國立交通大學

資訊科學與工程研究所

碩 士 論 文

空中寫字–以攝影機在空中的移動軌跡作為輸入中文的裝置

Writing in the air-Moving camera as Chinese word's Input device

研 究 生：陳奕安

指導教授：李嘉晃　教授

中 華 民 國　九 十 六　年 六 月

空中寫字-以攝影機在空中的移動軌跡作為輸入中文的裝置

研究生：陳奕安　　指導教授：李嘉晃　教授

國立交通大學資訊學院　資訊科學與工程研究所碩士班

摘要

近年來，手機與其他行動裝置的普及，使得它們儼然成為現代人最容易雖身攜帶的電子產品，這也使得行動裝置上的輸入模組更顯得不敷使用。本研究主要是利用行動裝置上的攝影鏡頭，搭配所開發出來的運動軌跡感測系統，以及分析感測到軌跡的辨識系統，達到以在空中移動攝影機作為輸入裝置，輸入中文字的功能。

Writing in the air-Moving camera as Chinese word's Input device

Student：Yi-An Chen　　Advisor：Prof. Chia-Hoang Lee

Department of Computer and Information Science
National Chiao Tung University

Abstract

Recently cellular phone or mobile device has become one of the most popular electronic consumer devices. This widespread usages have caused the urgent need of various and versatile input modules. This invention allows people to input Chinese writing by moving the mobile device in the air. The invention uses the camera equipped with the mobile device as well as develops a software to detect motion trajectory and recognition system for Chinese writing.

目錄

圖目錄

第一章　緒論

1.1 研究動機

近年來，由於通訊技術的成熟使得行動電話十分普及，行動電話儼然成為現代人最常隨身攜帶的電子設備。而在手機的使用上，也產生了一個新的問題：在這樣小的機器上要方便快速的輸入中文，變成了一件不容易的事。為了因應機器本身的大小，鍵盤必須要在按鍵大小和數量之間取得平衡。如此一來使得在手機上輸入中文字變得不太方便，於是行動電話上的中文輸入，便有了許多的變化方案。

1. 改善中文輸入法：

這類的做法，不去考慮鍵盤數量減少所帶來的影響，致力於改善現有的輸入法，藉由創新的編碼方式，使使用者輸入每個中文字所需要按的按鍵數盡量降到最低。缺點是，編碼的方式往往不夠直覺，使用者可能得花不少的時間去熟悉這樣的新輸入法。

2. QWERT 鍵盤：

(A)Nokia E61

(B)銓泰電子 BK600 藍芽對折式鍵盤

圖　1 Nokia 系列行動電話和外接式鍵盤

如上圖所示，直接把我們慣用的 QWERT 鍵盤內建在行動電話上面，或是直

接外接一個攜帶式鍵盤。如此一來自然是解決了使用者需要去學習新輸入法的困擾，但是帶來的問題，變成外接鍵盤本身過大，或是內建鍵盤的按鍵過小。這樣子仍然會帶來輸入上的不便利。

3. 手寫或語音輸入：

更甚，有些手機提供了手寫輸入或是語音輸入的功能。不過這兩項技術在目前產品的表現上並不佳，語音輸入本身很容易被環境干擾，而手寫辨識也受限於手機本身的狹小螢幕，使兩者的辨識效果不如預期。

目前的行動電話普遍都附帶有攝影機的功能。因此我們便想到利用這項裝置。利用攝影機捕捉到的畫面，我們是不是可以在空中寫字呢？另一方面，使用攝影機作為取代滑鼠等傳統用於桌上型電腦指向裝置的技術，也可以和這樣的想法作結合，使得利用攝影機做為輸入中文字的想法成為可能。

1.2 研究目標

我們預期的系統，主要分為兩個部份。第一個部份是藉由擷取到的攝影機畫面，偵測出攝影機在空中的位移量，也就是經由影像處理的方式，找出攝影機在空中的軌跡。第二部份，是對捕捉到的軌跡作辨識，找出最有可能的中文字。

我們在攝影機上，定義一個按鍵開關，使用者在空中寫字的時候，必須按下這個按鈕觸發影像的擷取。而在輸入方式上，也可以依照習慣，分為連續性的輸入和非連續性的輸入。連續性輸入意即為每一個字的筆劃輸入，是一次完成的，只需要按下按鈕到結束即可。非連續性輸入為每一個筆劃都視為分開的軌跡輸入。

軌跡本身可能十分的紛亂，因此我們第一部分的重點要放在如何減少攝影機

軌跡的變動情形。第二部份的重點在於，要建立一個提供辨識幫助的模型，畢竟軌跡本身是很難對應到中文字的，要用其他的方法來辨識，而且還得保持即時的運算量。畢竟若是輸入一個字如果還要等一段時間才能得到結果，這樣的系統在輸入文字上也就不太具有價值了。

1.3 論文結構

第一章為緒論，內容著重於說明研究的動機，目前各種為了因應可攜帶式電子產品的大小，所做出調整的輸入方式介紹，以及我們提出系統的大略架構解說。

第二章為背景知識，內容為簡單的介紹目前常用的攝影機影像行動偵測方法，如 edge detection，region detection 以及 optical flow。

第三章解說我們提出系統的第一部份，說明攝影機擷取影像的處理流程，各種消除雜訊，導正軌跡的處理方式，以及如何將軌跡分類的過程。

第四章解說我們提出系統的第二部份，說明如何把我們所擷取的軌跡資訊轉換為筆畫串列，並用特別的方式去做比對，以達到我們想要的辨識效果。

第五章為結論，總結我們系統的優缺點，以及未來可以改進的地方。

第二章 背景知識

一般常用來偵測 camera 在空中移動的方法，有 edge detection，region detection，optical flow，底下即對這些方法做大略的解說。

2.1 edge detection

edge 在灰階影像中代表的意義，為區域性灰階值有大變化的發生。比如 edge 常出現在物體和背景的交接處，因此藉由偵測 edge，我們可以得到物體的外型，進而應用於影像比對或是影像識別之中。

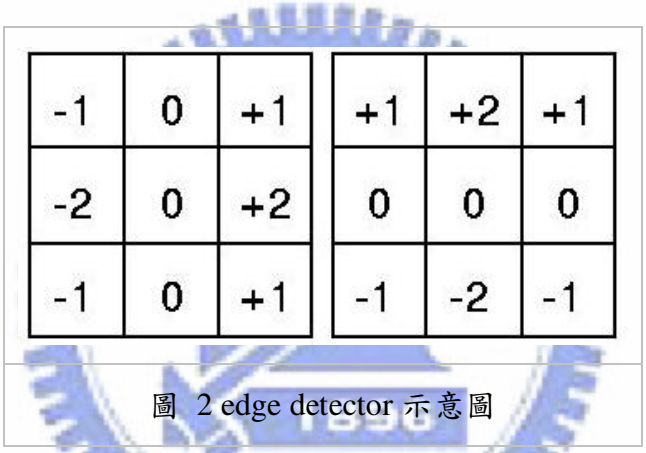

圖 2 edge detector 示意圖

以影像處理所常用的各種 edge detector，先將影像做抓取 edge 的處理。之後再針對已經處理後的影像，選取具有特殊特徵的 edge 區塊做為系統的特徵區域，之後再對連續的影像之間搜尋特徵區塊的變動位置，以得到攝影機的移動資訊。

而採用 edge detection 來偵測攝影機移動容易產生的問題為，特徵區塊選取的優劣，會對這方法的結果影響很大。且在擷取的畫面本身顏色分佈平滑的情況下，畫面的 edge 會相當不明顯，而容易導致較不好的比對結果。

2.2 region detection

region detection 目前主要的應用範圍常用於人體的偵測，比如人臉的偵測，先利用 Image segmentation 的方法，在 color space 中先將影像分為一塊塊的 homogeneous region，讓類似的顏色區塊得以分開，接下來再藉由臉的形狀以及膚色特徵去做辨識，以達到追蹤辨識的效果。

2.3 optical flow

optical flow 主要的應用目標，是用於追蹤畫面中的某個特定目標的移動。比如應用於讓攝影機隨著畫面中某個人的移動，而跟著轉動鏡頭追蹤。或是用於拍攝球賽的應用。

