國立交通大學

資訊科學與工程研究所

碩 士 論 文

探 討 彈 性 顯 示 方 式

Exploring The Flexible Display

研 究 生：王國懿

指導教授：李嘉晃 教授

中 華 民 國 九 十 五 年 六 月

探討彈性顯示方式

Exploring The Flexible Display

研 究 生：王國懿　　Student：Kuo-Yi Wang

指導教授：李嘉晃　　Advisor：Chia-Hoang Li

國 立 交 通 大 學

資 訊 科 學 與 工 程 研 究 所

碩 士 論 文

A Thesis

Submitted to Institute of Computer Science and Engineering

College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer Science

June 2006

Hsinchu, Taiwan, Republic of China

中華民國九十五年六月

探討彈性顯示方式

學生 ： 王國懿　　指導教授：李嘉晃 博士

國立交通大學電機資訊學院　資訊科學與工程研究所

中文摘要

如何取代傳統的顯示方式，如 LCD(Liquid Crystal Display)或 CRT(Cathode Ray Tube)，一直是人機互動(Human Computer Interaction)一個很重要的議題。 本論文透過以投影機(Projector)代替傳統顯示方式，透過雙手在投影機投出的影像下模擬滑鼠行為的方式，探討新的顯示方式的可行性、困難和帶給使用者的影響。

本論文以攝影機(Camera) – 投影機(Projector)為基礎的架構，提供一套以雙手模擬滑鼠行為的系統。 系統以投影機代替傳統顯示方式，將傳統顯示器的影像透過投影機投在桌面或牆上，利用攝影機捕捉右手在投影機投出影像中的移動作為滑鼠的游標移動；左手拿著標記(Token)當作滑鼠指令，如:左鍵雙擊、右鍵單擊，透過訊息傳遞方式與微軟系統溝通。透過這樣的方式，探討新的顯示方式的可行性、困難和帶給使用者的影響。

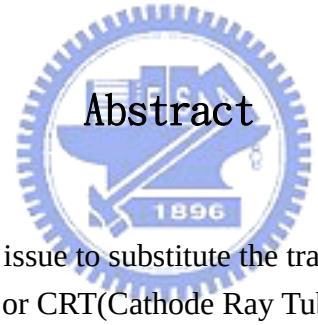
Exploring The Flexible Display

Student ： Kuo-Yi Wang　　Advisor：Prof. Chia-Hoang Lee

Institute of Computer Science and Engineering

College of Computer Science

National Chiao Tung University



Abstract

It is always an important issue to substitute the traditional display, like LCD(Liquid Crystal Display) or CRT(Cathode Ray Tube) in the HCI(Human Computer Interaction). The thesis provides a system which is based on the camera and projector to simulate the mouse actions.

The system uses projector to substitute the traditional displays. It projects the images form the traditional displays on the desk or the wall. The camera catches the motion of the right hand in the image as mouse cursor and tokens in the left hand as the instructions of the mouse, like double click or pop out menu. We can explore the difficulty and convenient for users through this system.

目錄

圖表目錄

第一章 緒論

1.1 研究動機

人機互動(Human Computer Interaction)領域一直致力於讓使用者和電腦之間有更直覺、更便利的互動模式。 其中，如何改良顯示方式，讓使用者擁有更寬敞的空間和更直接與電腦做互動，更是一大熱門話題。 目前主流傳統的顯示方式是利用 LCD(Liquid Crystal Display)或 CRT(Cathode Ray Tube)將畫面呈現出來，透過滑鼠和鍵盤對電腦做互動。 這種顯示方式的缺點是我們無法直接與電腦互動;而且 LCD 和 CRT 占掉大部分使用者使用面積。 為了改良此顯示方式的缺點，大部分人都採用以下兩種解決方法：

i、 利用特製的設備代替傳統的 LCD 或 CRT: Augmented Desk[1]系統利用一個特製的桌子，由桌子下面投出影像到桌面上，再利用攝影機捕捉使用者的動作去做相對應的指令。

ii、利用投影機代替傳統的 LCD 或 CRT： Augmented Surface[2]、AugmentedMaps[3]系統利用投影機將影像投到桌面上後，再利用標記(Token) 或者手對系統下達指令。

兩種方法都改善了傳統顯示方式的缺點，但是第一種方法的缺點在於還是需要特製的設備，也如同 LCD 或 CRT 一樣，占掉大部分使用者使用面積。 因此，本論文選擇探討透過投影機代替傳統顯示器，希望透過這種方式，可以提供使用者一個更方便及有彈性的使用環境。

1.2 研究目標

透過本論文探討，整理出利用投影機代替傳統顯示器所帶來的優點，以及此種顯示方式所需要的技術與遭遇的困難。

1.3 論文架構

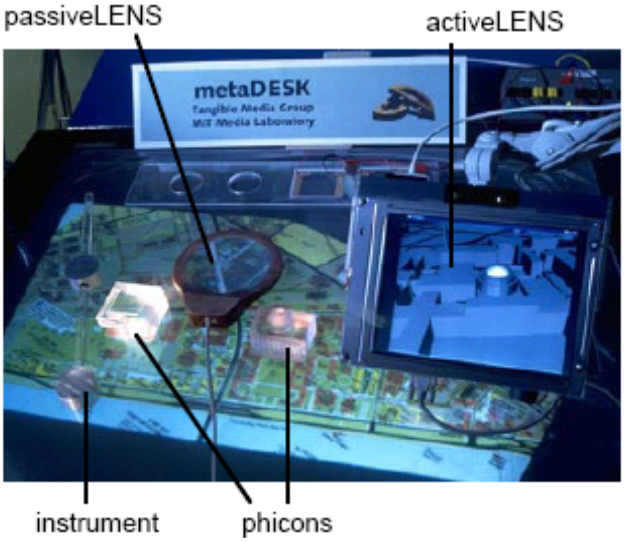
第一章為序論，內容主要為說明研究背景和文獻回顧和研究目標。 第二章為相關研究，內容包括有關論文相關主題的研究。 第三章系統架構與流程詳細介紹系統架構和實驗結果。 第四章問題探討則討論論文在實作中遇到的問題和解決的方法。 第五章結論為描述本篇論文的研究總結。

本論文實作系統為微軟(Microsoft)XP 系統，所以之後解說圖示都以微軟系統做說明。

第二章 相關研究

隨著科技越來越進步，電腦(Computer)已經成為生活不可或缺的一部份。 因此，如何可以與電腦更直覺、便利的做互動，也成為人機互動(Human Computer Interaction)一個很重要的研究方向了。

許多論文都已經提及這方面的構想與應用，例如早在 1997 年的 Tangible Bits: Towards Seamless interfaces between People，Bits and Atoms 這篇論文就已經提及到傳統人機互動方式缺點在於無法直覺的透過直接的動作，例如手勢或頭的擺動，與電腦做直接的互動。 因此，想創造一個新的互動環境的概念。 在此篇論文中透過 metaDESK 作為一個平台讓使用者可以透過設計好的標記(Tokens)讓多人同時做互動。 metaDESK 如圖 1 所示，事先設計一套環境，其中 instruments、phicons 都是設計好的標記，分別代表放大縮小的工具和某一個檔案或資料夾，透過 passiveLENS 可以讓使用者知道此標記和 passvieLENS 框選的部分在真實環境中所呈現的什麼？再透過設定好的標記讓使用者可以對這些檔案或資料夾做開啟或點選的動作，透過系統提供的環境，可以讓使用者體驗不一樣的互動方式。



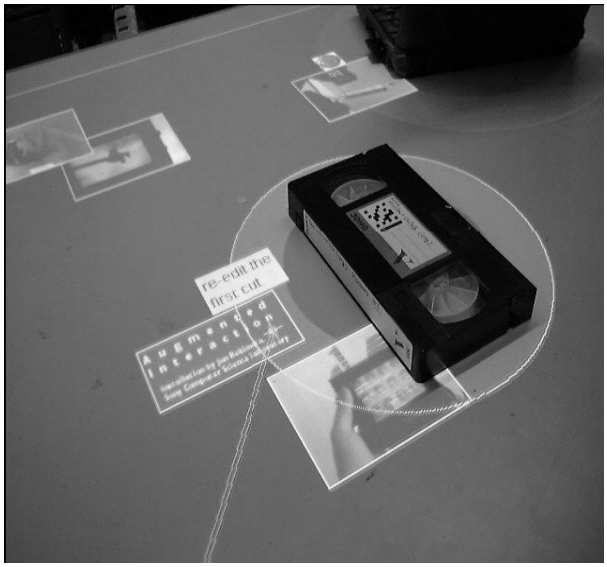
圖表 1 metaDesk 環境架構

到了 1999 年，Augmented Surface: A Spatially Continuous Work Space for Hybrid Computing Environments 這篇論文中，帶給使用者一個全新的互動方式。 這篇

論文希望帶來一種全新的多人會議(Multiple People Meeting)環境。 系統給予每一個物體(筆記型電腦或錄影帶)一個虛擬記號(Virtual Mark)，如圖 2 所示。

圖表 2 Virtual Mark

透過攝影機捕捉虛擬記號，可以知道每一個不同的物體的實際位置，除此之外，系統可以利用此虛擬記號對應到後端伺服器(Server) 上的虛擬物體，如圖 3 所示，攝影機辨識出錄影帶的虛擬記號後，透過後端伺服器將預先存起來有關此錄影帶的影像透過投影機投射在桌面上。



圖表 3 後端伺服器投影此錄影帶的相關影像

透過攝影機的捕捉，使用者可以透過拖曳(Drag)的動作將自己電腦中的資料拉到桌面上，利用投影機投影到桌面上，讓多個使用者可以同時瀏覽拖曳出來的影像，如圖 4 所示。

圖表 4 多人瀏覽拖曳出來的資料

此系統提供使用者一個完全直覺且方便的互動模式。 而到了2005年，Localisation and Interaction Augmented Maps 這篇論文中，同樣利用攝影機和投影機的組合，提供使用者一個瀏覽地圖的互動系統，如圖 5 所示。

