2013 臺灣國際科學展覽會
優勝作品專輯

作品編號 **110003**

參展科別 **電腦科學科**

作品名稱 **一種新的圖形導覽介面**

得獎獎項 **一等獎**

美國 ISEF 團隊正選代表:美國第 64 屆國際科技展覽會

就讀學校　國立新竹女子高級中學

指導教師　徐以誠、顏郁婷

作者姓名　吳思蓉、黃湘雯

關 鍵 字　導覽介面 、 影像處理

作者簡介

(右)大家好，我是吳思蓉，目前就讀新竹女中三年級。高一下學期很慶幸得參與專題研究，我的高中生活也就變得多彩多姿，不論是專業知識的拓展，又或語言表達都有明顯的成長，感謝互相扶持的夥伴和老師的鼓勵與指導，一路陪著我完成作品。帶著雀躍的心情參加這次的國際科展，期待分享自己的作品，結交來自各方的科學知音，相互切磋。

(左)很高興能認識大家，我是黃湘雯，目前就讀新竹女中三年級。我在研究專題的過程中，深刻了解到愈不遺餘力地投入，激發出的求知欲望愈加強烈。因為這一次的研究，培養了我對學習的正向態度，而且因為有一路相隨的夥伴和老師，所以我並不感到孤單。很開心能有如此難得的經驗，讓我能認識各地的朋友，在科學的互動中開拓我的視野。

摘要

利用電腦瀏覽圖形式資訊的時候，常受到螢幕空間大小的限制，沒有辦法在顯示圖形整體結構的同時也顯示細節部分。超廣角鏡頭是一種短焦距、大視角的相機鏡頭，鏡頭成像的時候，會有中間部分放大而周邊部分縮小的情形，藉由這個特性，我們發展出了一種新的圖形導覽介面，在瀏覽圖形式資訊的時候，有個圓形區域，該區域可隨著使用者的意願而自由移動，而區域內的圖形是以模擬超廣角鏡頭成像的方式呈現，且能夠與圓形區域外的圖形做銜接，如此，在瀏覽圖形式資訊的時候，除能夠顯示整體的結構外，也可以不開啟新視窗及無遮蔽的方式，即時地將想要觀察的部分做局部放大以展現細節。

Abstract

A Novel Interface for Image Browsing

The size of the most commonly-used screen often confines the display of the large graphic information. The drawback is that the screen can't show the whole structure and the details simultaneously. Nowadays, the most common solution includes scrollable view, multiple views and zooming. Although these ways are able to display the details of the graphic, there are still disadvantages. For example, users have to stare at the screen attentively when using scrollable view. The multiple views have the flaw that part of the area is obscured. As for zooming, it can't provide the overall structure at the same time.

A wide-angle lens is a very wide field of view with a short focal length. When forming the image, the wide-angle lens would magnify the middle part and shrink the outer part. With this feature, we advance a novel user interface for browsing graphics. There is a circle area which could be moved freely as users' will. The image of the circle area imitates the image of wide-angle lens, and is capable of conjuncting the outside of the circle. As a result, when browsing large graphic information, this function enables users to view the overall structure and the details in the timely manner, without opening any new window or being sheltered.

壹、研究動機與目的

利用電腦瀏覽圖形式資訊的時候，常受到螢幕空間大小的限制，沒有辦法在顯示圖形整體結構的同時也顯示細節部分，目前導覽介面所採用的方式有捲軸捲動、開啟多視窗、局部放大等方法，這些方法雖然可以呈現資訊的細節部分，但是仍有其不便之處，捲軸捲動的方式使得使用者的眼睛必須盯著仔細觀看；開啟多視窗的方法會有部分區域被遮蔽的情形發生；局部放大的方式則無法同時地呈現整體結構。

來自於日本的大頭狗依然是目前流行的商品之一，不論是玩偶或是文具、衣服、包包上都可以看到這些狗狗逗趣可愛的變形大頭照，經過查詢相關資料後，發現是利用超廣角鏡頭拍攝所造成的，為了放大影像的中心區域，使得周圍的區域產生了變形失真，雖然如此，但這樣的呈現方式似乎可以發展出一種新的圖形導覽介面，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，以不開啟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，這種導覽介面將可免除現有方法的缺點。

貳、器材及軟體

一、筆記型電腦。

二、數位相機 Nikon CoolPix 4500。

三、超廣角鏡頭 Nikon WC-E24、Nikon FC-E8。

四、Ulead PhotoImpact。

五、VisualStudio 2008。

參、研究過程

一、現有的圖形導覽介面

1. 最早的超廣角圖形導覽介面

超廣角鏡頭是一種短焦距、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，參考資料[4]將超廣角鏡頭的原理運用在圖形介面上，其所使用的轉換函數如下，其中 d 為正整數，根據原始影像中每一點的 x 與 y 座標，分別計算出該點新的位置，然後填入圖形介面上。

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

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

由圖 1 可以看出，他們的方法使得離焦點愈近的物件，能以較詳盡的方式呈現在螢幕上，而由焦點向外輻射，離焦點愈遠的物件則變小和變模糊，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，所需的計算量會較多。

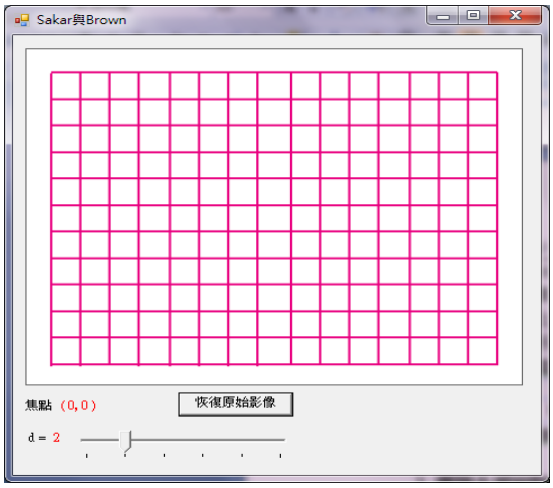


圖 1-1 原始影像

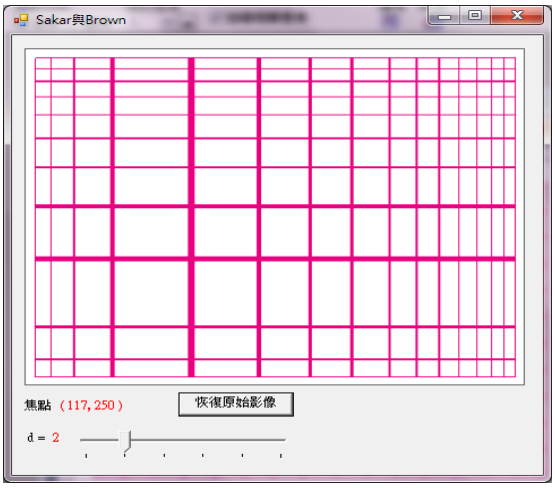


圖 1-2 Sakar 與 Brown 的圖形導覽介面

2. 常見的導覽介面

圖 2 捲軸捲動式的導覽介面

圖 3 開啟多視窗的導覽介面

圖 4 局部放大的導覽介面

當螢幕畫面的尺寸無法容納想要呈現的資訊時，目前導覽介面所採用的方式有捲軸捲動式、開啟多視窗、局部放大式等三種類型，這些方法雖然可以呈現資訊的細節部分，但是仍有其不便之處，捲軸捲動的方式，必須拉動垂直或橫向捲軸，才能查看目前超出檢視範圍的資訊，使用者的眼睛也必須盯著留意；開啟多視窗的方法，除了必須重複地操作視窗的開啟與關閉外，也會有部分區域被遮蔽的情形發生；局部放大的

方式，無法同時地呈現整體結構。

二、廣角鏡頭的變形機制

1. 鏡頭的種類

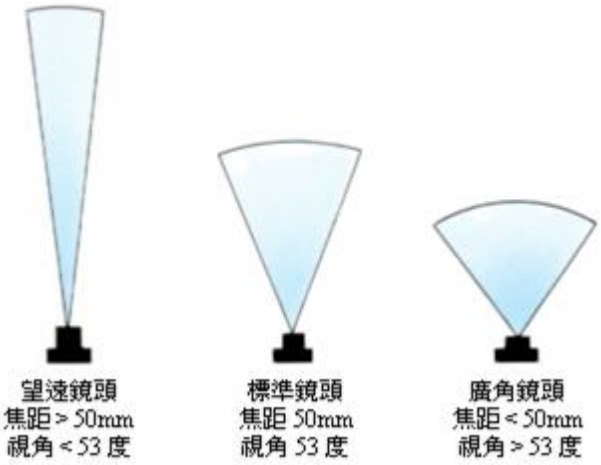


圖 5 相機鏡頭種類

以鏡頭焦距的長短來分類，有望遠鏡頭、標準鏡頭、廣角鏡頭等三大類。望遠鏡頭為焦距 50mm 以上的長焦距鏡頭，能夠拉近遠距離的景物，視角較小，景深也會變短，適合拍攝遠距離的大自然景觀，自然生態的動植物及人物的特寫鏡頭；標準鏡頭的焦距為 50mm，其呈現的影像與肉眼所見的景物相似（視角為 53 度），因為最大光圈值較大，所以適合拍攝一般風景或人物肖像照片；廣角鏡頭為焦距 50mm 以下的短焦距鏡頭，其拍攝視野較寬（即視角較大），從觀景窗看到的景物呈現於照片上會會變小，不過景深較深長。適合拍攝寬闊的全景、風景照、俯瞰景觀等景物。

2. 校正樣板的製作與拍攝裝置

廣角鏡頭雖然可以取得較大範圍的影像，但是無論製作技術有多麼地精良，廣角鏡頭都會有光學失真變形的情形，為了利用這個特性來發

展出一種新的圖形導覽介面，我們打算先設計些校正樣板，然後，以超廣角鏡頭來拍攝這些校正樣板，希望能夠找到其成像與變形的模式。

校正樣板可用來尋找成像前後的對應關係，圖 6-1 的網狀格子點是最直覺的選擇，經過拍攝後，我們發現了一些變形的特性，於是又設計了圖 6-2 的校正樣板來驗證這些特性。