從連續的變動影像中，可以求得影像 pixel 的 optical flow，從影像平面空間觀察 optical flow 的變化量，便可以推導出目標物體在空間中移動的訊息。反之我們也可藉此推斷攝影機的移動。

Optical flow 的定義為影像中各個 pixel 的亮度梯度(gradient)，在空間中一個物體經由攝影機投影於平面上，除了因目標物體和攝影機的相對運動會造成影像的 pixel 光度產生變化之外，還有其他的因素會改變 pixel 的光度。例如環境光源的改變也會影響之。

而往往擷取整張影像作為求取 optical flow 的資訊並不一定能計算出正確的值；真正正確的 optical flow 通常是出現在影像中有移動且是特徵的區域。在變動稍微快速的連續圖像中，optical flow 常常會有較不好的效果。

第三章　系統架構---攝影機軌跡擷取

本系統主要可分為兩個主要的部份：

一、先將攝影機所擷取到的影像做 Image Processing，取得並記錄攝影機在空中移動的軌跡。

二、再依據所記錄處理後的軌跡，去辨識出和哪些字最為相似。

第三章的部份我們將先介紹第一部分的架構以及流程。整個系統架構圖可以簡單的以下圖表示之。

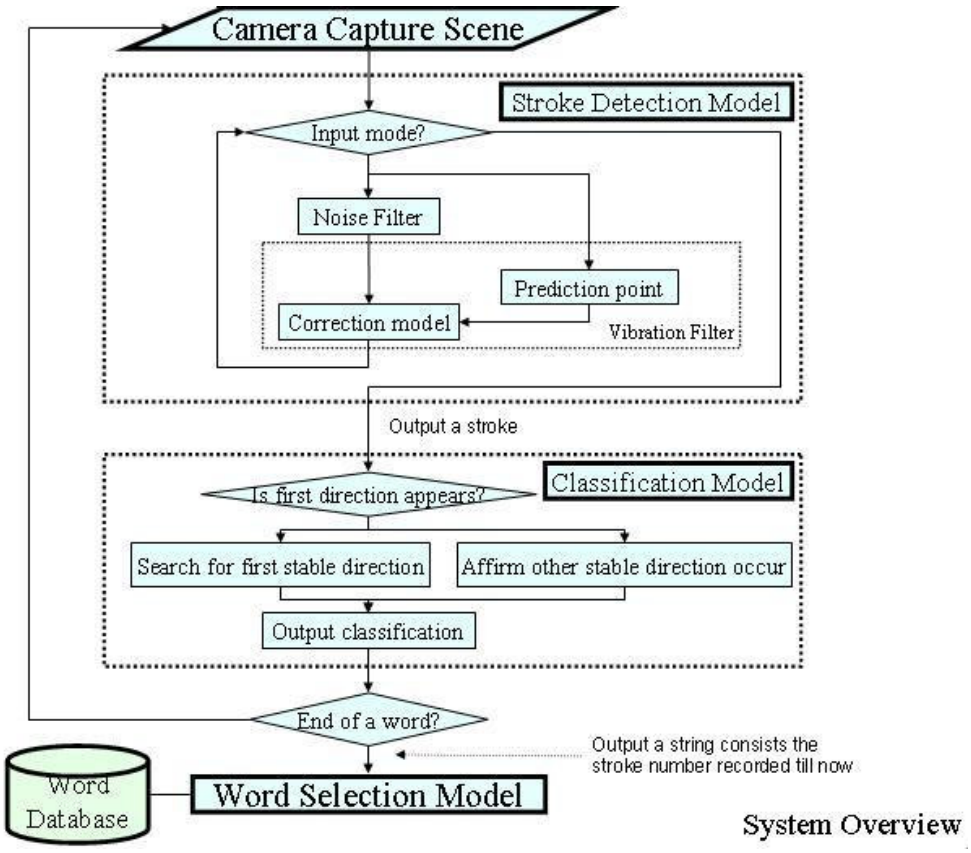

圖 3　System Overview

第一部分的系統大致上可以分為 Camera Capture Scene，Stroke Detection Model，Classification Model，第二部分主要的內容是 Word Selection Model，和一個已經建構好的 Word Database。

以流程圖來看，我們的系統一開始先要求使用者按下一個按鈕，以代表開始紀錄攝影機的軌跡（連續筆劃和非連續筆劃的輸入，這邊的差別在於按鈕次數的多寡）。在系統紀錄完軌跡之後，便會將軌跡資訊傳給 Classification Model，Classification Model 會把軌跡資訊轉換成為我們所定義的筆劃數字串列。在每一筆筆劃數字串列被辨識出來後，系統會考慮目前累積的筆劃數字串列長度，自動去調整比對 Word Database 裡面的資料，找出最相似的候選中文字。

3.1 Camera Capture Scene

這部份的章節內容是說明對攝影機影像的處理，目的為藉由拍到的連續影像，偵測出攝影機在空中的位移，產生系統所需要的最初步軌跡資訊。不同於前面文獻探討所提出的三種方式，我們採用另外一種更直覺，計算更簡便的方式來實做。

這部分的系統流程如下圖所示：

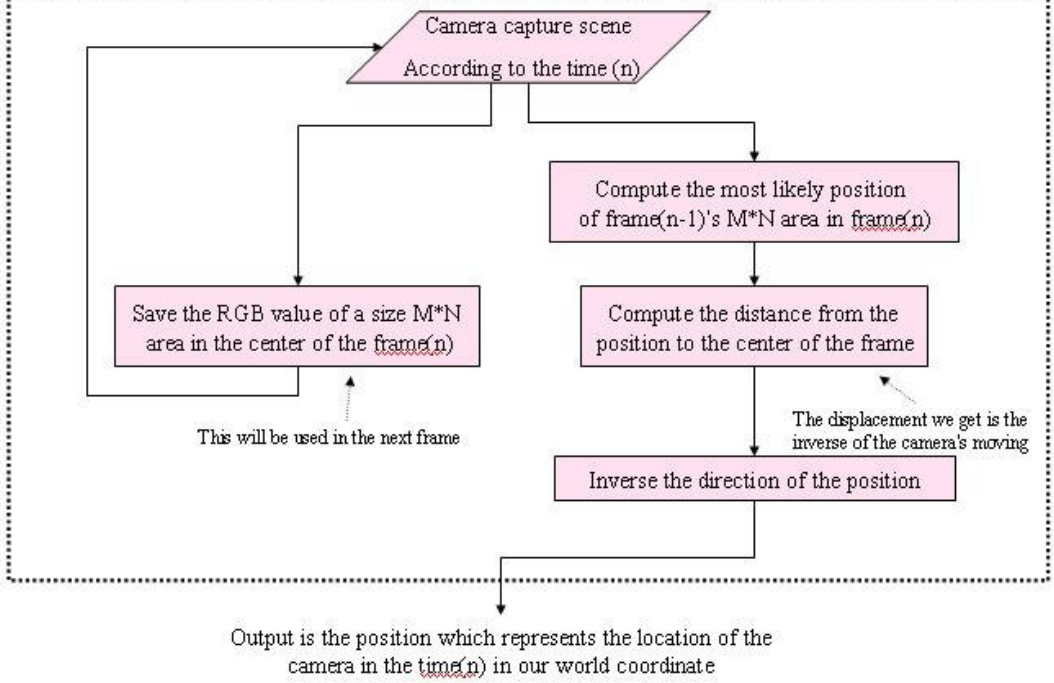

圖 4　Camera Capture Scene

首先我們假設攝影機每一個 frame 的大小都是 X * Y 的 size，我們以畫面中心取出 M * N 大小的一塊區域作為特徵區域(M < X，N < Y)，藉由計算出本來在 frame T 中心的 M * N 區塊，在下一個 frame T+1 中的位置，我們就可以大略估算出 Camera 在這個時間內於空中的平面位移。

上述的處理是基於偵測 Motion Vector 的概念為主軸，而為了讓這樣概念可以以 real-time 的處理速度實現在我們的系統上，必須加上一些額外的前處理。