圖表 5 Augmented Map 環境架構

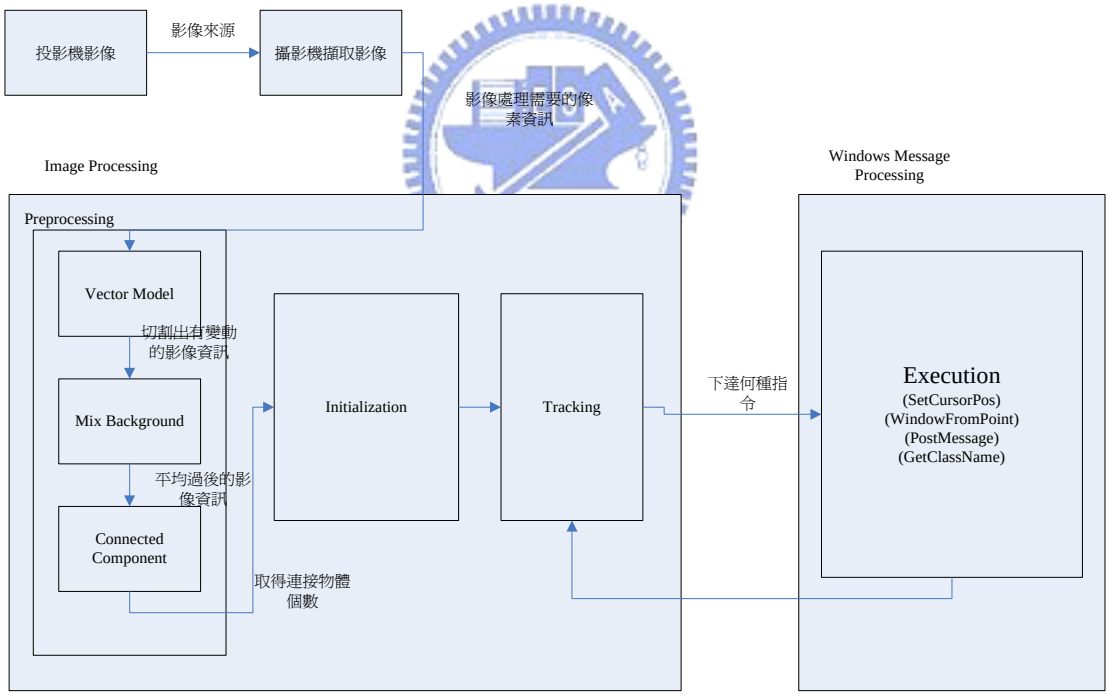
圖 5 所示為一紙製街道地圖，其中藍色框選區域為系統利用投影機所投射出的直升機，並由系統所控制。 圖中紅色框選區域為一顯示布幕，使用者可以透過該裝置上方突起處指向地圖上欲瀏覽的地方，並透過系統利用投影機投出欲瀏覽地區相關資訊於此裝置上。 同時，系統會自動將直升機慢慢移動到使用者所指之區域。 除了利用顯示布幕，使用者也可以透過 PDA 指向欲瀏覽的區域，達成上述顯示裝置的功能；若使用環境提供網路服務，亦可透過系統自動將該地點相關資訊傳送至 PDA 供使用者查詢。

第三章 系統架構與流程

本章節中，將詳細的介紹系統的架構與流程。 系統主要分成兩大部分 Image Processing 和 Windows Messages Processing。 Image Processing 又可以細分為 Preprocessing、Initialization 和 Tracking 三個部分。 在 3.1 節中，以系統架構圖來幫助瞭解系統主要的架構和流程。 在 3.2 節中針對 Image Processing 這部分做詳細的說明。 在 3.3 節中針對 Windows Messages Processing 這部分做詳細的說明。 在 3.4 節針對實驗結果做說明和呈現。

3.1 系統架構

系統流程圖如圖 6 所示，可以分成兩大部分:



圖表 6 系統架構流程圖

1. Image Processing - 處理、分析從攝影機擷取出來的像素(Pixel)資訊，提供 Windows Messages Processing 模組(model)所需要的資訊。
2. Windows Messages Processing - 從 Image Processing 模組接受下達的指令，透過 Windows Messages Processing 模組傳遞訊息給微軟系統，達成使用者所

下的指令。

本系統第一步透過 Image Processing 的步驟，處理攝影機擷取出來的影像，紀錄手和每一種標記的面積，並以面積的大小辨識不同標記。另外，設定軌跡球原點的位置，關於軌跡球會在第四章詳細說明。

透過攝影機的捕捉分析，追蹤右手在投影機投出影像中的位置，並將右手最高點當成參考點，系統判斷參考點相對於軌跡球原點的距離和方向，來設定系統游標移動的速度和方向。 當系統發現有兩隻手出現，即有兩個物體面積超過在 Initialization 所設定手的面積，以右手控制參考點，左手觸發滑鼠按鍵事件，並判斷使用者手持何種標記，對系統下達指令。 Windows Messages Processing 處理 Image Processing 所傳達的指令，透過微軟提供的指令與微軟系統溝通，達成使用者所下達的指令。

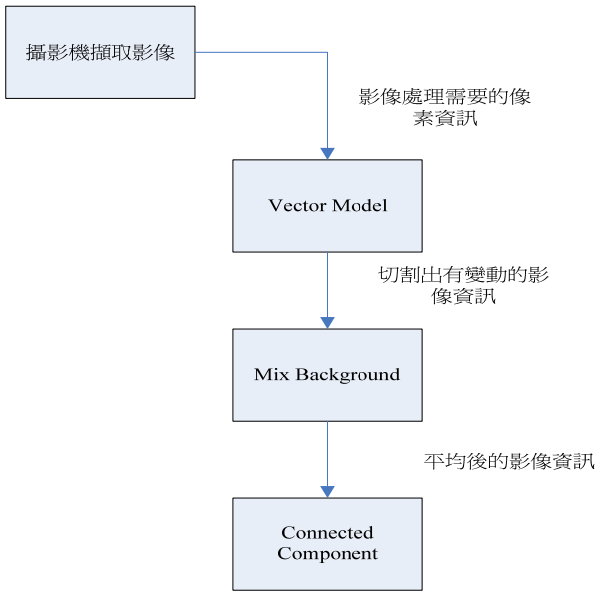
3.2 Image Processing

在此模組中，主要的內容有 Preprocessing、Initialization 和 Tracking 三個部分。將會在 3.2.1 詳細介紹 Preprocessing 的架構和流程。 在 3.2.2 詳細介紹 Initialization 的架構和流程。 3.2.3 詳細介紹 Tracking 的架構與流程。

3.2.1 Preprocessing

如流程圖 7 所示，此模組分成三個步驟:

1. Vector Model
2. Mix Background。
3. Connected Component

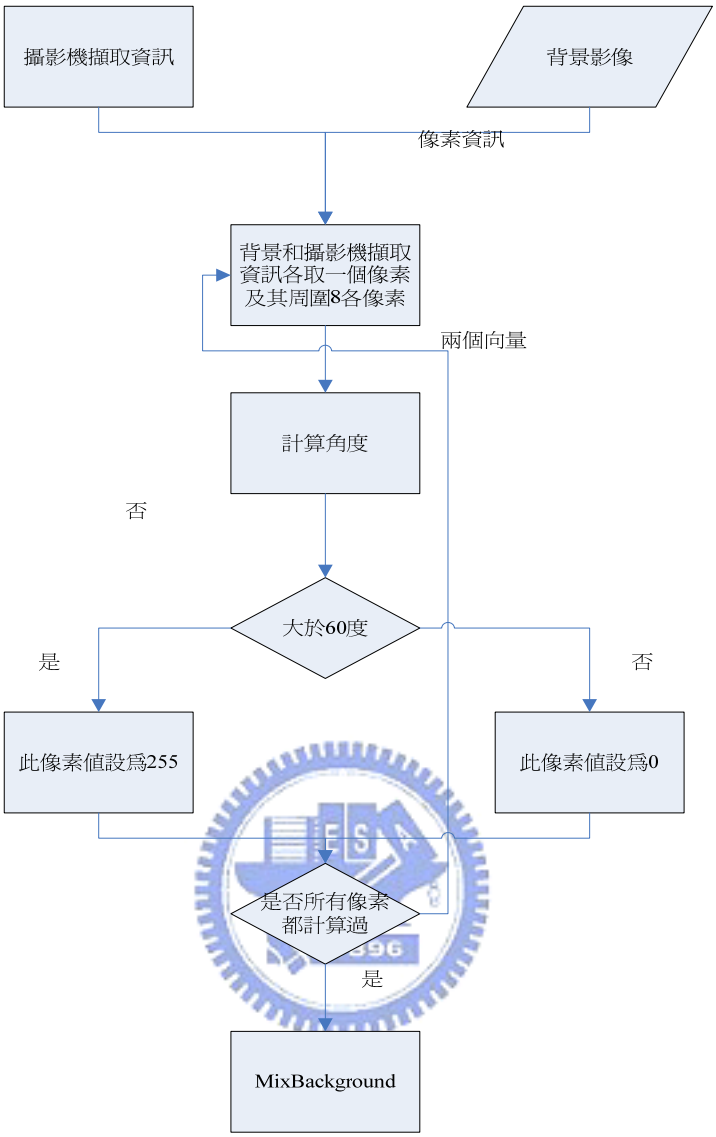


圖表 7　Preprocessing 流程圖

首先,將攝影機擷取出的影像透過 Vector Model 切割出移動的物體。 Vector model 主要概念是計算兩個向量的夾角，利用計算出來角度去判斷兩個向量的相似程度。 這個概念應用在許多方面，例如搜尋引擎就經常利用 Vector Model 將檢索詞彙與文件相關程度作為兩個向量，找出所夾擠的角度，若角度越大代表兩份文件的相似度越高。 將這觀念套在影像處理中， 主要是要處理兩張影像相似程度。

在影像處理領域中，經常會利用去背景(Background Subtraction)的技術去取出正在移動的物體(Motion Objects)。 簡單的說，去背景的方法就是先儲存一張影像作為背景，接下來每一張影像都會和此預先儲存好的背景作比較，看每一個像素是否有變化，來判斷是否有移動的物體。 如果我們只用單純的 pixel-by-pixel 的比較，可能會因為光線或攝影機輕微晃動等因素造成誤差。 因此，為了可以解決上述的問題，本系統採取了 Vector Model 的方法。

圖 8 所示，有別於去背景 pixel-by-pixel 的方法，系統對於每一個像素，會取其鄰近的 8 個點,即一個 3*3 的遮罩(Mask)。 利用這 9 個像素作為 Vector Model 的兩個向量(此次擷取到影像的 9 個像素和預先存取好背景影像中和此次擷取到的影像同樣位置的 9 個像素)，計算出這兩各向量所夾擠的角度。 若大於預先設定的門檻(系統預設是 60 度)(Threshold)，則表示此像素相對於背景是移動的。



圖表 8 Vector Model 流程圖

透過 Vector Model 的方法，因為參考的資訊比 pixel-by-pixel 的方法多了 8 倍，可以切割出更精確移動中的物體，以便於之後系統的處理。在 Mix Background 中，我們將把投影機產生的問題減少到最小，詳細部分會在第四章說明。 當系統在處理經由攝影機擷取出來的影像時，會因為投影機和攝影機的問題，無法直接對擷取出來的影像做分析。 因此必須先將擷取出來的影像做處理，本系統處理的方法為 Mix Background。

