設計與實作直覺式手勢動作識別系統

朱鴻棋
朝陽科技大學
資訊與通訊研究所
助理教授
hcchu@cyut.edu.tw

鄭元欽
朝陽科技大學
資訊與通訊研究所
研究生
s9830620@cyut.edu.tw

摘要

隨著感測技術的進步，資訊設備操作介面也隨之改變。從傳統的指令操作演進成直覺式操作。帶給使用者不必事先學習的便利操作感受。目前在智慧型手機上已開發出幾種直覺式的使用者介面系統，如畫面直式/橫式自動偵測切換。但是，此系統無法處理複雜的自動化功能。因此，我們提出一種新的直覺式手勢動作識別系統，藉由手機中的重力感應器記錄手勢動作，並透過我們設計的手勢識別演算法識別手勢藉此實現直覺式手勢動作識別系統。

關鍵詞：手勢識別、重力感應器、加速規。

Abstract

As sensing technologies have developed, information equipment operating interfaces have changed. Thus, traditional instruction operation has evolved into intuitive operation, providing users a convenient operation experience without the need of learning in advance. At present, several kinds of intuitive user interface systems have been developed in intelligent mobile phones, such as picture vertical/horizontal automatic sensing switch and mobile phone face/back sensing switch. However, this system cannot process complicated action sensing. Therefore, this study developed a new intuitive gesture recognition system, where a G-sensor inside a phone records gestures, and then the gestures are identified by our gesture recognition algorithm in order to realize the intuitive gesture recognition system.

Keywords: gesture recognition, G-sensor, accelerometer.。

1. 前言

近年來對於手勢識別技術在各個領域已有許多研究，如電腦視覺技術[1]、智能手套[2]、慣性動作追蹤系統[3]以及移動式設備[4,5]。其各個技術中使用的設備也各不相同，在電腦視覺技術相關研究中，其透過運用紅外線感測技術來測量肢體動作與手勢動作，如 Wii Remote[6,7]。在智能手套方面的研究中，主要應用多種感測器(加速規)來測量精細的手勢動作[2]。其中加速規(Accelerometer)大多數應用於慣性系統的研究中使用，且系統結合陀螺儀(gyroscopes)與磁力計(magnetometers)來達到動作檢測功能。在移動設備方面，其手勢識別上主要使用加速規作為其感測元件來識別手勢動作[3,4]。然而，上述的電腦視覺技術，智能手套、慣性動作追蹤系統，需要大量的資料運算與特殊資料擷取設備，故對於移動式設備而言將不適用。

而在移動式設備中，由於智慧型手機的快速發展，手機的尺吋越來越小以及具備的功能也越來越多，在硬體設備上也具備了多種感測器，如重力感應器(G-Senosr)、陀螺儀、磁力計以及光感應器等元件。在智慧型手機作業系統的部分有 Android、Window mobile、iOS4 等，其中由 Google 所發表 Android 系統近幾年來也成為熱門的應用目標平台，此系統具備下列優點：

(1) 開放式作業系統：定義為使用者能夠根據自己需求添加功能或修改作業系統，且作業系統為公開原始碼不需要任何授權即可自行更改。
(2) 軟體開發自由度高；即為任何使用者皆可設計此作業系統的應用程式。且透過 Market 服務能分享或販售應用程式。
(3) 整合 Google 服務：如 Gmail 與 Google map 等雲端服務整合在作業系統中。

由於上述優點，所以使用者皆能自行開發應用程式在此系統上。但是，運用在手機上的手勢識別基礎互動上存在許多技術挑戰。首先

是手勢識別缺乏標準化與廣泛採用的手勢詞彙。第二是自發性互動需要被立即參與，例如當使用者在智慧型手機上輸入手勢後經過詞彙內預先定義的手勢識別後，其手勢相對應的結果必須要立即執行。第三是智慧型手機平台上受到成本與系統資源高度的限制，系統資源包括計算能力與電池電量等。

