臺灣二 OO 四年國際科學展覽會

科　　別：電腦科學科

作品名稱：讓視域更遼闊
--在有限的螢幕空間上顯示更多的圖形式資訊

得獎獎項：電腦科學科第二名
英特爾電腦科學獎第一名
候補作品

學　　校：國立新竹女子高級中學

作　　者：龔怡嘉、劉育珍

英文摘要(Abstract)：

To Broaden Your Vision
How to Provide More Graphic Information on a Limited Screen

Browsing the global structure of a large graph in limited screen space has the drawback that details are often too small to be seen. The most common solution provides a scrollable view. This shows full details at the region currently visible through the view, but hides the rest of the global structure. Alternatively, zooming into a part of the graph does show local details but misses the overall structure of the graph. The multiple views approach, one view of the entire graph and the other of a zoomed portion, has the advantage of seeing both local details and overall structure, but has the drawback that parts of the graph adjacent to the enlarged area are not visible at all in the enlarged views.

A fisheye camera lens is a very wide angle lens that magnifies nearby objects while shrinking distant objects. It seems to be a tool for seeing local detail and global structure simultaneously. By means of exploring fisheye camera lens, we develop a user interface for browsing graphs using program analog of fisheye lens. Thus, our method seems to have all the advantages of the other approaches without suffering from any of the drawbacks.

中文摘要：

讓視域更遼闊

在有限的螢幕空間上顯示更多的圖形式資訊

在利用電腦螢幕來瀏覽圖形式資訊的時候，常常受限於螢幕的空間，沒有辦法在顯示資訊整體結構的同時顯現細節部分的資料，目前的使用者介面所採用的方法有放大（zoom in）、捲動（scrolling）、開啓多個視窗（multiple view）等方法，這些方法雖然可以呈現出資料的細節部分，但是仍有其個別的缺點存在，放大的方式會有遮蔽的情形；捲動的方式無法同時地呈現整體結構；開啓多個視窗的方法使得使用者的眼睛必須在這些視窗間來回的移動，造成麻煩。

魚眼鏡頭是一種短焦距、大視角的相機鏡頭，鏡頭成像的時候，越接近鏡頭中心的物體會越放大而越遠的部分會越縮小，藉著發掘魚眼鏡頭的成像函數，我們發展出了一種新的使用者介面，在瀏覽圖形式資訊的時候，能夠顯示整體的結構，並隨著滑鼠游標的移動，以不開啓新視窗及無遮蔽的方式，即時地將想要觀察的部分局部放大以展現細部的資料，這種使用者介面將具備現有方法的優點而無其缺點。

一、研究動機與目的：

在利用電腦螢幕來瀏覽圖形式資訊的時候，常常受限於螢幕的空間，沒有辦法在顯示資訊整體結構的同時顯現細節部分的資料，目前的使用者介面所採用的方法有放大（zoom in）、捲動（scrolling）、開啓多個視窗等方法，這些方法雖然可以呈現出資料的細節部分，但是仍有其個別的缺點存在，放大的方式會有遮蔽的情形；捲動的方式無法同時地呈現整體結構；開啓多個視窗的方法使得使用者的眼睛必須在這些視窗間來回的移動，造成麻煩。

魚眼鏡頭是一種短焦距、大視角的相機鏡頭，鏡頭成像的時候，越接近鏡頭中心的物體會越放大而越遠的部分會越縮小，這個特性似乎可以用來發展出一種新的使用者介面，在瀏覽圖形式資訊的時候，能夠顯示整體的結構，並隨著滑鼠游標的移動，以不開啓新視窗及無遮蔽的方式，即時地將想要觀察的部分局部放大以展現細部的資料，這種使用者介面將具備現有方法的優點而無其缺點。

二、研究過程：

1. 相關文獻探討

魚眼鏡頭是一種短焦距、大視角的相機鏡頭，除了在攝影上的使用外，其他如醫學內視鏡、針孔攝影機等也都有用到。Furnas[1]運用魚眼鏡頭的原理，提出作用在文字模式上的魚眼視域（fisheye views）方法，後來，Sakar 與 Brown[2]除了將其方法做了修正外，更進一步地應用在圖形介面的使用上，Sakar 與 Brown 所使用的魚眼視域轉換函數如下，根據原始影像中每一點（像素 pixel）的 x 與 y 座標分別計算出該點新的位置。

$$G(x) = \frac{(d+1)x}{dx+1}$$

$$G(y) = \frac{(d+1)y}{dy+1}$$

由圖 1-1 與圖 1-2 可看出 Sakar 與 Brown 的方法使得離焦點愈近的物件能以較大也較詳盡的方式呈現在螢幕上，而由焦點向外輻射，離焦點愈遠的物件便有變小和變模糊的傾向，雖然圖形的整體結構有顯現出來，但是每選定一個新的焦點，整個影像中的每一點均需重新計算其新的位置，所需的計算量將會十分的驚人。

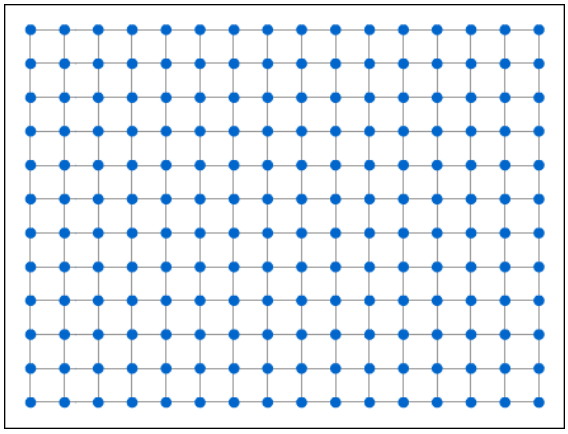

圖 1-1 原始影像

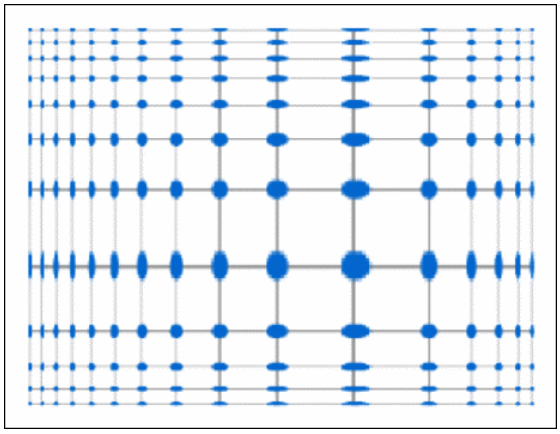

圖 1-2 經過 Sakar 與 Brown 的魚眼視域函數轉換後

2. 研究魚眼鏡頭找尋新的方案

Sakar 與 Brown 的方法將整個圖形式資訊由焦點以矩形的方式分垂直與水平方向往外輻

射，與我們看到的魚眼鏡頭成像圖應是圓形的有很大的差異，所以打算重新的檢視與研究魚眼鏡頭的成像方式，希望能尋找出新的方案。

我們繪製了些如圖 2-1 的網狀圓點，並加上一些輔助線，以圖 2-2 中的魚眼鏡頭拍攝，拍攝後的影像如圖 2-3 所示，爲了方便與原圖作比較，需作適當的調校，圖 2-4 是經過調校後的拍攝影像，影像的中間部分會放大而四周部分會縮小變形，由圖中我們可以發現，通過鏡頭光軸（中心點）直線上間隔距離相等的各點，經過魚眼鏡頭成像後仍在通過光軸的同一直線上，而且此直線與水平軸的夾角維持不變，但是各點的間隔距離有所改變，離光軸越近，間隔越大，離光軸越遠，則間隔距離越小。

