第十五屆旺宏科學獎

成果報告書

參賽編號：SA15-593

作品名稱：筆跡晃動變形修正系統

姓名：蘇柏穎

關鍵字：防手震、移動估計、人機互動

摘　要

在過去的五年中，行動裝置之手寫輸入法、滑行輸入法等基於筆跡的輸入系統有巨大進展。但仍無法解決物理晃動造成之筆跡變形。為了解決人們在不穩定的環境中書寫筆跡的困難，本研究提出了“筆跡變形修正系統”。 通過結合內置感應器的數據，系統估計裝置的運動軌跡，並提出的屏幕-手指互動模型恢復失真的筆跡。 該系統嘗試了不同的移動估計算法，也套用濾波器以去除雜訊和去除漂移。 為了得到更精確的感應器讀值，本研究使用了一種不需要額外儀器的校準方法。 翻轉後靜置設備 6 次，校準演算法將計算感應器誤差模型的最適合參數值並且套用。本研究提出之筆跡失真修正系統在三代的演進後，已成功在現實世界中測試證明有效性和實用性。

Abstract

Recently, input methods based on handwriting like traditional handwriting recognition system or Swype has made great progress. But drawing distortion problem due to device shaking is still unresolved. To solve this common issue, the study developed “Drawing Distortion Recovery System”. By combining data from phone sensors, the system estimates mobile motion in space and recover distorted handwriting using the proposed screen-finger motion model. To calibrate sensors, the system collects sensor values of the device in six poses and calculate calibration constants. The project also tried different approaches to estimate motion. After three generations of evolution, the system can successfully recover distorted strokes and be able to handle different situations. The system was proved to be effective and useful under daily use.

目　錄

壹、 研究動機

隨著智慧型手機市場的發展，行動裝置的普及率已經迅速增長。手機從以前單純的語音通訊、文字訊息的功能，演變至今除了能繪畫、作筆記，甚至還能夠進行手寫辨識。人機互動關係也愈趨多樣化；從按鈕到滑動方塊再到現在流行的手勢輸入，說明人們操作行動裝置的演進與複雜化，同時也帶動手寫功能的發展，且新的裝置也紛紛加入高精細的繪圖功能。

根據 T.M.T. Do 等人於 2011 年所做的調查[1]，在(表一)中的黃色區域顯示，人們搭乘交通工具時有大量的時間使用智慧型手機傳簡訊、上網或發電子郵件。而在這種不穩定(晃動)的環境中(如公車上)，人們在使用基於筆跡的輸入法時，常因晃動導致輸入筆跡變形而無法辨識。

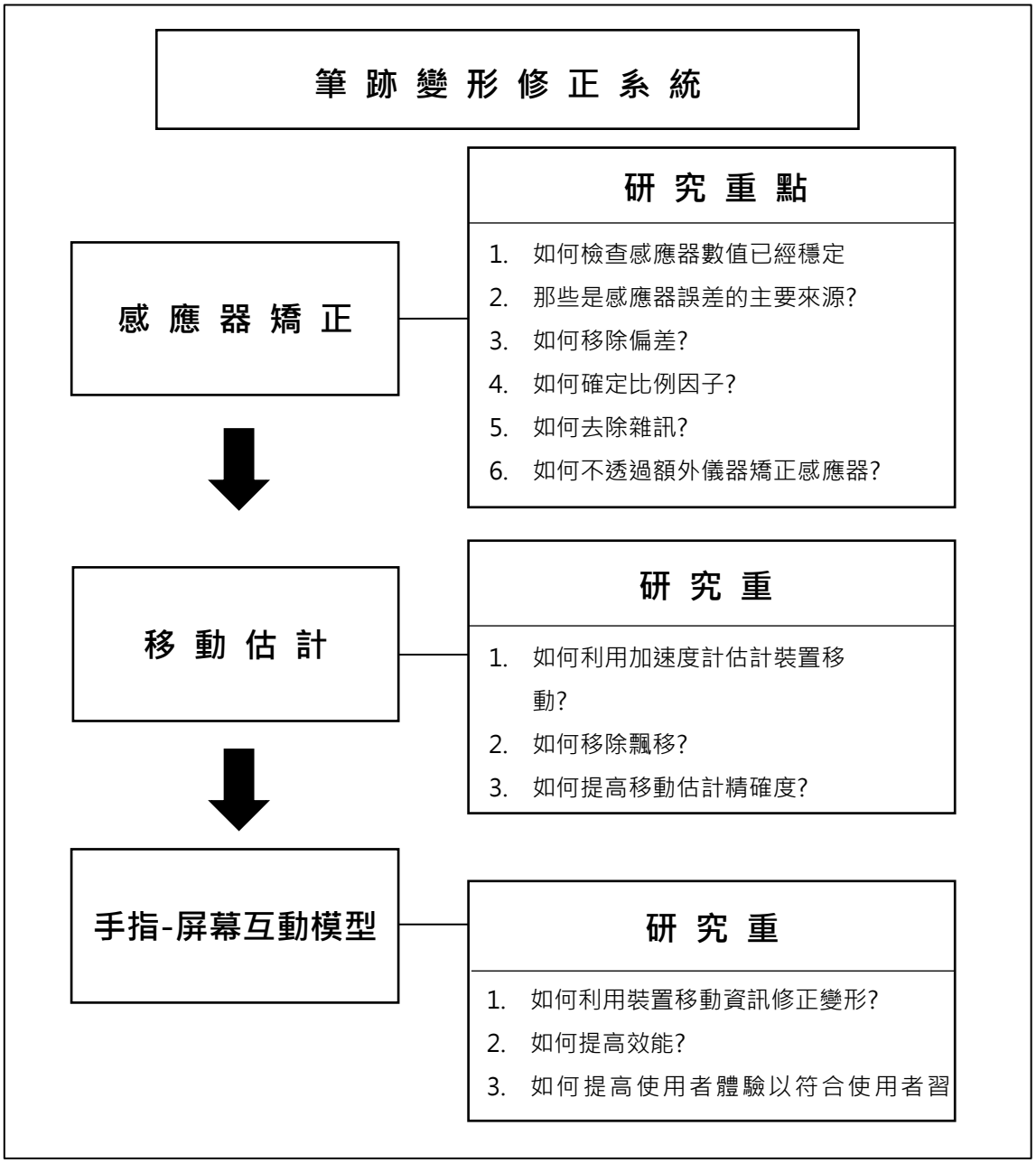
(表一) 收集時間內各地方使用軟體的頻率 (單位: 平均事件數/小時)

地點 / 軟體	家住	公司	朋友家	朋友公司	廳餐	動運	交通工具	日假	逛街購物	閒休	他其	值平均
簡訊	0.41	0.51	0.63	0.72	0.52	0.93	0.33	0.09	0.21	0.15	0.75	0.48
語音通訊	0.18	0.29	0.34	0.30	0.27	0.26	0.57	0.14	0.33	0.51	0.40	0.24
網路	0.05	0.09	0.30	0.02	0.19	0.09	0.52	0.43	0.22	0.29	0.16	0.08
多媒體	0.06	0.07	0.06	0.05	0.02	0.03	0.17	0.22	0.08	0.29	0.09	0.07
時鐘	0.07	0.01	0.06	0.02	0.01	0.01	0.07	0.01	0.04	0.00	0.02	0.05
相機	0.02	0.02	0.03	0.05	0.03	0.03	0.07	0.81	0.09	0.00	0.09	0.03
電子郵件	0.09	0.11	0.17	0.00	0.10	0.32	0.07	0.06	0.00	0.00	0.20	0.03
行事曆	0.02	0.04	0.04	0.04	0.05	0.07	0.15	0.00	0.03	0.00	0.11	0.04
語音聊天	0.06	0.05	0.05	0.04	0.00	0.23	0.05	0.35	0.06	0.00	0.05	0.06
地圖	0.01	0.01	0.01	0.01	0.06	0.03	0.04	0.07	0.01	0.07	0.05	0.01
運動追蹤器	0.03	0.04	0.04	0.00	0.00	0.10	0.04	0.00	0.00	0.00	0.05	0.03
多媒體串流	0.06	0.05	0.00	0.00	0.00	0.04	0.01	0.00	0.00	0.00	0.05	0.05