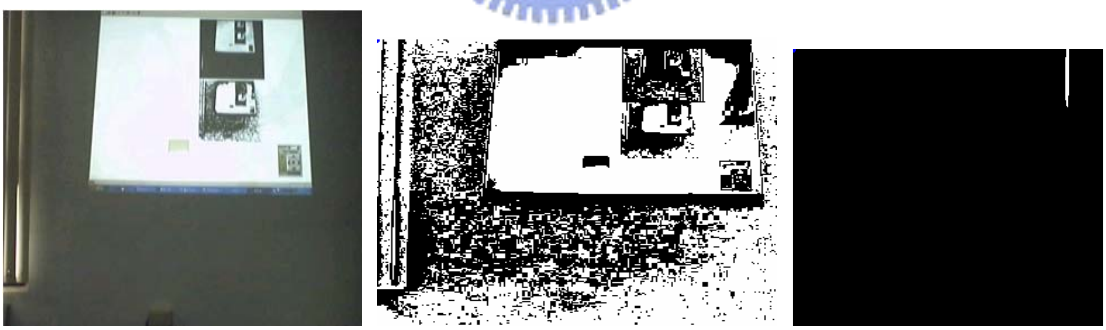
Mix Background，顧名思義，就是和背景作混合。 在介紹 Vector Model 中提到去背景的概念，此處所指的背景，為 Vector Model 步驟中，預先儲存的背景。

其處理的公式如下:

$$CurrentFrame = \frac{\sum GCI - \bar{I}_{current}}{\sigma_{current} / \sigma_{last}} + \bar{I}_{last}$$

CurrentFrame : each Frame captured by camera
GCI : the value of gray level for each pixel
$\bar{I}_{current}$: average value of all the pixels in each Frame
$\bar{I}_{last}$: average value of all the pixels of background
$\sigma_{current}$: the variance of GCI
σ_{last} : the variance of the frame of background

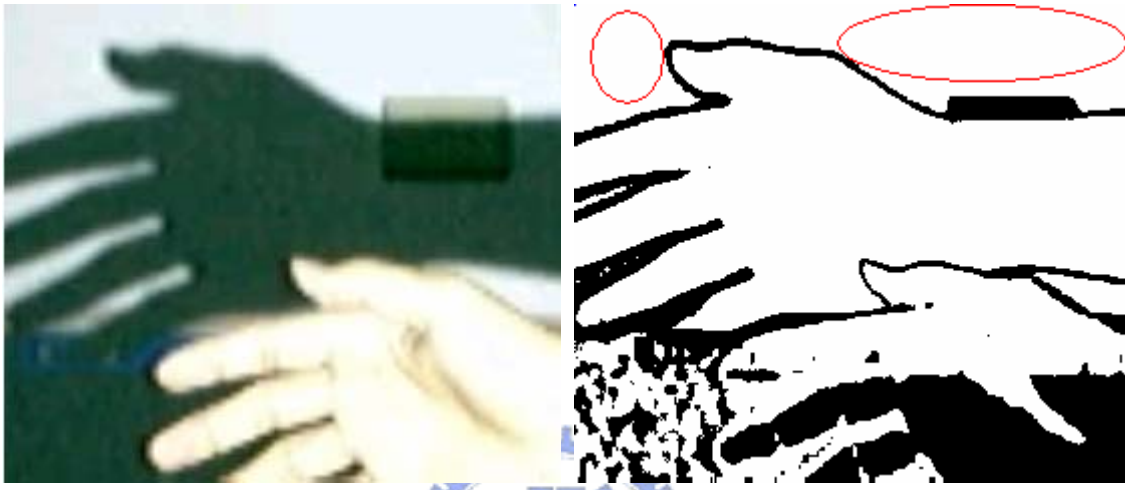
混合的意思是系統會對每一次攝影機擷取出來的影像和系統預先儲存好的背景做每一個像素平均(Average)的動作。 如果這一次擷取出來的影像因為干擾而和預先儲存好的背景差太多的話，透過公式，會將這一次擷取出來的影像的變化量和預先存取好的背景做平均，讓變化量降低。 如圖 9 中為攝影機擷取的影像，影像因為攝影機和投影機所造成的問題，造成大量的白色閃光。 圖 10 為未經過 Mix Background 處理的影像，圖 11 為經過 Mix Background 處理的影像，投影機所產生的問題已經大為改善。



左：圖表 9 攝影機擷取影像
中：圖表 10 未經 Mix Background 處理影像
右：圖表 11 經由 Mix Background 處理影像

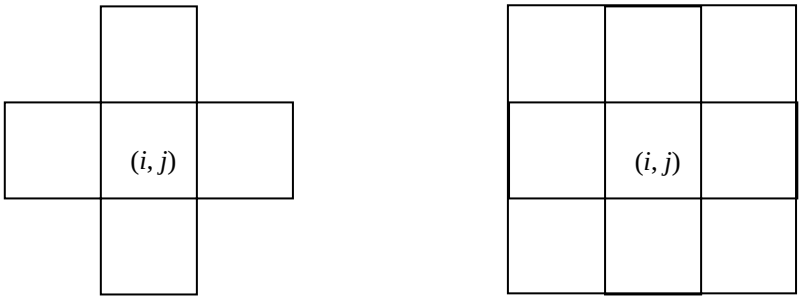
此公式的缺點在於只適用於某些特殊情形。 因為大量不規則的閃動會和顏色較淡的背景做平均，讓擷取出來的影像不會因為不規則的閃動而影響影像的處理。 因此，若在本系統中出現一個暗的物體，例如：手或肢體的影子，因為影子的顏色會比背景的暗，因此在經過平均的動作後，反而會讓手或肢體的影子變

成閃動的影像，影響影像的處理，如圖 12 為攝影機擷取的影像，圖 13 為經過 Mix Background 處理的畫面。 從圖 13 中可觀察到原本應該為黑色的部分(紅色框選部分)，因為出現一個比背景暗的物體(此處是手的影子)，經過平均過後，使得黑色部分變成不均勻的黑白分佈，造成處理上的困難。 因此，這是在利用這公式上面的一個缺點。



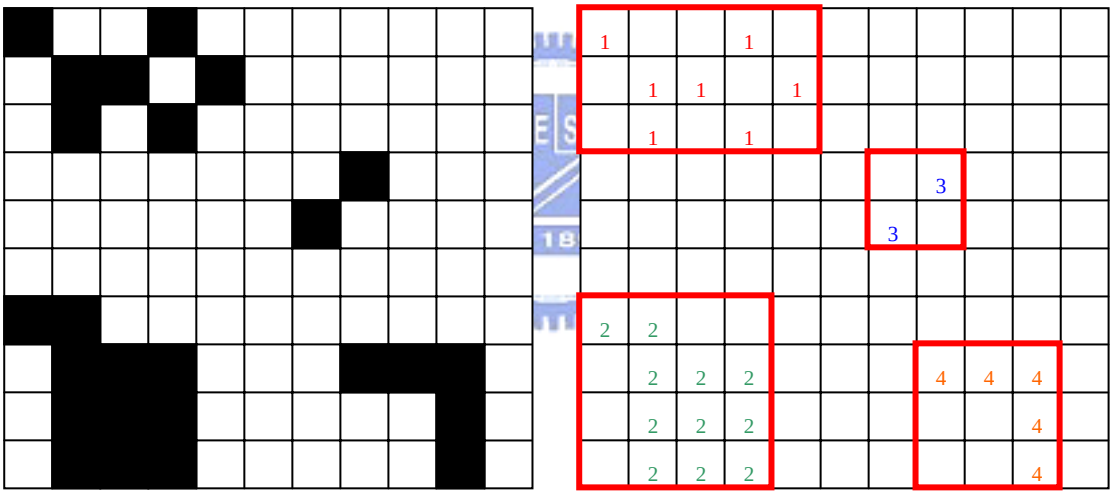
左：圖表 12 攝影機擷取影像
右：圖表 13 因陰影造成問題

經由 Vector Model、Mix Background 步驟之後，我們可以得到所有可能變化的像素，這些像素混雜著雜訊、運動的背景以及任何可能的候選者。 為了擷取出有意義的區域，我們必須將個別連接的物體一一標記出來。 為了定義連接物體，我們假設個別的連接物體在二質化(Binary，即非黑及白的像素)的影像上是連接的像素全體，而兩個像素連接的定義為此兩像素為 4-Neighbors(假設兩點其中一點座標為(i，j)，則另一點必在(i+1,j)、(i-1,j)、(i,j+1)、(i,j-1)四個座標之中)或 8-Neighbors(假設兩點其中一點的座標為(i，j)，則另一點必在(i+1,j+1)、(i+1,j)、(i+1,j-1)、(i,j+1)、(i,j-1)、(i-1,j+1)、(i-1,j)及(i,j-1)八個座標之中)，如圖 14 所示。



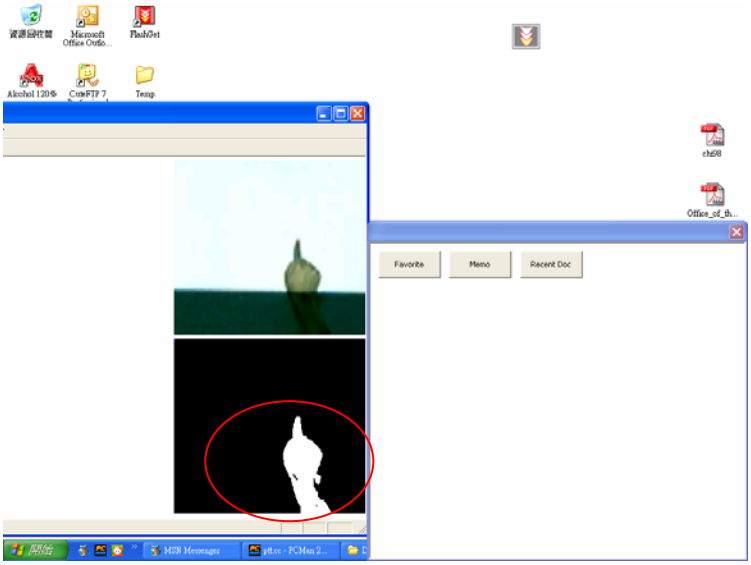
圖表 14　4-Neighbors 與 8-Neighbors 示意圖

我們從圖上任何一點開始標記連接物體，對所有互相成 8-Neighbors 連接的像素以一個獨特的數字標記，代表其互相連接，每當發現一個未標記的像素時，便以一新數字做標記，依此反覆做到整張二質化影像的所有點都被標記為止，如圖 15 所示。