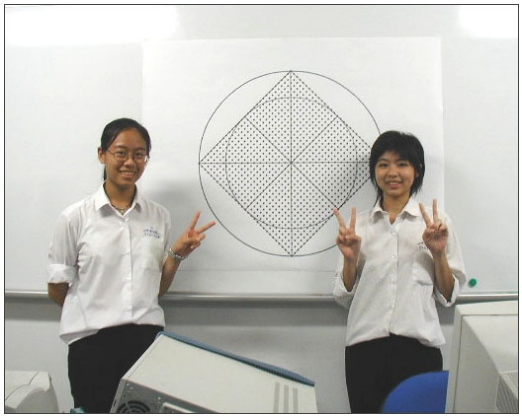

圖 2-1 原始影像



圖 2-2 魚眼鏡頭 Nikon AF Fisheye 16mm/2.8D

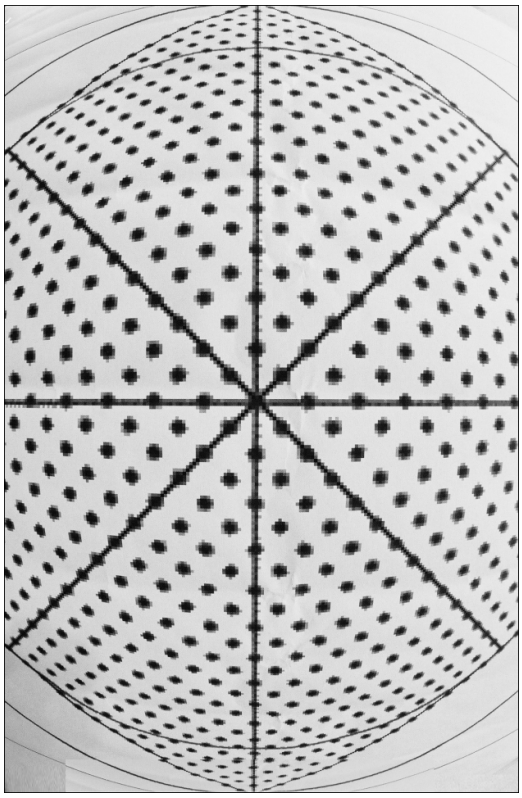

圖 2-3 原始影像經魚眼鏡頭拍攝後的結果

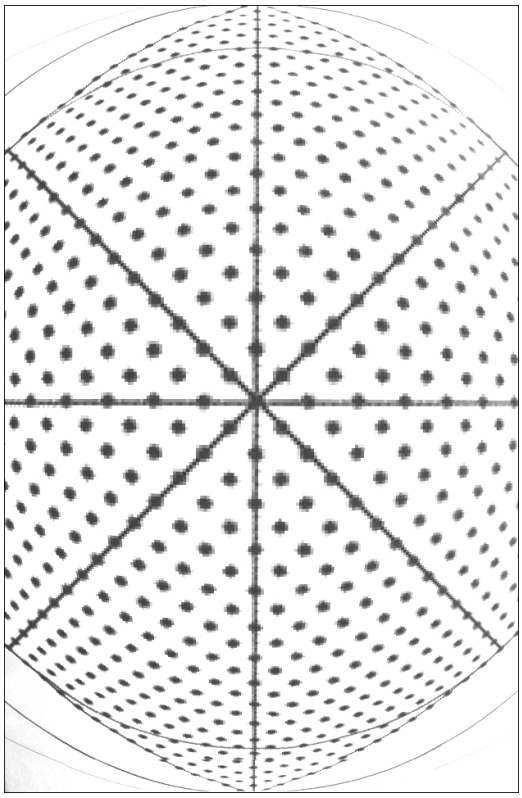

圖 2-4 調校後的拍攝影像

這種間隔距離的改變，是否有遵循著某種數學函數嗎？在圖 3 中，我們將圖 2-1 置於 x 軸的下方，圖 2-4 置於與 y 軸的左邊，嘗試著繪製出原圖與經過魚眼鏡頭成像後的關係函數圖，以便了解魚眼鏡頭的成像原理，同時也作爲發展符合我們要求的使用者介面的基礎。