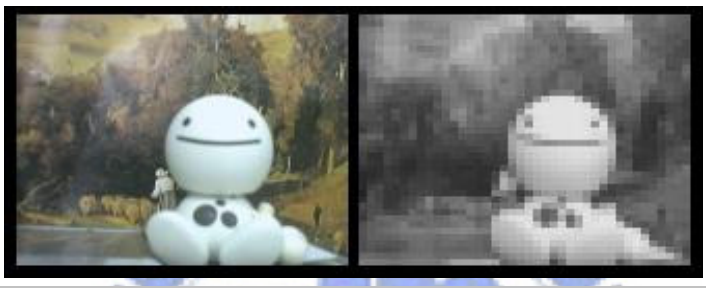

圖 5　將擷取的影像做灰階化和區塊化的處理

基於降低運算量的考量，在將擷取到的 frame 灰階化之後，更進一步的把畫面切割成一塊塊的 block，而每一個 block 的灰階值所代表的是整個 block 涵蓋的 pixel 之平均灰階值。而 block 的大小，也可以依照系統需求進行調整，比如說一般拿著攝影機在空中移動，我們會發現左右移動的範圍通常比上下來的大。因此在此考量下我們所採用的 block 並不是正方形，而是使用矩形來實做，藉由縮小 block 上下所涵蓋的 pixel 數目，來增加偵測攝影機上下移動時的精密度。

在把原本的擷取到的畫面區塊化之後，在下一個 frame 之中找最相似的特徵區塊時，採用 MSE 的計算方式。和特徵區域差異最小的區域就是最有可能的區域。這部份用下面的公式表示之：

$$MSE(dx,dy)=\frac{1}{MN}\sum_{m=x}^{x+M-1}\sum_{n=y}^{y+N-1}\left[Frame_T(m,n)-Frame_{T+1}(m+dx,n+dy)\right]^2$$

(dx,dy)所代表的是在 frame T+1 中移動的 xy 位置，已藉此找出最小的 MSE 值。因此攝影機的移動向量及可用下式表示：

$$\overrightarrow{MV} = (MV_x, MV_y) = \min_{(dx,dy)\in R^2} MSE(dx, dy)$$

我們藉由比較特徵區域所涵蓋的 block 之間灰階值的 MSE，找出原本位處在上一張 frame 正中間的特徵區塊，在目前的 frame 中的位置。要記得的是，畫面移動的方向恰好是攝影機移動方向的反方向。所以我們把畫面移動的方向做反向的處理，就能夠得到攝影機的移動方向向量了。

3.2 Stroke Detection Model

經由前述的軌跡擷取過程，我們已經能夠藉由 frame 與 frame 之間的影像處理，而得到 Camera 在空中的平面位移資訊。但是這邊產生的新問題是，我們並不能保證經過上述的處理後所得到的攝影機位移資訊是百分之百正確。

假使我們只是單純的記錄下我們所偵測到的每一個 Motion Vector(以此來代表攝影機的移動)，那麼我們會發現有時候會有這樣的偵測結果發生：

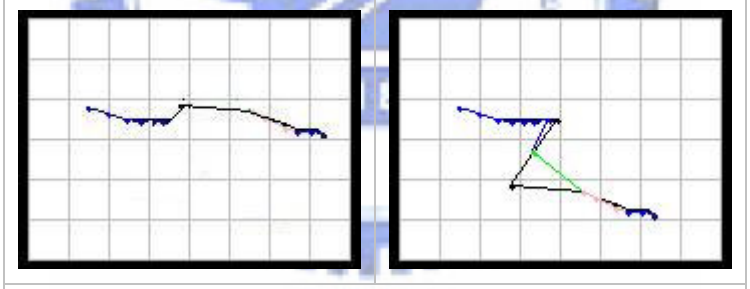

圖 6　非預期的失誤情形

如上圖所示，我們本來預期的軌跡路線如左是一直線，但是卻因為中間出現的一個偵測失誤，使得線段中間出現很大的轉折，進而可能影響到後面的筆劃判斷。

這種錯誤出現的原因，往往是攝影機擷取到的背景過於單調，比如說是一整片白色的牆壁；或是具有一致性或是重複性，比如擁有相同紋路的一整片地磚、天花板，過於接近的色塊重複性的出現，使原本以為是特徵的區塊不再具有特徵

的性質。

如此一來，想要單單藉由影像處理而得到正確無誤的軌跡資訊，就變得十分的困難，如果我們增加上面的章節所敘述的特徵區域大小，或是多增加幾個特徵區域來比較，或是縮小 block 的大小，這麼做確實能夠使軌跡偵測的準確率上升。但是隨之衍生出來的新問題是，我們必須做出運算時間大幅增加的犧牲，甚至會變成無法即時判斷出輸入的軌跡是哪些字了。以系統要能夠即時運算為重要前提考量下，我們只能藉由其他的方式技巧儘可能的消除這一類的錯誤。這便是 Stroke Detection 此部分模組所需要克服的問題。

所以那麼接下來我們要做的處理，就是利用其他的數學方法及技巧來消除可能發生的偵測錯誤，我們增加了一個 Noise Filter，一個 Vibration Filter，和一個 Correction Model 來解決這個問題。這部份的流程圖如下所示：

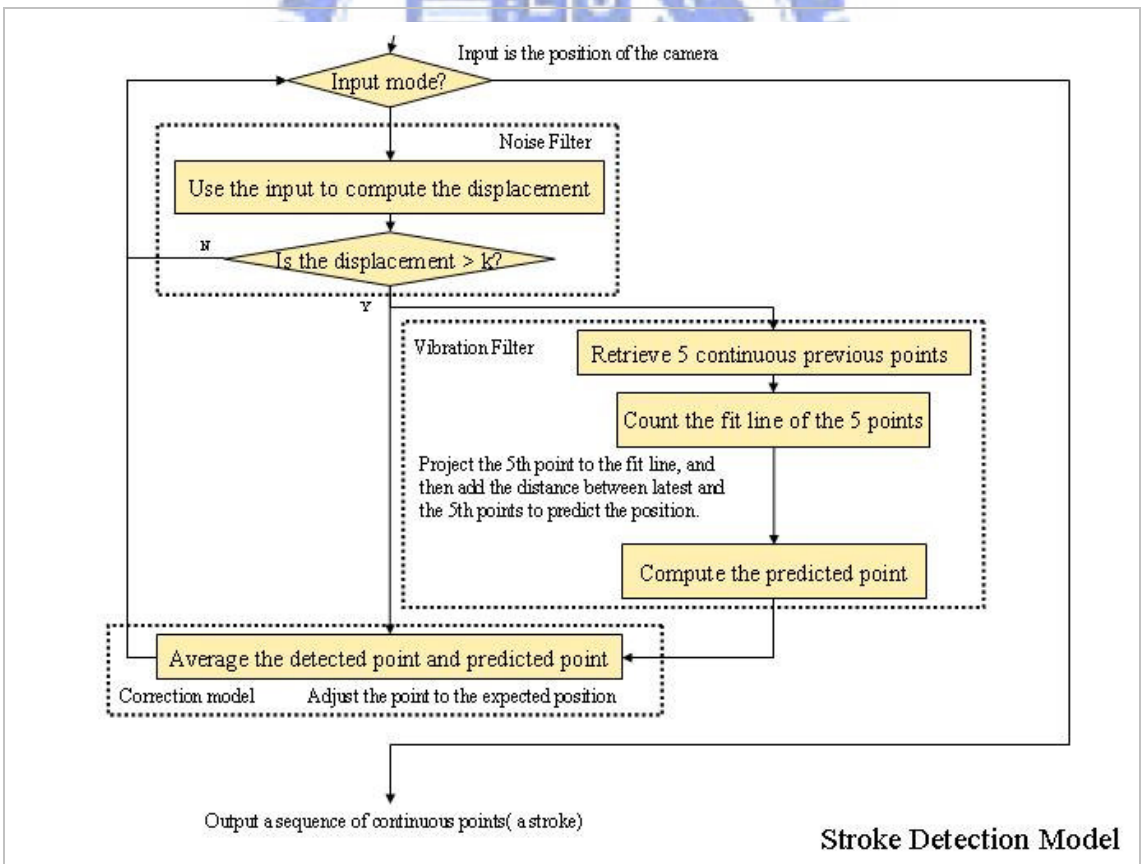

圖 7　Stroke Detection Model

3.2-1 Noise Filter

首先我們要面對的問題是，當 Camera 靜止時，手部的任何無意識抖動仍然會使攝影機在空中產生位移，但是紀錄這樣的資訊對我們的目標是沒有意義的。因此在 Noise Filter 之中，我們簡單的定義一個值 k，並且在每一次我們偵測攝影機位移的位置後，計算和上一個紀錄點的位置，只有在攝影機的位移距離要大於這個值的條件下，系統才會儲存所偵測到的新位置。