為了解決這種令人擾人的問題，與提高行動裝置的整體使用體驗，本研究提出全新的演算法和解決方案，以解決裝置晃動造成的不良影響。

貳、 研究目的

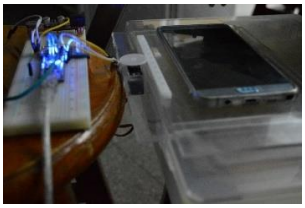
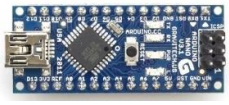
本研究之目的為提出一個全新的演算法修正因裝置晃動導致的筆跡變形系統，而本研究之系統主要利用常見於智慧型裝置的內建加速度計、指北針與陀螺儀，為了取得更好的效果，感應器必須在事前做矯正，這個矯正的流程必須有效、快速且不需要額外器材，接著使用一個精準、穩定的移動估計方法估計裝置的移動狀況。最後，提出手指與螢幕的互動模型，並且應用於修正因為晃動變形的筆跡，以下(圖一)為筆跡變形修正系統研究流程與問題介紹。



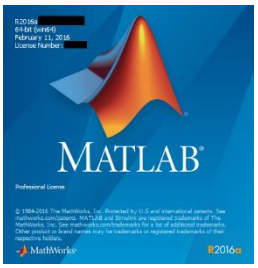
(圖一) 筆跡變形修正系統流程圖

參、 研究設備及器材

(表二) 研究設備及器材一覽

裝　　置	實體照片	規　　格
Microsoft Surface Pro 3		CPU:第 4 代 Intel Core i5-4300U 1.90 GHz, 記憶體: 8GB RAM DDR3-1600, SSD:256GB
Samsung Galaxy Note 5		螢幕: 2560 X 1440 QHD Super Amoled, 處理器: Exynos 7420 2.1GHz, 記憶體 4g LP-DDR4
Periodic Motion Generator		週期運動產生器由一個可移動的平台、Arduino、伺服馬達和一些齒輪所構成。上方放置的手機 被固定在一個平台上。這個移動平台被限制在一軸上移動。平台已經過水平矯正以確保平台不會像任何一個方向傾斜。至於移動的方式，如頻率、震幅等...在實驗時設定。
Arduino Nano v3.0		微控制器: Atmel ATmega328

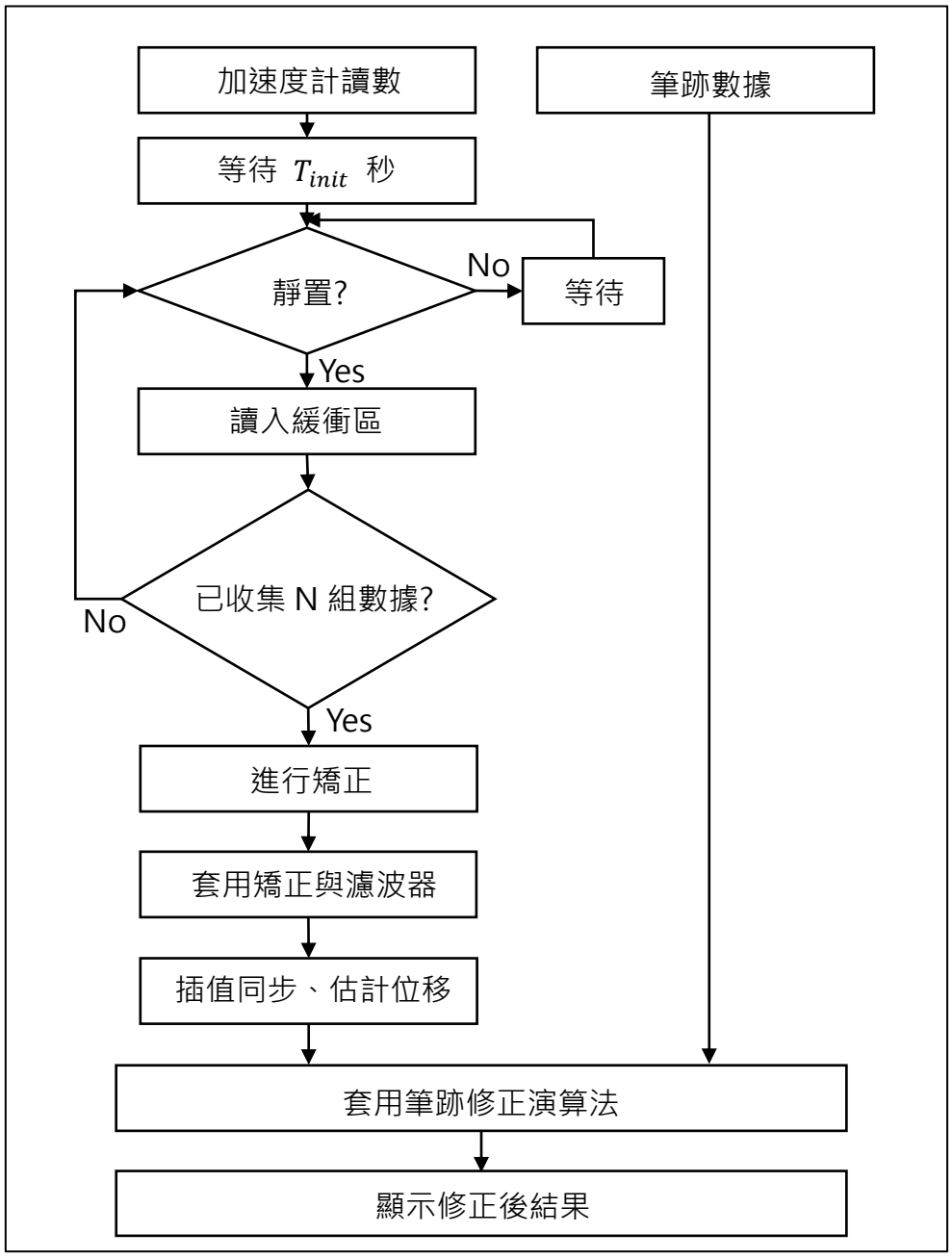
(表三) 研究軟體一覽

軟　　體	圖　　片	版　　本
Matlab		R2016a
Android Studio		v2.1
Arduino Software (IDE)		V2.0

肆、 研究過程或方法

一、 筆跡變形修正系統運作流程圖:

(圖二)為本研究提出之筆跡變形修正系統運作流程；在初始化階段，系統等待T_{init}秒，接著系統會提示使用者翻轉裝置 N 次以收集足夠的感應器矯正數據。Levenberg-Marquardt 演算法在此被用來得到感應器誤差模型正確的參數數值，並且經過感應器校正後，本系統嘗試使用三種不同的移動估計演算法: Euler integration, RK4 integration 和 Spring-mass-damper model 利用加速度計數據估計裝置移動。最後，變形得筆跡透過筆跡修正演算法得到修正。



(圖二) 筆跡變形修正系統

二、 感應器矯正:

(一) 感應器誤差模型:

在感應器誤差模型中，加速度計讀數的雜訊也需要被納入考慮。 因此，完整的感應器誤差模型寫成:

$$a^o = K^a(a^s + b^a + v^a) \quad \text{(式 1)}$$

a^o : 3x1 模型中準確的加速度數值，K^a: 3x3 比例因子矩陣，a^s : 3x1 測得的讀數矩陣，b^a : 3x1 偏差常矩陣，v^a : 3x1 感應器雜訊矩陣

(二) 定義加速度計能量函數:

1. 誤差模型中，有 6 個參數需要被確定。參數向量可以表示為:

$$\lambda^a = \left[s_x^a, s_y^a, s_z^a, b_x^a, b_y^a, b_z^a\right] \quad \text{(式 2)}$$