另外，大多數對於識別手勢軌跡方法是透過加速規擷取的三軸加速度變化值，經由歐幾里得距離(Euclidean Distance)$\sqrt{(x^2+y^2+z^2)}$計算後獲得加速規的總位移距離，最後藉由三軸加速度變化值與總位移距離來當作手勢軌跡的條件。但是只使用上述的兩項條件來識別手勢軌跡並不準確，並且會造成識別手勢軌跡失敗率增加的問題產生。

在智慧型手機上，雖然近幾年已針對直覺式的使用者介面系統，如畫面直式/橫式自動偵測切換。但是，此系統無法以此來完成複雜的自動化控制。

因此，我們提出一種新的直覺式使用者介面系統，藉由手機中的重力感應器記錄手勢動作，並透過我們設計的手勢識別演算法識別手勢藉此實現此系統。其中我們設計的手勢識別演算法能夠解決上述的挑戰問題，藉由此演算法能提高手勢辨識的精確度與提升成功識別率。

本文其餘的內容組織架構如下，第二部份將描述手勢識別技術的相關研究。第三部份提出我們的系統架構以及手勢識別演算法設計。第四部份是我們的實驗結果與分析。最後，為本文的結論。

2.相關工作

在一般的慣性動作追蹤系統中，動作方向的檢測是藉由感測器來測量[3]。可使用的感測器有加速規、陀螺儀與磁力計等。加速規的功能主要是測量加速度，而陀螺儀的功能主要是測量旋轉速度，磁力計的功能則是用來測量運動方向。因此，在慣性測量單元(Inertial Measurement Units, IMUs)上通常由上述的三個元件所組成，其功能是藉由使用陀螺儀或磁力計來測量動作方向的改變[3]。

然而，在慣性測量系統中存在兩個缺點，第一個是陀螺儀的價格非常昂貴。第二個是磁力計容易受到其他電子設備干擾產生誤差。因此，在[3]提出在動作追蹤是藉由使用三個三軸加速規來測量動作，其系統中提出的演算法是結合動作方程式(Extrapolation)方法與最小平方方法(Least Squares Method, LSM)以及最小平方問題(Least Squares Problem, LSP)，並採用 Lagrange multipliers 來將計算後的誤差值減至最低[1]。

在[4,5]中，提出一個只需使用單個三軸加速規來識別手勢動作的高效率識別演算法，稱為 uWave。其演算法對於每一個手勢模式只需要一個訓練樣本作為識別條件，並且允許使用者採用個人化手勢作為實體操作的識別。然而，uWave 演算法主要的核心是動態時間校正(dynamic time warping, DTW) [8]。動態時間校正被廣泛研究並使用在語音辨識系統上。其 uWave 演算法利用加速度量化(Acceleration quantization)、動態時間校正與 Template adaptation 組成，藉由對兩時間序列使用動態時間校正演算法來尋找同樣時間點時的最佳對應值。藉由最佳對應值的誤差大小做為識別手勢的條件。

在[2]中，智能手套中使用五個二軸加速規來進行手勢識別，其加速規是裝置於手套上的五個手指頂端。因此,，藉由五個加速規能夠精確的識別手指動作。然而，在移動式設備上受到成本與尺寸大小的考量。因此，在硬體設備上無法具備數個加速規。

但在過去大多數的手勢識別研究中主要注重於檢測手部動作的輪廓，而不注重手指動作。另外，針對手勢識別上的相關研究已有許多研究與探討是使用電腦視覺技術[1]，如 Wii 遙控器(Wii Remote)[6,7]與 Kinect[9]。在 Wii 遙控器體感運作原理主要可分為兩項功能，分別是指向定位及動作感應。指向定位是透過紅外線感應來追蹤定位座標，運作方式是由光學感應條(Sensor Bar)內的 LED 發射紅外線，在 Wii 搖控器前端的紅外線 COMS 感測器來接收光學感應條所發射的紅外線光點來判斷光學感應條與搖控器之間的位置與距離。另外，藉由接收到的紅外線光點透過紅外線室內定位技術來完成定位使用者的相對位置。以藉此來達到指向定位。動作感應則可偵測三維空間當中的移動及旋轉，主要是透過 Wii 遙控器內建的 ADX330 鏡片(Accelerometer)所收集到的 x 軸、y 軸以及 z 軸等三軸電壓變化值來判斷傾斜度與移動方向。因此，結合指向定位及動作感測可以達成所謂的體感操作。