圖表 15　連接物體標示，

在發現所有連接物體資訊之後，透過系統事先預設的門檻值，就可以找出系統想要找出的物體，進而做接下來的步驟。 圖 16 紅色框選部分所示，為經過 Connected Component 從處理完 Vector Model 和 Mix Background 找出的連接物體。



圖表 16　Connected Component 找出的連接物體

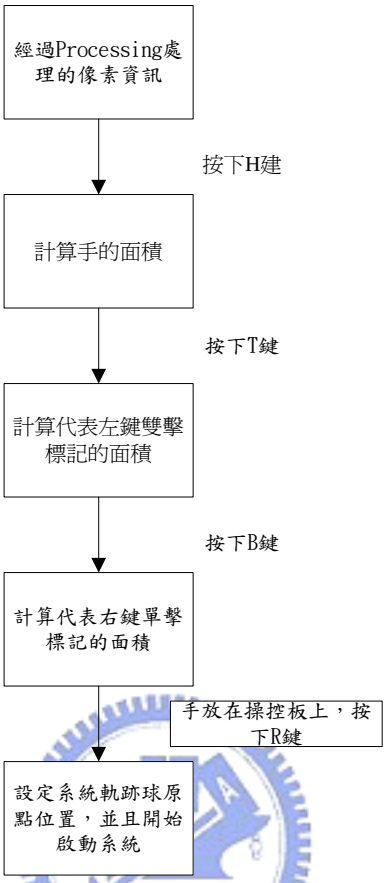
3.2.2 Initialization

在處理完 Preprocessing 的部分之後，系統會設定手和標記的面積，以便之後系統要做 Tracking 和 Execution。本系統中，設計了兩種標記，圖 17 代表的滑鼠左鍵的雙擊(Double Click)，而圖 18 代表滑鼠的右鍵的單擊。

左：圖表 17　代表滑鼠左鍵雙擊標記

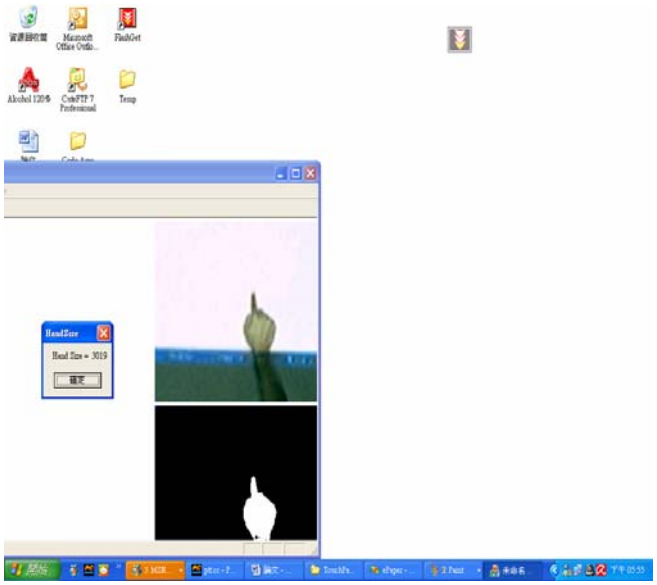
右：圖表 18　代表滑鼠右鍵單擊標記

Initialization 的流程圖如圖 19 所示。



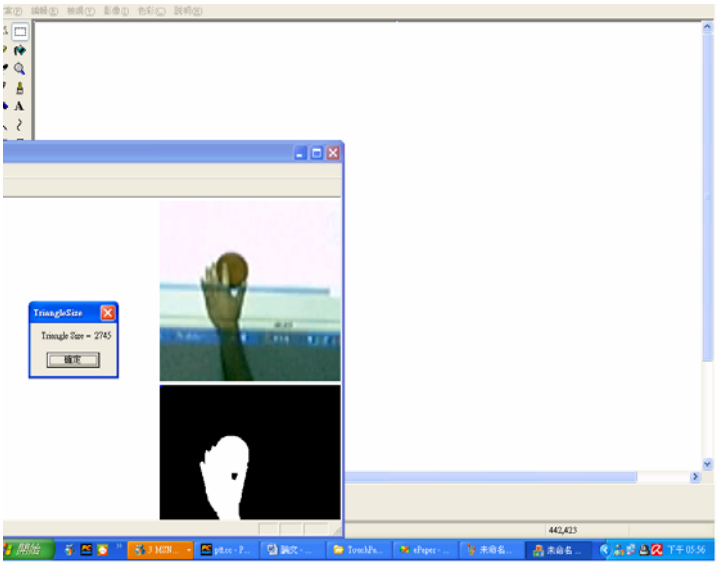
圖表 19 Initialization 流程圖

首先，系統會先偵測手的面積，當手確定放在攝影機可以擷取到的位置後，如圖 20 所示，按下鍵盤 H 鍵，系統則會得知手的面積。



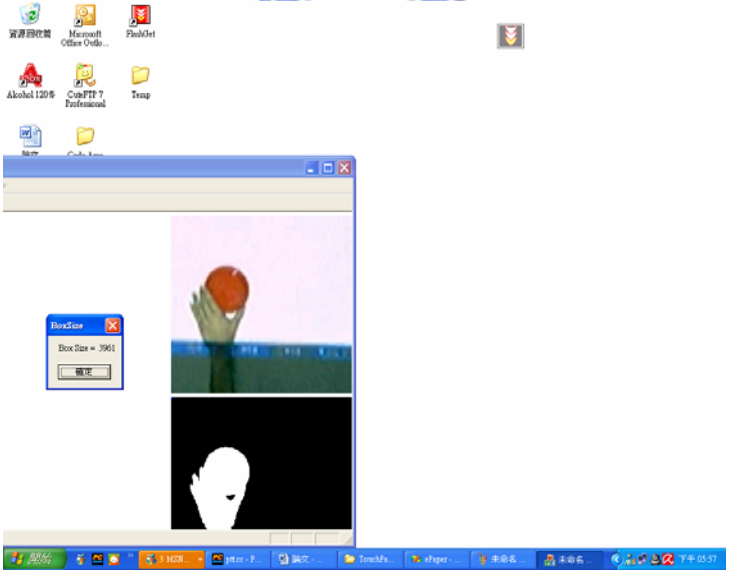
圖表 20 偵測手部面積

接下來手拿著代表滑鼠左鍵雙擊的標記，放在攝影機可以擷取到的位置後，如圖 21 所示。 按下鍵盤 T 鍵，系統則會得知代表滑鼠左鍵雙擊標記的面積(按下 T 鍵時所得到的面積 － 之前所得到手的面積)。



圖表 21 偵測代表滑鼠左鍵雙擊標記的面積

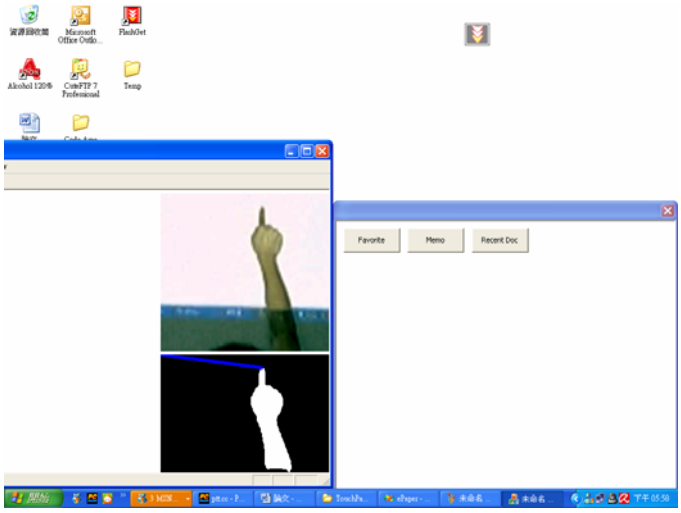
最後手拿著代表滑鼠右鍵單擊的標記，放在攝影機可以擷取到的位置後，如圖 22 所示，按下鍵盤 B 鍵，系統則會得知代表滑鼠右鍵單擊標記的面積。