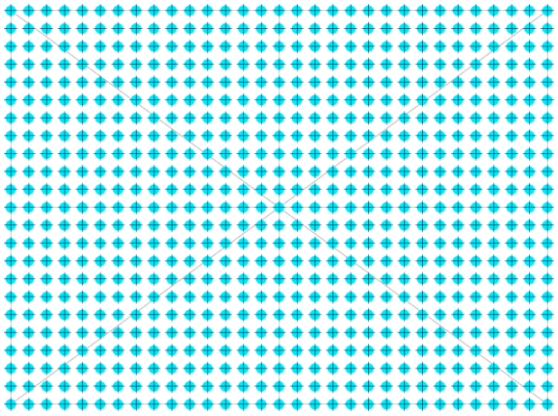


圖 6-1 校正樣板 1

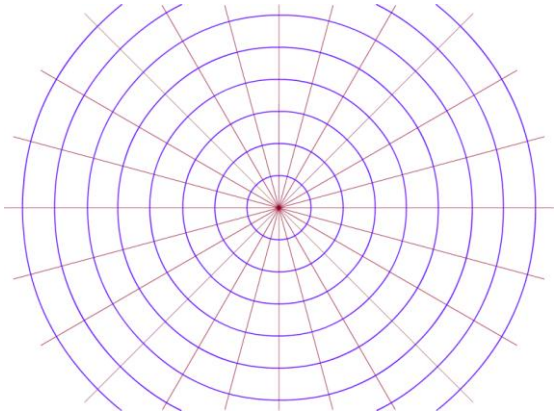


圖 6-2 校正樣板 2

整個拍攝設備與裝置如圖 7 所示，所使用的器材有：數位相機、超廣角鏡頭、相機腳架、校正樣板、水平儀、鉛垂線、筆記型電腦，所拍攝出來的校正樣板影像為圖 8-1 與圖 8-2。在正式拍攝前有兩件事情是必須特別注意的地方，第一，置放校正樣板的腳架平台以及相機機身必須做好水平與垂直調校；第二，相機鏡頭的中心點須與校正樣板的中心點重合。

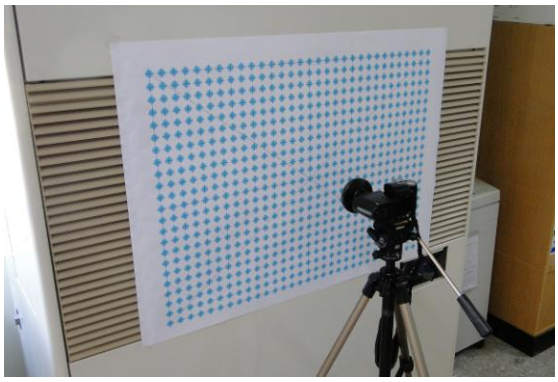


圖 7 校正樣板 2

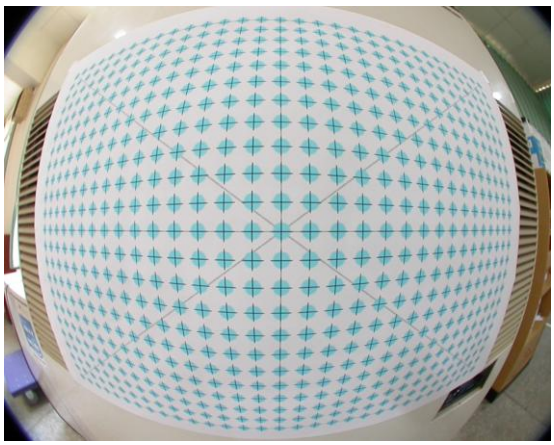


圖 8-1 校正樣板 1 的影像

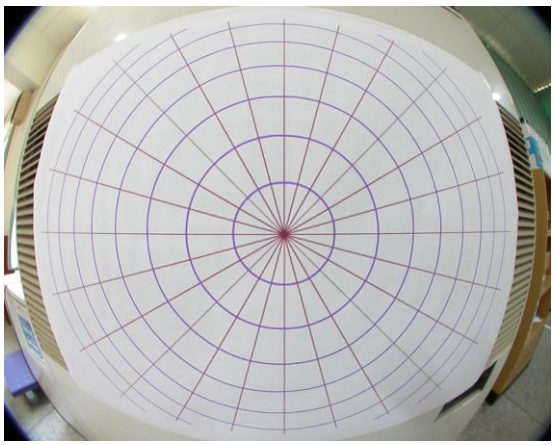


圖 8-2 校正樣板 2 的影像

3. 變形特性

觀察後我們可以發現，如圖 9-1 所示，校正樣板 1 上通過中心點的直線，經過超廣角鏡頭成像後仍為通過中心點的直線，其他沒有通過中心點的直線則會彎曲，而且離中心點越遠彎曲的越嚴重。由圖 9-2 中可以了解造成直線彎曲的原因，校正樣板 1 中同一直線上的點，經過超廣角鏡頭成像後，離中心點越遠的點，其距離被縮減的越多。

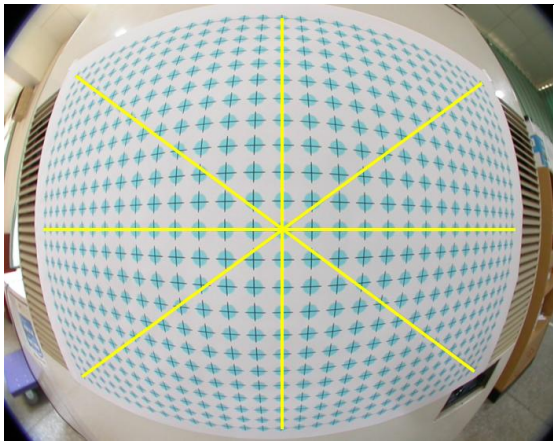


圖 9-1 通過中心點的直線

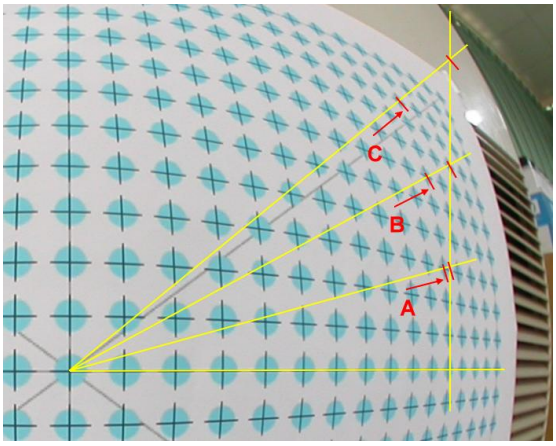


圖 9-2 離中心點越遠距離縮減越多

校正樣板 2 是由等間隔距離的同心圓及等夾角的直線所構成，我們可從圖 9-3 中發現，間隔距離相等的同心圓經過超廣角鏡頭成像後仍為同心圓，但是間隔距離有所改變，離中心點越近，間隔越大，離中心點越遠，則間隔距離越小；從圖 9-4 中發現，通過中心點等夾角的直線經

過超廣角鏡頭成像後仍為通過中心點的直線，且此直線與水平軸的夾角維持不變。

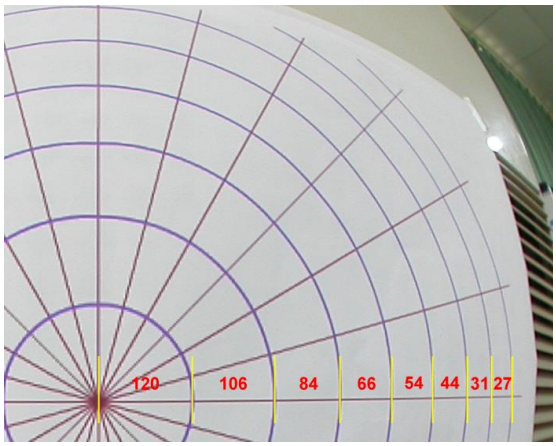


圖 9-3 同心圓間隔距離的改變情形

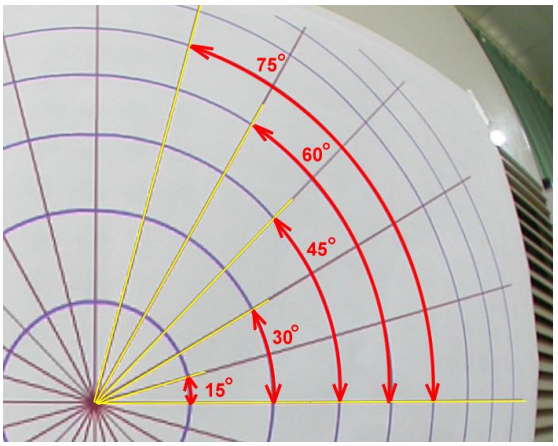


圖 9-4 直線與水平軸的夾角維持不變

三、廣角鏡頭的變形與校正

1. 變形與校正的構想

由超廣角鏡頭的變形特性看來，極座標系統比較適合用來表示超廣角鏡頭物與像的關係函數，假設 P 點為原始影像的某一點，在中心點為 O 的極座標系統中其座標為 (r,θ)，經過超廣角鏡頭成像後映射至 P’ (r',θ')，由於原始影像中各點與中心點的連線與水平軸的夾角經過超廣角鏡頭成像後仍維持不變，所以 $\theta=\theta'$，再來的工作就是要找出 $r'=f(r)$ 成像函數，有了 $f(r)$ 成像函數，我們除了可以利用程式將原始影像產生類似超廣角鏡頭變形的效果，也可以利用其反函數 $f^{-1}(r')$ 來校正超廣角鏡頭所拍攝的變形影像，整個超廣角鏡頭的變形與校正構想如圖 10 所示。

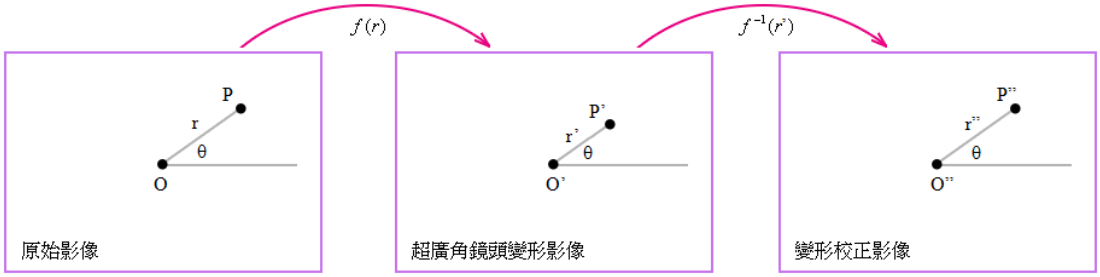


圖 10 超廣角鏡頭的變形與校正構想

2. 尋找成像與校正函數

由圖 9-3 看出，等間隔距離的同心圓在超廣角鏡頭拍攝後間隔距離的改變情形，這種間隔距離的改變，是否遵循著某種數學函數呢？我們嘗試著繪製出其函數圖形，以便了解鏡頭的成像方式，由圖 11 看來，多項式函數應該可以滿足需求。為了求得多項式函數的各項係數，在校正樣板 1 上隨機挑選了 12 個取樣點，將其成像前後的座標當作已知資料，然後使用 *Excel* 的趨勢線來尋找成像函數 $f(r)$ 及其反函數 $f^{-1}(r')$，由取樣點的平均誤差看來，較高階的多項式函數會比較精準些。