2. 實際值可通過以下公式計算:

$$a^o = h(a^s, \lambda^a) = K^a(a^s + b^a) \quad \text{(式 3)}$$

3. 這裡使用了一個定理: 靜態時加速度計所測量的加速度值的大小必須等於重力。(式 4)中所定義的能量函數表示重力的大小$\| g \|$和誤差模型輸出的大小的差。

$$L(\lambda^a) = \sum_{k=1}^{N}(\| g \|^2 - \| h(a_k^s, \lambda^a) \|^2)^2 \quad \text{(式 4)}$$

Algorithm 1 感應器校正

Require: $\mathbf{T_{init}}$(wait before start acquiring data),$\boldsymbol{a^s}$

datasetNum (number of cycles to collect data)

1: Wait for $\mathbf{T_{init}}$ seconds

2: **for i** = 0: **datasetNum**

3: **if static == true**

4: $\mathbf{dataset_i}$ ← average of array of a^s

7: **end**

8: **end**

9: **Params** ← optimize using dataset and Levenberg − Marquardt algorithm

10: $\boldsymbol{a^0}$ ← calibrate $\mathbf{a^s}$ using **Params**

三、 Euler Integration:

Euler Integration 是從加速度計算速度和位置的最簡單的方法。

(一) 下面為 Euler Integration 運作流程:

1. 狀物體的運動由(式 5)與顯示:

$$\frac{dx}{dt} = v \quad \text{(式 6)}$$

$$\frac{dv}{dt} = a \quad \text{(式 7)}$$

2. 已知的初始位置和速度:

$$\mathrm{x}(0) = x_0 \qquad \mathrm{v}(0) = v_0 \quad \text{(式 8)}$$

3. 在本系統中，每當使用者下筆$\mathrm{x}(0)$就會被重置，當裝置被偵測靜置時，積分就會以初速度$\mathrm{v}(0) = 0$開始。

4. 當$\Delta\mathrm{t}$ 足夠小時(實際上為感應器採樣間隔):

$$\mathrm{dt} \approx \Delta\mathrm{t} \quad \text{(式 9)}$$

5. 位置對時間的微分是速度

$$\frac{dx}{dt} \approx \frac{x(t+\Delta\mathrm{t})-x(t)}{\Delta\mathrm{t}} \quad \text{(式 10)}$$

6. 速度對時間的微分是加速度:

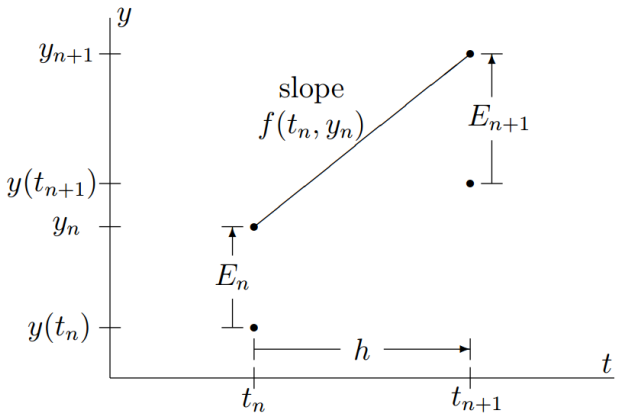
$$\frac{dv}{dt} \approx \frac{v(t+\Delta\mathrm{t})-v(t)}{\Delta\mathrm{t}} \quad \text{(式 11)}$$

7. 由下列兩式從加速度計算速度與位置:

$$\mathrm{x}(t+\Delta\mathrm{t}) \approx \mathrm{x}(\mathrm{t}) + v(t)\Delta\mathrm{t} \quad \text{(式 12)}$$

$$\mathrm{v}(t+\Delta\mathrm{t}) \approx \mathrm{v}(\mathrm{t}) + a(t)\Delta\mathrm{t} \quad \text{(式 13)}$$

(二) Euler Integration 的局部截尾誤差(Local truncation error, LTE):



(圖三)一次歐拉積分過程

1. 首先，下式給出由 Euler Integration 於y_0估計y_1之值:

$$y_1 = y_0 + \mathrm{hf}(t_0, y_0) \quad \text{(式 14)}$$

其中 f 為此位置函數的導數，$\mathrm{h} = t_{n+1} - t_n$

2. 設$y_1 = y(t_0 + h)$，利用泰勒級數展開y_1點得出精確解:

$$\mathrm{y}(t_0 + \mathrm{h}) = y(t_0) + hy'(t_0) + \frac{1}{2}h^2y''(t_0) + O(h^3) \quad \text{(式 15)}$$

3. 這時將精確值$\mathrm{y}(t_0 + \mathrm{h})$減去 Euler Method 的估計值$y_0$得到局部截尾誤差

$$\mathrm{LTE} = \mathrm{y}(t_0 + \mathrm{h}) - y_1 = \frac{1}{2}h^2y''(t_0) + O(h^3) \quad \text{(式 16)}$$

4. 由上式可知，當步長 h 很小時，Euler Integration 的局部截尾誤差與h^2成比例。

Algorithm 2 Euler Integration

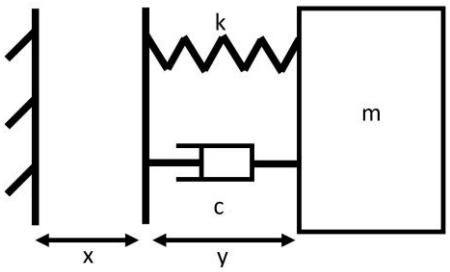
Require: $\mathbf{v_p}$ (previous velocity), $\mathbf{p_p}$ (previous position), $\boldsymbol{a}$ (measured acceleration), $\boldsymbol{\alpha}$ (high pass filter coefficient), dt (time elapsed since previous measurement), SD (static detector status)

1: $\mathbf{v_p} += \boldsymbol{a} * \mathbf{dt}$
2: $\mathbf{p_p} += \mathbf{v_p} * \mathbf{dt}$
3: **if** SD == true
4: set $\mathbf{v_p}$ and $\boldsymbol{a}$ zero
5: **end**

四、 Mass-Spring-Damper Model:

Mass-Spring-Damper Model 是個物理模型，其靈感來自 Mass-Spring-Damper System。這樣的系統與屏幕在手上搖晃時的行為非常相似[5]，此模型將屏幕看成一個透過彈簧與阻尼連接滯留

空中的物體，透過加速度計模擬器統受力，其中與屏幕平行的兩軸的彈簧、阻尼是相互獨立的，(圖四)顯示系統在一維的情況。



(圖四) 一維 Spring-mass-damper physical model

(一) 系統的行為依賴以下阻尼比:

$$\zeta = \frac{c}{2\sqrt{km}} \quad \text{(式 17)}$$

(二) 當 $\zeta < 1$，系統呈現欠阻尼狀態並不斷來回震盪；當$\zeta > 1$，系統呈現過阻尼狀態，這會使系統花費過多時間回到平衡狀態；當設$\zeta = 1$，系統呈現臨界阻尼狀態，是最佳的情況。而$c = 2\sqrt{km},\ \zeta = 1$。臨界阻尼系統方程式可表示為:

$$\ddot{y} + 2\sqrt{km}\dot{y} + ky = -A(t) \quad \text{(式 18)}$$

(三) 透過將$-A(t)$與系統沖激響應函數$H(t)$計算卷積可以得到位移:

$$Y(t) = H(t) * -A(t) \quad \text{(式 19)}$$

(四) 臨界阻尼狀態的 Mass-Spring-Damper System 的沖激響應函數為:

$$H(t) = te^{-t\sqrt{k}} \quad \text{(式 20)}$$

Algorithm 3 Mass-Spring-Damper Model

Require: mBuffer(measured acceleration array in circular buffer), SD (static detector status), CONST_K (constant), bufferSize, sampleDelay(time elapsed since previous measurement)

```
for i = 0: bufferSize
    t ← (1 − i / bufferSize) * sampleDelay;
    impulseResponse ← t e^(−t√CONST_K)
    sum+= (mBuffer[i] * impulseResponse); //convolve buffer with impulseResponse
end
return sum
```

五、 Runge-Kutta Integration:

一階 Euler Integration 在加速度二次積分後會產生較大的偏差，相較之下 Runge-Kutta Integration 有著高階泰勒級數的精確度，而其中經典四階 Runge-Kutta Integration 是最常用的。

(一) Runge-Kutta Integration 需要下列四個方程式:

$$y_{n+1} = y_n + \frac{\Delta t}{6}(k_1 + 2k_2 + 2k_3 + k_4) \quad \text{(式 21)}$$

其中

$$\frac{dy}{dt} = f \quad \text{(式 22)}$$

$$k_1 = f(t_n, y_n) \quad \text{(式 23)}$$

$$k_2 = f(t_n + \frac{\Delta t}{2}, y_n + \frac{\Delta t}{2}k_1) \quad \text{(式 24)}$$

$$k_3 = f(t_n + \frac{\Delta t}{2}, y_n + \frac{\Delta t}{2}k_2) \quad \text{(式 25)}$$

$$k_4 = f(t_n + \frac{\Delta t}{2}, y_n + \frac{\Delta t}{2}k_3) \quad \text{(式 26)}$$

y可以表示為位置或是速度

(二) RK4 方法的局部截尾誤差(Local truncation error, LTE)為$O(h^5)$，當h足夠小時，RK4 比 Euler method 有更小的 LTE。

Algorithm 4 RK4 積分

Require: $\mathbf{v_p}$ (previous velocity), $\mathbf{p_p}$ (previous position), $\boldsymbol{a}$ (measured acceleration),dt (time elapsed since previous measurement), SD (static detector status)

1: //RK4 divides the integration process into 4 steps，4 positions.

2: V_1(velocity at position X_1) ← $\mathbf{v_p} + \mathbf{0} * \mathbf{0}$ **//the 1st point at t = 0**

3: V_2(velocity at position X_2) ← $\mathbf{v_p} + V_1 * \mathbf{dt} * \mathbf{0.5}$ **//the 2nd point at t = 0.5 * dt**

4: V_3(velocity at position X_3) ← $\mathbf{v_p} + V_2 * \mathbf{dt} * \mathbf{0.5}$ **//the 3st point at t = 0.5 * dt**

5: V_2(velocity at position X_2) ← $\mathbf{v_p} + V_3 * \mathbf{dt}$ **//the 4th point at t = dt**

6: $V_{all} \leftarrow \frac{1}{6}(V_1 + 2V_2 + 2V_3 + V_4)$

7: $\mathbf{p_p}$+= V_{all} * dt;

8: $\mathbf{v_p}$+= **a** * dt;

9: **return** $\mathbf{p_p}$

六、 基於 Exponential smoothing 的濾波器:

(一) 低通濾波器:

低通濾波器為使用一階指數平滑公式達到保留低頻信號的效果,(式 27)是一階指數平滑化公式:

$$S_t = \alpha \cdot X_t + (1-\alpha) \cdot X_{t-1} \quad \text{(式 28)}$$

(二) 高通濾波器:

可將原始信號減去低頻部分便得號高通濾波的效果:

$$S_t = X_t - (\alpha \cdot X_t + (1-\alpha) \cdot X_{t-1}) \quad \text{(式 29)}$$

七、 靜置感測器:

此感測器用於校準感應器誤差的數據將影響感應器矯正的準確度；因此，區分裝置是否處於靜態或是動態非常重要，給定時間間隔 t_w，在時間 t 的變異幅度透過以下計算:

$$\text{變異幅度} = \zeta(t) = \sqrt{\left[VAR_{t_w}(a_x^t)\right]^2 + \left[VAR_{t_w}(a_y^t)\right]^2 + \left[VAR_{t_w}(a_y^t)\right]^2} \quad \text{(式 30)}$$

通過檢查$\zeta(t)$比閾值高或低可以決定靜置感應器是否被觸發。

八、 感應器時間戳同步:

裝置中每一個感應器的更新時間並不同步，更新頻率也各不相同；為了使系統運算時各感應器的時間戳同步，本系統利用了共同環形緩衝區(Ring Buffer)及樣條插植法。

(一) 共同環形緩衝區:

設一共同緩衝區，所有感應器傳入的資料皆載入環形緩衝區中；其中 N 代表觸控感應器於緩衝區的資料陣列，新數據進入緩衝區的同時會移除舊數據以維持緩衝區大小不變，共同緩重區除了使數據抽取更加方便，也更容易的進行插值或設定濾波器觀察窗。

(二) 樣條插植:

在進行移動估計與套用筆跡修正演算法時需要進行各感應器數據時間戳的同步，這個同步的過程需要利用共同環形緩衝區與樣條插值方法實現，而此系統在時間戳同步採用的插值法為三次樣條插值(Cubic Spline Interpolation)，因此不會發生抖動(龍格現象)。

九、 異常偵測:

本研究所開發的系統聚焦於裝置小範圍移動的一般狀況，因此將部分特殊狀態視為異常。以下情況被視為異常:

(一) 加速度	>	4m/s^2
(二) 速度	>	1m/s
(三) 位移量	>	4cm
(四) 加速度	同號	超過 1 秒

十、 手指-屏幕互動模型:

(圖五) 本研究提出的手指-屏幕互動模型

顯示本研究提出的手指-屏幕互動模型包含四個部分:

(一) 繪畫層:

繪畫層是固定於世界座標上的一張畫布，人們意圖在這張畫布上寫下筆跡，這張畫布在空間中的位置及角度接再下筆的一瞬間被確定，並且在同一筆畫中不會有任何改變。

(二) 感應器層:

作為連結繪畫層及觸控層之間的橋樑，感應器層利用裝置內建的加速度計、陀螺儀以及指南針計算感應層平面與世界座標之間的相對位置與角度，在現代普遍的智慧型裝置內置的感應器使用微機電(MEMS)原理，屬於 strapdown system，因此感應器層與觸控層始終保持平行一體。

(三) 觸控層:

觸控層是本系統與手指直接物理接觸的媒介，用來記錄包括觸控位置及壓力大小的資訊，此層與感應器層終保持平行一體。

(四) 手指:

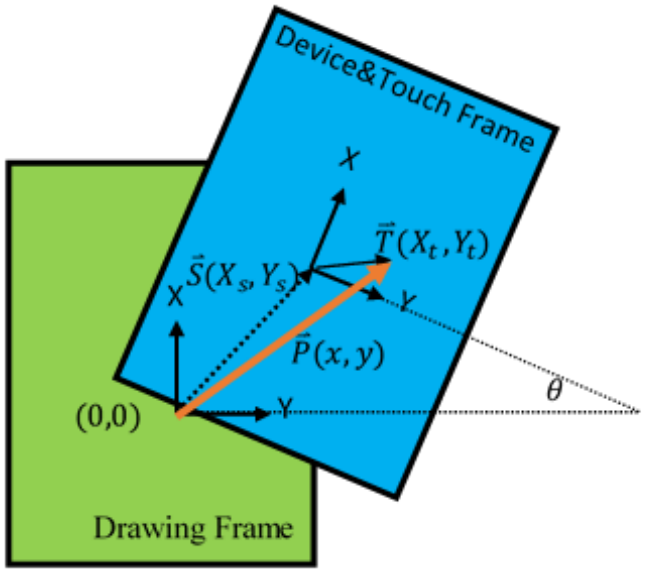
使用者的手指進行手寫，意圖在繪畫層上寫下理想的筆跡，但是在特別惡劣的不穩定環境下，手指本身也會產生使用者無法控制的運動。

十一、 筆跡修正演算法(第一代)

第一代的手指-屏幕互動模型中，所有的層都相互平行於地面。

(一) 第一代考慮的手指-屏幕互動模型:

1. 繪畫層: 使用者意圖繪畫的層，平行於地表，始終固定。
2. 感應器層: 裝有加速度計、陀螺儀及指南針。平行於地表，隨裝置平移旋轉但局限於 2 維平面上。
3. 觸控層: 用來取得觸控感應點數據，始終與感應器層相連。
4. 手指: 第一代筆跡修正演算法不考慮使用者意料之外的手指運動。