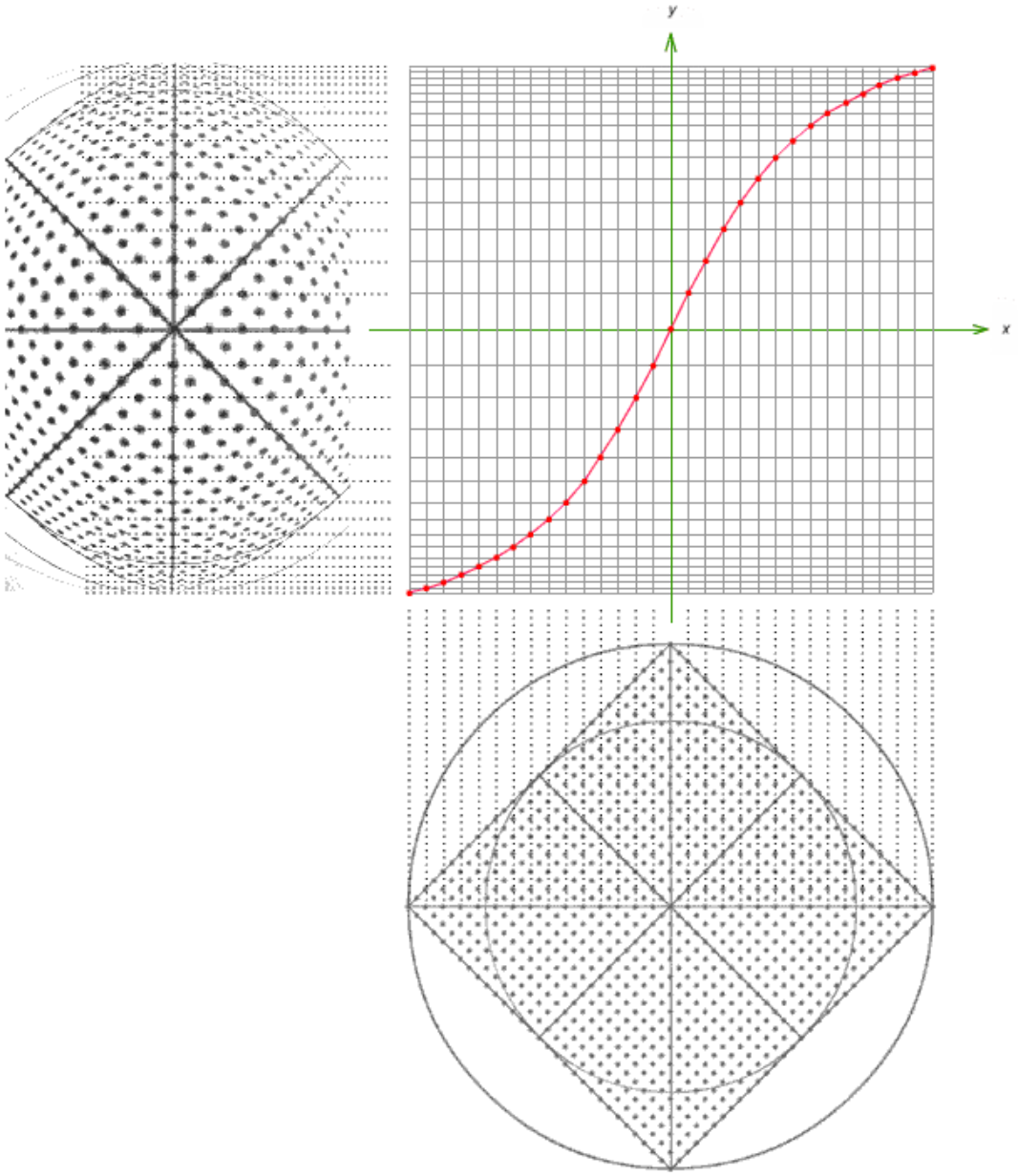

圖 3 魚眼鏡頭物與像的關係圖

3. 建構出我們的成像函數

由於原始影像中各點與鏡頭光軸（中心點）的連線與水平軸的夾角經過魚眼鏡頭成像後仍維持不變，僅是與鏡頭光軸（中心點）的距離有所改變，所以極座標系統（polor coordinates）比較適合來表示魚眼鏡頭的物與像關係函數。圖 4 中，P 點爲原始影像在魚眼鏡頭涵蓋範圍內的某一點，在以鏡頭光軸（中心點）O 點爲極的極座標系統中其座標爲 (r,θ)，經過魚眼鏡頭成像後映射至 $P'(f(r),\theta)$，現在我們的首要工作就是要找出 $f(r)$ 函數，其定義域與值域都介於 $[0,R]$ 區間，R 是魚眼鏡頭涵蓋範圍的半徑。

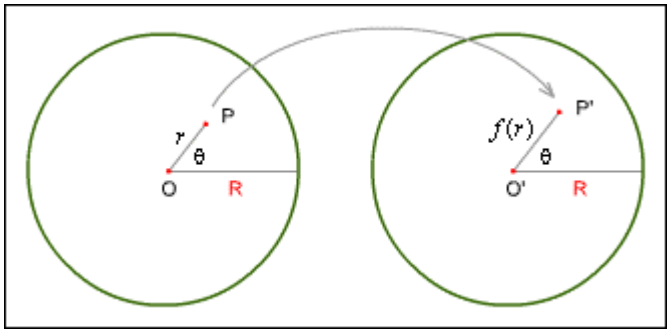

圖 4 P 點經魚眼鏡頭成像映射至 P' 點示意圖

圖 5-1 爲指數函數 $f(r)=1-a^{-r}$ 在定義域爲 $[0,\infty)$ 區間時候的圖形（a 爲實數且 $a>1$），其值域介於 $[0,1)$ 區間，與圖 3 中的魚眼鏡頭物與像的關係圖蠻類似的，但仍需將此函數做下列的修正以符合定義域與值域都介於 $[0,R]$ 區間要求：

$$f(r)=R\times\left(\frac{1-a^{-r}}{1-a^{-R}}\right)$$

圖 5-2 爲修正後的函數在定義域爲 $[0,R]$ 區間的圖形，其值域也會介於 $[0,R]$ 區間。

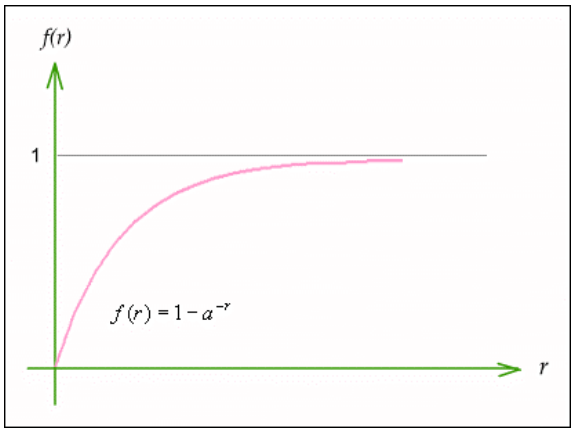

圖 5-1 指數函數 $f(r)=1-a^{r}$

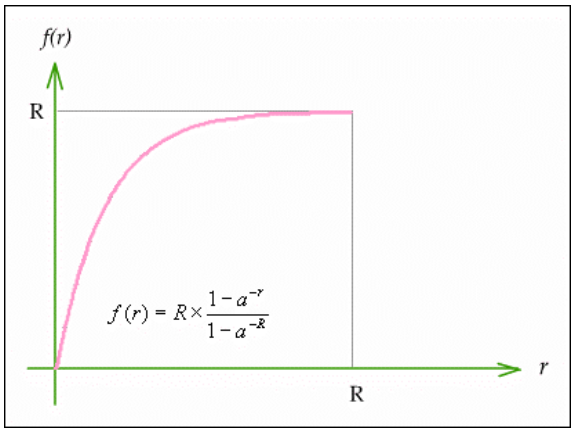

圖 5-2 修正後的函數圖形

4. 模擬魚眼鏡頭將影像局部放大的方法

要在圖形式資訊上模擬魚眼鏡頭顯現局部放大的功能，僅需針對鏡頭所涵蓋範圍內的資訊作處理，範圍外的部分就維持在原來的位置上即可。圖 6 中 O 點爲魚眼鏡頭的光軸（中心點），在整個影像中的平面直角座標爲 (x_0,y_0)，P 點爲魚眼鏡頭半徑爲 R 之涵蓋範圍內的某一點，設其在整個影像中的平面直角座標爲 (x,y)，可以計算出 P 點在以 O 點爲極的極座標系統中之座標 (r,θ)，其中 r 值爲

$$r=\sqrt{(x-x_0)^2+(y-y_0)^2}$$

由於反三角函數 arcsin 的值域介於 $[-\frac{\pi}{2},\frac{\pi}{2}]$ 區間，故 θ 值需作下列的調整，使其範圍介於 $[0,2\pi]$ 區間

$$\theta=\sin^{-1}\left(\frac{|y-y_0|}{r}\right) \qquad \text{當}\quad x\ge x_0\ ,\ y\ge y_0$$

$$\theta=\pi-\sin^{-1}\left(\frac{|y-y_0|}{r}\right) \qquad \text{當}\quad x\le x_0\ ,\ y\ge y_0$$

$$\theta=\pi+\sin^{-1}\left(\frac{|y-y_0|}{r}\right) \qquad \text{當}\quad x\le x_0\ ,\ y\le y_0$$

$$\theta=2\pi-\sin^{-1}\left(\frac{|y-y_0|}{r}\right) \qquad \text{當}\quad x\ge x_0\ ,\ y\le y_0$$