圖表 22 偵測代表滑鼠右鍵單擊標記的面積

在完成手和標記的設定之後，將右手放在經由投影畫面中的操控區域(將在第四章詳細說明)上，如圖 23 所示，按下 R 鍵，則此時右手的最高點，設定成軌跡球

的原點。 當 Initialization 所有步驟完成時，系統會設定一條藍線連接著手的最高點，此點代表著系統參考使用者如何移動滑鼠的參考點。



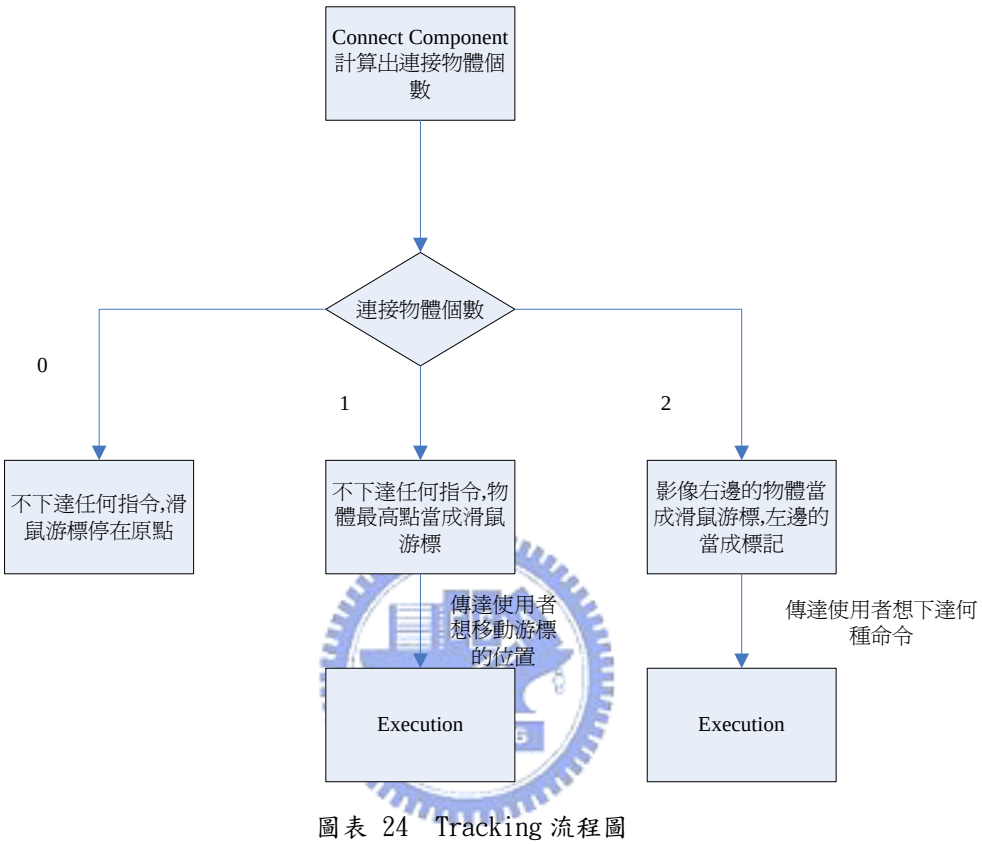
圖表 23 設定軌跡球原點位置

本系統面積的算法如下

1. 求出手的面積
2. 代表滑鼠左鍵雙擊的標記面積: 手拿著代表滑鼠左鍵雙擊標記整個物體面積 - 手的面積。
3. 代表滑鼠右鍵單擊的標記面積: 手拿著代表滑鼠右鍵單擊標記整個物體面積 - 手的面積。
4. 算出代表滑鼠左鍵雙擊標記面積的上限及下限: (2 算出來的面積 - 手的面積) / 3(記作 Temp)，則代表滑鼠雙擊標記面積的下限為 2 求出的面積 - Temp。 利用(3 算出來的面積 - 2 算出來的面積) / 3(記作 Temp1)， 則代表滑鼠雙擊標記面積的上限為 2 求出的面積 + Temp1。
5. 算出代表滑鼠右鍵單擊標記面積的上限及下限: 代表滑鼠右鍵單擊標記面積的下限為 3 - Temp1。 代表滑鼠右鍵單擊標記的上限為 3 + Temp1。
6. 系統則會把介於滑鼠左鍵雙擊標記面積範圍下限和上限之間面積的值，都代表滑鼠左鍵雙擊標記。 介於代表滑鼠右鍵單擊標記面積範圍下限和上限之間的值，代表滑鼠右鍵單擊。

3.2.3 Tracking

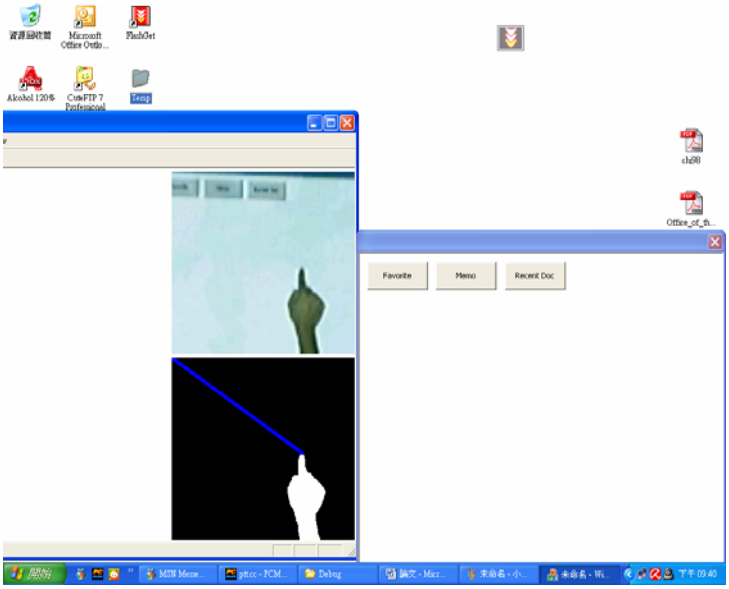
在完成了 Processing 有關影像和系統的一些初始化後，系統將開始模擬使用者利用右手操控系統游標。 Tracking 模組的流程圖如圖 24 所示：



圖表 24 Tracking 流程圖

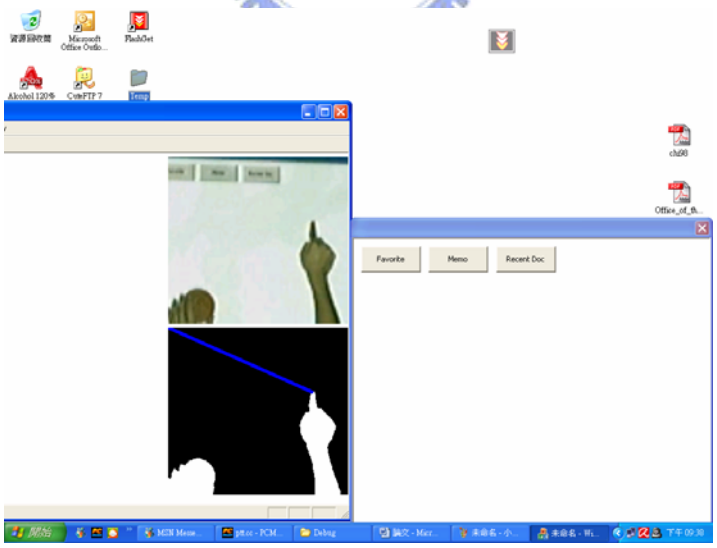
首先，系統利用 Connected Component 的方法計算出目前有幾個物體大於在 Initialization 所設定手的面積，會有下面三種情形:

1. 零個: 本系統判斷攝影機擷取到的影像中，若沒有半個物體面積超過手的面積，則畫面上不會有任何影像更動。 在此情況下，本系統會設定系統游標停留在軌跡球原點的位置。
2. 一個: 本系統判斷攝影機擷取到的影像中，若有一個物體面積超過手的面積，表示本系統開始驅動系統事件。 在此情況下，系統會將此物體當成使用者的右手，視其最高點為參考點。 本系統會偵測參考點相對於軌跡球原點的距離和方向，設定系統游標移動的速度和方向，然後傳達使用者想移動系統游標的位置給 Execution 與微軟系統作溝通，如圖 25 所示。



圖表 25　Tracking 偵測到一個物體

3. 兩個：本系統判斷攝影機擷取到的影像中， 若偵測到兩個物體超過手的面積，在畫面右邊的物體代表參考點，左邊的代表使用者欲下達的指令。 然後傳達使用者想移動系統游標的位置和使用者欲下達的指令給 Execution 與微軟系統作溝通，如圖 26 所示。



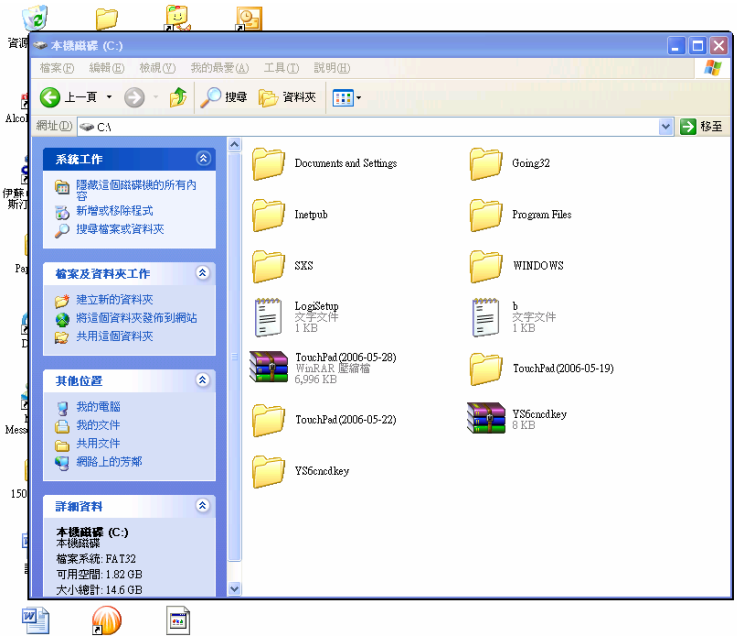
圖表 26　Tracking 偵測到兩個物體

本系統假設最多一次只會兩個物體大於手的面積，因此，在 Tracking 模組中並不會有除了上述三個狀況以外的情形發生。

3.3 Windows Messages Processing

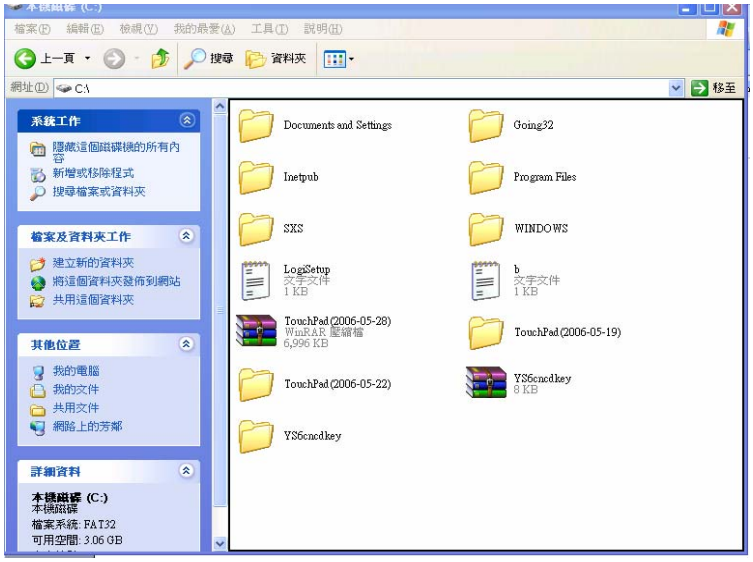
此模組主要處理有關微軟溝通的部分。本系統透過使用者的右手控制系統游標，左手觸發按鈕滑鼠事件，例如:按下滑鼠左鍵、右鍵。 因此，如何透過傳遞訊息的方式與微軟系統作溝通來完成這樣的操控行為，成為論文中另外一個很重要的部分。

因此，我們必須瞭解微軟(Microsoft)系統是如何讓滑鼠訊息去驅動視窗系統事件，例如:左鍵雙擊(Double Click) 則會開啟檔案。 首先，先介紹視窗架構。微軟是採用容器(Container)的方式建構視窗架構，接下來會以一幅畫的例子配合實際微軟視窗架構的圖(本例利用 C:\這個視窗)搭配說明。首先，我們可以看到如圖 27 所示，



圖表 27 被黑框框住的 C:\槽視窗

C:\被一個黑色的框框住，可以把黑框想像成畫布，有了畫布我們才可以利用畫筆在畫布上繪圖。 同樣的，微軟視窗架構同樣需要底層的容器，之後才可以附加新的物件在容器上。 如圖 28 所示