(圖六) 第一代手指-屏幕互動模型

(二) 將觸控層映射到繪畫層:

1. 下筆當下所有層坐標軸平行共原點。設定一共用原點$(0,0)$ 。
2. 設觸控層相對於原點移動(X_s, Y_s) ，設 $\vec{S}$ 為觸控層原點於繪畫層的位置向量
3. 設觸控層上一觸控點(X_t, Y_t)，其在觸控層上之位置向量為$\vec{T}$
4. 設觸控層相對繪畫層以螢幕中心點逆時針(+)旋轉θ

5. 假設從原點至下一採樣點之間，感應器層與觸控層相對繪畫層進行了一個含有移動與旋轉的線性變換。以下為了方便，使用齊次座標(Homogenous Coordinates)。

(1) 設使用者於繪畫層意圖的筆跡位置向量:

$$\vec{P} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad \text{(式 31)}$$

(2) 設T'轉到繪畫層坐標系之向量，其中可以知道:

$$\vec{P} = \vec{S} + \overrightarrow{T'} \quad \text{(式 32)}$$

(3) 欲知道$\vec{P}$，先將觸控位置向量$\vec{T}$轉到繪畫層坐標系，為$\overrightarrow{T'}$

$$\overrightarrow{T'} = \begin{bmatrix} \cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \vec{T} \quad \text{(式 33)}$$

即可得到:

$$\vec{P} = \vec{S} + \overrightarrow{T'} = \vec{S} + \begin{bmatrix} \cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \vec{T} = \begin{bmatrix} X_s \\ Y_s \\ 1 \end{bmatrix} + \begin{bmatrix} \cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_t \\ Y_t \\ 1 \end{bmatrix} = \begin{bmatrix} X_s + X_t cos(\theta) + Y_t sin(\theta) \\ Y_s + Y_t cos(\theta) - X_t sin(\theta) \\ 1 \end{bmatrix} \quad \text{(式 34)}$$

6. 將繪畫層筆跡顯示在螢幕上

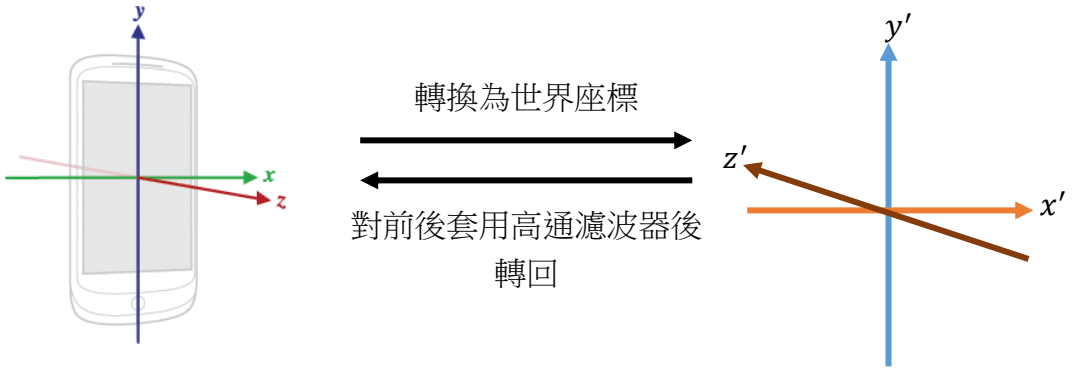
十二、 筆跡修正演算法(第二代)

第二代的修正算法大幅的改進第一代效果與效能。首先，第二代在加速度讀值中對於與指向地心方向垂直、與移動載具加速度方向相同的分量套用高通濾波器，使使用者在存在加速度的移動載具上也能使用本系統。第二代也解除了三個層(觸控螢幕層、感應器層以及假想的使用者繪畫層)相互平行的假設。完整利用三軸加速度值進行三維空間的線性變換。最後，第二代改採用線上(online)的修正方法，使系統效能大幅提升，並使筆跡持續與手指位置相黏，使用者的感受更加簡單易用。

(一) 第二代的手指-屏幕互動模型:

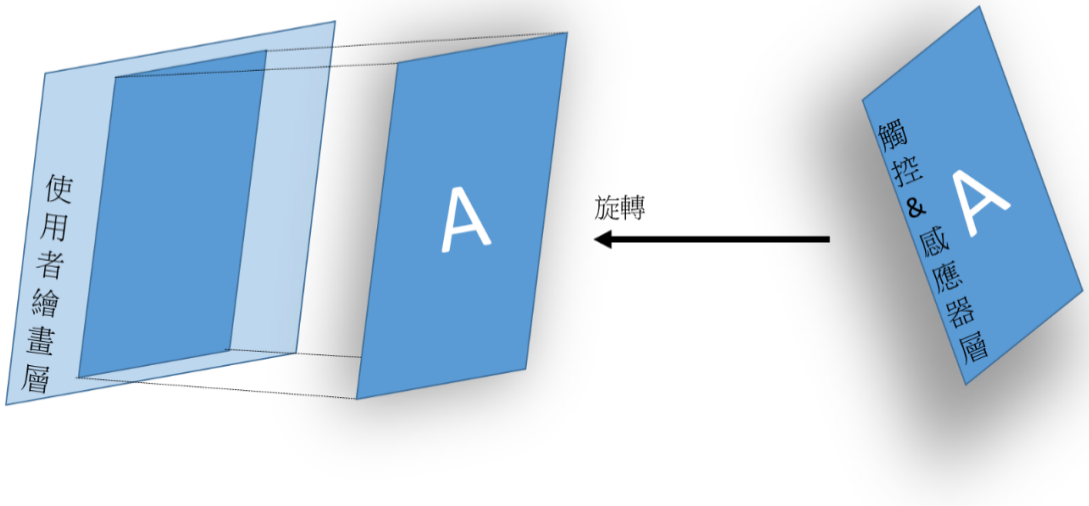
1. 繪畫層: 使用者意圖繪畫的層，由下筆瞬間被確定，始終固定
2. 感應器層: 裝有加速度計、陀螺儀及指南針。初始時原點重疊、坐標軸平行，隨裝置平移旋轉。
3. 觸控層: 用來取得觸控感應點數據。始終與感應器層相連
4. 手指: 第二代筆跡修正演算法不考慮使用者意料之外的手指運動。

(二) 對於與指向地心方向垂直、與移動載具加速度方向相同的分量(前後)套用高通濾波器:



(圖七) 對移動載具加速度方向相同的分量(前後)套用高通濾波器

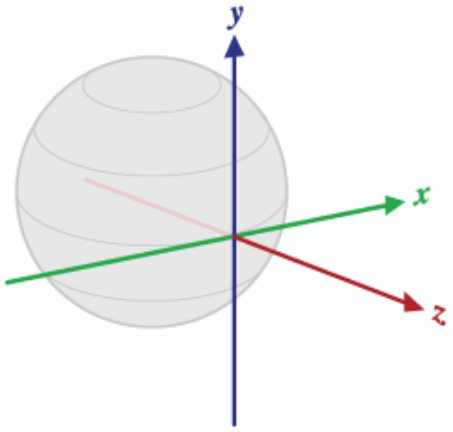
(三) 將觸控層映射到繪畫層:



(圖八) 進行三維觸控點的線性變換

1. 下筆當下所有層坐標軸平行共原點，設定一共用原點(0,0)。
2. 因為人們不垂直寫字，故不考慮 Z 軸移動；設觸控層相對於原點移動(X_s, Y_s) ，設$\vec{S}$為觸控層原點於繪畫層的位置向量。
3. 設觸控層上一觸控點(X_t, Y_t)，其在觸控層上之位置向量為$\vec{T}$。
4. 為了防止特殊情形下環架鎖定(Gimbal lock)的發生，這裡所有的旋轉皆以四元數(Quaternion)表示。設下筆瞬間所有重疊層的絕對方向q_0，下一觸控點的絕對方向q_1。

5. 這裡 Quaternion 中的旋轉向量座標如下定義:



(圖九)世界座標的定義

X 軸:Y 與 Z 軸之外積方向(Y × Z)，Y 軸:指向地磁北方為正，Z 軸:指向天空為正(重力的反向)