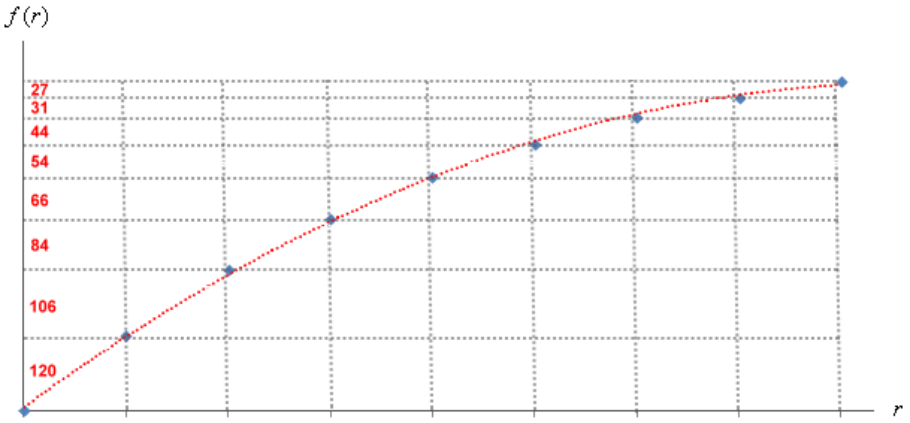


圖 11 同心圓等間隔距離的改變情形

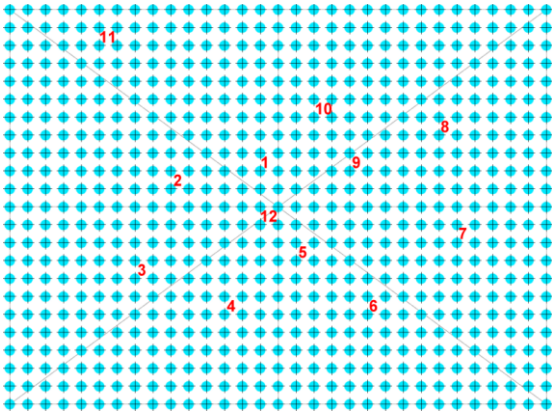


圖 12-1 校正樣板上的取樣點

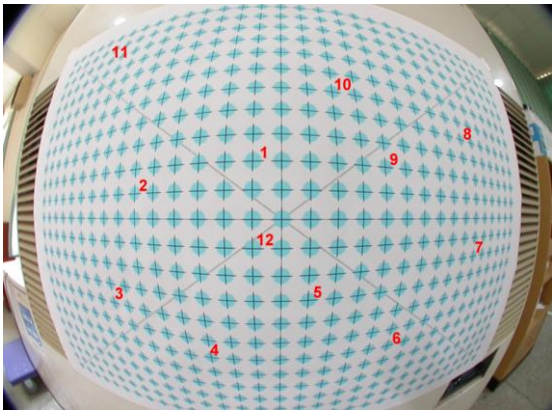


圖 12-2 各取樣點的成像點

表 1

取樣點編號	取樣點位置	取樣點極座標 (r,θ)	成像點位置	成像點極座標 (r',θ)
1	(-26 , 52)	(58.1, 117°)	(-42 , 85)	(94.8, 116°)
2	(-156 , 25)	(158.0, 171°)	(-214 , 37)	(217.2, 170°)
3	(-208 , -104)	(232.6, 207°)	(-243 , -121)	(271.5, 206°)
4	(-78 , -156)	(174.4, 243°)	(-102 , -201)	(225.4, 243°)
5	(26 , -79)	(83.2, 288°)	(39 , -117)	(123.3, 288°)
6	(130 , -156)	(203.1, 310°)	(154 , -184)	(239.9, 310°)
7	(260 , -53)	(265.3, 348°)	(278 , -53)	(283.0, 349°)
8	(234 , 103)	(255.7, 24°)	(260 , 116)	(284.7, 24°)
9	(104 , 52)	(116.3, 27°)	(151 , 76)	(169.0, 27°)
10	(52 , 130)	(140.0, 68°)	(74 , 186)	(200.2, 68°)
11	(-260 , 233)	(349.1, 138°)	(-249 , 230)	(339.0, 137°)
12	(-26 , -27)	(37.5, 226°)	(-42 , -42)	(59.4, 225°)

表 2

多項式階數	成像函數 $f(r)$	取樣點平均誤差
2	$1.6446r - 2.0117\times10^{-3}r^2$	5.71
3	$1.8596r - 3.9914\times10^{-3}r^2 + 4.1065\times10^{-6}r^3$	3.87
4	$1.6512r - 4.8465\times10^{-4}r^2 - 1.3108\times10^{-5}r^3 - 2.5544\times10^{-8}r^4$	3.31

表 3

多項式階數	校正函數 $f^{-1}(r')$	取樣點平均誤差
2	$3.5901\times10^{-1}r'+1.9320\times10^{-3}r'^2$	6.80
3	$5.8830\times10^{-1}r'-1.1839\times10^{-4}r'^2+4.2814\times10^{-6}r'^3$	4.43
4	$8.3215\times10^{-1}r'-4.0733\times10^{-3}r'^2+2.3301\times10^{-5}r'^3-2.8171\times10^{-8}r'^4$	4.35

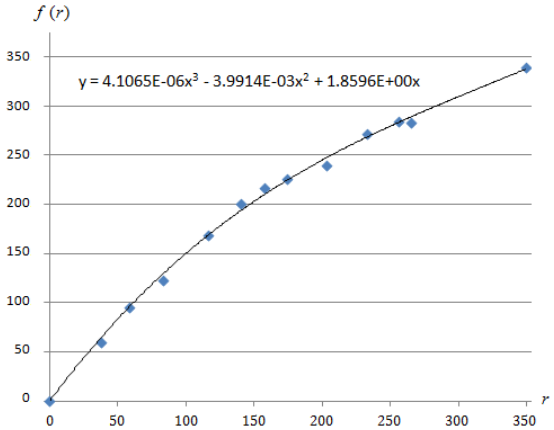


圖 13-1 三階多項式成像函數 $f(r)$

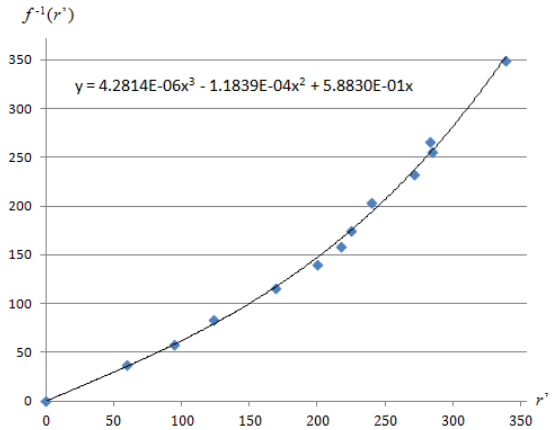


圖 13-2 三階多項式校正函數 $f^{-1}(r')$

四、模擬超廣角鏡頭的圖形導覽之方法

雖然利用超廣角鏡頭的成像函數，可將原始影像產生類似超廣角鏡頭變形的效果，但是其變形的作用範圍為整張影像，放大的部分也僅限於影像的中心區域，無法移動到其他的區域做細節上的導覽。

我們所構想的圖形導覽方式，如圖 14 所示，是希望在導覽的圖形上有一個圓形區域，該區域可隨著使用者的意願而自由移動，而區域內的圖形，除了是以模擬超廣角鏡頭成像的方式呈現外，而且能夠與圓形區域外的圖形做銜接，如此，在瀏覽圖形式資訊的時候，除能夠顯示整體的結構外，也可以不開啟新視窗及無遮蔽的方式，即時地將想要觀察的部分做局部放大以展現細節。

圖 14 模擬超廣角鏡頭的圖形導覽之方法

要達到我們構想中的導覽方法，必須修正成像函數 $f(r)$，使其定義域與

值域都介於[0, R]區間，R 是超廣角鏡頭涵蓋之圓形導覽區域的半徑。由於 R 是可變動的，會隨著使用者的喜好而進行調整，所以比較好的做法是先求出正規化後的成像函數，其定域與值域都介於[0,1]區間，然後依 R 值做等比例的放大即可。圖 15 為正規化後的三階多項式成像函數，是先將各取樣點及其成像點之極座標中的 r 與 r' 進行正規化處理後，再利用 *Excel* 的趨勢線求得。

表 4

取樣點編號	r		r'	
	原數據	正規化	原數據	正規化
1	58.1	0.120	94.8	0.253
2	158.0	0.326	217.2	0.579
3	232.6	0.479	271.5	0.723
4	174.4	0.360	225.4	0.600
5	83.2	0.171	123.3	0.329
6	203.1	0.419	239.9	0.639
7	265.3	0.547	283.0	0.754
8	255.7	0.527	284.7	0.758
9	116.3	0.240	169.0	0.450
10	140.0	0.289	200.2	0.533
11	349.1	0.720	339.0	0.903
12	37.5	0.077	59.4	0.158

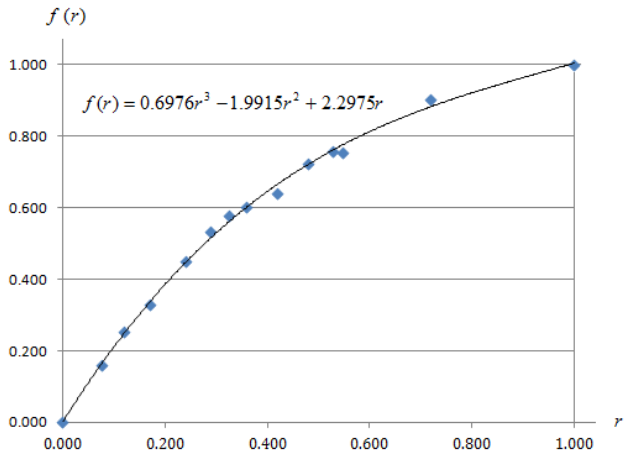


圖 15 正規化後的三階多項式成像函數

有了正規化後的成像函數後，即可在圓形導覽區域內產生模擬超廣角鏡頭成像的圖形呈現方式，其步驟如下，其中 R 為圓形區域的半徑，O 為圓形

區域的中心點，$f(r)$ 為正規化後的超廣角鏡頭成像函數。

- 針對圓形區域內的每一點 $p(x,y)$，推算出以 O 為中心點的極座標 $p(r,\theta)$。
- 將 p 與 O 的距離 r，進行正規化為 r/R。
- 將 r/R 代入正規化後的成像函數 $f(r)$，得到 $f(r/R)$。
- $R\times f(r/R)$ 即為成像點到 O 的距離 r'。
- 將成像點的極座標為 $p'(r',\theta)$；轉換為平面直角坐標 $p'(x',y')$。