經由這樣的簡單過程，我們便簡單有效的先消除了一部份對系統無用的雜訊。

3.2-2 Vibration Filter

這部分重要的想法為，首先，在一般的情況下，我們拿著攝影機在空中移動輸入文字時，不會是無目的的在空中隨處飄移；相反的，通常都是有某種目標的方向性移動，比如說，這次的移動目的就是要輸入橫的筆劃，或是豎，或是撇、捺。以此想法假設下，我們可以用 Least Squares Fitting 的方式來導正所偵測到的軌跡。

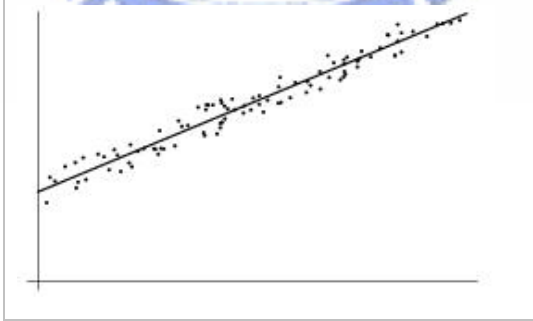

圖 8　Least Squares Fitting 的示意圖

利用 Least Squares Fitting 的目標是，我們先將已經偵測到的前數個點拿來使用（我們所採用的是前五個已記錄的點），計算出一條最有可能的軌跡前進路徑。

前面偵測記錄到的點，在某種程度下可以反映出目前使用者所想要移動的方向，因此若是下一個偵測到的位置，過於偏離原先的態勢，我們便可以用這條計算出來的假想直線來進行修正。使用的公式如下所示：

$f(a,b) = a + bx$　f(a,b)代表計算出來的直線方程式。

$$\begin{bmatrix} n & \sum_{i=1}^{n} x_i \\ \sum_{i=1}^{n} x_i & \sum_{i=1}^{n} x_i^2 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} y_i \\ \sum_{i=1}^{n} x_i y_i \end{bmatrix}$$ 如此我們可以算出 a,b 的值。

利用前五個點所算出來的 f(a,b)便可以幫助我們局部的對偵測點進行矯正，但是得記得的是，不能使這個結果過度干涉偵測系統的結果。

3.2-3 Correction Model

把偵測到的點，和上一個已記錄點的距離算出來，並且投影到上一節提到的直線 f(a,b)上，得到的這個位置，我們稱之為預測點。把預測點和實際測得的點作平均，便可以得到最後被系統採納記錄的位置。

一般的情形下，我們採用各佔一半的權重，但是在特殊的情況下，我們會對權重預測點作出調整，以免過度影響軌跡的真實性。這會在後面的需要做調整的部份被提及。

3.3 Classification Model

我們先看一張經過上述的 Camera Capture Scene 以及 Stroke Detection Model 的方式處理後，在空中寫出“木”所擷取並記錄的軌跡路線。

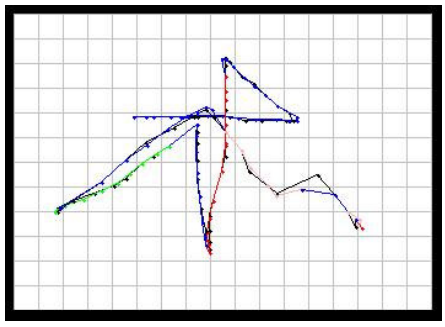

圖 9 以連續輸入方式所輸入的“木”

以人的角度來說，這個字還勉強看的出來是在寫什麼字。但是我們問題就在，電腦應該是看不懂的。這一節的重點即為我們要怎麼把這樣的筆劃軌跡資訊依照定義的方式做分類。我們的系統有了這樣的資訊，才能夠把軌跡和 Word Database 之間結合，進行比對辨識的工作。

就如先前所提到的，我們提供了兩種輸入模式：連續的筆劃輸入，以及非連續的筆劃輸入。非連續的筆劃輸入，所指的是每一次按下按鈕時，只輸入一個筆畫。連續的筆劃輸入則是按下按鈕後一次輸入多筆的筆畫，兩者在軌跡的辨識上就有些許不同的作法。但是主要的流程是相似的，後面將針對這兩種輸入方式在 Classification 部分不同的地方作分開解說。