6. 假設從原點至下一採樣點之間，感應器層與觸控層相對繪畫層進行了一個含有移動(X_s, Y_s)與旋轉q_{01}的線性變換。

(1) 設使用者於繪畫層意圖的筆跡位置向量:

$$\vec{P} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \quad \text{(式 35)}$$

(2) 設T'轉到繪畫層坐標系之向量，其中可以知道:

$$\vec{P} = \vec{S} + \overrightarrow{T'} \quad \text{(式 36)}$$

(3) 設定初始及下一觸控點之 Quaternion:

$$q_0 = q_0[0] + q_0[1]i + q_0[2]j + q_0[3]k \quad \text{(式 37)}$$

$$q_1 = q_1[0] + q_1[1]i + q_1[2]j + q_1[3]k \quad \text{(式 38)}$$

(4) 取兩四元數之差:

$$q_{01} = q_1 q_0^{-1} = q_1 \frac{q_0^*}{\|q_0\|^2} \quad \text{(式 39)}$$

(5) 因為從 Android Rotation Vector 取出之四原數的範數皆為 1，故:

$$q_{01} = q_1 q_0^* \quad \text{(式 40)}$$

其中:

$$q_0^* = q_0[0] - q_0[1]i - q_0[2]j - q_0[3]k \quad \text{(式 41)}$$

(6) 接著透過q_{01}旋轉$\vec{T}$得到$\overrightarrow{T'}$:

$$\overrightarrow{T'} = q_{01}\vec{T}{q_{01}}^{-1} \quad \text{(式 42)}$$

用旋轉矩陣的形式表示:

$$\vec{T'} = \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} (1-2q_{01}[2]^2-2q_{01}[3]^2) & 2(q_{01}[1]q_{01}[2]+q_{01}[0]q_{01}[3]) & 2(q_{01}[1]q_{01}[3]-q_{01}[0]q_{01}[2]) \\ 2(q_{01}[1]q_{01}[2]-q_{01}[0]q_{01}[3]) & (1-2q_{01}[1]^2-2q_{01}[3]^2) & 2(q_{01}[2]q_{01}[3]+q_{01}[0]q_{01}[1]) \\ 2(q_{01}[1]q_{01}[3]+q_{01}[0]q_{01}[2]) & 2(q_{01}[2]q_{01}[3]-q_{01}[0]q_{01}[1]) & (1-2q_{01}[1]^2-2q_{01}[2]^2) \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \quad \text{(式 43)}$$

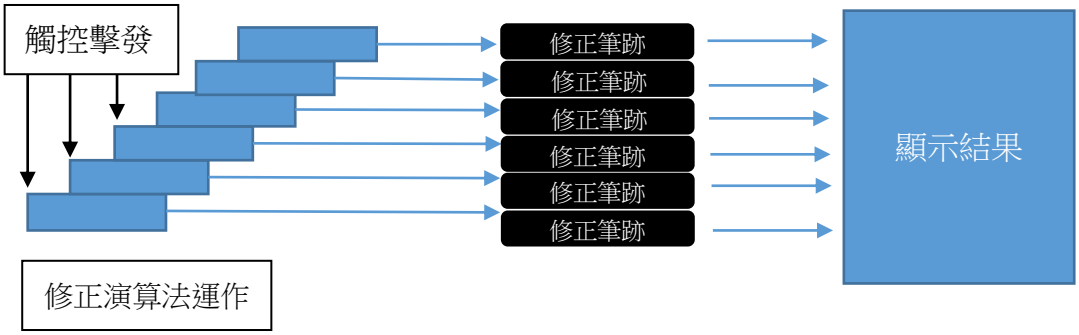
(7) 最後得到:

$$\vec{P} = \begin{bmatrix} X_s \\ Y_s \\ 0 \end{bmatrix} + \begin{bmatrix} (1-2q_{01}[2]^2-2q_{01}[3]^2) & 2(q_{01}[1]q_{01}[2]+q_{01}[0]q_{01}[3]) & 2(q_{01}[1]q_{01}[3]-q_{01}[0]q_{01}[2]) \\ 2(q_{01}[1]q_{01}[2]-q_{01}[0]q_{01}[3]) & (1-2q_{01}[1]^2-2q_{01}[3]^2) & 2(q_{01}[2]q_{01}[3]+q_{01}[0]q_{01}[1]) \\ 2(q_{01}[1]q_{01}[3]+q_{01}[0]q_{01}[2]) & 2(q_{01}[2]q_{01}[3]-q_{01}[0]q_{01}[1]) & (1-2q_{01}[1]^2-2q_{01}[2]^2) \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \quad \text{(式 44)}$$

(8) 將繪畫層筆跡顯示在螢幕上

(四) 改採用線上(online)的修正方法:

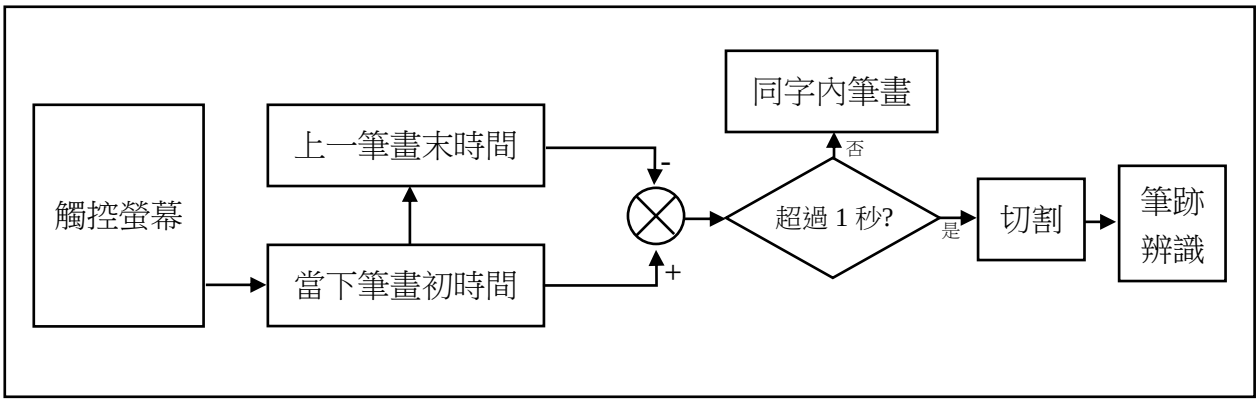
第二代利用多線程處理使處理速度、介面反應速度皆大幅提升。



(圖十) 多線程處理

(五) 同字內筆畫偵測:

為了使本系統擁有區分筆畫是否屬於同一字母的能力(多筆畫字如 A、B、J...)，本系統設計一基於筆畫時間差之機制切割並群組屬於不同字之筆畫。



(圖十一) 同字內筆畫偵測流程圖

(六) 使筆跡持續與手指位置相黏:

系統持續移動已修正之筆跡使其末端與使用者最新的觸控座標相黏。此有助於讓使用者了解最新修正狀況並提升使用效果、且更加直覺。

(圖十二) 一、二帶顯示方式比較 (左:第一代、右:第二代)

十三、 筆跡修正演算法(第三代)