O 點經過魚眼鏡頭局部放大後對映到 $O'(x_0',y_0')$，由於 O 與 O' 兩點的位置相同，故

$$x_0'=x_0$$

$$y_0'=y_0$$

P 點則對映到 P' 點，P' 點在以 O' 點爲極的極座標系統中座標爲 (r',θ')，其中

$$r' = R \times \left(\frac{1 - a^{-r}}{1 - a^{-R}} \right) \quad （a 爲實數且 a > 1）$$

$$\theta' = \theta$$

那麼， P' 點在整個影像中的平面直角座標爲 (x', y') ，其中

$$x' = x_0^{'} + r' \cos\theta'$$

$$y' = y_0^{'} + r' \sin\theta'$$

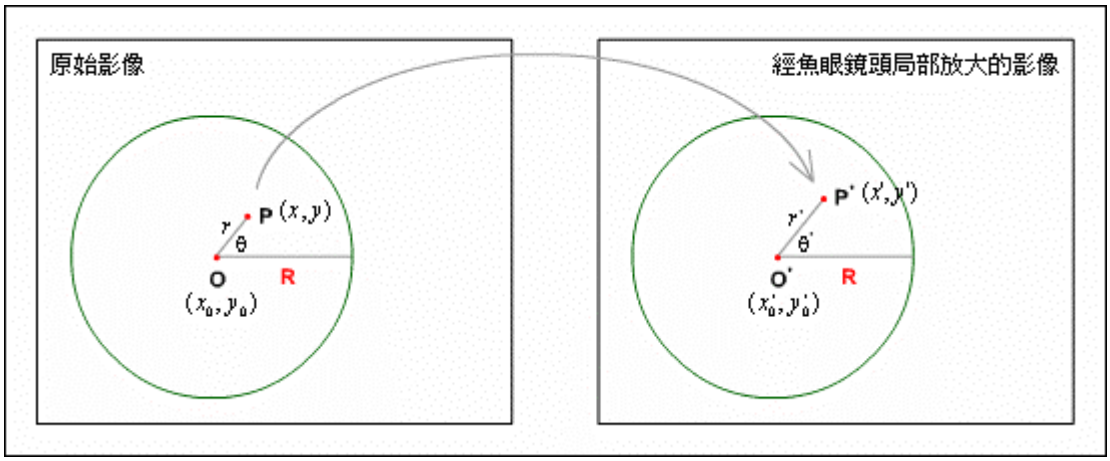

圖 6 原始影像與經過魚眼鏡頭局部放大後影像間對映示意圖

5. 原型程式實作

對於我們來說，以 VB.NET 來實作原型程式是最方便不過了，它提供了適用於多媒體與繪圖的繪圖裝置介面 GDI+（Graphics Device Interface），可讓程式設計人員不需注意特定顯示裝置的硬體詳細資料，只要透過呼叫 GDI+類別所提供的方法，便可以開發出與裝置無關的應用程式並在螢幕顯示資訊。

在圖 7 的表單中插入兩個 PictureBox 控制項，左邊的用來存放原始影像，利用滑鼠事件 MouseDown 來觸發程式，執行的過程中呼叫 GDI+的方法 GetPixel 與 SetPixel 將模擬魚眼鏡頭將影像局部放大的結果置入右邊的 PictureBox。

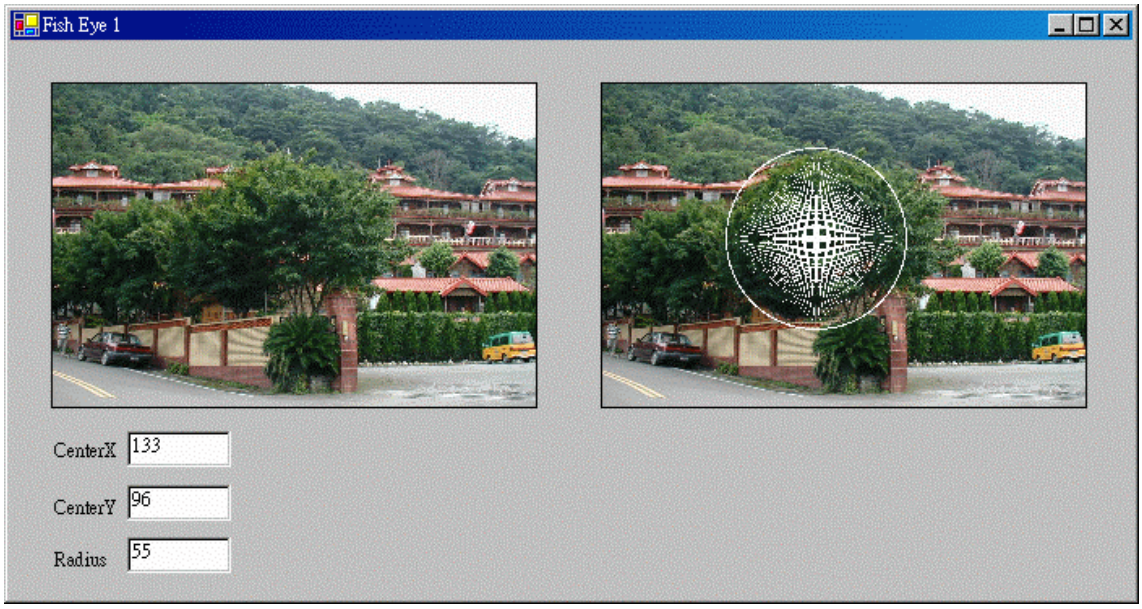

圖 7 模擬魚眼鏡頭將影像局部放大

乍看圖 7 的結果，似乎頗令人沮喪的，是不是哪裡發生了錯誤？仔細想想後，發現上圖的結果是必然的現象，因爲原始影像中各像素經過運算對映至新的位置，會有好幾個點對映至同一點的情形，當然也會有些地方沒有原始影像中的像素對映到，像圖 7 右圖中的白色部分就是。底下附上原始程式，雖然它的執行結果未臻理想，但成功地使用 GDI+的方法 GetPixel 與 SetPixel 直接對 Bitmap 影像做像素操作，收穫也算是蠻大的。

