte
            For i As Integer = 0 To nx - 1
                For j As Integer = 0 To ny - 1
                    Dim dd As Integer = Math.Abs(CInt(B1(i, j)) - B2(i, j))
                    If dd > Th Then D(i, j) = 1
                Next
            Next
            Return BWImg(D)
        End Function
End Class

Imports System.Drawing.Drawing2D '匯入 2D 繪圖模組
Public Class Form2
    '開啟視窗，建立一個鏤空半透明外觀的視窗
    Private Sub Form2_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        ClipWindow() '鏤空視窗
    End Sub
    '鏤空視窗
    Private Sub ClipWindow()
        '影像辨識區透空處理
        Dim path As New GraphicsPath
        Dim pt(9) As Point
        pt(0).X = 0 : pt(0).Y = 0 '左上角
        pt(1).X = Me.Width : pt(1).Y = 0 '右上角
        pt(2).X = Me.Width : pt(2).Y = Me.Height '右下角
        pt(3).X = 0 : pt(3).Y = Me.Height '左下角
        pt(4).X = 0 : pt(4).Y = Me.Height - 10
        pt(5).X = Me.Width - 10 : pt(5).Y = Me.Height - 10
        pt(6).X = Me.Width - 10 : pt(6).Y = 10
        pt(7).X = 10 : pt(7).Y = 10
        pt(8).X = 10 : pt(8).Y = Me.Height - 10
        pt(9).X = 0 : pt(9).Y = Me.Height - 10
```

```
        path.AddPolygon(pt) '以多邊形的方式加入 path
        Me.Region = New Region(path) 'Region 視窗區域
    End Sub
    '拖曳視窗功能，定義視窗位置
    Dim mdp As Point
    Private Sub Form2_MouseDown(sender As Object, e As MouseEventArgs) _
        Handles Me.MouseDown
        mdp = e.Location
    End Sub
    Private Sub Form2_MouseMove(sender As Object, e As MouseEventArgs) _
        Handles Me.MouseMove
        If e.Button = Windows.Forms.MouseButtons.Left Then
            Me.Location += e.Location - mdp
        End If
    End Sub
    '拖曳右下角功能，定義視窗寬高大小
    Private Sub Label1_MouseDown(sender As Object, e As MouseEventArgs) _
        Handles Label1.MouseDown
        mdp = e.Location
    End Sub
    Private Sub Label1_MouseMove(sender As Object, e As MouseEventArgs) _
        Handles Label1.MouseMove
        If e.Button = MouseButtons.Left Then
            Label1.Location += e.Location - mdp
            Me.Width = Label1.Right : Me.Height = Label1.Bottom
            Me.Refresh()
            ClipWindow()
        End If
    End Sub
End Class
```