而在微軟 Xbox 360 的周邊設備 Kinect 上，其機身上具有 3 種鏡頭，分別為 RGB 彩色攝影機、紅外線發射器與紅外線 CMOS 攝影機，透過紅外線發射器和紅外線 CMOS 攝影機構

成 3D 深度感應器。其 3D 深度感應器為 Kinect 主要偵測使用者動作的元件。因此，Kinect 透過上述元件可提供一次擷取三種元素，分別是彩色影像、3D 深度影像以及聲音訊號。

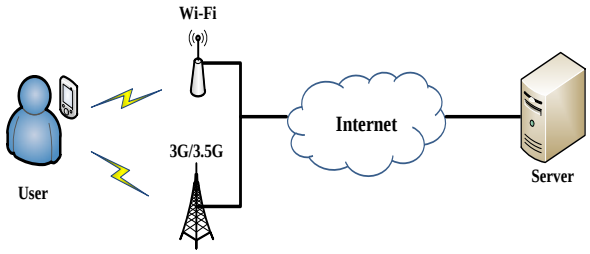
Kinect 感測技術主要是使用 PrimeSensor 技術，透過此技術產生深度影像[10]。而 Kinect 運作可區分為動作識別(Movement Tracking)、語音識別(Voice Recognition)與內置馬達(Motor)。在動作識別部分，Kinect 透過機器中內建的紅外線 VGA 鏡頭發射出紅外線脈衝光，在掃瞄範圍內藉由紅外線 COMS 攝影機接收反射回來的紅外線，並透過 PS1080 感測晶片來分析判斷使用者位置，其感測晶片運作首先對所有掃描到的物件進行 Depth Field 的標示，以不同顏色和使用者與 Kinect 之間距離來分別標示不同物件。即以不同的距離遠近將使用者標示為不同顏色。當標示完成之後，再針對使用者身體部分與背景物件作區隔，並透過圖像辨識系統來正確判斷使用者的姿態，並形成 3D 深度影像。藉由將 3D 深度影像資料轉換成骨架圖，透過骨架追蹤系統來辨識使用者的動作。