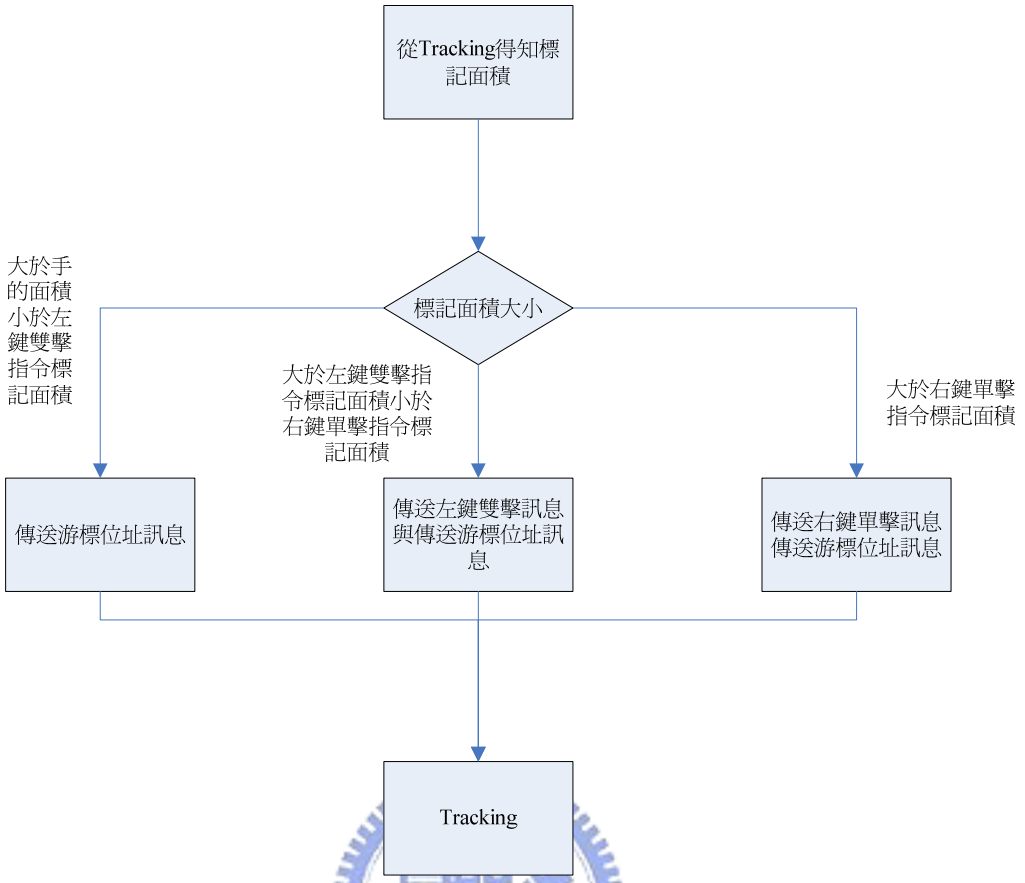
圖表 28　被黑框框住資料夾物件

此時被黑框框住的視窗如同在畫布上畫一座山。　在微軟視窗架構的觀念中，就是在底層的容器中加一個新的物件，利用一層疊一層的原理，架構出視窗。 對於每一個放在容器中的物件，微軟系統都會提供一個事件處理者(Handler)，事件處理者會紀錄這個物件的辨識碼(ID)、物件類別(Class Name)和在物件中包含哪些資料。 因此，傳達滑鼠事件給微軟系統，步驟如下:

1. 知道滑鼠指定位置所屬的物件(放於容器中的某物件)。
2. 找出處理這個物件的事件處理者。
3. 透過事件處理者，傳遞訊息告訴事件處理者想要對滑鼠指定位置下達的指令。

利用這三個步驟，就可以透過傳遞訊息的方法，與任意視窗中的任意物件進行溝通。 至於如何透過傳遞訊息的方式，會在第四章詳細介紹。

在瞭解微軟系統如何讓滑鼠訊息去驅動視窗系統事件的流程後，本系統中的 Windows Messages Processing 中主要包括 Execution 模組。 Execution 模組主要處理經由 Tracking 知道使用者所要下達的命令，透過 Execution 模組與微軟系統溝通，其流程圖如圖 29 所示



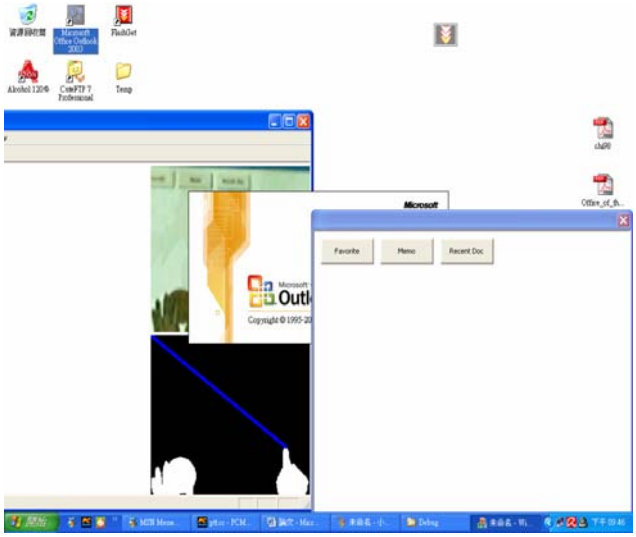
圖表 29 Execution 流程圖

若使用者拿的是代表滑鼠左鍵雙擊的標記，系統則會傳送 Double Click 的訊息。若使用者拿的是代表滑鼠右鍵單擊的標記，系統則會傳送跳出系統選單的訊息。當 Execution 傳達訊息給微軟系統後，本系統繼續監聽 Tracking 模組是否傳送新的訊息，讓使用者移動到下一個想要下達指令的檔案或資料夾。

3.4 實驗結果

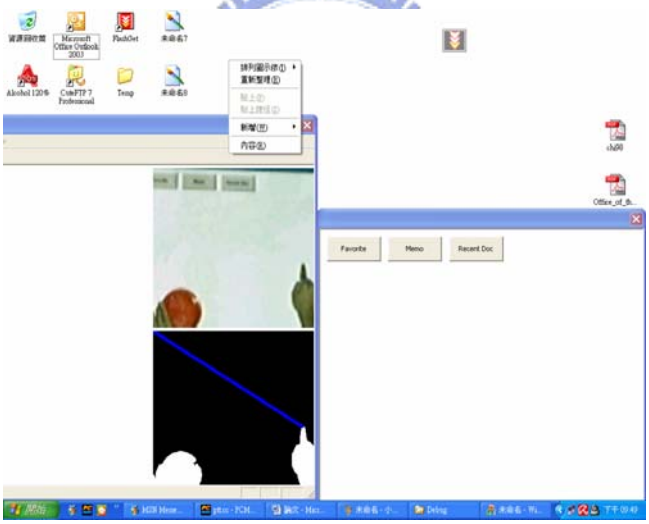
接下來介紹本系統環境架設和實驗結果。 本系統利用投影機投出由電腦螢幕的畫面，然後透過攝影機去擷取影像，透過影像處理分析再做後端的處理。 為了系統架設方便，我們將投影機直接放在桌上，攝影機放在投影機上面，將影像投在牆壁上，使用者手貼著牆壁做系統的操控。 而系統實驗結果是透過特定的標記可以讓使用者對指定的檔案或資料夾下達左鍵雙擊或右鍵單擊的指令。如圖

30 所示，透過左鍵雙擊指令，系統可以開啟 OutLook 應用程式 。



圖表 30 開啟 OutLook 應用程式

如圖 31 所示，透過右鍵單擊指令，系統可以開啟系統選單。



圖表 31 開啟系統選單

第四章 問題探討

本章節將探討系統在實作中所遭遇的問題，及其解決的方法，並將問題分成硬體問題和軟體問題兩個部分討論。 其中硬體問題包括畫面閃爍、細部特徵難擷取和可見光干擾效應。 軟體問題包括了訊息問題、系統延遲問題、場景變動問題和討論操控方式。 本章節會在 4.1 介紹硬體問題，在 4.1.1 詳細介紹畫面閃爍帶來的問題，在 4.1.2 詳細介紹細部難擷取帶來的問題，在 4.1.3 詳細介紹可見光干擾效應帶來的問題。 在 4.2 介紹軟體問題，在 4.2.1 詳細介紹訊息問題，在 4.2.2 詳細介紹系統延遲訊息，在 4.2.3 詳細介紹場景變動問題，最後在 4.2.4 討論有關操控方式所帶的問題。

4.1 硬體問題

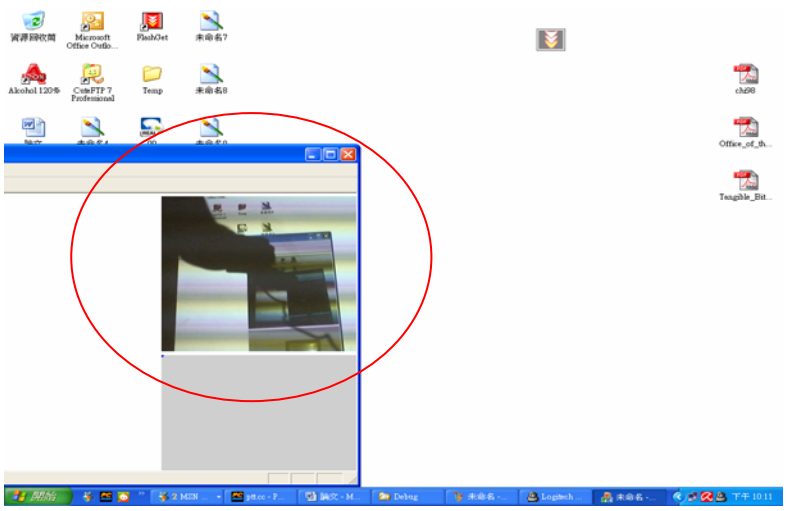
硬體問題主要是因為攝影機、投影機等周邊設備所產生的問題，我們列出了三個主要的問題:

1. 畫面閃爍:因為投影機投影模式問題，造成攝影機擷取到閃爍的畫面。。
2. 細部特徵難擷取:增加投影機與攝影機的距離帶來的問題。
3. 可見光干擾效應:投影機光線對使用者和系統處理上帶來的問題。

4.1.1 畫面閃爍

本系統利用攝影機將影像擷取出來，發現一個嚴重的問題。 投影機是利用固定頻率掃描的方式投出影像，如同 CRT 利用掃描線的方式呈現影像；但是因為視覺暫留的原理，人不會覺得投影機投出的畫面是不連續的。

攝影機擷取影像時，也是利用一秒擷取固定數量的影像數，利用視覺暫留的原理，讓使用者不覺得影像不連貫。 然而利用攝影機擷取投影機投出影像的時候，攝影機擷取到的會是不連貫的影像，造成攝影機擷取到的畫面有嚴重閃爍的問題，如圖 32 中紅色框選部分。



圖表 32 畫面閃爍問題

問題產生的原因主要是受限於硬體本身的技術而並非程式或系統所能解決。 而且閃爍的方式是以不規則的頻率發生，因此也無法透過演算法達成有效的處理。因此，在閃爍的影像中，要利用影像處理的方法抓出特徵值，進而做相關處理，會變成非常的困難。