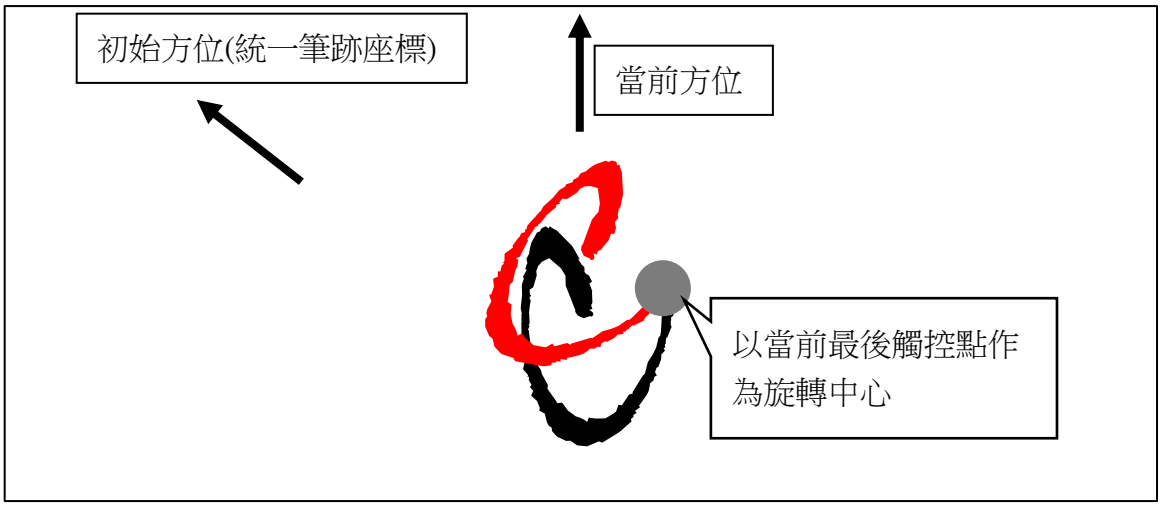
第三代的修正算法以第二代算法為基礎，進行大量的性能優化，同時加入筆跡平滑化使修正結果更加美觀。

(一) 性能優化

第三代的大幅性能進展來自於累積先前運算結果，而非如過去幾代在每一次重繪事件中重新運算全畫面的筆跡修正結果。此全新的累加算法主要內容如下:

1. 筆跡的宏觀旋轉

裝置旋轉時，整體筆跡需要往反方向進行相應的旋轉以抵銷旋轉晃動。於下筆瞬間得知初始絕對方向q_0(全域統一的筆跡座標)，在每一次重繪事件中得知當下絕對方向q_1，取兩方向之差並將所以歷史筆跡旋轉至當前方向。取 x-y 軸之筆跡顯示。



(圖十三) 將筆跡旋轉至當前方位

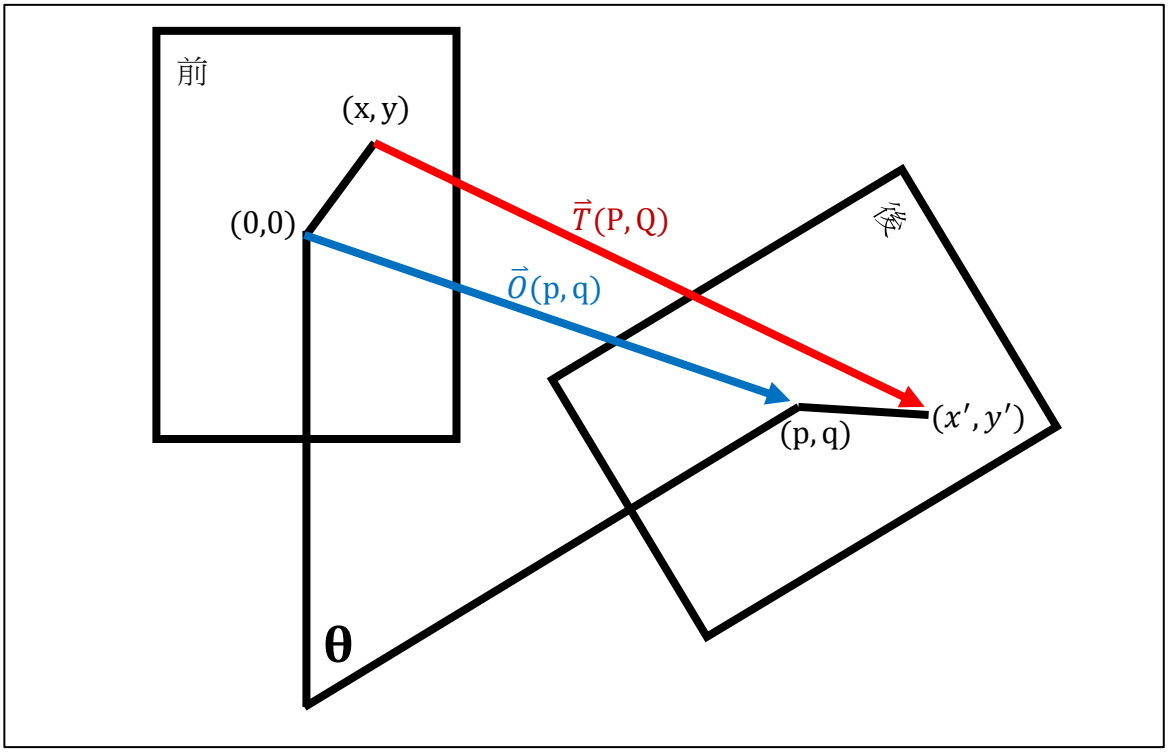
2. 統一筆跡座標

為了使同一筆畫中的所以觸控點座標統一，本系統以下筆當下方位作為筆跡統一的方位。為了使裝置偏離初始方向後的筆跡變化量之方向符合統一座標之要求，於下筆瞬間得知初始絕對方向q_0 (全域統一的筆跡座標)，在每一次重繪事件中得知當下絕對方向q_1，取兩方向之差，對每一個筆跡變化向量反向旋轉並黏接至上一筆跡點得到新筆跡座標。此修正最明顯的作用在於將因旋轉變成曲線的直線拉直。

3. 修正因為觸控點與裝置感應具存在距離造成之估計誤差

由於觸控點與裝置感應器(加速度計、陀螺儀及指北針)與裝置觸控點存在距離，因此透過以下修正以符合實際狀況。

(1) 設觸控座標(x, y)，旋轉後觸控座標移至(x', y')，由加速度計估計之位移$\vec{O}(p, q)$，觸控座標於空間中實際發生之位移$\vec{T}(P, Q)$，過程中裝置由方向感應器得知順時針旋轉了θ度，其旋轉矩陣R。



(圖十四) 計算觸控點空間中實際位移

(2) 以測得之位移表示$\vec{O}$表示旋轉後觸控座標(x', y')

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \vec{O} + R \begin{bmatrix} x \\ y \end{bmatrix}$$

(3) 整理上式得知正確位移$\vec{T}$

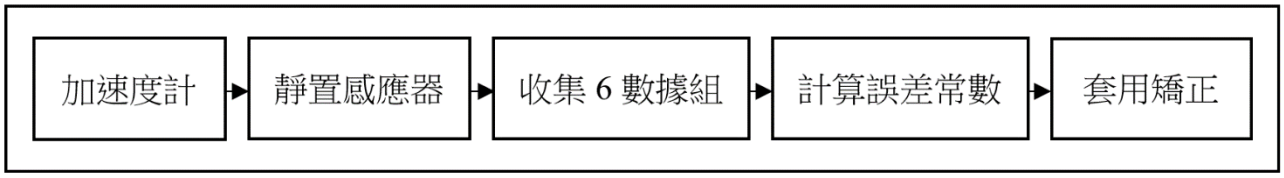
$$\vec{T} = \begin{bmatrix} P \\ Q \end{bmatrix} = \begin{bmatrix} x' \\ y' \end{bmatrix} - \begin{bmatrix} x \\ y \end{bmatrix} = \vec{O} + R \begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} x \\ y \end{bmatrix}$$