綜合上述的智能手套方法與電腦視覺技術方法並不適用於成本與資源受限的設備上。

3. 系統架構與手勢識別演算法設計

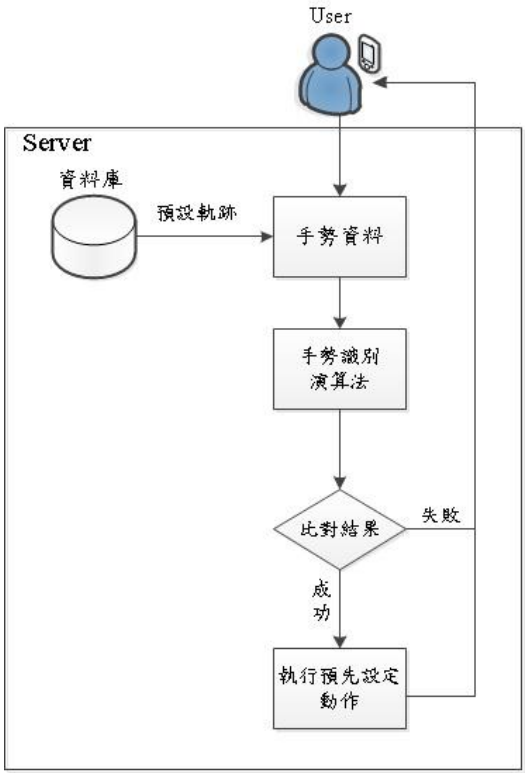
3.1 系統架構

圖一為系統運作環境。使用者身上配有手機，其中手機具備各種感測元件，如光感測元件、重力感應器以及磁力計等元件。另外，在伺服器上具有資料庫以及手勢識別演算法，在資料庫中主要儲存數組手勢動作以及對映的執行動作，而手勢識別演算法的功能是計算使用者傳送的軌跡資料以及手勢識別。當使用者藉由手機上的重力感應器輸入手勢軌跡資料後，透過 Wi-Fi 或 3G/3.5G 網路把輸入之手勢軌跡傳送給伺服器做判斷，伺服器會藉由手勢識別演算法來計算取得識別條件資料，並透過讀取資料庫中預先設定好的數組手勢動作軌跡來判斷手勢，最後手勢識別結果會傳送回使用者顯示於手機上並執行手勢所對映的相關動作。



圖一、系統運作環境

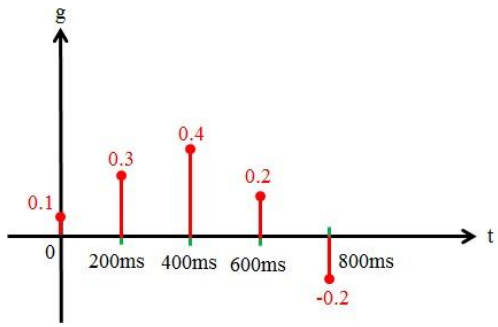
圖二為系統運作流程圖。當使用者傳送手勢軌跡資料給伺服器後，伺服器透過輸入手勢資料與讀取資料庫內的數組軌跡條件並執行手勢識別演算法，以藉此判斷輸入之手勢資料與資料庫內預設的數組軌跡資料是否相符，如果手勢資料比對成功則將資料庫中的手勢動作對映的執行動作回傳給使用者並執行動作。如果比對失敗則直接回傳結果給使用者。



圖二、系統運作流程

3.2 手勢識別演算法

在我們演算法的設計中，首先在擷取手勢資料時設定以每隔 200ms 來收集紀錄軌跡資料。收集完成之後，其記錄的軌跡資料可分為 *x* 軸、*y* 軸與 *z* 軸三部份。針對三軸資料分別做運算處理。假設 *x* 軸加速度為{0.1, 0.3, 0.4, 0.2, -0.2}，如圖三所示。



圖三、x 軸之加速度取樣值

透過

$$\Delta\alpha_n = 0.098/(0.01/a_n - a_{n-1}) \qquad (1)$$

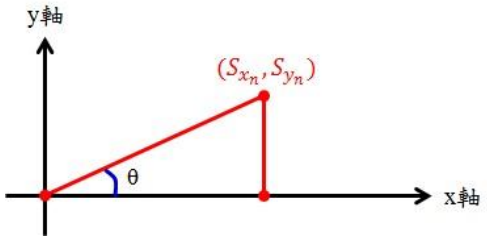
其中$\Delta\alpha_n$為區間加速度變化量，獲得 x 軸在 200ms 時加速度的變化量為：
$\Delta\alpha_1 = 0.098/(0.01/0.3 - 0.1)$

其餘依此類推。獲得加速度變化量後透過：

$$S = \iint_{t_{n-1}}^{t_n} (\Delta\alpha_n * dt)dt \qquad (2)$$

其中S為位移距離。因此可獲得每個時間點位移距離S。另外，其 y 軸與 z 軸依照上述所使用公式(1)與公式(2)可獲得每個時間點位移距離S。

當 x 軸、y 軸與 z 軸位移距離經計算獲得後，x 軸和 y 軸平面所計算的距離透過三角函數運算可獲得其夾角角度θ，如圖四所示。



圖四、三角函數之夾角θ

圖四中，S_{x_n}為 x 軸位移距離，S_{y_n}為 y 軸位移距離，θ為夾角角度。同理，y 軸與 z 軸平面、x 軸與 z 軸平面也可透過三角函數算出其夾角角度θ。

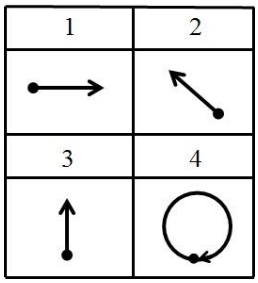
經過上述計算後，將上述 x 軸、y 軸以及 z 軸中所有時間點位移的距離S，相同時間點上 x 軸與 y 軸平面、y 軸與 z 軸平面、x 軸與 z 軸平面的夾角角度θ等資料儲存於資料庫中，做為判斷相同軌跡的條件使用。