五、原型程式實作

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 16 的表單中左邊導覽控制區內的 ComboBox 可以選擇導覽圖形；TrackBar 可以控制圓形導覽區域的半徑，PictureBox 顯示縮小的全圖，右邊的 PictureBox 則是模擬超廣角鏡頭的圖形導覽介面，利用滑鼠事件 MouseDown 來觸發程式的運算，使得圓形導覽區域可隨著滑鼠位置的改變而移動。

圖 16 模擬超廣角鏡頭的圖形導覽之方法

六、程式的修正

由圖 16 中可以看到「臺北捷運路線圖」上的文字會有字體變形的情況，變形扭曲的文字容易使瀏覽者在判讀時造成困擾或錯誤，如圖中的「大橋頭」、「劍潭」、「中山國小」等。此外，圓形導覽區域中圖形的放大率是固定的，有時這固定的放大率仍無法滿足觀察資訊細節的需求。

1. 修正字體變形的問題

線性函數可以使影像做等比例的縮放，不會有字體變形的問題，若能如圖 17 所示將原來的成像函數與線性函數相結合，那麼字體變形的問題應該就可以獲得改善。若 為導覽區域中心部分以等比例呈現的半徑，則當 時，採用線性函數將影像做等比例的放大；當 時，採用原來的函數來成像。

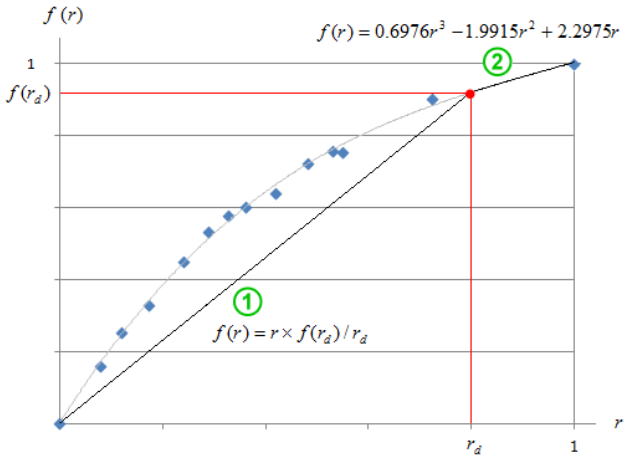


圖 17 原成像函數與線性函數結合後的函數

圖 18 為修正字體變形後的執行畫面，圓形導覽區域中心部分的字體變形已獲得改善。

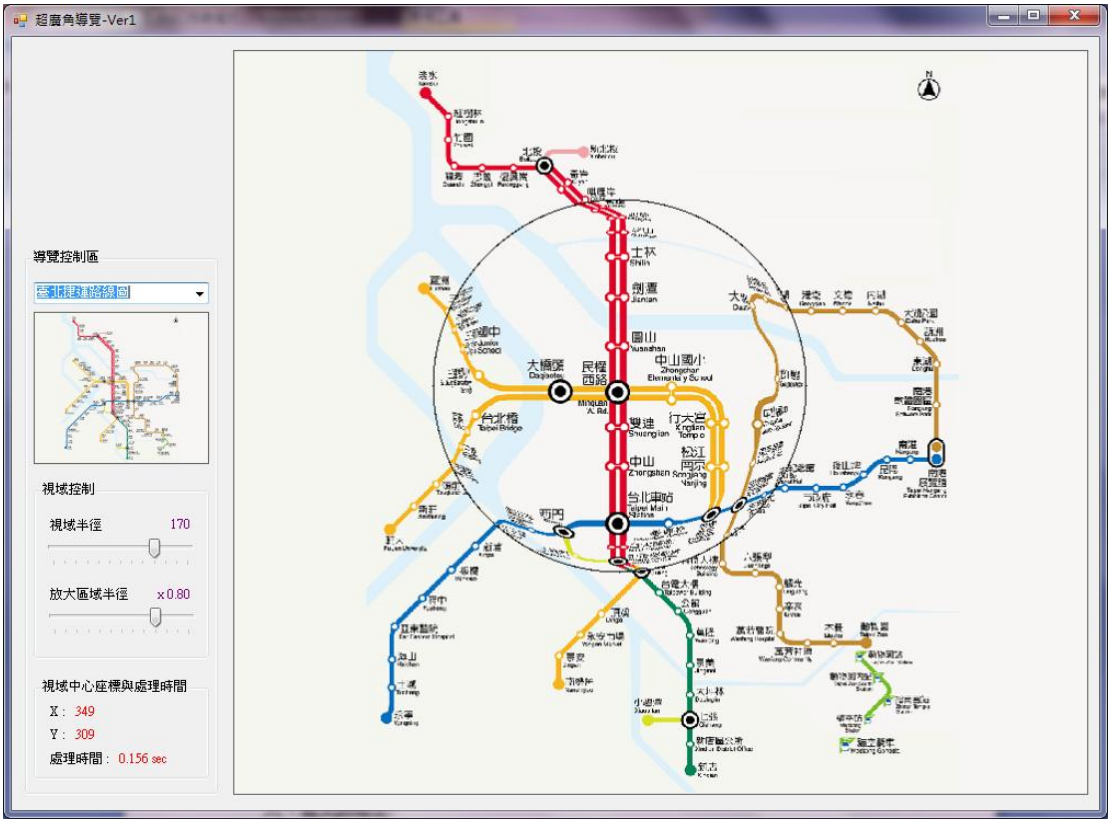


圖 18 修正字體變形後的圖形導覽

2. 修正中心區域放大倍率的問題

我們另外發現到，圓形導覽區域中圖形雖然有放大了，但是其放大倍

率是固定的，有時這固定的放大率仍無法滿足觀察資訊細節的需求。為了滿足使用者可以自行調整導覽區域中心部分的放大倍率，我們設計了如圖 19 所示的成像函數，假設導覽區域中心部分的放大倍率為 s，導覽區域中心部的半徑為 r_d，則當 $0 \le r \le r_d$ 時，採用線性函數 $f(r) = s \times r$ 將影像做等比例的放大；當 $r_d < r \le 1$ 時，採用線性函數 $f(r) = r \times (1 - s \times r_d)/(1 - r_d)$，以等比例的方式將導覽區域與外部的圖形做銜接，當然，為了不超出導覽區域的邊界，必須 $s \times r_d < 1$。

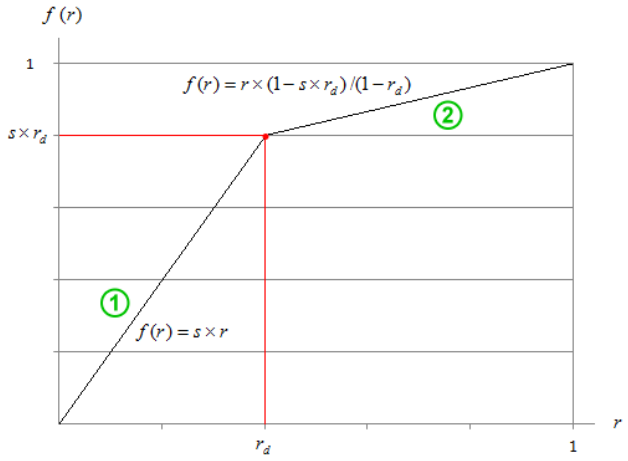


圖 19 可調整中心區域放大倍率的函數

圖 20 為修正導覽區域中心部分字體變形扭曲及可調整放大倍率後程式的執行畫面，導覽區域中心部分的字體已較圖 18 中大了許多。

圖 20 修正可調整中心區域放大倍率後的圖形導覽

肆、研究結果

一、Windows Form 視窗應用程式實作

Windows Form 是最基礎的視窗應用程式，我們以 Visual Basic 2008 來實作，左邊是導覽控制區，右邊是模擬超廣角鏡頭的圖形導覽介面。控制區內的 ComboBox 可以選擇導覽圖，三個 TrackBar 分別可以控制圓形導覽區域的半徑、中心放大區域的比例及放大倍率。圖形導覽介面是採用 PictureBox 實作，利用滑鼠事件 MouseDown 來觸發程式的運算，使得圓形導覽區域可隨著滑鼠位置的改變而移動。