3.3-1 非連續的輸入方式

在非連續的輸入中，我們依據軌跡之間的方向，將軌跡線段歸納成五類。

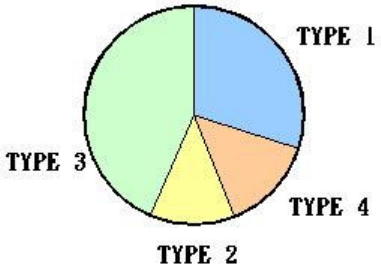

圖 10 非連續性輸入的軌跡分類圖

除了上圖所展示的四類之外，只要是包含有轉折的軌跡，即被歸類為第五類軌跡。所以非連續輸入，總共有五種軌跡。下圖將舉一些實際的例子來幫助瞭解：

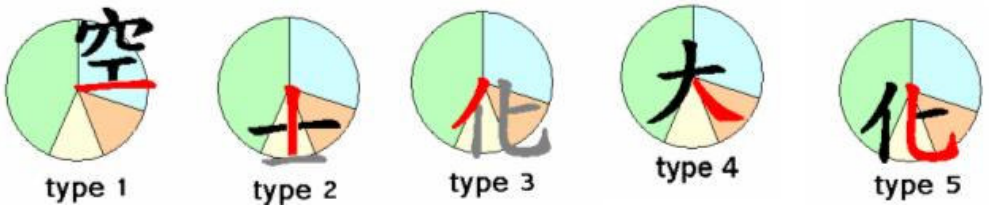

圖 11 五種種類的筆畫例子

經過對軌跡方向的定義之後，我們需要一個過程將系統紀錄好的軌跡資訊，分類成這五種其中之一的模組。流程圖如下：

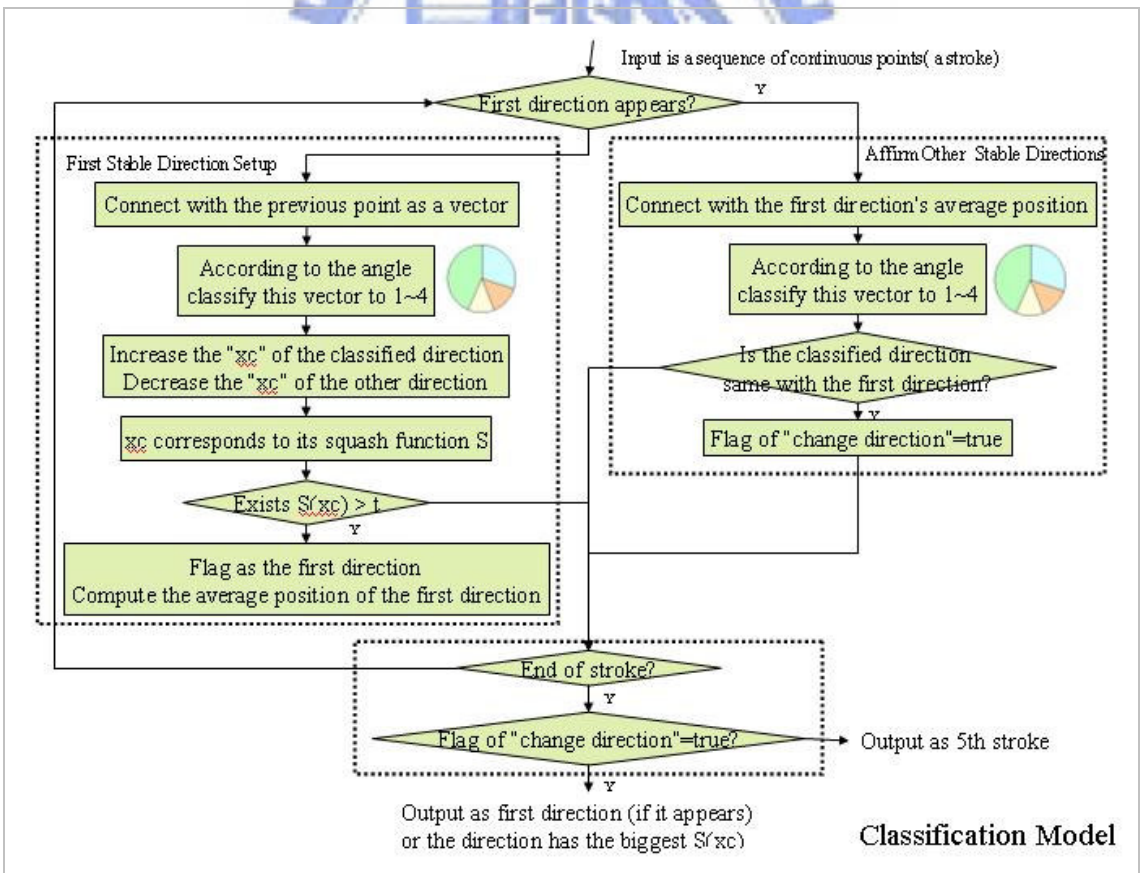

圖 12 Classification Model(非連續)

非連續輸入的流程分成兩個部份。第一個部份系統要先找出軌跡的方向，這部份我們使用 squash function 的概念，首先給 1~4 四種 vector 一樣的起始值，而隨著 evidence（這邊的 evidence 所指的是 sequential 的軌跡分成一段段的向量依序輸入）的輸入，依據 evidence 本身的方向，使同方向的 vector 的值上升，不同方向的值下降。而隨著四個 vector 本身的數值改變，對應到 squash function 的值會有指數的變化，而這個數值代表的是這個軌跡被分類成哪種方向的可能性最高。以下圖解釋 squash function 的對應情形：

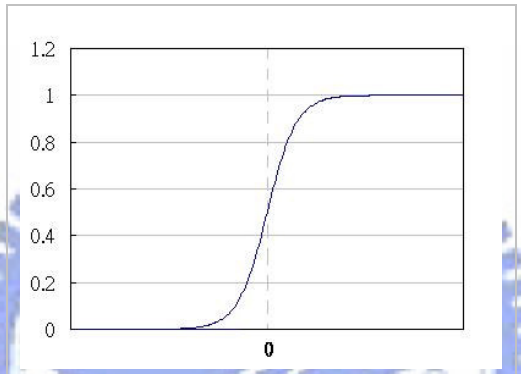

圖 13 $f(x)=\frac{1}{1+e^{-x}}$

用這樣的想法，系統再加上一個定義好的值 t，只要當某個方向 vector 對應 f(x)的值大於 t，就把這個方向 vector 當做這個軌跡的第一個特徵方向向量。如果我們只把筆畫分類成四種，那麼這邊的工作就已經結束了。但是中文字常常是存在有第五種形式的筆畫：也就是含有轉折的筆畫，我們定義為第五種軌跡。

為了正確的辨識出這種軌跡，我們在確定軌跡的第一個特徵方向向量之後，計算出這一個軌跡的方向向量中心點，並且在後續的處理之中，用這一個中心點去和後續的偵測點連起來，再去計算這線段的方向向量是否和第一個特徵方向向量相異，當做軌跡的轉折是否存在的判斷依據。

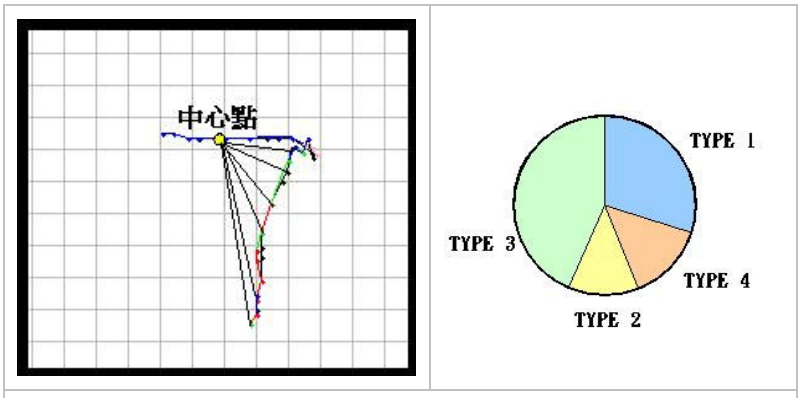

圖 14　第五種軌跡的判斷

由上圖所示，第一個被偵測到的特徵方向向量為 TYPE 1，之後算出中心點後，藉由該特徵線段的中心點去檢查剩下的點，可以發現範圍跨越了 TYPE 4 和 TYPE 2，因此這個輸入軌跡便會被系統辨別為第五類軌跡。

3.3-2 連續的輸入方式

以“水”這個字為例，若是以連續的方式輸入這個字，我們可以得到下面的這個軌跡結果：

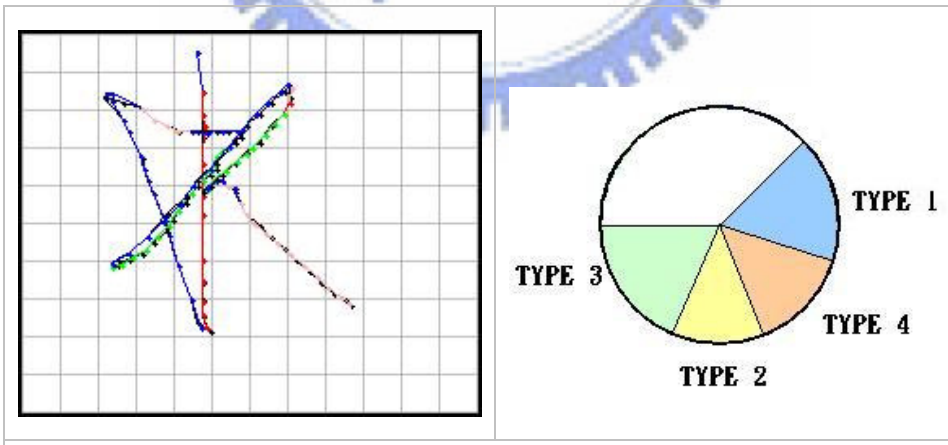

圖 15　水的連續輸入軌跡圖

在連續的輸入模式下，我們的改變在於不再有第五種代表轉折的筆畫，因為系統無法單單從一串軌跡的記錄資料中，分辨出哪些筆畫應該是連在一

起，哪些筆畫應該是分開的，更甚，哪些筆劃只是實筆之間的連帶產生的虛筆。不過這部份的問題，我們可以留待 Word Selection Model 的部分來解決。目前我們只要專注於將連續的軌跡一一拆開來辨識即可。

相較於非連續的輸入，我們必須把四種特徵方向的範圍重新定義。把整個左上到右上的區塊切除不定義的原因是，一般中文字經過我們的觀察，鮮少有往左上以及往右上方向的筆畫。因此為了避免筆劃和筆劃之間的軌跡使辨識的工作更加地複雜化，我們可以捨棄掉這部份的資訊以降低我們的辨識難度。

而必須注意的是，在前面所提及的 Vibration Model，必須在連續輸入的前提下進行小幅度的修正。因為 Vibration Model 的用意是減少主要路徑以外的偵測點的產生，而連續輸入會因為常常變換軌跡的主要軌跡方向，過於採信 Vibration Model 所提供的預測點往往會使得軌跡發生難以轉彎而因此漏失資訊的現象。因此我們可以在這邊加上一個條件：當系統察覺到軌跡似乎有轉向的跡象時，降低 Vibration Model 所提供預測點的權重，讓實際的偵測點佔更大的比例，如此一來當軌跡的特徵方向改變時，Vibration Mode 將不再能強硬的矯正軌跡的方向，上述的問題也能夠獲得解決。