原始程式碼

```
Dim tmpImg1 As Bitmap
Dim tmpImg2 As Bitmap
Dim CR, Cx, Cy As Integer
Const a = 1.013
Private Sub PictureBox1_MouseDown(ByVal sender As Object, ByVal e As System.Windows.Forms.MouseEventArgs) _
    Handles PictureBox1.MouseDown
    Dim BlackColor As Color
    Dim Imgwidth, Imgheight, x, y, Newx, Newy, Difx, Dify As Integer
    Dim r, fr, rf, Raidx, Raidy As Double
    Dim Mousex, Mousey As Integer
    Mousex = CInt(e.X)    '偵測取得滑鼠 Click 的 x 座標
    Mousey = CInt(e.Y)    '偵測取得滑鼠 Click 的 y 座標
    tmpImg1 = PictureBox1.Image.Clone    '原始影像變數
    Imgwidth = tmpImg1.Width    '影像的寬度
    Imgheight = tmpImg1.Height    '影像的高度
    tmpImg2 = tmpImg1.Clone
    Cx = Mousex    '魚眼鏡頭光軸中心的 x 座標
    Cy = Mousey    '魚眼鏡頭光軸中心的 y 座標
    TextBox1.Text = CStr(Mousex)    '魚眼鏡頭光軸中心的 x 座標顯示在 TextBox1 上
    TextBox2.Text = CStr(Mousey)    '魚眼鏡頭光軸中心的 y 座標顯示在 TextBox2 上
    CR = CInt(TextBox3.Text)    '由 TextBox3 取得魚眼鏡頭的半徑
    For x = 1 To Imgwidth
        For y = 1 To Imgheight
            tmpImg2.SetPixel(x - 1, y - 1, BlackColor.White)    '目標影像填成白色基底
        Next
    Next
    For x = 1 To Imgwidth
        For y = 1 To Imgheight
            r = Math.Sqrt((x - Cx) ^ 2 + (y - Cy) ^ 2)
            If r > CR + 2 Then    '魚眼鏡頭涵蓋範圍外的部分就維持在原來的位置上
                tmpImg2.SetPixel(x - 1, y - 1, tmpImg1.GetPixel(x - 1, y - 1))
            End If
            If (r <= CR + 2) And (r >= CR) Then    '繪製魚眼鏡頭的外框
                tmpImg2.SetPixel(x - 1, y - 1, BlackColor.White)
            End If
            If (r < CR) And (r > 0) Then    '計算魚眼鏡頭涵蓋範圍內各點的對映點
                Difx = x - Cx
                Dify = Cy - y
                Raidx = Math.Acos(Difx / r)
                Raidy = Math.Asin(Dify / r)
                fr = CR * (1 - Math.Pow(a, -r)) / (1 - Math.Pow(a, -CR))
                Newx = Cx + CInt(fr * Math.Cos(Raidx))
                Newy = Cy - CInt(fr * Math.Sin(Raidy))
                tmpImg2.SetPixel(Newx - 1, Newy - 1, tmpImg1.GetPixel(x - 1, y - 1))
            End If
            If r = 0 Then    '魚眼鏡頭光軸中心點直接對映至原來的位置
                tmpImg2.SetPixel(x - 1, y - 1, tmpImg1.GetPixel(x - 1, y - 1))
            End If
        Next
    Next
    PictureBox2.Image = tmpImg2
End Sub
```

6. 第一次修正程式

爲了修正上述模擬魚眼鏡頭局部放大程式會使得目標影像中某些像素沒有被對映到的缺點，必須採取另類的思考模式及可行的方法，若針對魚眼鏡頭成像之目標影像中的每一個像素分別至原始影像中取點，就可以解決這個問題了，不過這個方法牽涉到反

函數的運算，現將 $P'(x',y')$ 至原始影像中取點 $P(x,y)$ 的方法描述如下。

$O'(x_0^{'},y_0^{'})$ 直接由 $O(x_0,y_0)$ 取點，兩點的座標相同，故

$$x_0^{'}=x_0$$

$$y_0^{'}=y_0$$

$P'(x',y')$ 在以 $O'(x_0^{'},y_0^{'})$ 爲極的極座標系統中座標爲 (r',θ') ，其中

$$r'=\sqrt{(x'-x_0^{'})^2+(y'-y_0^{'})^2}$$

$$\theta'=\sin^{-1}\left(\frac{|y'-y_0^{'}|}{r}\right) \qquad \text{當} \quad x'\geq x_0^{'} \ ， \ y'\geq y_0^{'}$$

$$\theta'=\pi-\sin^{-1}\left(\frac{|y'-y_0^{'}|}{r}\right) \qquad \text{當} \quad x'\leq x_0^{'} \ ， \ y'\geq y_0^{'}$$

$$\theta'=\pi+\sin^{-1}\left(\frac{|y'-y_0^{'}|}{r}\right) \qquad \text{當} \quad x'\leq x_0^{'} \ ， \ y'\leq y_0^{'}$$

$$\theta'=2\pi-\sin^{-1}\left(\frac{|y'-y_0^{'}|}{r}\right) \qquad \text{當} \quad x'\geq x_0^{'} \ ， \ y'\leq y_0^{'}$$

P 點在以 O 點爲極的極座標系統中之座標 (r,θ) ，由於 $r'=R\times\left(\frac{1-a^{-r}}{1-a^{-R}}\right)$ ， a 爲實數且 $a>1$ ，所以

$$r=\log_a\left(\frac{R}{R-r'(1-a^{-R})}\right)$$

$$\theta=\theta'$$

P 點在原始影像中的平面直角座標爲 (x,y)