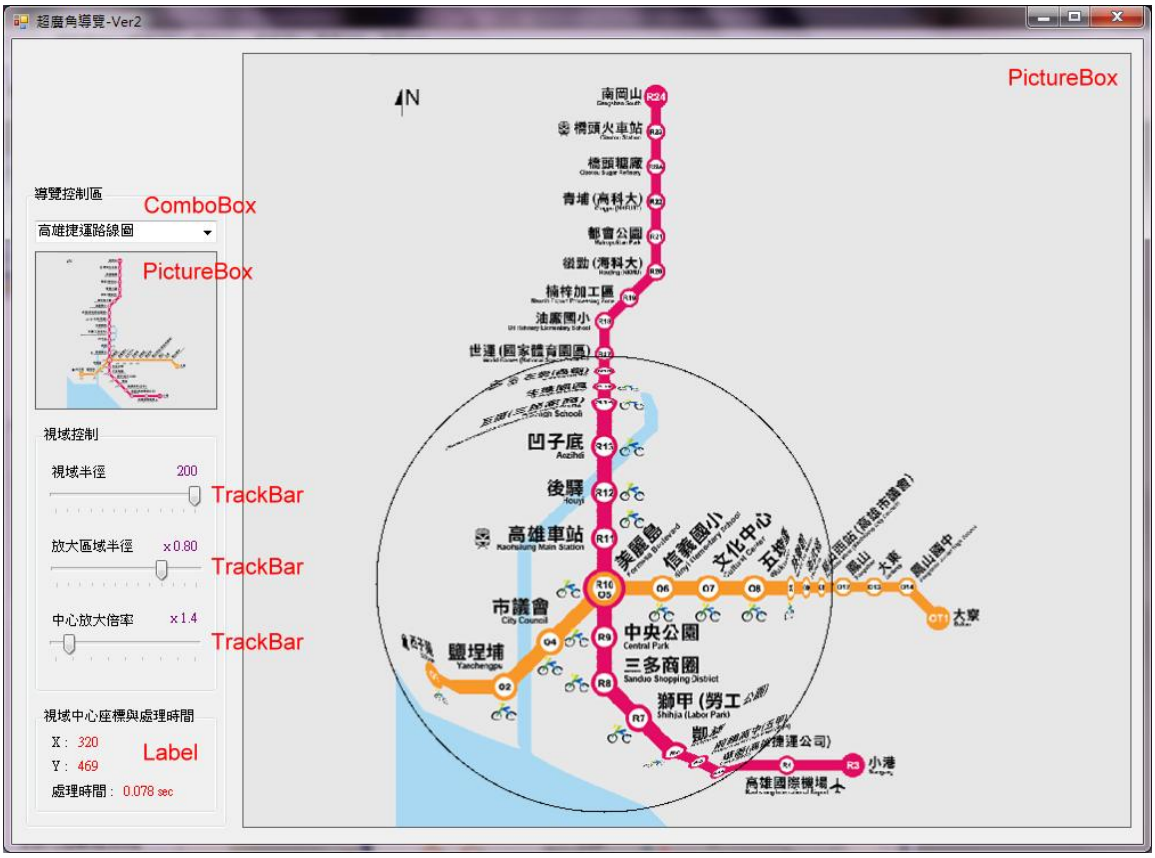


圖 21-1 高雄捷運路線導覽圖

圖 21-2 大雪山森林遊樂區導覽圖

圖 21-3 太平山森林遊樂區導覽圖

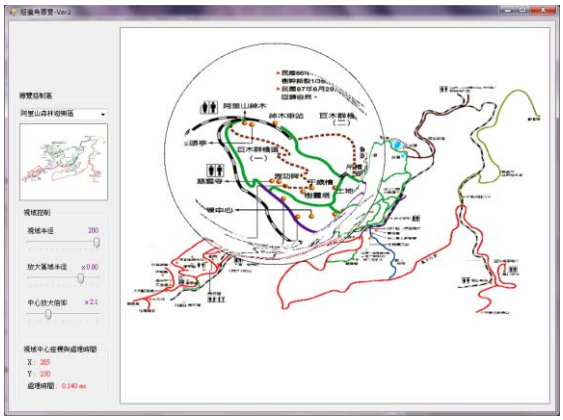


圖 21-4 阿里山森林遊樂區導覽圖

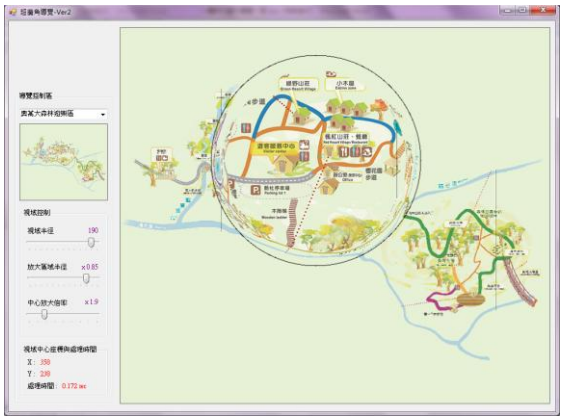


圖 21-5 奧萬大山森林遊樂區導覽圖

二、WPF 瀏覽器應用程式實作

WPF（Windows Presentation Foundation）是微軟公司開發用來強化應用程式使用者介面與多媒體內容顯示的視覺效果，讓開發人員在一致性的設計模式中建置整合多媒體和文件。在 WPF 開發平台上，它的核心應用程式開發功能包含 XAML、控制項、2D 圖形、3D 圖形、動畫、媒體、資源、配置、文件、資料繫結、安全性、能夠充分運用新圖形顯示硬體的功能，發揮處理圖形資料的效能，有效降低耦合度，讓開發人員各司其職分工合作，以降低系統更新及維護成本。

WPF 分為二種專案應用程式型態，視窗應用程式及瀏覽應用程式，因此就可以解決視窗應用程式一個版本，網路應用程式又是另一個版本而造成浪費許多時間重複撰寫相同邏輯程式碼的時間，讓開發者不用為了部署成 Windows Applications 和 Browser Applications 成兩個版本撰寫。我們嘗試著將圖形導覽程式改寫為 WPF 瀏覽器應用程式，這樣即可利用現今電腦平台的瀏覽器來執行超廣角圖形導覽的功能。

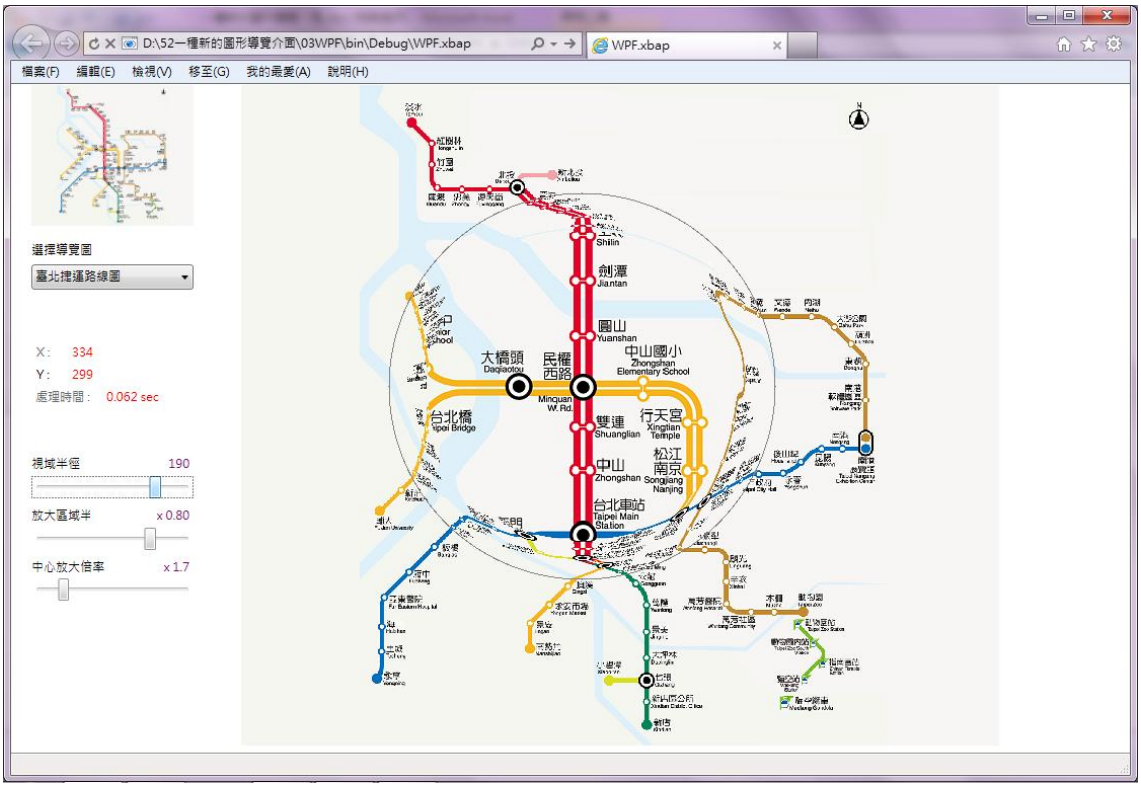


圖 22 運用 WPF 瀏覽器應用程式實作的圖形導覽介面

伍、討論及結論

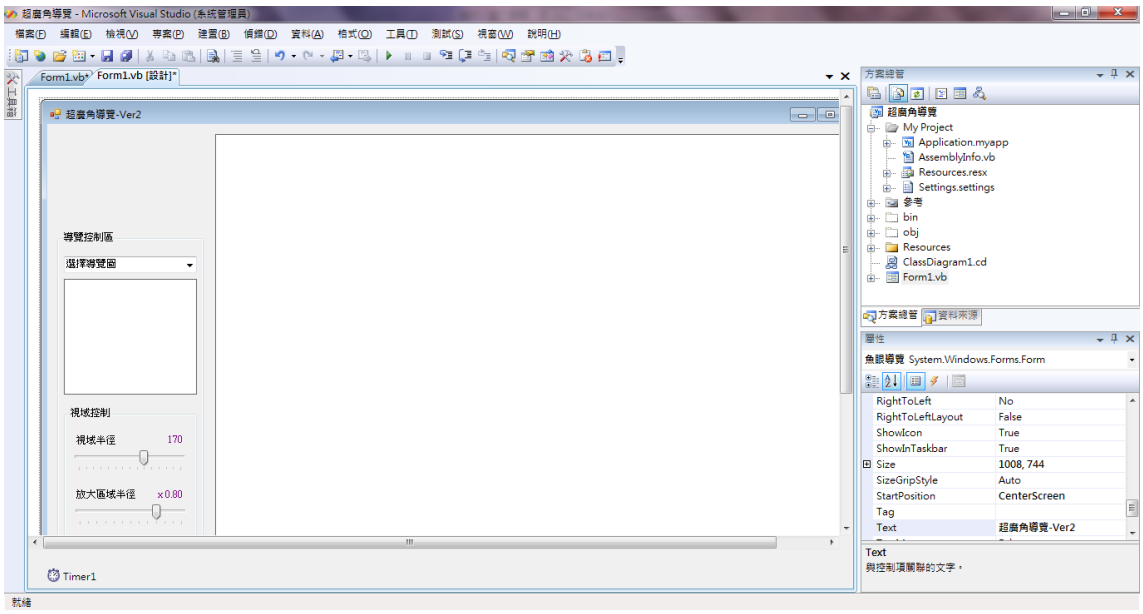
產生一幅780×680像素大小的導覽圖形，在CPU為Intel Core i5-560 2.66GHz的筆電上執行，平均約花費0.08秒，在操作圓形導覽區域的移動時，雖然還算可以跟得上操作者的步伐，但若能更順暢些就好了，查詢過相關資料，這種處理速度，幾乎已經是透過繪圖裝置介面GDI+類別來設計圖形處理程式的最快極限，解決之道，就是採用DirectX，DirectX以不透過Windows API的方式，盡可能的直接操作硬體，來達成高速執行運作。目前DirectX對我們來說是一項很困難的挑戰，不過相信在將來終有完成的一日。

現今最熱門的通訊裝置，如智慧型手機、平板電腦、輕薄型筆電等，圖形式資訊在其系統中扮演了非常重要的角色，這類型的設備為了輕巧與攜帶方便，通常螢幕的空間都非常有限，相信本研究所發展出來的圖形導覽介面，會有很好的發揮空間。

陸、參考資料

1. 董大偉、許雅婷著,“Visual Basic 2005 程式設計與案例剖析,” 旗標出版股份有限公司, 2006。
2. 鍾國亮著,“影像處理與電腦視覺,” 台灣東華書局股份有限公司, 2006。
3. 繆紹綱譯,“數位影像處理,” 台灣培生教育出版股份有限公司, 2003。
4. Manojit Sarkar, and Marc H. Brown., “Graphical fisheye views,” Communications of the ACM, 37(12), pp. 73-84, 1994.。