4. 實驗結果與分析

4.1 實驗結果

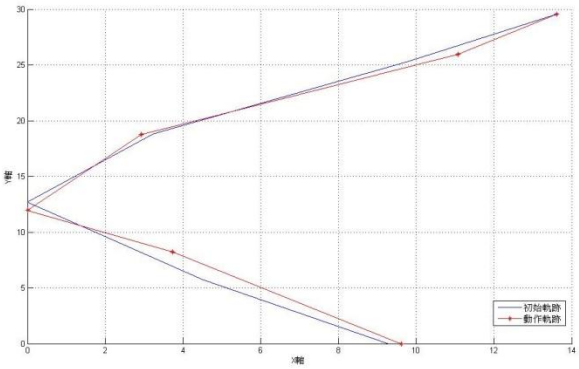
在實驗中，使用擷取手勢軌跡資料的設備為 HTC Desire，其作業系統為 Android 2.2，手機內建的重力感應器晶片使用 BMA150，對於 HTC 加速度擷取的值為正負 9.8g 之間。其中 $1g = 9.8\,{}^{m}/_{s^2}$。伺服器主機為 DELL-Optiplex 745MT，CPU 配備為 Intel Core(TM)2 Qual Q8400，記憶體為 4GB。

圖五為實驗所擷取的手勢動作，透過圖中四種手勢動作來擷取手勢資料並重繪手勢軌跡圖。其中，1 為向右移動，2 為右下往左上，3 為往上移動，4 為由左為起點右為結束點畫圓。

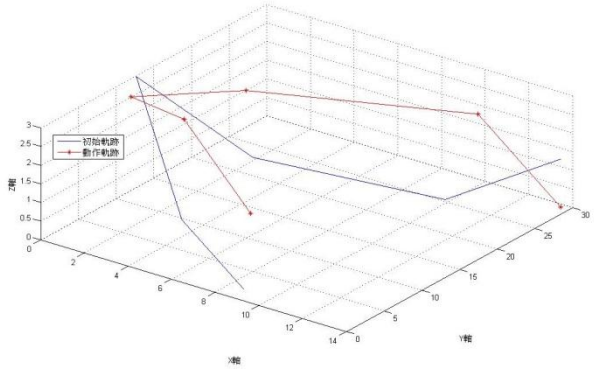


圖五、手勢動作(圓點為開始，箭頭為結束)

使用手勢動作 1 所擷取計算比對後重繪之雙軸軌跡圖，如圖六所示。圖六發現在 x 軸與 y 軸平面上之軌跡線條具有些許誤差。原因為每次擷取軌跡資料時，受到些許晃動或動作些微誤差而導致記錄的手勢資料具有些許誤差。另外，以三軸重繪後之軌跡圖，發現動作軌跡與初始軌跡的 z 軸值差異較大，如圖七所示。原因為在使用手機擷取手勢資料時，其擷取的加速度值會受到地心引力的重力影響，而影響到記錄的軌跡資料。因此，手勢動作 1 之軌跡資料中 z 軸受到地心引力的重力影響導致軌跡些許誤差。