$$x=x_0+r\cos\theta$$

$$y=y_0+r\sin\theta$$

下面是第一次修正後的程式及其執行出來的結果，頗令人滿意的，不是嗎？

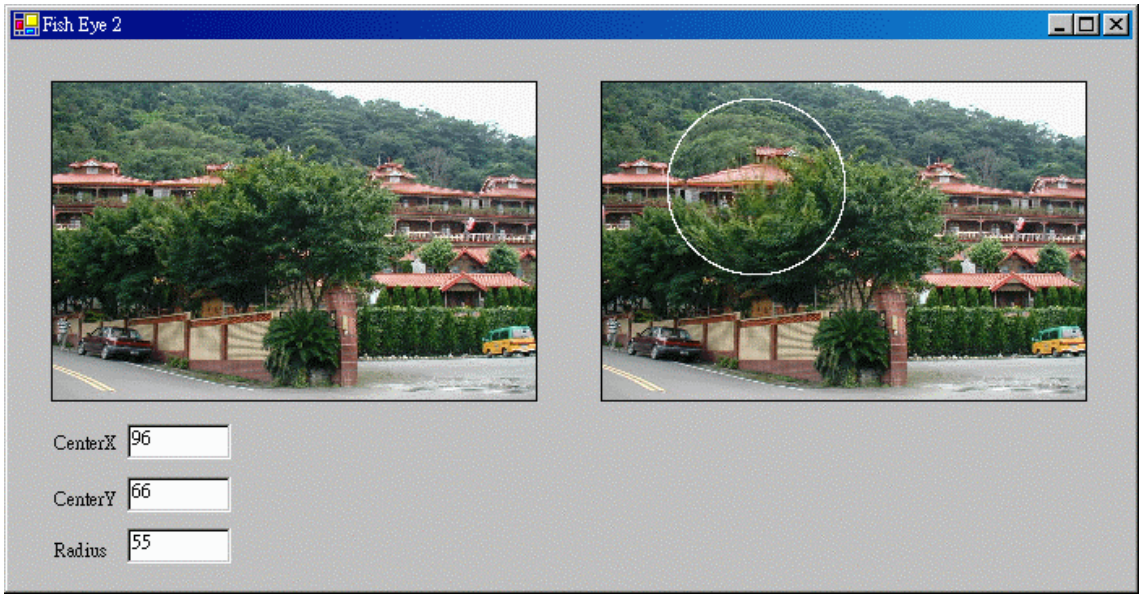

圖 8 模擬魚眼鏡頭將影像局部放大程式經第一次修正後執行的結果

第一次修正後的程式碼

```
Dim tmpImg1 As Bitmap
Dim tmpImg2 As Bitmap
Dim CR, Cx, Cy As Integer
Const a = 1.013
Private Sub PictureBox1_MouseDown(ByVal sender As Object, ByVal e As System.Windows.Forms.MouseEventArgs) _
    Handles PictureBox1.MouseDown
    Dim BlackColor As Color
    Dim Imgwidth, Imgheight, x, y, Newx, Newy, Difx, Dify As Integer
    Dim r, fr, Raidx, Raidy As Double
    Dim Mousex, Mousey As Integer
    Mousex = CInt(e.X)    '偵測取得滑鼠 Click 的 x 座標
    Mousey = CInt(e.Y)    '偵測取得滑鼠 Click 的 y 座標
    tmpImg1 = PictureBox1.Image.Clone    '原始影像變數
    Imgwidth = tmpImg1.Width    '影像的寬度
    Imgheight = tmpImg1.Height    '影像的高度
    tmpImg2 = tmpImg1.Clone
    Cx = Mousex    '魚眼鏡頭光軸中心的 x 座標
    Cy = Mousey    '魚眼鏡頭光軸中心的 y 座標
    TextBox1.Text = CStr(Mousex)    '魚眼鏡頭光軸中心的 x 座標顯示在 TextBox1 上
    TextBox2.Text = CStr(Mousey)    '魚眼鏡頭光軸中心的 y 座標顯示在 TextBox2 上
    CR = CInt(TextBox3.Text)    '由 TextBox3 取得魚眼鏡頭的半徑
    For x = 1 To Imgwidth
        For y = 1 To Imgheight
            tmpImg2.SetPixel(x - 1, y - 1, BlackColor.White)    '目標影像填成白色基底
        Next
    Next
    For x = 1 To Imgwidth
        For y = 1 To Imgheight
            fr = Math.Sqrt((x - Cx) ^ 2 + (y - Cy) ^ 2)
            If fr > CR + 2 Then    '魚眼鏡頭涵蓋範圍外的部分就維持在原來的位置上
                tmpImg2.SetPixel(x - 1, y - 1, tmpImg1.GetPixel(x - 1, y - 1))
            End If
            If (fr <= CR + 2) And (fr >= CR) Then    '繪製魚眼鏡頭的外框
                tmpImg2.SetPixel(x - 1, y - 1, BlackColor.White)
            End If
            If (fr < CR) And (fr > 0) Then    '經由反函數的計算該由原始影像中何處取點
                Difx = x - Cx
                Dify = Cy - y
                Raidx = Math.Acos(Difx / fr)
                Raidy = Math.Asin(Dify / fr)
                r = (Math.Log((CR * Math.Pow(a, CR)) / (CR * Math.Pow(a, CR) - fr * (Math.Pow(a, CR) - 1)))) / (Math.Log(a))
                Newx = Cx + CInt(r * Math.Cos(Raidx))
                Newy = Cy - CInt(r * Math.Sin(Raidy))
                tmpImg2.SetPixel(x - 1, y - 1, tmpImg1.GetPixel(Newx - 1, Newy - 1))
            End If
            If fr = 0 Then    '魚眼鏡頭光軸中心點直接由原來的位置取點
                tmpImg2.SetPixel(x - 1, y - 1, tmpImg1.GetPixel(x - 1, y - 1))
            End If
        Next
    Next
    PictureBox2.Image = tmpImg2
End Sub
```

7. 第二次修正程式

當在台北捷運地圖上使用模擬魚眼鏡頭將影像局部放大的的時候，發現台北捷運地圖上的文字會產生字體變形的情況，變形扭曲的文字容易使瀏覽者在判讀時造成困擾或錯誤，如圖 9 所示，「台北車站」、「台大醫院」、「中正紀念堂」等字體是不是很難辨識呀？

線性函數可以使影像以等比例的方式來縮放，不會有字體變形的問題，若能如圖 10 所示將原來的成像函數與線性函數相結合，在魚眼鏡頭所涵蓋的部分範圍內採用線性函數，那麼字體變形的問題應該就可以獲得某種程度的改善。

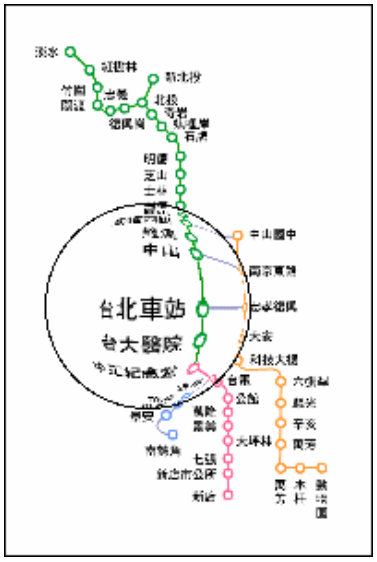

圖 9 文字產生了變形的情況

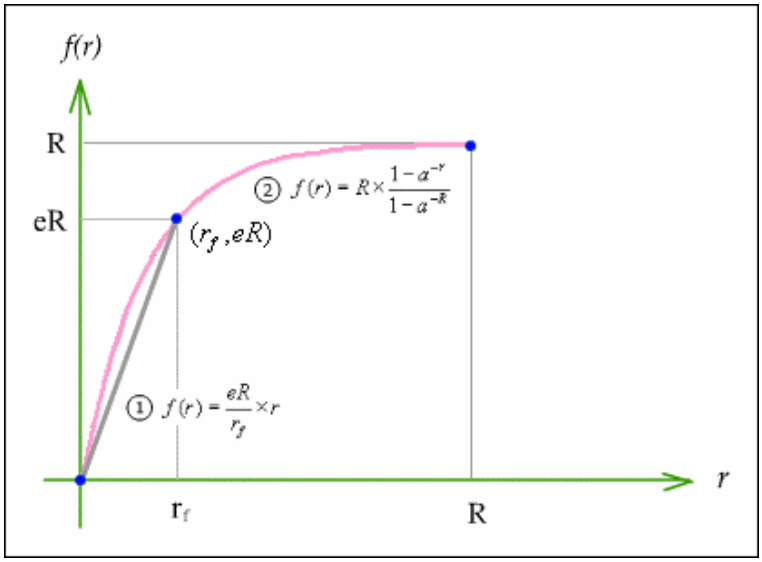

圖 10 原成像函數與線性函數結合後的函數圖形

結合線性函數後，新的模擬魚眼鏡頭將影像局部放大的成像函數如下，當 $0 \le r \le r_f$ 時，採用線性函數將影像做等比例的放大；當 $r_f < r \le R$ 時，採用原來的魚眼鏡頭函數來成像，圖 11 爲此新函數成像的簡單示意圖。

$$f(r) = \frac{eR}{r_f} \times r \qquad \text{當} \quad 0 \le r \le r_f$$

$$f(r) = R \times \frac{1 - a^{-r}}{1 - a^{-R}} \qquad \text{當} \quad r_f < r \le R$$