附錄：一種新的圖形導覽介面原始程式碼



附錄圖 1 程式設計畫面

```
' 引用類別庫

Imports System.Drawing

Imports System.Drawing.Imaging

Imports System

Imports System.Windows.Forms


Public Class 超廣角導覽

    Const intervalLimit = 50

    Const distanceLimit = 5


    Dim viewScale As Double        ' 圓形導覽區域放大倍率

    Dim innerRatio As Double

    Dim viewRadius As Integer      ' 圓形導覽區域半徑

    Dim srcImg, targetImg, viewImg As Bitmap

    Dim MouseX, MouseY, PreMouseX, PreMouseY As Integer

    Dim tmpDistance As Double


    ' 產生原始無變形導覽圖型

    Public Function makeTargetImg(ByVal inImg As Bitmap, ByVal outImgWidth As Integer, ByVal
      outImgHeight As Integer) As Bitmap
```

```
        Dim outImg As New Bitmap(outImgWidth, outImgHeight, PixelFormat.Format24bppRgb)

        Dim inData As BitmapData = inImg.LockBits(New Rectangle(0, 0, inImg.Width, inImg.Height),
            ImageLockMode.ReadOnly, PixelFormat.Format24bppRgb)

        Dim outData As BitmapData = outImg.LockBits(New Rectangle(0, 0, outImg.Width, outImg.Height),
            ImageLockMode.ReadWrite, PixelFormat.Format24bppRgb)

        Dim ptrIn As IntPtr = inData.Scan0

        Dim ptrOut As IntPtr = outData.Scan0

        Dim bytesIn, bytesOut As Long

        Dim rgbvalIn(), rgbvalOut() As Byte

        bytesIn = inData.Stride * inImg.Height

        ReDim rgbvalIn(bytesIn - 1)

        System.Runtime.InteropServices.Marshal.Copy(ptrIn, rgbvalIn, 0, bytesIn)

        bytesOut = outData.Stride * outImg.Height

        ReDim rgbvalOut(bytesOut - 1)

        System.Runtime.InteropServices.Marshal.Copy(ptrOut, rgbvalOut, 0, bytesOut)

        Dim col, row, getX, getY As Integer

        Dim R, G, B As Integer

        Dim ratio As Double = inImg.Width / outImgWidth

        For row = 1 To outImg.Height

            For col = 1 To outImg.Width

                getX = Int(col * ratio)

                getY = Int(row * ratio)

                R = rgbvalIn(inData.Stride * (getY - 1) + (getX - 1) * 3 + 2)

                G = rgbvalIn(inData.Stride * (getY - 1) + (getX - 1) * 3 + 1)

                B = rgbvalIn(inData.Stride * (getY - 1) + (getX - 1) * 3 + 0)

                rgbvalOut(outData.Stride * (row - 1) + (col - 1) * 3 + 2) = R

                rgbvalOut(outData.Stride * (row - 1) + (col - 1) * 3 + 1) = G

                rgbvalOut(outData.Stride * (row - 1) + (col - 1) * 3) = B

            Next

        Next

        System.Runtime.InteropServices.Marshal.Copy(rgbvalOut, 0, ptrOut, bytesOut)  ' Copy the RGB
            values back to the outImg

        inImg.UnlockBits(inData)  ' Unlock the bits.

        outImg.UnlockBits(outData)

        Return outImg

    End Function
```

```
' 產生超廣角導覽圖型

Public Function fisheyeImg(ByVal inImg1 As Bitmap, ByVal inImg2 As Bitmap, ByVal mouseX As Integer,
  ByVal mouseY As Integer, ByVal viewRadius As Integer, ByVal viewScale As Double) As Bitmap

    Dim outImg As New Bitmap(inImg2.Width, inImg2.Height, PixelFormat.Format24bppRgb)

    Dim inData1 As BitmapData = inImg1.LockBits(New Rectangle(0, 0, inImg1.Width, inImg1.Height),
      ImageLockMode.ReadOnly, PixelFormat.Format24bppRgb)

    Dim inData2 As BitmapData = inImg2.LockBits(New Rectangle(0, 0, inImg2.Width, inImg2.Height),
      ImageLockMode.ReadOnly, PixelFormat.Format24bppRgb)

    Dim outData As BitmapData = outImg.LockBits(New Rectangle(0, 0, outImg.Width, outImg.Height),
      ImageLockMode.ReadWrite, PixelFormat.Format24bppRgb)

    Dim ptrIn1 As IntPtr = inData1.Scan0

    Dim ptrIn2 As IntPtr = inData2.Scan0

    Dim ptrOut As IntPtr = outData.Scan0

    Dim bytesIn1, bytesIn2, bytesOut As Long

    Dim rgbvalIn1(), rgbvalIn2(), rgbvalOut() As Byte

    bytesIn1 = inData1.Stride * inImg1.Height

    bytesIn2 = inData2.Stride * inImg2.Height

    ReDim rgbvalIn1(bytesIn1 - 1)

    ReDim rgbvalIn2(bytesIn2 - 1)

    System.Runtime.InteropServices.Marshal.Copy(ptrIn1, rgbvalIn1, 0, bytesIn1)

    System.Runtime.InteropServices.Marshal.Copy(ptrIn2, rgbvalIn2, 0, bytesIn2)

    bytesOut = outData.Stride * outImg.Height

    ReDim rgbvalOut(bytesOut - 1)

    System.Runtime.InteropServices.Marshal.Copy(ptrOut, rgbvalOut, 0, bytesOut)


    Dim col, row As Integer

    Dim R, G, B As Integer

    Dim innerRadius As Integer = CInt(innerRatio * viewRadius)

    Dim nowRadius, getRadius As Double

    Dim difX, difY As Integer

    Dim raidX, raidY As Double

    Dim ratio As Double = inImg1.Width / outImg.Width 'srcImg 與 targetImg 比例

    Dim srcCenterX, srcCenterY, getCol, getRow As Integer


    srcCenterX = CInt(mouseX * inImg1.Width / outImg.Width)

    srcCenterY = CInt(mouseY * inImg1.Width / outImg.Width)


    For row = 1 To outImg.Height
```

```
For col = 1 To outImg.Width
    difX = col - mouseX
    difY = mouseY - row
    nowRadius = Math.Sqrt(difX ^ 2 + difY ^ 2)
    If nowRadius > viewRadius + 1 Then '--- view 外部區域
        R = rgbvalIn2(inData2.Stride * (row - 1) + (col - 1) * 3 + 2)
        G = rgbvalIn2(inData2.Stride * (row - 1) + (col - 1) * 3 + 1)
        B = rgbvalIn2(inData2.Stride * (row - 1) + (col - 1) * 3 + 0)
    Else
        If nowRadius <= innerRadius And nowRadius > 0 Then '--- view 中心區域
            raidX = Math.Acos(difX / nowRadius) '極座標角度
            raidY = Math.Asin(difY / nowRadius) '極座標角度
            getRadius = nowRadius / viewScale * ratio
            getCol = srcCenterX + CInt(getRadius * Math.Cos(raidX))
            getRow = srcCenterY - CInt(getRadius * Math.Sin(raidY))
            R = rgbvalIn1(inData1.Stride * (getRow - 1) + (getCol - 1) * 3 + 2)
            G = rgbvalIn1(inData1.Stride * (getRow - 1) + (getCol - 1) * 3 + 1)
            B = rgbvalIn1(inData1.Stride * (getRow - 1) + (getCol - 1) * 3 + 0)
        Else
            If nowRadius < viewRadius And nowRadius > innerRadius Then '--- view 連接區
域
                raidX = Math.Acos(difX / nowRadius) '極座標角度
                raidY = Math.Asin(difY / nowRadius) '極座標角度
                getRadius = ((nowRadius - innerRadius) * (viewRadius - innerRadius /
viewScale) / (viewRadius - innerRadius) + (innerRadius / viewScale)) * ratio
                getCol = srcCenterX + CInt(getRadius * Math.Cos(raidX))
                getRow = srcCenterY - CInt(getRadius * Math.Sin(raidY))
                R = rgbvalIn1(inData1.Stride * (getRow - 1) + (getCol - 1) * 3 + 2)
                G = rgbvalIn1(inData1.Stride * (getRow - 1) + (getCol - 1) * 3 + 1)
                B = rgbvalIn1(inData1.Stride * (getRow - 1) + (getCol - 1) * 3 + 0)
            Else
                If nowRadius <= viewRadius + 1 And nowRadius >= viewRadius Then '---
view 邊界
                    R = 0
                    G = 0
                    B = 0
                Else
```

```
                                If nowRadius = 0 Then '--- view 中心點
                                    getCol = srcCenterX
                                    getRow = srcCenterY
                                    R = rgbvalIn1(inData1.Stride * (getRow - 1) + (getCol - 1) * 3 + 2)
                                    G = rgbvalIn1(inData1.Stride * (getRow - 1) + (getCol - 1) * 3 + 1)
                                    B = rgbvalIn1(inData1.Stride * (getRow - 1) + (getCol - 1) * 3 + 0)
                                End If
                            End If
                        End If
                    End If
                    rgbvalOut(outData.Stride * (row - 1) + (col - 1) * 3 + 2) = R
                    rgbvalOut(outData.Stride * (row - 1) + (col - 1) * 3 + 1) = G
                    rgbvalOut(outData.Stride * (row - 1) + (col - 1) * 3) = B
                Next
            Next
            System.Runtime.InteropServices.Marshal.Copy(rgbvalOut, 0, ptrOut, bytesOut)  ' Copy the RGB values back to the outImg
            inImg1.UnlockBits(inData1)  ' Unlock the bits.
            inImg2.UnlockBits(inData2)  ' Unlock the bits.
            outImg.UnlockBits(outData)
            Return outImg
        End Function

        ' 選擇導覽圖
        Private Sub cobSelect_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles cobSelect.SelectedIndexChanged
            Select Case cobSelect.SelectedItem
                Case "臺北捷運路線圖"
                    srcImg = My.Resources.臺北捷運路線圖
                Case "高雄捷運路線圖"
                    srcImg = My.Resources.高雄捷運路線圖
                Case "大雪山森林遊樂區"
                    srcImg = My.Resources.大雪山森林遊樂區
                Case "太平山森林遊樂區"
                    srcImg = My.Resources.太平山森林遊樂區
```

```
            Case "阿里山森林遊樂區"
                srcImg = My.Resources.阿里山森林遊樂區
            Case "奧萬大森林遊樂區"
                srcImg = My.Resources.奧萬大森林遊樂區
        End Select
        targetImg = makeTargetImg(srcImg, picNavigate.Width, picNavigate.Height)
        viewRadius = 170
        tbRadius.Value = viewRadius / 10
        lblRadius.Text = viewRadius
        innerRatio = 0.8
        tbinnerRatio.Value = CInt(innerRatio * 100)
        lblinnerRatio.Text = "x " & Format(innerRatio, "0.00")
        viewScale = Math.Round(srcImg.Width / targetImg.Width, 1)
        If viewScale > 5 Then viewScale = 5
        If viewScale < 1 Then viewScale = 1
        tbScale.Value = viewScale * 10
        lblScale.Text = "x " & Format(viewScale, "#.0")
        picShrink.Image = targetImg
        picNavigate.Image = targetImg
    End Sub

    ' 程式開始執行時
    Private Sub 超廣角導覽_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Load
        viewRadius = 170
        tbRadius.Value = viewRadius / 10
        lblRadius.Text = viewRadius
        innerRatio = 0.8
        tbinnerRatio.Value = CInt(innerRatio * 100)
        lblinnerRatio.Text = "x " & Format(innerRatio, "0.00")
        viewScale = 1
        tbScale.Value = viewScale * 10
        lblScale.Text = "x " & Format(viewScale, "#.0")
        lblX.Text = 0
        lblY.Text = 0
        PreMouseX = -10
```

```
        PreMouseY = -10
        Timer1.Interval = intervalLimit
        Timer1.Enabled = False
    End Sub

    ' 變更導覽區域中心放大倍率
    Private Sub tbScale_Scroll(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles tbScale.Scroll
        viewScale = CDbl(tbScale.Value / 10)
        lblScale.Text = "x " & Format(viewScale, "#.0")
    End Sub

    ' 變更導覽區域中心放大區域的半徑
    Private Sub tbinnerRatio_Scroll(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles tbinnerRatio.Scroll
        innerRatio = CDbl(tbinnerRatio.Value / 100)
        lblinnerRatio.Text = "x " & Format(innerRatio, "0.00")
    End Sub

    ' 變更導覽區域半徑
    Private Sub tbRadius_Scroll(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles tbRadius.Scroll
        viewRadius = CInt(tbRadius.Value * 10)
        lblRadius.Text = viewRadius
    End Sub

    ' 滑鼠離開圖形導覽介面範圍
    Private Sub picNavigate_MouseLeave(ByVal sender As Object, ByVal e As System.EventArgs) Handles picNavigate.MouseLeave
        lblX.Text = 0
        lblY.Text = 0
        picNavigate.Image = targetImg
        picShrink.Image = targetImg
        PreMouseX = -10
        PreMouseY = -10
        If Timer1.Enabled = True Then Timer1.Enabled = False
    End Sub
```

```
    ' 滑鼠於圖形導覽介面範圍移動
    Private Sub picNavigate_MouseMove(ByVal sender As Object, ByVal e As
      System.Windows.Forms.MouseEventArgs) Handles picNavigate.MouseMove
        MouseX = CInt(e.X)
        MouseY = CInt(e.Y)
        lblX.Text = MouseX
        lblY.Text = MouseY
        If Timer1.Enabled = False Then Timer1.Enabled = True
    End Sub

    ' 時間片段計時終了
    Private Sub Timer1_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
      Timer1.Tick
        tmpDistance = ((MouseX - PreMouseX) ^ 2 + (MouseY - PreMouseY) ^ 2) ^ 0.5
        If targetImg IsNot Nothing And tmpDistance > distanceLimit Then
            Timer1.Enabled = False
            Dim startTime As DateTime = DateTime.Now()
            viewImg = fisheyeImg(srcImg, targetImg, MouseX, MouseY, viewRadius, viewScale)
            picNavigate.Image = viewImg
            PreMouseX = MouseX
            PreMouseY = MouseY
            Dim endTime As DateTime = DateTime.Now()
            Timer1.Enabled = True
            Dim spendTime As TimeSpan = endTime.Subtract(startTime)
            lblSpend.Text = String.Format("{0:n3}", spendTime.TotalMilliseconds / 1000) & " sec"
        End If
    End Sub

End Class
```