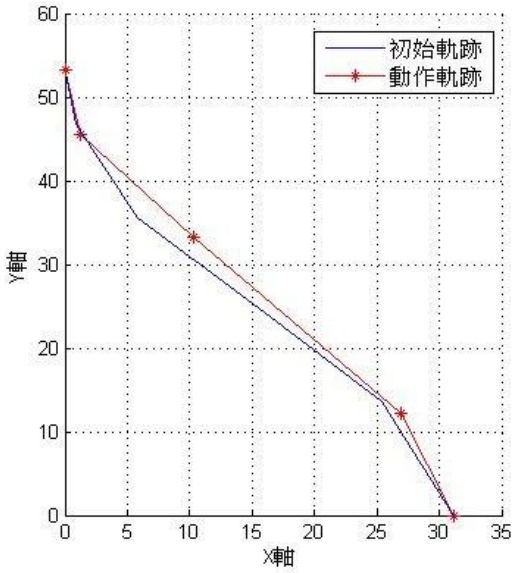


圖六、手勢動作 1 之(x,y)軸軌跡

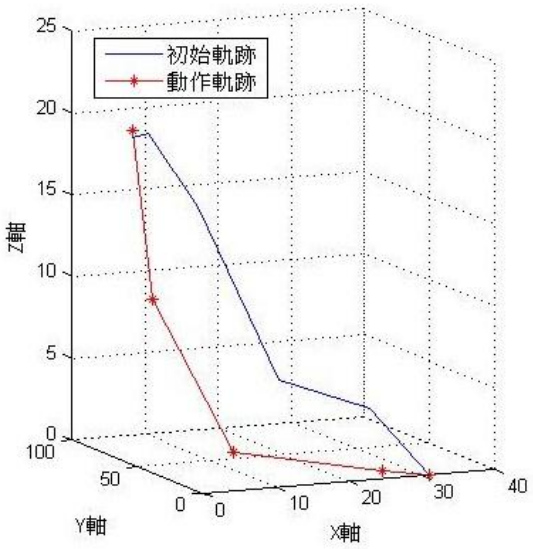


圖七、手勢動作 1 之(x,y,z)軸軌跡

圖八為使用手勢動作 2 所擷取計算比對後之雙軸軌跡圖，重繪之結果顯示因擷取手勢資料時受到晃動或動作些微誤差而有些許誤差。另外，透過三軸重繪之軌跡圖，發現動作軌跡與初始軌跡所繪之軌跡線誤差較大，如圖九所示。因 z 軸受到嚴重地心引力的重力影響導致軌跡誤差提升。

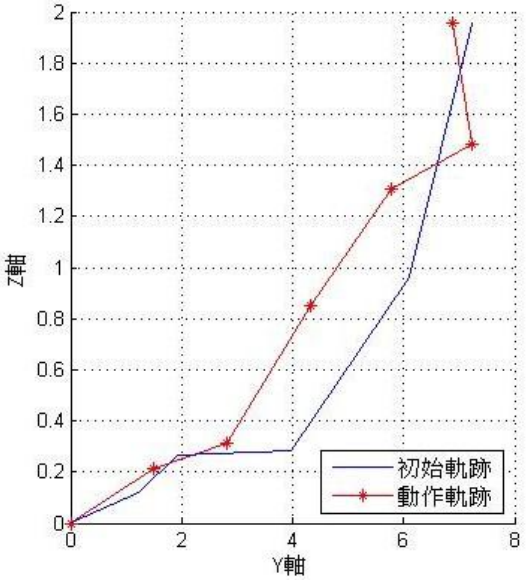


圖八、手勢動作 2 之(x,y)軸軌跡

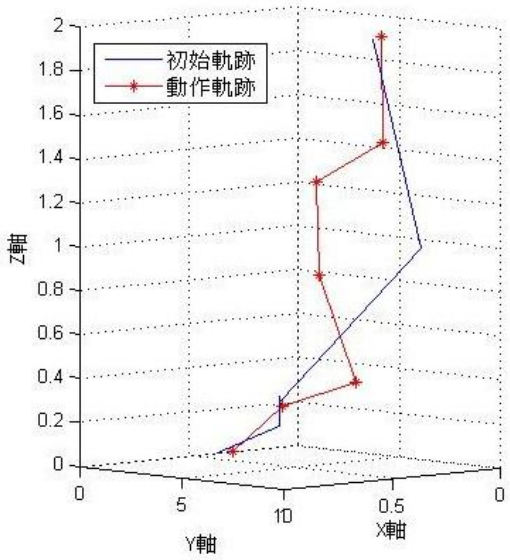


圖九、手勢動作 2 之(x,y,z)軸軌跡

使用手勢動作 3 所擷取計算以及重繪後雙軸軌跡圖，如圖十所示。透過圖十可發現動作軌跡與初始軌跡具有誤差，原因為使用之手勢動作為由下至上移動，因此，在擷取手勢資料時 *x* 軸、*y* 軸以及 z 軸受到嚴重地心引力的重力影響而導致軌跡資料誤差提升。另外，透過三軸重繪之軌跡圖發現動作軌跡與初始軌跡所繪之軌跡線條不同，如圖十一所示。因為 *x* 軸、*y* 軸以及 z 軸受到嚴重地心引力的重力影響導致軌跡誤差提升。