因此，連續輸入的流程圖較為簡單：

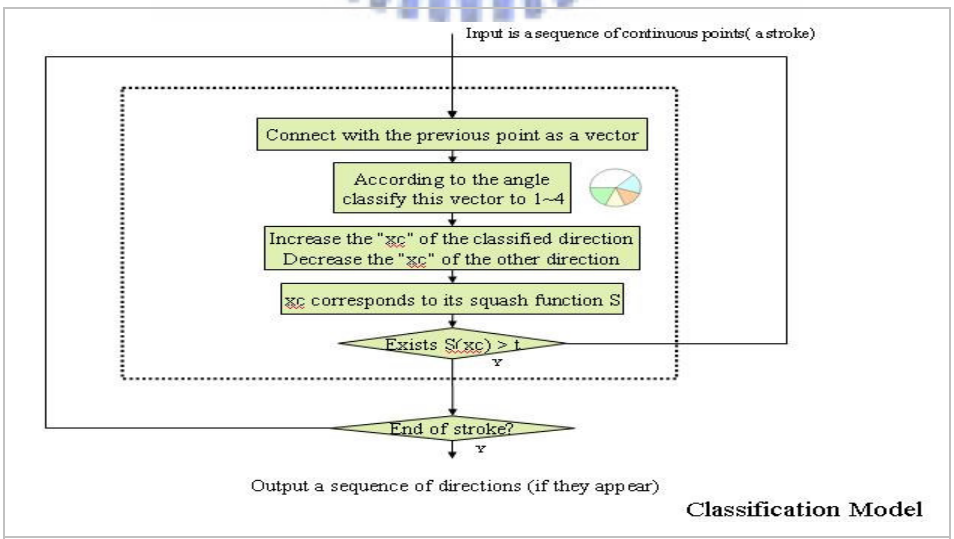

圖 16　Classfication Model(連續)

3.3-3 Conclusion

經過上述的處理之後，不論是非連續輸入或是連續輸入，我們已經把從攝影機擷取到的軌跡資訊轉換成我們想要的筆劃數字串列資訊了。接下來 Word Selection Model 的工作，就是要將處理好的筆畫分類資訊，在 Word Database 中比對，辨識出最為相似的候選字。

第四章　系統架構---字庫的辨識比對

經由上一章的處理後，我們已經得到了分類後的筆劃數字串列資訊。剩下的工作便是定義出一種方法來準確的辨識出我們真正想要寫的字。這章主要為介紹 Word Database 和 Word Selection Model。

4.1 Word Database

在介紹 Word Selection Model 之前，我們需要先瞭解所 Word Database 裡面的內容以及各個中文字該如何定義，才能夠接著考慮要採用怎樣的比對方法。

Word Database 簡單的說，就是我們為了這系統編寫的一本字典，裡面重新定義了每個中文字在我們的系統中所需要的編碼。以下即依據連續性輸入和非連續性輸入不同的需求，介紹編碼的方式。

4.1-1 非連續性的輸入方式

以下便用簡單的例子解說我們如何把一般字 encode 成我們需要的資訊串列：

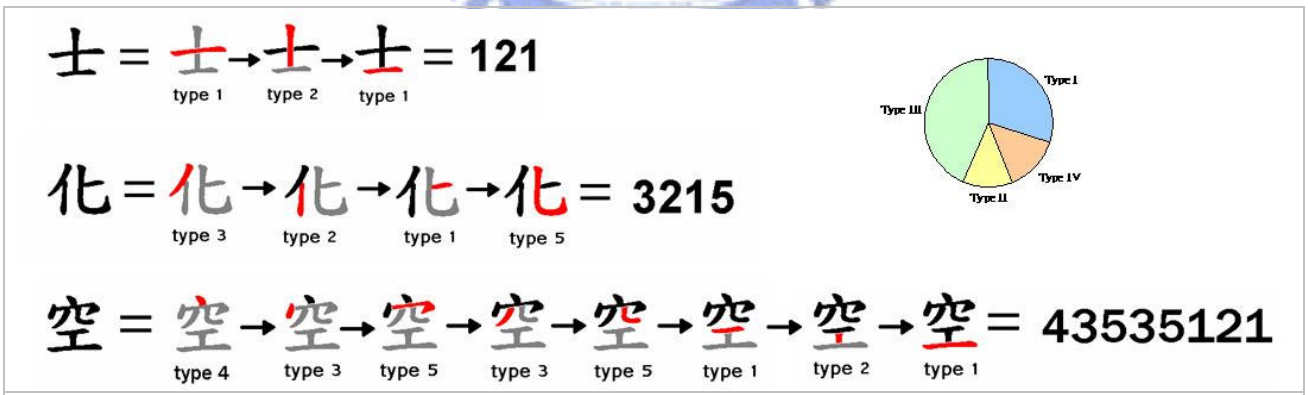

圖 17　非連續性輸入的 encode 方式

非連續性輸入的定義方式，共有五種不同的軌跡。依上圖所示，“士”這個字，第一劃是一橫，所以 encode 為 2。第二劃是一豎，encode 為 1。第

三劃又是一橫，所以“士”在非連續輸入的 Word Database 中被 encode 為“121”。同理，“化”被 encode 為“3215”，“空”被 encode 為“43535121”。

我們把常用字藉由上述的方式 encode，輸入 Word Database 中，當作辨識用的資料依據。

4.1-2 連續性的輸入方式

在這種輸入模式下，我們需要多加考量的是筆畫和筆畫之間的虛筆，也要 encode 到 Word Database 中。因為系統本身是很難去分辨哪些是實筆哪些是虛筆，所以把它們通通 encode 起來也是一種解決的方式。

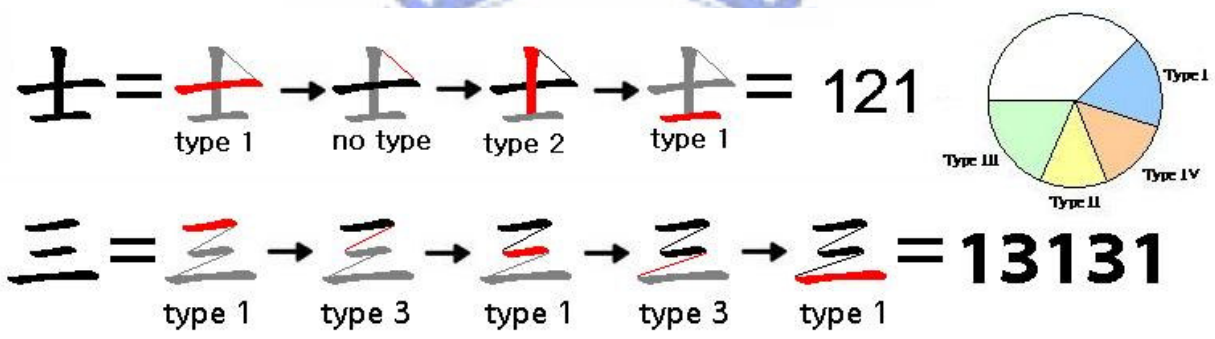

圖 18　連續性輸入的 encode 方式

“士”這個字，在連續性的輸入中，因為虛筆的部份是被我們當成不需要定義的部份，所以 encode 和非連續輸入的結果一樣都是“121”。

但是“三”這個字，就不一樣了。“三”是三個實筆兩筆虛筆所組成的字，因此 encode 起來變成了“13131”。這個方向的虛筆是不能夠被去除的，不然會連帶影響到撇這類實筆的輸入上的辨別。

可以簡單觀察到的是，連續性的筆劃輸入，不僅僅加深了從攝影機擷取影像資訊的處理難度，也使得 encode 中文字到 Word Database 的複雜度上升。間接地也加大了辨識上的難度。

4.2 Word Selection Model

最後的這部份，我們要依據 Word Database，和由軌跡辨識出來的特徵方向字串做比對，找出最相似的候選字。

舉例來說，我們要處理的問題是像這樣子：

輸入的軌跡辨識出來的字串：3122344，真正想寫出來的字在 Word Database 中，被 encode 成 312434（我）。下圖為系統實際運作的狀況：