評語

作品有很好的創意及實用性，有機會推廣成流行的應用軟體元件。但互動的設計尚待加強，最好能夠在 iPad 及 iPhone 上來呈現作品，會比較吸睛。

I. Introduction

Magnification of an image on a computer or handheld device is typically carried out in one of two ways: (1) by enlarging the entire image proportionally, resulting in only part of the image being displayed; or (2) by enlarging a small region of the image centered at the point-of-interest, resulting in neighboring region being obstructed. In either way, disorientation becomes a problem when searching through different parts of the image. Some cognitive loads are required to remember the relative position of the magnified region relative to the entire image. As a result, in this project we developed and tested a fish-eye lens inspired image magnification method that can enlarge the region of interest without occluding neighboring regions.

II. Fish-eye Lens Visual Characteristics

Fig. 1

Our idea comes from pictures of dogs (Fig. 1) that have big heads but small bodies. We find out that they were shot by fish-eye lenses. Fish-eye lenses provide extremely wide angle of view, and it is used for taking hemispherical images. (Fig. 2(a)) The fish-eye lenses captured image within the field of view is distorted spherically. (Fig. 2(b)) Magnification power increases gradually toward center of the image, the center region is shown much larger than the neighboring regions. (Fig. 2(c)) This feature provides an image to view the details and show the overall structure simultaneously.

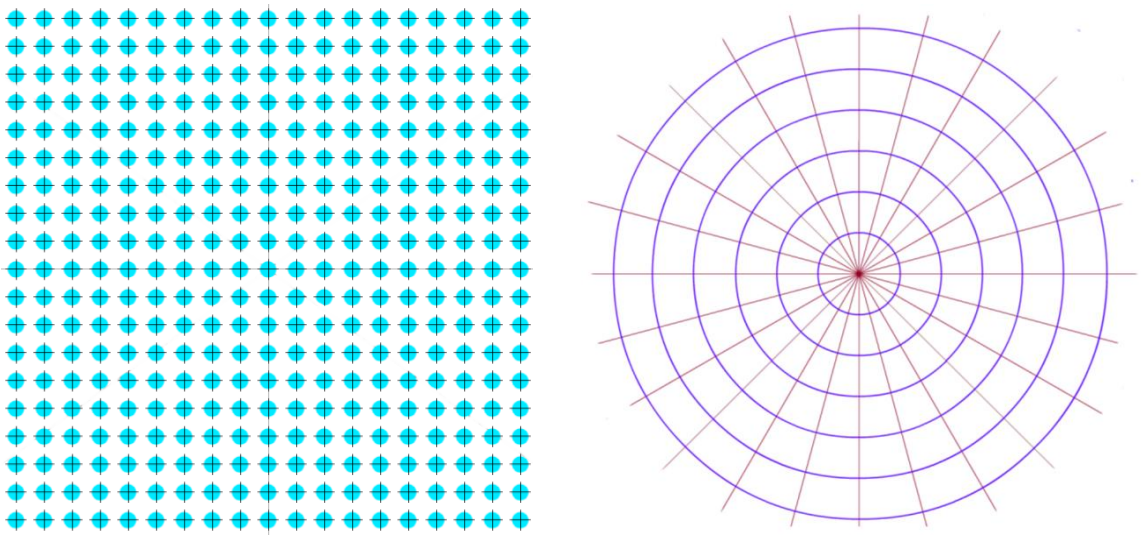
Fig. 2 (a)

Fig. 2(b)

Fig. 2(c)

III. Physical Distortion Property

To observe the characters of fish-eye lenses; first, we devise a plan paper (a) with regularly spaced grid points on it, and shot it by the fish-eye lens. Through the observation, we devised a second plan paper, which has (b) equally spaced concentric circles and lines passing through the origin.



(a) (b)

Fig. 3 Plan papers

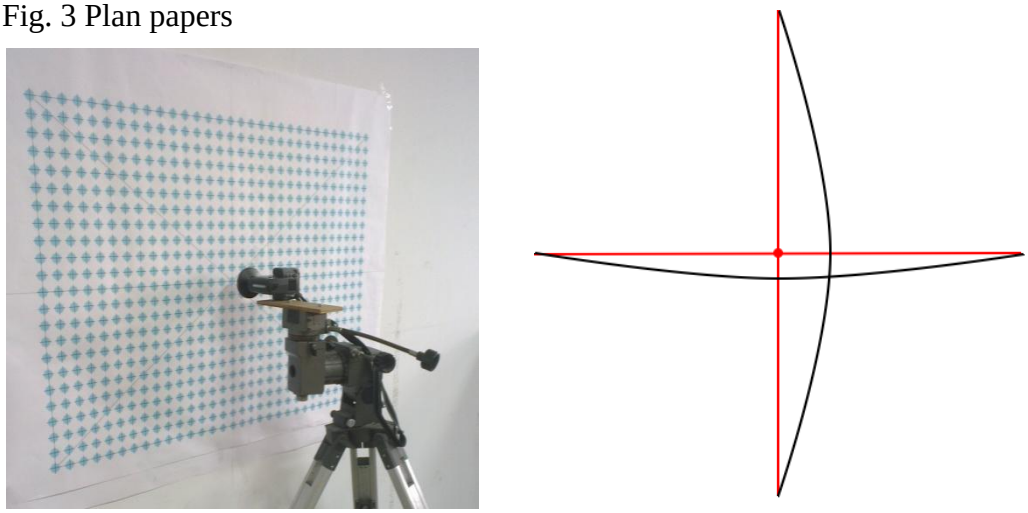
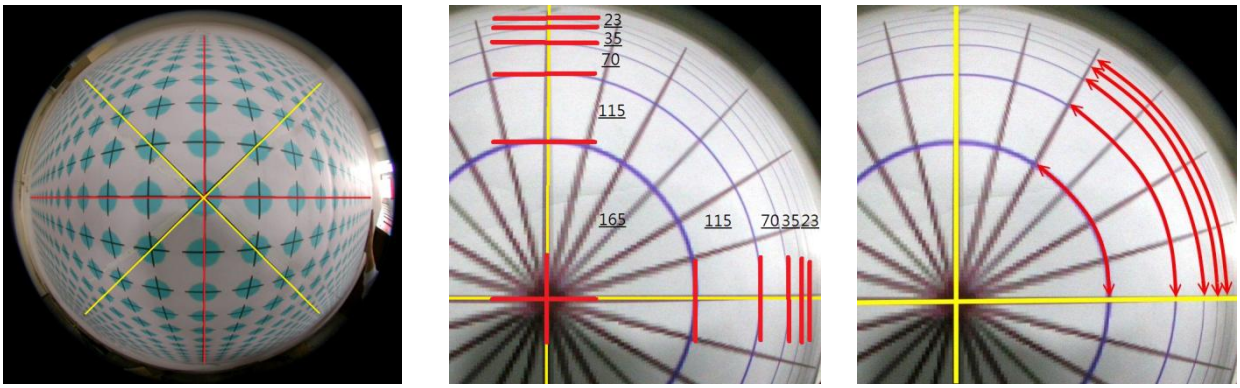