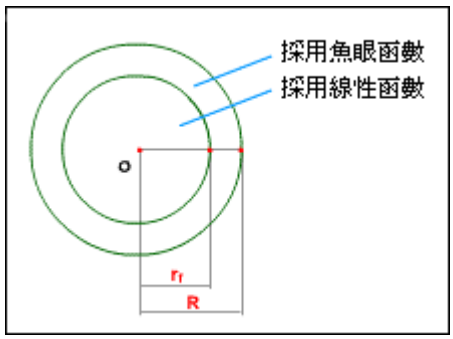

圖 11 新的函數成像示意圖

經過再次地修正後，我們將整個程式的運作方式做個整理，其中需要用到的參數與符號有 $O(x_0, y_0)$ 爲原始影像中魚眼鏡頭光軸中心、$P(x, y)$ 爲原始影像中的某一點、r_f 爲原始影像中魚眼鏡頭涵蓋範圍內線性放大部分的半徑、$O'(x_0', y_0')$ 爲目標影像中魚眼鏡頭光軸中心、$P'(x', y')$ 爲目標影像中的某一點、R 爲魚眼鏡頭涵蓋範圍之半徑、eR 爲目標影像中魚眼鏡頭涵蓋範圍內線性放大部分的半徑。首先，目標影像中的 $O'(x_0', y_0')$ 直接由原始影像中的 $O(x_0, y_0)$ 取點，所以

$$O'(x_0', y_0') = O(x_0, y_0)$$

再來針對目標影像中的每一點 $P'(x', y')$，先計算出該點以 $O'(x_0', y_0')$ 爲極的極座標系統中的座標 $P'(r', \theta')$，依據 r' 值分別作下列的處理

當 $r' > R$ 時

$P'(x', y')$ 直接由 $P(x, y)$ 取點

$$P'(x', y') = P(x, y)$$

當 $R \ge r' > eR$ 時

$$r' = f(r) = R \times \frac{1 - a^{-r}}{1 - a^{-R}}$$

所以 $r = \log_a \left(\frac{R}{R - r'(1 - a^{-R})} \right)$

魚眼成像時θ角維持不變，故 $\theta = \theta'$

$P'(x', y')$ 由 $P(x, y)$ 取點，其中 x 、 y 分別爲

$x = x_0 + r\cos\theta$

$y = y_0 + r\sin\theta$

當 $eR \geq r' > 0$ 時

因爲 $eR = R \times \dfrac{1 - a^{-r_f}}{1 - a^{-R}}$

所以 $r_f = -\log_a\left(1 - e(1 - a^{-R})\right)$

由於 $r' = f(r) = \dfrac{eR}{r_f} \times r = \dfrac{eR}{-\log_a\left(1 - e(1 - a^{-R})\right)} \times r$