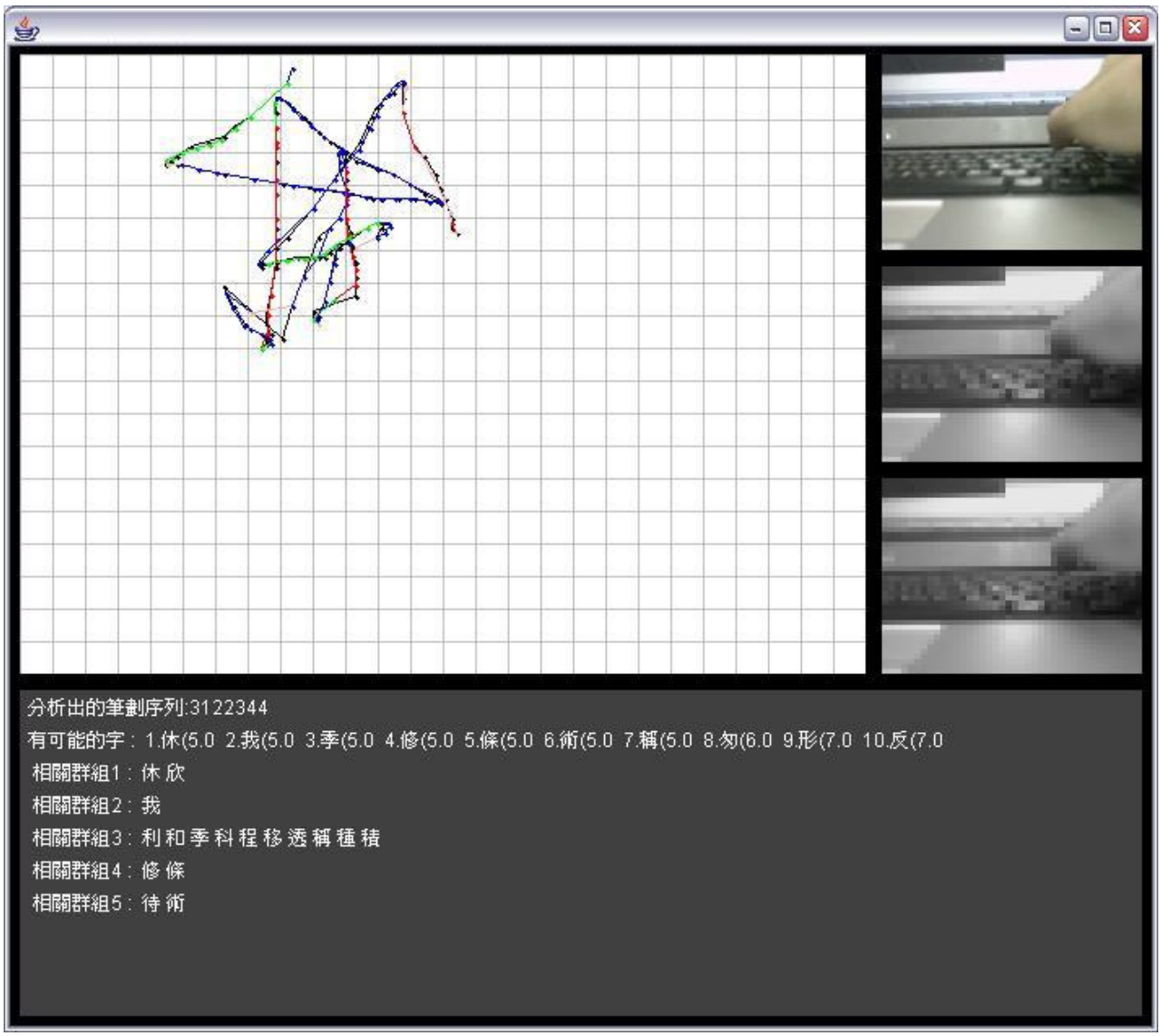

圖 19　系統中的實際範例

單單只看偵測到的軌跡串列，實在是連人都很難辨識出這個字就是“我”。而系統分析出的筆劃序列為 3122344，和 Word Database 中的 312434 差距也不小。不僅中間多了一個 2，少了一個 4，最後還多了一個 4。

顯然這樣的字串比對問題我們無法用傳統的字串比對演算法來解決，而且我們的系統還有必需要能夠即時處理這項大前提。過於繁複的精密計算，系統是沒有辦法承擔的。

4.2-1 相似度計算方式

於是我們想到了，平常常見的智力測驗題目中，常常有一種題型，先給考生俯視圖，以其其他方向所看到的圖，然後問題本身是詢問哪個立體圖最正確。

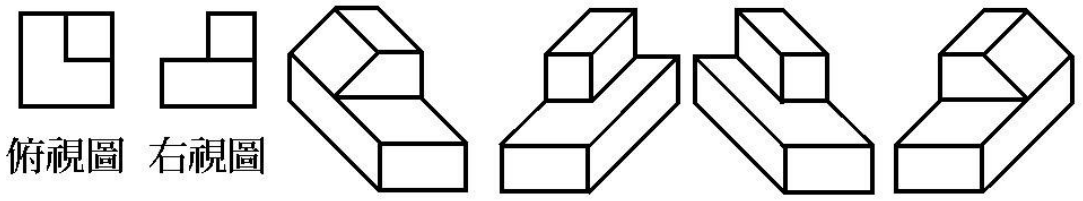

圖 20　常見的立體圖問題

同樣的道理可以應用在我們面臨的問題上面，3122344 和 312434，乍看之下真是不知從何比對起。但是我們如果先一次只看兩個數字，比如先只看 2 和 3 的部份，它們就變成了 3223 和 323。只看 2 和 4 的話就變成了 2244 和 244。藉由這樣的概念我們的問題一下子便簡化了許多。

於是我們定義了兩種計算用參數：

1. 1，2，3，4 各個數字出現的總次數。
2. 將原字串先處理成只剩下兩種數字後，12/21，13/31，14/41，23/32，24/42，34/43，這樣的字串連續出現的總次數。

第一種參數很容易理解，至於第二種參數為什麼要採用 12/21 這種而非 11/22，我們利用下表來解釋。

	11	12	21	22	1	2
111	2→1	-	-	-	3→2	-
112	1→0	1	-	-	2→1	1
121	0→1	1→0	1→0	-	2	1→0
122	-	1	-	1→0	1	2→1
211	1→0	-	1	-	2→1	1
212	-	1→0	1→0	0→1	1	1→0
221	-	-	1	1→0	1	2→1
222	-	-	-	2→1	-	3→2

圖 21 容錯性的比較表

這個圖表的內容為計算比較假設原本應該是要出現連續的三個數字中間的那個數字系統沒有偵測到，或是本來中間應該沒有多的數字，系統卻誤判使得多了一個數字的這兩類情況下，11/12/21/22 的總數變化情形。例如原本 121 誤判成 11，或是 23 誤判成 243 的情況。

從圖表中可以觀察到，11/22 對於系統發生上述兩類失誤時總數變化過於敏感，我們想要找的是一種較能容忍系統失誤而不至於產生劇烈變化的參數，這便是我們選擇 12/21 的主要理由。

而為什麼要採用這兩個參數的原因，和大家都玩過的一個小遊戲的思考方式有關。那就是使用?A?B 來玩的猜四個數字遊戲。A 在這個小遊戲中代表的是一種含有“位置資訊”的符號，B 所代表的是“不含有位置資訊”，但是有提供正確性參考的符號。

相似地，我們把“將原字串處理成只剩下兩種數字後，12/21，13/31，14/41，23/32，24/42，34/43，連續出現的總次數”這項特徵當成是我們的 A，把“1，2，3，4 各個數字出現的總次數”這項特徵當成是我們的 B。再給予 A 高於 B 的權重（畢竟數字出現的順序是很重要的資訊，我們把 A * 4 + B