Fig. 4 (a) apparatus of photographing (b) correct the center of an fish-eye image

When captured with a fish-eye lens, the resulting images showed that:

1. distortion is symmetric along the optical axis and decreases non-linearly away from origin. (Fig. 5(a))
2. the extent of bow gets greater and the distance between two neighboring concentric circles gets shorter as the concentric circles move away from the center. (Fig. 5(b))
3. the degree angle along the same radial line remain the same. (Fig. 5(c))



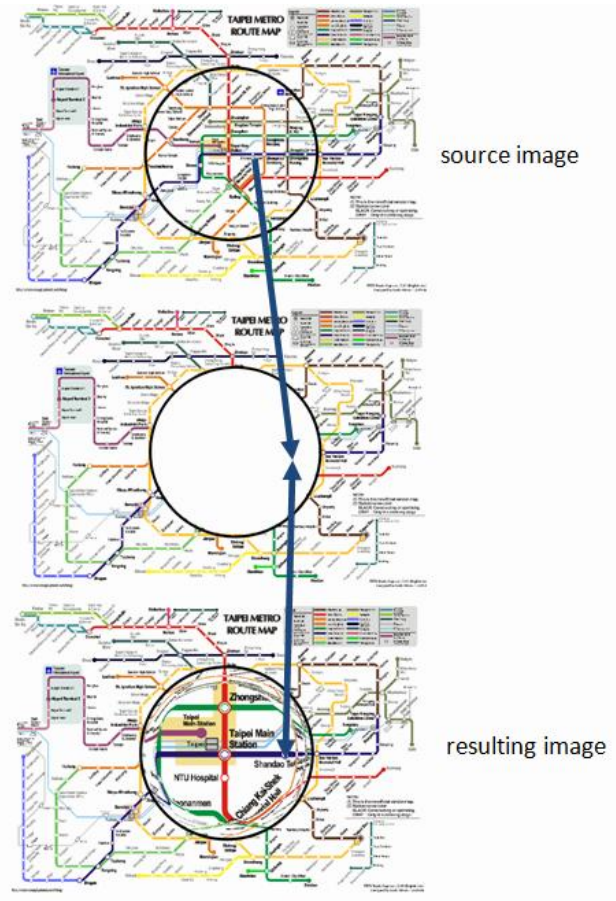
(a) (b) (c)

Fig. 5 Distortions on images captured by fish-eye lens

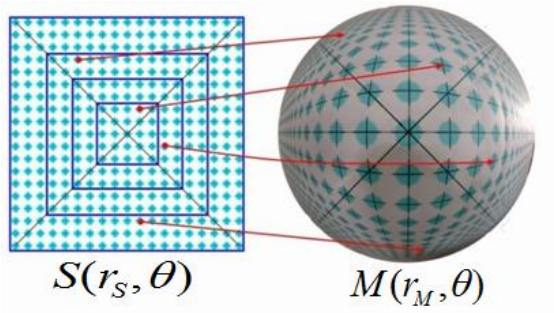
IV. Derivation of Fish-eye Lens Inspired Magnification Method

To find out the mapping function that emulates a fish-eye lens, we map pixels in a source image to resulting image, and derive F^{-1} from F to allow for inverse mapping of a pixel in source image for any point in resulting image. Polar Coordinates system is used to represent each point, because the angles remain unchanged after transformation.

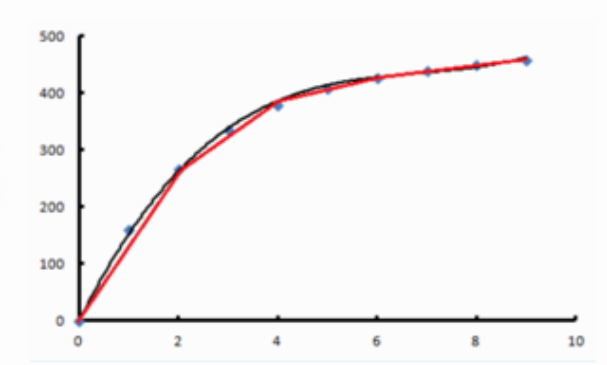
1. Procedure:



Experimentations are carried out to discuss how many sample points be chose can absolutely represent the rest of the data points. We separate all data points on the plan paper into four parts according to the changing distance between the points. Then, data points are sampled proportionally from four concentric regions. Results showed that 52 data points were experimentally determined to be suitable for matching image points from source images to resulting images.



(a) mapping between source image and resulting image



(b) distances change between the points in four parts

Fig. 6

And with these data points, we can formulate a mapping function by using least square fitting. The function defined by 52 data points has the least mean absolute error, which is adopted for our application. (Fig. 7)

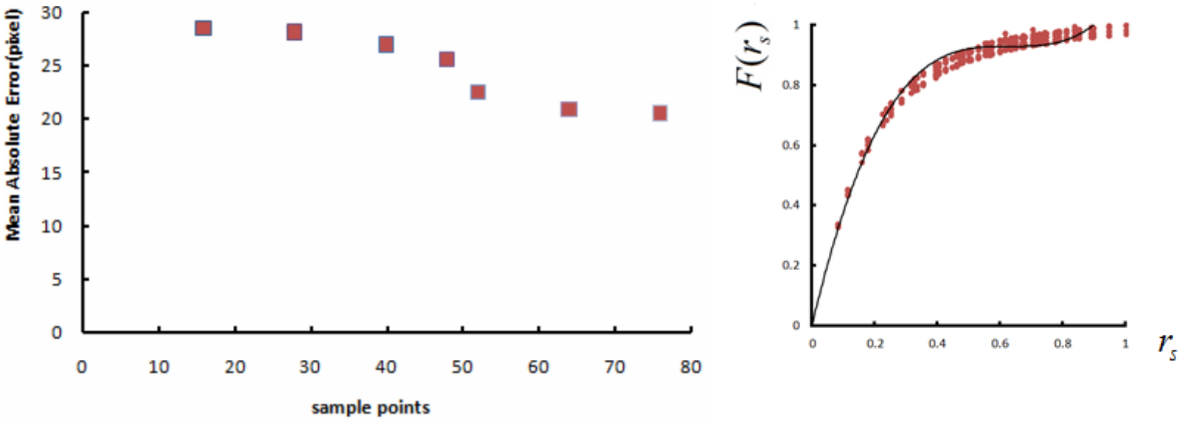
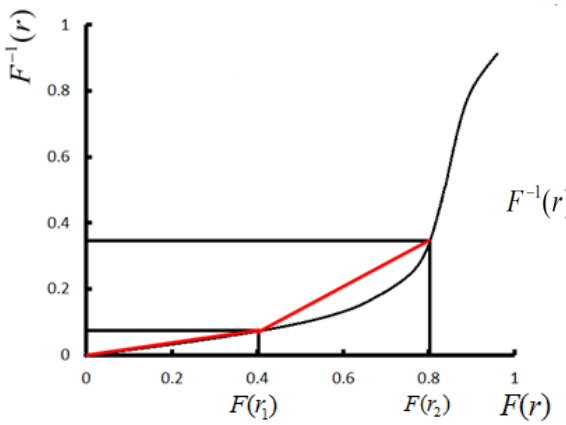


Fig. 7 (a) the least mean absolute error of different number sample points
(b) normalized mapping function

We aim to make a circular region be able to move according to the users' will, and make the magnified region blend seamlessly with the neighboring regions. In order to achieve the goal, we compute the normalize inverse function that reverse the mapping process. That is to map a pixel in the enlarged region back to the original image and retrieve the pixel value.

Through the mapping function, we are able to emulate a fish-eye lens like magnification result. However, the center of the magnified region is spherically distorted, we then constructed a piecewise linear function. The inverse function was converted to a partially piecewise linear function. Divided the function into three segment, this enabled us to magnified the point of interest and allow different magnification power in the circular region. The center of the magnified region has the largest magnification level based on the slope of it and is flatten, while the outer region adopted the original function, hoping to blend in with the rest of the image.



$$F^{-1}(r)=\begin{cases}\dfrac{F^{-1}(r_1)}{F(r_1)}\times F(r), if\ F(r)\le F(r_1)\\ F^{-1}(r_1)+\dfrac{F^{-1}(r_2)-F^{-1}(r_1)}{F(r_2)-F(r_1)}\times(F(r)-F(r_1)), if\ F(r_1)<F(r)<F(r_2)\\ F^{-1}(r), if\ F(r)>F(r_2)\end{cases}$$

Fig. 8 piecewise linear inverse function

(a) (b) (c)

Fig. 9 Exemplary images (a) Original Image.
(b) Fish-eye lens like magnification.
(c) Fish-eye lens like magnification without distortion in central region.

V. Results

Our new magnifier has three important characteristics:

1. central region of magnification is not distorted
2. central region has higher magnification power than the surrounding region

3. seamless integration with rest of the original image.

As oppose to a fish-eye lens captures image, the focused region is flat as with

other magnifiers. It is easier to focus in seek and search activity by the gradually changed magnification power. And user always know where in the image he is magnifying, so less chance of disorientation.

Fig. 10 Comparison between our magnifier and other magnifiers

Lastly, we conducted experiments to compare our new spot magnifier with other magnification methods in seek and search activities with “subway maps” and “Where’s Waldo?” tasks. Participants were to perform the search tasks with both magnifiers. At the end of the search activities, participants were asked to answer questionnaires about the usefulness of two magnifiers. It was observed that participants who used our magnifier first spend less time locating the object to be found. It was also found that seamless integration of the magnified region with the source image helped participants familiarize the given image and not being disoriented due to partial occlusions.

VI. Conclusions

1. By using image processing technique, a fish-eye lens like digital magnifier was

successfully developed.

2. The magnifier has all the advantages of a fish-eye lens image, but no ill-effect of image distortion in the central part of the magnified region.

3. Field experiment showed that with the new magnifier, users are less stressful in locating a familiar object in the given image than a conventional magnifier.

VII. References

[1] Chung, K.-L. (2008). “Introduction to image processing and computer vision,” Dong-Hwang Publisher.

[2] Gonzalez, R.C. and Woods, R.E. (2002). “Digital Image Processing,” 2nd Ed., Prentice-Hall.

[3] Sarkar, M. and Brown, M.H. (1994). “Graphical fisheye views,” CACM, 37(12), 73-84.

[4] Sweller, J., Ayres, P., and Kalyuga, S. (2011). “Cognitive Load Theory,” Springer.