故 $r = \dfrac{-\log_a\left(1 - e(1 - a^{-R})\right)}{eR} \times r'$

魚眼成像時θ角維持不變，故 $\theta = \theta'$

$P'(x', y')$ 由 $P(x, y)$ 取點，其中 x 、 y 分別爲

$x = x_0 + r\cos\theta$

$y = y_0 + r\sin\theta$

圖 12 是原成像函數與線性函數相結合後的修正程式執行出來的結果，其程式碼也顯示出來以供參考。

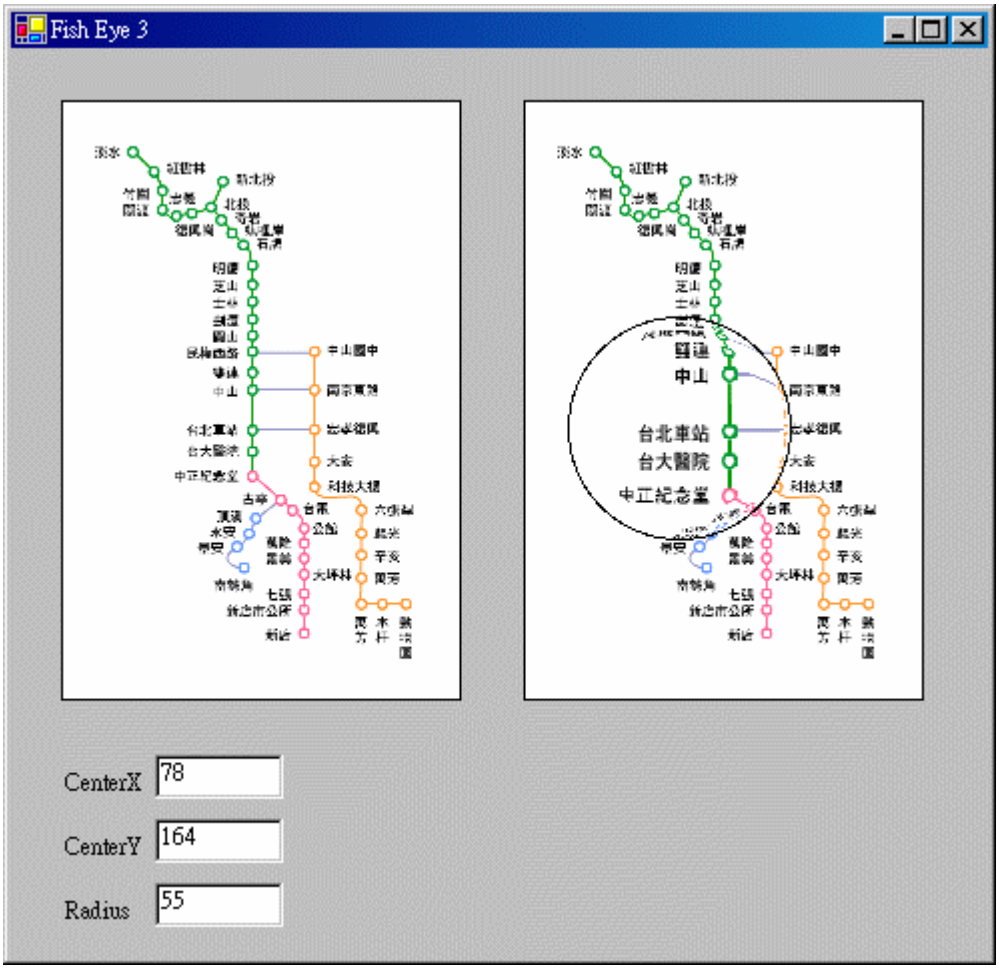

圖 12 模擬魚眼鏡頭將影像局部放大程式經第二次修正後執行的結果

第二次修正後的程式碼

```
Dim tmpImg1 As Bitmap
Dim tmpImg2 As Bitmap
Dim CR, Cx, Cy As Integer
Const a = 1.018
Const nCR = 4 / 5
Private Sub PictureBox1_MouseDown(ByVal sender As Object, ByVal e As System.Windows.Forms.MouseEventArgs) _
    Handles PictureBox1.MouseDown
    Dim BlackColor As Color
    Dim Imgwidth, Imgheight, x, y, Newx, Newy, Difx, Dify As Integer
    Dim r, fr, rf, Raidx, Raidy As Double
    Dim Mousex, Mousey As Integer
    Mousex = CInt(e.X)    '偵測取得滑鼠 Click 的 x 座標
    Mousey = CInt(e.Y)    '偵測取得滑鼠 Click 的 y 座標
    tmpImg1 = PictureBox1.Image.Clone    '原始影像變數
    Imgwidth = tmpImg1.Width    '影像的寬度
    Imgheight = tmpImg1.Height    '影像的高度
    tmpImg2 = tmpImg1.Clone
    Cx = Mousex    '魚眼鏡頭光軸中心的 x 座標
    Cy = Mousey    '魚眼鏡頭光軸中心的 y 座標
    TextBox1.Text = CStr(Mousex)    '魚眼鏡頭光軸中心的 x 座標顯示在 TextBox1 上
    TextBox2.Text = CStr(Mousey)    '魚眼鏡頭光軸中心的 y 座標顯示在 TextBox2 上
    CR = CInt(TextBox3.Text)    '由 TextBox3 取得魚眼鏡頭的半徑
    rf = -1 * (Math.Log(1 - nCR * (1 - Math.Pow(a, -CR)))) / (Math.Log(a))
    For x = 1 To Imgwidth
        For y = 1 To Imgheight
            tmpImg2.SetPixel(x - 1, y - 1, BlackColor.Black)    '目標影像填成黑色基底
        Next
    Next
    For x = 1 To Imgwidth
        For y = 1 To Imgheight
            fr = Math.Sqrt((x - Cx) ^ 2 + (y - Cy) ^ 2)
            If fr > CR + 2 Then    '魚眼鏡頭涵蓋範圍外的部分就維持在原來的位置上
                tmpImg2.SetPixel(x - 1, y - 1, tmpImg1.GetPixel(x - 1, y - 1))
            End If
            If (fr <= CR + 2) And (fr >= CR) Then    '繪製魚眼鏡頭的外框
                tmpImg2.SetPixel(x - 1, y - 1, BlackColor.White)
            End If
            If (fr < CR) And (fr > 0) Then    '經由反函數的計算該由原始影像中何處取點
                Difx = x - Cx
                Dify = Cy - y
                Raidx = Math.Acos(Difx / fr)
                Raidy = Math.Asin(Dify / fr)
                If fr < nCR * CR Then    '線性函數
                    r = (fr * rf) / (nCR * CR)
                Else    '魚眼函數
                    r = (Math.Log((CR * Math.Pow(a, CR)) / (CR * Math.Pow(a, CR) - fr * (Math.Pow(a, CR) - 1)))) / (Math.Log(a))
                End If
                    Newx = Cx + CInt(r * Math.Cos(Raidx))
                    Newy = Cy - CInt(r * Math.Sin(Raidy))
                    tmpImg2.SetPixel(x - 1, y - 1, tmpImg1.GetPixel(Newx - 1, Newy - 1))
            End If
            If fr = 0 Then    '魚眼鏡頭光軸中心點直接由原來的位置取點
                tmpImg2.SetPixel(x - 1, y - 1, tmpImg1.GetPixel(x - 1, y - 1))
            End If
        Next
    Next
    PictureBox2.Image = tmpImg2
End Sub
```

三、討論與應用：

1. 討論

瀏覽圖形式資訊的時候，希望在顯示整體結構的同時，能夠隨著滑鼠游標的移動，以模擬魚眼鏡頭的方式，即時地將想要觀察的部分局部放大以展現細部的資料，此種使用者介面應該是可行的，實作時只要將觸發程式執行的事件由 PictureBox1（存放原始

影像)的 MouseDown 改爲 PictureBox2(存放目標影像)的 MouseMove，然後將 PictureBox1 隱藏起來即可，不過執行的結果有點令人沮喪，無論如何精簡與加速程式，或多或少都有反應延遲的情形。

探究其原因，可由參考資料[3]知道，GDI+是 Windows 系統中繪圖用的 API（Application Programming Interface），其描繪速度相當的慢，能夠表現的部分也顯得相當的貧瘠，解決之道，就是採用 DirectX，DirectX 以不透過 API 的方式，盡可能的直接操作硬體，來達成高速執行運作。目前 DirectX 對我來說是一項很困難的挑戰，不過我相信在將來終有完成的一日。

2. 應用

現今最熱門的行動式通訊（mobile communication），圖形式資訊在其系統中扮演了非常重要的角色，爲了攜帶方便，通常使用者端的設備如筆記型電腦、個人數位助理、掌上型電腦、全球衛星定位系統、WAP 手機，其螢幕空間都非常的有限，我們發展出的使用者介面將會對其有相當大的助益。

四、結語：

由 Sakar 與 Brown 的「Graphical fisheye views」方法中得到了啓示，藉著重新檢視魚眼鏡頭而找出模擬的成像函數，並透過程式的實作來修正，最後才形成了我們模擬魚眼鏡頭將影像局部放大的使用者介面，雖然內心充滿著喜悅，但也苦惱於其執行速度略慢的問題，DirectX 或許是解決之道，不過在沒有以 DirectX 實作出來前，不敢妄自推測。

在一些公眾場所常見到「哈哈鏡」，自己與朋友們常因鏡中扭曲變形的影像而互相取笑，可能可以用類似的研究模式而發展出影像處理上的哈哈鏡，讓大家隨時有自娛娛人的機會。

五、參考資料：

1. George W. Furnas., “Generalized fisheye views,” Proceeding of Human Factors in Computing System, CHI’86, ACM Press, pp. 16-23, 1986.。
2. Manojit Sarkar, and Marc H. Brown., “Graphical fisheye views,” Communications of the ACM, 37(12), pp. 73-84, 1994.。
3. 藤田伸二著, “DirectX+VB.NET 3D 電玩程式設計實例入門,” 博碩文化股份有限公司, 2003.。

評語及建議事項

此作品具原創性，科學態度與精神佳，思考程序完整，作者可在多想想當"魚眼鏡頭材質變化時"所呈現出的影像爲何？作品有應用性，亦與理論來支持，作者表達能力佳。補充"建議作者定義"視域函數”來將作品的優異性量化。