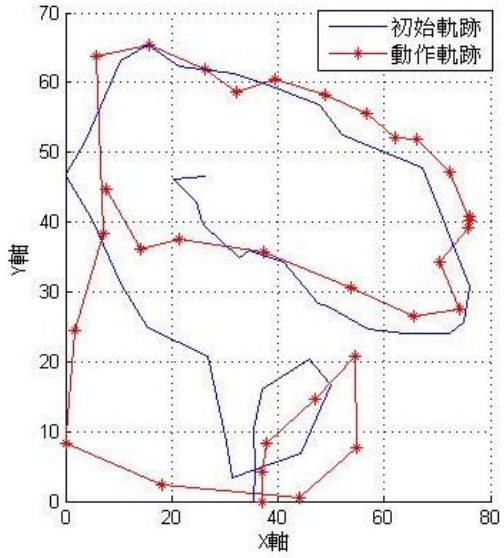


圖十、手勢動作 3 之(x, y)軸軌跡

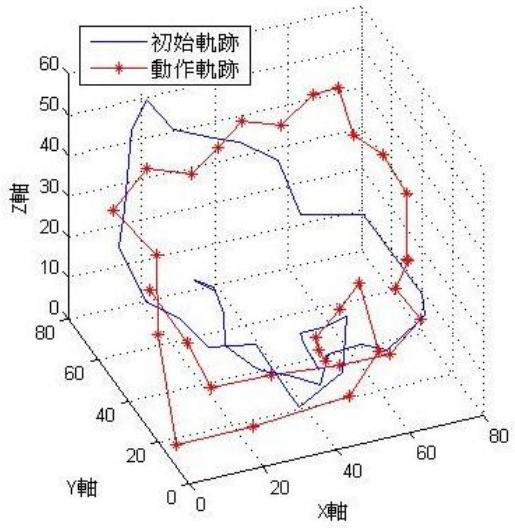


圖十一、手勢動作 3 之(x, y, z)軸軌跡

圖十二與圖十三為使用手勢動作 4 所擷取計算重繪後之軌跡圖。藉由這兩張圖可發現動作軌跡與初始軌跡誤差較高，誤差較高的原因為使用之手勢動作具有由下至上移動，因此在擷取手勢資料時 x 軸、y 軸以及 z 軸受到嚴重地心引力的重力影響而導致軌跡資料誤差提升。



圖十二、手勢動作 4 之(x, y)軸軌跡



圖十三、手勢動作 4 之(x, y, z)軸軌跡

4.2 實驗分析

透過實驗結果得知，當使用 HTC Desire 手機擷取手勢軌跡資料時，其擷取的 x、y、z 軸三軸的加速度值會嚴重受到地心引力的重力影響而導致軌跡資料誤差提升。原因為 Android 2.2 版本作業系統中，其作業系統所提供針對擷取三軸加速規值的 API，主要功能為單純取得硬體設備中重力感應器的加速度值，其所取得的加速度值包含地心引力的重力值與手勢移動的加速度值。因此，實驗中所使用之手勢動作，其中 2、3 與 4 的手勢軌跡資料會受到很大的影響導致誤差提升。

表一為每個手勢經過五次測試後，經過公式(3)分析的平均誤差率。透過表一可發現除手勢動作 1 所擷取的手勢軌跡資料誤差率為 11.35%，而其他剩餘的手勢的誤差率偏高，原因為在擷取手勢資料時因受到嚴重地心引力重力影響的關係，導致所擷取的 x、y、z 軸的加速度值嚴重受到影響。因此，對於手勢識別上會造成失敗率偏高。

$$Error_{avg} = \left\| 1 - \left(S_{new-total} / S_{pre-total} \right) \right\| \quad (3)$$