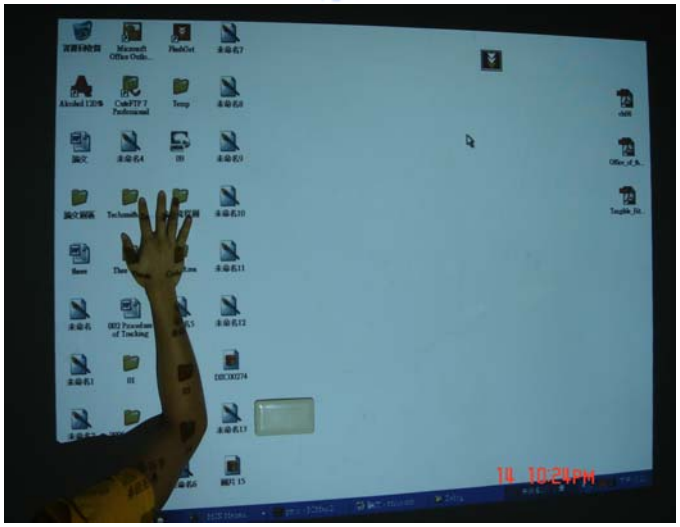
4.1.2 細部特徵難擷取

經過實驗證明，若增加投影機與攝影機之間的距離，可以改善畫面閃爍的問題。 但因為距離增加的關係，攝影機擷取物體影像的大小也會相對地因為距離的增加而減小。 而物體影像減小所帶來的問題為物體原有的特徵將會難以判斷或是消失。 讓系統更難透過影像處理的方法分析出所要得到的資訊。 如圖 33 所示，紅色框選的部分為攝影機擷取投影機投出的畫面，可以看出在上一節所提出的閃爍幾乎已經解決，但是因為距離拉遠的關係，攝影機擷取的畫面中我們幾乎沒法對物體做細部特徵上的分析，例如:每隻手指的位置即所指的方向。

圖表 33 擷取影像物體太小難以辨識

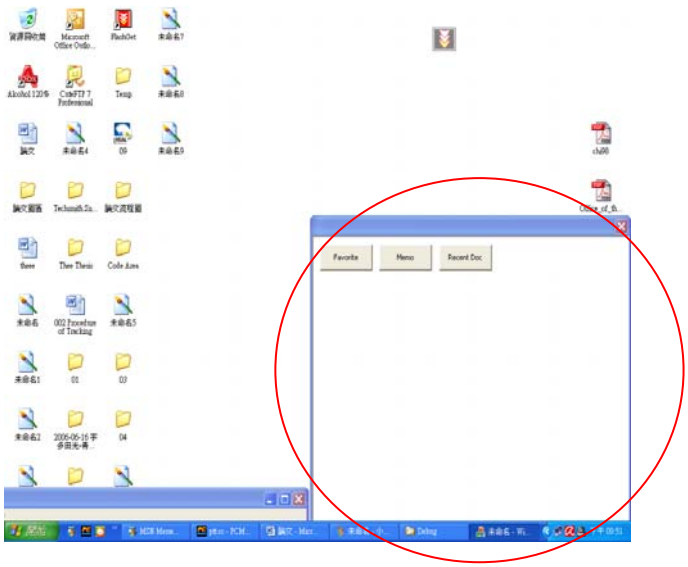
4.1.3 可見光干擾效應

投影機投影的原理，是透過 R、G、B 三種不同顏色光線的組合，產生物體的影像。 因此，當使用者將手或標記放在投影機所投出的影像畫面上，必會受到投影出來的光線所干擾，除此影響之外，亦改變了手原來的顏色。 因此，欲針對物體的顏色和形狀特徵做判斷，都變的很困難，也加深了利用投影機代替傳統顯示器構想的困難度，如圖 34 所示，畫面中的手經過投影機投出來光線的影響，不僅失去原本手的顏色，手的形狀也會因為改變。



圖表 34 投影光線改變了手原本的顏色和形狀

為了解決因為硬體所產生的問題，本系統提出了操控區域的概念。 操控區域是一塊白色的區域，如圖 35 紅色框選部分。



圖表 35　操控區域

在本系統的擺設中，攝影機只擷取操控區域的影像。雖然此時仍會有畫面閃爍和可見光干擾效應，但操控區域為單純的白色影像，因此投出來的畫面只剩閃爍的影像，再透過 Mix Background 的方法，可以將閃爍問題降到最低。

4.2 軟體問題

軟體問題探討本系統遇到所有非硬體所產生的問題。 我們列出了四個主要的問題:

1. 訊息問題: 利用傳送訊息方式與微軟系統溝通發生的問題。
2. 系統流暢度: 系統執行的速度問題。
3. 場景變動: 攝影機擷取畫面變化太複雜所產生的問題。
4. 討論操控方式: 探討使用者操控本系統的方式。

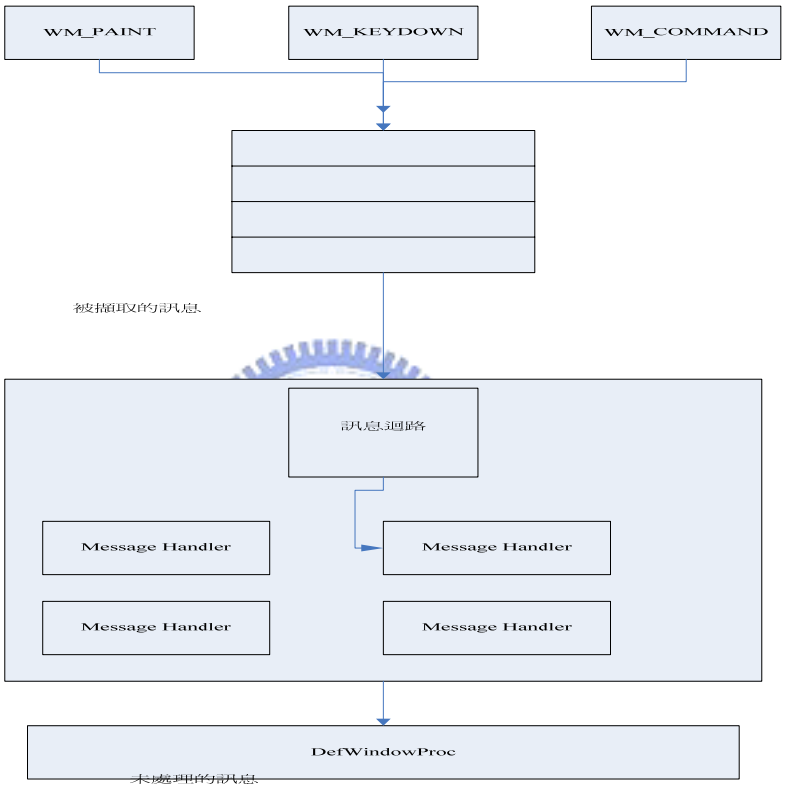
4.2.1 訊息問題

訊息問題只要是要探討當透過傳遞訊息與微軟系統溝通的時候所發生的問題，我們列出兩個主要的問題:

1. 訊息延遲(Delay)

2. 無法處理的訊息

探討這兩個問題之前，我們先瞭解微軟系統是如何處理接收到的訊息。如流程圖 36 所示，當使用者送出訊息的時候，如 WM_PAINT。 微軟系統會先將所有訊息先放到一個訊息佇列(Message Queue)中，再將訊息透過訊息迴路將存在訊息佇列中的訊息分派給指定的訊息處理(Message Handler)，最後，訊息處理會將訊息送到正確的函式(function)處理訊息。



圖表 36　微軟訊息傳遞流程

在瞭解微軟系統如何處理訊息之後，首先，探討訊息延遲的原因。

本系統利用 PostMessage 的方式傳送訊息到訊息佇列中而告知此時我們希望透過標記下達的指令為何。

- 右鍵單擊:

```
::PostMessage(g_hwndFoundWindow, WM_RBUTTONDOWN, MK_LBUTTON, lParam
::PostMessage(g_hwndFoundWindow, WM_RBUTTONUP, MK_LBUTTON, lParam);
```

- 左鍵雙擊:

```
::PostMessage(g_hwndFoundWindow, WM_LBUTTONDOWN, MK_LBUTTON, lParam);
::PostMessage(g_hwndFoundWindow, WM_LBUTTONUP, MK_LBUTTON, lParam);
::PostMessage(g_hwndFoundWindow,WM_LBUTTONDBLCLK,MK_LBUTTON,lParam);
```

PostMessage 處理傳遞訊息的方式類似網路的 UDP(Universal Datagram Protocol)模式，即訊息傳送出去以後不管訊息有沒有被處理都繼續執行接下來的流程。 因此，本系統偶而會發生下達指令以後，訊息可能因為優先權(Priority)不夠高，一直被擺在訊息佇列中，等到所有優先權高的訊息處理完才處理使用者下達的指令，或者因為 PostMessage 沒有檢查訊息是否真的有被執行，導致訊息根本沒有被放到訊息佇列中，讓使用者下達的命令根本不會觸發。

雖然有另外一種 SendMessage 的方式傳遞訊息。 SendMessage 的處理方式類似網路的 TCP(Transmission Control Protocol)模式，即訊息送出去以後會等訊息確定被處理以後才繼續接下來的流程。 但是，如果系統利用此方法傳遞訊息的話，將會造成系統為了要等待每一個訊息的處理，而失去了系統即時性的優點。所以，本系統最終還是採用 PostMessage 的方法來處理訊息傳遞的工作。

PostMessage 傳遞訊息的方式與微軟系統溝通，透過這種方式，要處理滑鼠拖曳(Drag)指令會變的很困難。 一般拖曳的動作可以分成下面幾步:

1. 在想要被拖曳的檔案或資料夾上面按單擊滑鼠左鍵。
2. 按著滑鼠左鍵不放，將檔案或資料夾拖曳到想要的地點。
3. 放開滑鼠左鍵。

照著這樣的流程利用 PostMessage 的方法應該如下:

1. ::PostMessage(g_hwndFoundWindow, WM_LBUTTONDOWN, MK_LBUTTON, lParam);
2. 移動滑鼠到想要的地點。
3. ::PostMessage(g_hwndFoundWindow, WM_LBUTTONUP, MK_LBUTTON, lParam);