當成最後的分數），用這樣的技巧，我們便能辨識出我們需要的候選字。

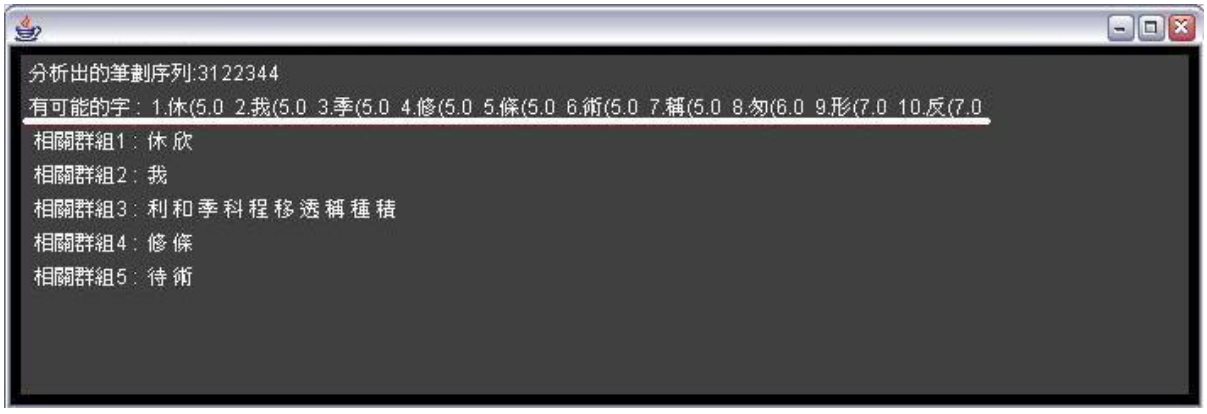

圖 22 候選字示意圖

我們先看上圖白色線段的部份，那代表 3122344 拿去系統中分析後，選出來最有可能的前十個字，“(”後面的數字代表他們的分數。如圖中所示，“我”的相似度分數為 5.0，詳細的計算過程如下

A 部分的分數：共多了一個 21 ➔ 1*4=4

B 部分的分數：共多了一個 4 ➔ 1*1=1

因此結果便是 4+1=5 分。

以式子來表達的話，可以這樣寫：

$$\arg\min_{w} \Pr(w \mid data) = \arg\min_{w} \frac{\Pr(data \mid w)\Pr(w)}{\Pr(data)} = \arg\min_{w} \Pr(data \mid w)\Pr(w)$$

其中，w 代表每個中文字，data 代表偵測到的筆劃數字串列。Pr(data|w)代表的就是上述兩項特徵參數計算後的結果，另外算上 Pr(w)的原因是，我們的比對會在中文字一邊輸入的時候就一邊進行，因此例如當使用者輸入到第五劃的時候，我們也會把所有少於或是超過五劃的字拿出來進行比對。但是自然的，比對的長度就不是這些字原本被 encode 在 Word Database 中的長度了。因此我們做一個假設，當使用者寫到第五劃的時候，原本就是五劃的字的可能性，稍微比少於或多於五劃的字的可能性來的高。但是基於怕過分影響 Pr(data|w)的部份，Pr(w)這參數能做的影響其實很小，但結果顯示加入此參數有助於提高辨識的準確率。

4.2-2 群組化

前面的系統範例圖中，在第一排的候選字下面，多了五個相關群組的選字，是有特殊用途的。

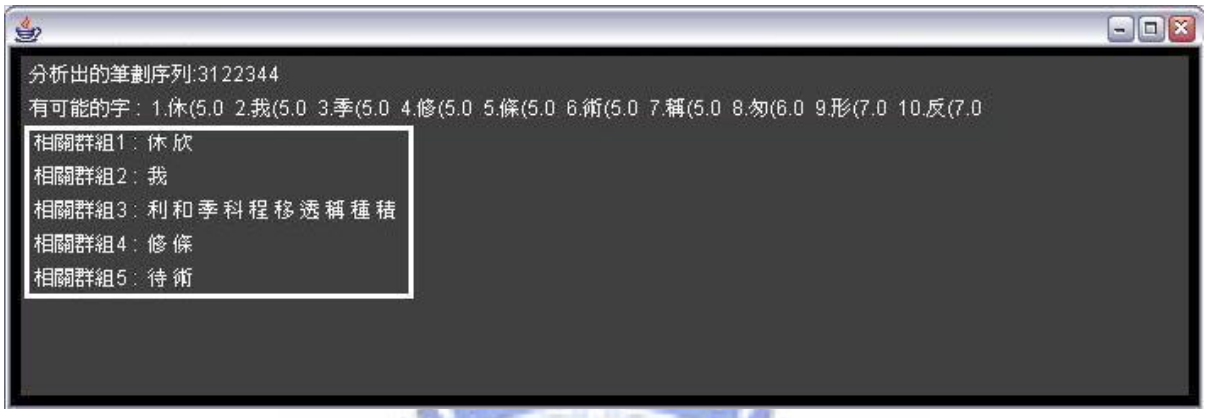

圖 23 群組化示意圖

我們觀察到，連續輸入，以及非連續輸入，都有一個共同的問題存在，那就是輸入十劃以上的中文字時過於耗時。非連續性的輸入由於每一筆軌跡都是獨立輸入，在準確度上還不會有影響。但是連續輸入在這個狀態下，系統多偵測到或是少偵測到軌跡的風險，也隨著筆劃數的增加越來越高。

因此，我們便想著，是不是能夠有更快速的輸入方式，使我們就可以用選字的方式選出我們要寫的字。

我們在觀察之後發現，中文字有一項很特別的特徵，即是部首的概念。很多字雖然乍看之下前面的部首幾筆筆劃都一樣，但是只要一寫完那部份，之後要接的是橫，是豎，是撇，是捺，就已經可以決定出我們要寫的是什麼字了。因此在 Word Database 中，我們可以先將已建檔的中文字，根據前幾筆筆劃資訊簡單的分類。

圖 24 群組化後的字組

以上圖的內容來解說，首先由於“休”是被排在第一順位的候選字，因此和“休”擁有共同的前五劃的字，會被我們找出來，“休”的前四劃是“32132”，和欣是一樣的。同理下面四個群組也是這樣被找出來的，我們很容易可以發現，起手寫的部首相似的字很容易被找出來。如此一來我們便不需要把一個字從頭到尾輸入完畢，只需要寫前幾個筆劃，就可以找到要輸入的字了。

4.3 Conclusion

因此，經過上面兩章的模組互相配合，我們的系統已經可以成功的在空中藉由攝影機達到輸入中文字的目的。

第五章 結論與展望

5.1 結論

在整個研究的過程中，我們用了許多小技巧，和各式各樣的方法來解決所遭遇到的各種問題。大多時候，我們都秉持著“避免產生過大影響，但是容許些許的誤差”的方式去解決問題。比如，以攝影機的影像為輸入來偵測攝影機移動的問題上，以現有的技術和硬體環境，我們受到的限制十分的大，且擷取攝影機移動軌跡的效果仍然是無法做到很完美。但是這方面的缺陷，我們便在下一個把攝影機軌跡分類的模組，想辦法減少誤差帶來的影響。甚至在和 Word database 的比對方法上，我們也下了很多工夫去想出一種容錯率高的比對方式，而又不能加重系統的計算負擔。最後的部份，為了能夠使使用者能夠更快速的得到自己想輸入的字，我們也想出了把中文字群組化的解決方式。

5.2 未來工作

系統中尚有許多可以改進的地方，比如四種或是五種筆劃的分類邊界，可以用模糊的權重方式做分類，而不是絕對的二分法；擷取攝影機移動軌跡的系統是否可以改良，使得偵測的部份更佳完善；辨別相似度的分數計算部份是否有更好的方法可以使用；或是在最後的選字部份，加上 Language Model 的幫助，使電腦選字更加的有智慧。

參考文獻

[1] Wang, J. and Canny, J. TinyMotion: camera phone based interaction methods. CHI '06 Extended Abstracts, 339-344, 2006.

[2] Rohs, M. Real-world interaction with camera-phones. Proc. UCS. IPSJ Press (2004).

[3] Ballagas, R., Rohs, M., Sheridan, J. G., and Borchers, J. Sweep and point & shoot: Phonecam-based interactions for large public displays. CHI '05: Extended abstracts of the 2005 conference on Human factors and computing systems. ACM Press (2005).

[4] M.E. Munich, "Visual Input for Pen-based Computers," PhD thesis, California Inst. of Technology, Pasadena, Jan. 2000.

[5] M. Nakai, N. Akira, H. Shimodaira, and S. Sagayama. Substroke Approach to HMM-based On-line Kanji Handwriting Recognition. In Proc. of Int. Conf. on Document Analysis and Recognition (ICDAR'01), pages 491–495, Sep 2001.