其中 $Error_{avg}$ 為平均誤差率，$S_{pre-total}$ 為初始軌跡之位移距離總和，$S_{new-total}$ 為動作軌跡之位移距離總和。

表一、平均誤差率

手勢動作	1	2	3	4
平均誤差率	12.35%	57.55%	107%	45.49%

另外，Google 於 2010 年 12 月 7 日發表 Android 2.3 作業系統，此版本系統中除了修正 UI 與新增支援硬體設備外[11]，針對擷取重力感應器的值提供了新的 API[12]，此 API 主要的功能與 Android 2.2 版本類似。但是，API 最大的差別在於擷取的加速度值不包含地心引力的重力。因此，我們所設計的系統未來移植至 Android 2.3 系統將可提升擷取手勢軌跡資料精確度。

5. 結論

本篇主要提出一種新的直覺式手勢識別系統，此系統中主要經由我們設計的手勢識別演算法來實現直覺式手勢識別。然而，在實作過程中遇到了一項挑戰，挑戰為在手機上所擷取的加速度值會受到地心引力的重力影響而導致誤差率提高。

因此，未來工作中主要是將目前設計的直覺式手勢識別系統移植至 Android 2.3 版，藉以解決上述提到的挑戰以及提高識別成功率。

參考文獻

[1] Y. Wu and T. S. Huang, "Vision-Based Gesture Recognition: A Review," ***in: Proceedings of the International Gesture Workshop on Gesture-Based Communication in Human-Computer Interaction***: SpringerVerlag, 1999.

[2] J. K. Perng, B. Fisher, S. Hollar, K. S. J. Pister, "Acceleration sensing glove (ASG)," ***The Third International Symposium on Wearable Computers***, 1999.

[3] C.-W. Yi, C.-M. Su, W.-T. Chai, J.-L. Huang, T.-C. Chiang, "G-Constellations: G-Sensor Motion Tracking Systems," ***IEEE 71st Vehicular Technology Conference (VTC 2010-Spring)***, 2010.

[4] J. Liu, Z. Wang, and L. Zhong, J. Wickramasuriya and V. Vasudevan, "uWave: Accelerometer-based Personalized Gesture Recognition and Its Applications," ***IEEE International Conference on Pervasive Computing and Communications***, 2009.

[5] J. Liu, L. Zhong, J. Wickramasuriya and V. Vasudevan, "uWave: Accelerometer-based personalized gesture recognition and its applications," ***Pervasive and Mobile Computing***, Volume 5, Issue 6, December 2009, Pages 657-675.

[6] T. Petric, A. Gams, A. Ude, L. Zlajpah, "Real-time 3D marker tracking with a WIIMOTE stereo vision system: Application to robotic throwing," ***IEEE 19th International Workshop on Robotics in Alpe-Adria-Danube Region (RAAD)***, 2010.

[7] P.-W. Chen, K.-S. Ou, K.-S. Chen, "IR Indoor Localization and Wireless Transmission for Motion Control in Smart Building Applications based on Wiimote Technology," ***In Proceedings of SICE Annual Conference***, Taiwan, 2010.

[8] C. S. Myers, L. R. Rabiner, "A cpmparative study of several dynamic time-warping algorithms for connected word recognition." ***The Bell System Technical Journal***, vol. 60, pp. 1389-1409, 1981.

[9] Kinect for Xbox 360, http://www.microsoft.com/presspass/presskits/

[10] PrimeSense Supplies 3-D-Sensing Technology to "Project Natal" for Xbox 360, http://www.microsoft.com/presspass/press/2010/mar10/03-31primesensepr.mspx

[11] 維基百科, Android http://zh.wikipedia.org/zh-tw/Android#.E4.BD.9C.E6.A5.AD.E7.B3.BB.E7.B5.B1

[12] Android Developers, http://developer.android.com/reference/android/hardware/SensorEvent.html