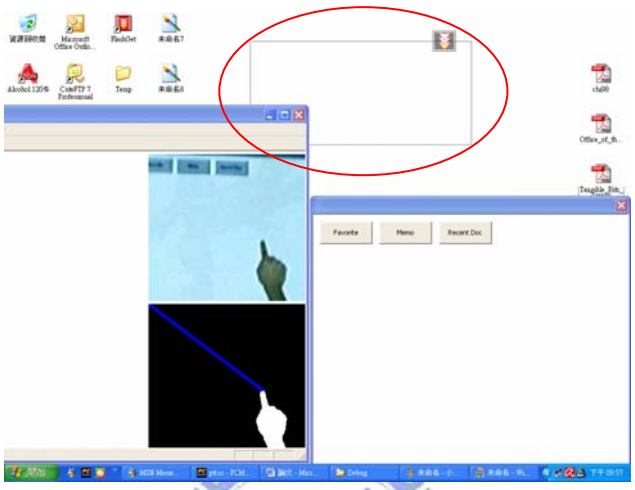
照這樣的流程發現檔案只能稍微的抖動一下，沒辦法順利的拖曳。 因此，我們為了避免流程錯誤，我們測試框選數個檔案和資料夾，而並非只針對單一檔案或資料夾做拖曳動作，流程如下:

1. ::PostMessage(g_hwndFoundWindow, WM_LBUTTONDOWN, MK_LBUTTON, lParam);

2. 移動滑鼠框選欲拖曳的檔案和資料夾到指定位置

3. ::PostMessage(g_hwndFoundWindow, WM_LBUTTONUP, MK_LBUTTON, lParam);

照這樣的流程發現我們可以利用 PostMessage 框選出欲拖曳的檔案和資料夾，如圖 37 紅色框選部分



圖表 37 利用傳遞訊息做框選的動作

但同樣的，我們只能稍微的讓檔案們抖動，並沒有辦法移動他們。 因此，如果要利用 PostMessage 的方法模擬滑鼠的拖曳動作，必須瞭解整個拖曳動作的流程，其中包括在微軟系統是如何處理拖曳這個動作?回傳的是什麼參數？我們該怎麼去處理這一串的流程?而不再只是單純的利用 PostMessage 傳遞訊息給訊息佇列，整個過程冗長且繁瑣，因此，在本系統中，並沒有處理拖曳這個動作。

4.2.2 系統流暢度

系統的流暢度，是人機互動領域很在意的事情，因為這即意味著一個系統對使用者來說”是不是好用”一個重要的指標。

本系統透過傳遞訊息的方式去驅動視窗事件，達成使用者下達的指令。 因此，本系統不斷處理龐大的影像處理運算和訊息傳遞事件，造成本系統會佔據系統大部分的資源，例如：中央處理器(CPU)和快取記憶體(Random Access

Memory)。因此，造成本系統，開啟檔案所需要的時間較透過滑鼠直接點選所需要的時間長。

4.2.3 場景變動

為了偵測前景(Foreground)物體，系統會利用去背景等類似方法處理。 但在利用投影機代替傳統顯示方式的環境中，透過此方法會因場景的變動造成環境過於複雜。 在此環境中，除了移動的雙手外，移動、開啟或關閉的檔案亦會造成背景的變動，透過影像處理的分析，可能會因為這些變動，造成分析錯誤。

為了解決這問題，本系統讓操控區域永遠呈現在最上面(TOPMOST)，即不管是否開啟或關閉檔案或應用程式，操控區域不會因此被遮蔽住。 透過這種方式，本系統的前景永遠只會有移動的雙手，解決了因為場景變動所造成的問題。

4.2.4 討論操控方式

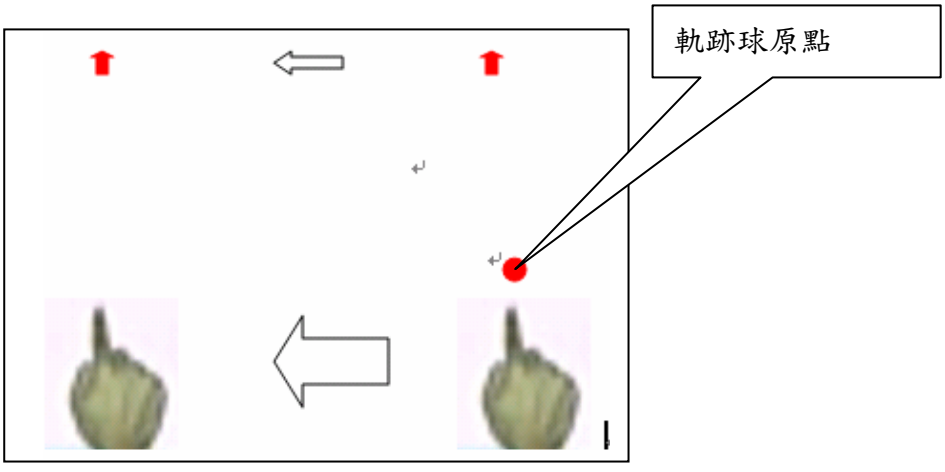
在這部分主要探討兩個問題:

1. 直覺操控方式所帶來的問題
2. 操用操控版的優缺點

首先探討第一個問題 ，當經由投影機投出來在桌面或者牆壁的影像點選或開啟指定地方的資料夾或檔案時，直覺的方法是利用右手點選此特地地方的資料夾或檔案，再利用標記對系統下達點選或開啟等指令。 但此種方法有一個缺點，當點選某特定地方的時候，使用者的肢體，如 :手或軀幹，會擋住投影機投出來的影像，造成使用者使用上的不方便，如有些檔案或資料夾會因為被擋住而看不到或投影出來的強光造成使用者不舒服，如圖 38 所示。

圖表 38　使用者遮蔽了投影畫面

考量到這個因素，我們採用軌跡球的滑鼠操控模式。 軌跡球的操控模式有別一般滑鼠的方式在於一般滑鼠的移動方式是採取絕對距離的方式，即目標離系統游標越遠，我們必須移動較大的位移量讓滑鼠游標可以移動到指定的目標位置，因此會產生因肢體擋住投影畫面的問題。 而軌跡球的操控方式是採取相對距離的方式，即會有一個原點的概念，所有的位移都是參考此原點，如圖 39 中，紅色圓圈當成軌跡球原點，紅色箭頭當成滑鼠游標，當使用者往原點左邊移動時，滑鼠游標也往左邊移動。

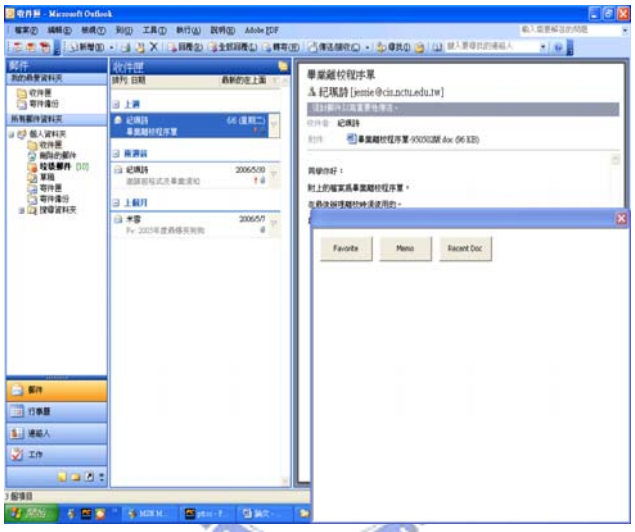


圖表 39　軌跡球操控方式

系統還加上速度的概念，即當離原點的擺動距離越大的時候，滑鼠游標會產生較快的移動速度，減少使用者使用上的時間。 再加上之前所提出操控區域的概念，

利用這樣的操控方式，不僅解決了肢體擋住投影畫面的問題，也減少使用者為了點選或開啟到遙遠目標位置不必要的走動或是力氣。

系統為了要解決投影機產生的問題，提出了操控版的概念。 而這樣的概念所產生的缺點就是會影響使用者檢視檔案的完整度和方便性。 如圖 40 所示，當使用者開了一個 OutLook(微軟作業系統提供的應用程式)時，右下角有一部份被操控版遮蔽，造成使用者無法瀏覽到被遮蔽的部分，只有利用縮小視窗或者移動視窗的方法才能解決。



圖表 40　操控區域遮蔽了應用程式

第五章 結論

本論文中，提出以投影機代替傳統的顯示方式，經由新的顯示方式可以帶給使用者更便利、更直覺的使用環境。 透過利用空手(Free Hand)和標記(Token)模擬滑鼠行為的系統，證明利用此方式，確實帶給使用者便利及直覺的使用環境。 在本論文中，利用投影機代替傳統顯示方式的優點:

1. 帶給使用者更寬敞的空間: 傳統顯示方式仍須一台機器用以顯示影像，但投影機可以架在空中或者置於架子上，並不會佔用使用者桌上的空間，讓使用者有一個更寬敞、便利的使用環境。

2. 直覺的使用方式: 傳統顯示方式，是以滑鼠控制游標為主，透過此種操控方式，使用者較無法直覺的與電腦做互動。 而本論文中，透過空手和標記控制游標，更是達到使用者與電腦最直接的互動。

藉由本論文的討論，投影機代替傳統顯示方式，仍會遭遇很多問題。 雖然本系統透過一些方法初步解決了這些問題，但仍有許多限制尚未改善。 因此，未來如何解決這些問題，是必須的。 也唯有解決這些問題，才能完全的利用投影機代替傳統顯示方式，創造一種全新的使用環境，讓使用者可以更直覺與方便的電腦做互動。

參考文獻

[1] Hiroshi Ishii and Brygg Ullmer, Tangible Bits: Towards Seamless Interfaces between People, Bits and Atoms, Published in the Proceedings of CHI '97, March 22-27, 1997, © 1997 ACM

[2] Jun Rekimoto and Masanori Saitoh, *"Augmented Surfaces: A Spatially ContinuousWorkspace for Hybrid Computing Environments"*, Proceedings of CHI'99, pp.378-385, 1999

[3] Gerhard Reitmayr, Ethan Eade, Tom Drummond,Localisation and Interaction for Augmented Maps, Mixed and Augmented Reality, 2005. Proceedings. Fourth IEEE and ACM International Symposium

[4] Takahiro Nishi, Yoichi Sato+, Hideki Koike*, Interactive Object Registration and Recognition for Augmented Desk Interface, *Extended Abstract of ACM SIGCHI 2001*, pp. 371-372, April 2001.

[5] Beth M. Lange, Mark A. Jones, and James L. Meyers.Insight Lab: An immersive team environment linking paper, displays, and data. In *CHI'98 Proceedings*, pp. 550-557, 1998.

[6] Gonzalez Woods 著, 謬紹綱編譯, 數位影處處理, 普林斯頓出版社出版

[7] Jeff Prosise 著, 集思廣譯工作室編譯, Windows 程式設計使用 MFC 第二版, 文魁資訊股份有限公司出版

[8] 位元文化編著, 精通視窗程式設計 Visual c++.Net 2003, 文魁資訊股份有限公司