(二) 筆跡平滑化

第三代將離散的觸控點序列以三次貝茲曲線(Cubic Bézier curves)使平滑，實際使用更加美觀。

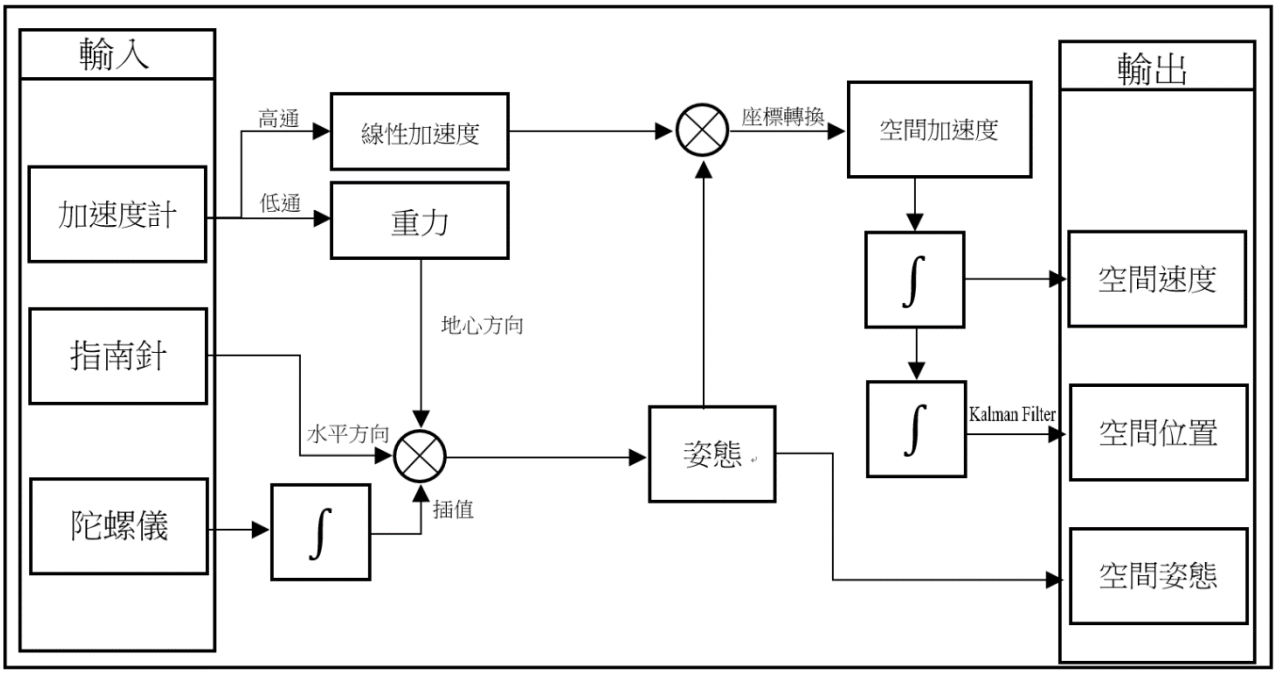
十四、　完整流程:

(一) 感應器矯正:



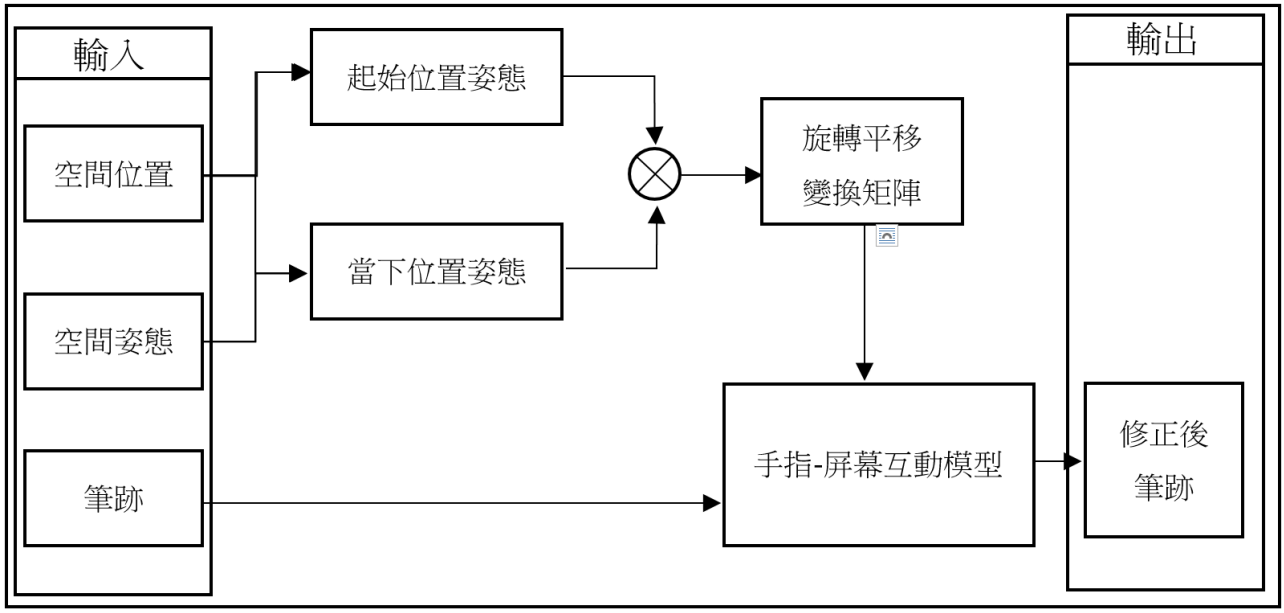
(圖十五) 感應器矯正流程圖

(二) 移動估計模組:



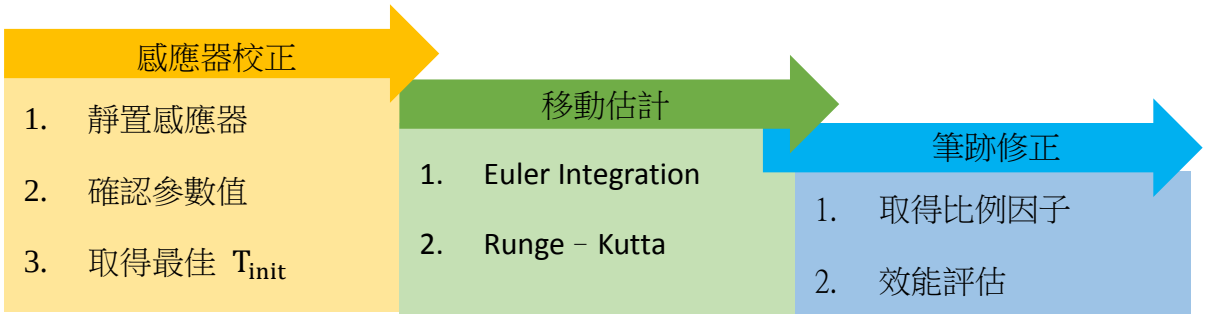
(圖十六) 移動估計模組流程圖

(三) 筆跡變形修正模組:



(圖十七) 筆跡變形修正模組流程圖

伍、 研究結果及討論



(圖十八) 實驗流程

一、 感應器校正:

(一) 調整靜置感應器閥值:

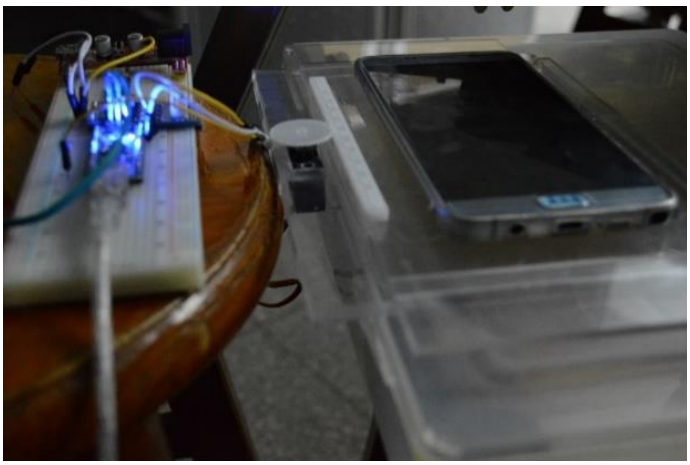
靜置感應器的觸發是依賴觀察窗內三軸加速度數值變異數數值是否小於一個的給定的閥值。這個閥值若太高，靜置感應器不容易被觸發；相反的，若閥值太低，靜置感應器則太容易被觸發。本實驗透過嘗試不同的閥值大小嘗試取得靈敏度與準確度的平衡。本實驗所使用的週期運動產生器將會使裝置產生兩種不同的狀態:

I. 裝置被靜置於平台上

II. 裝置被週期運動產生器移動

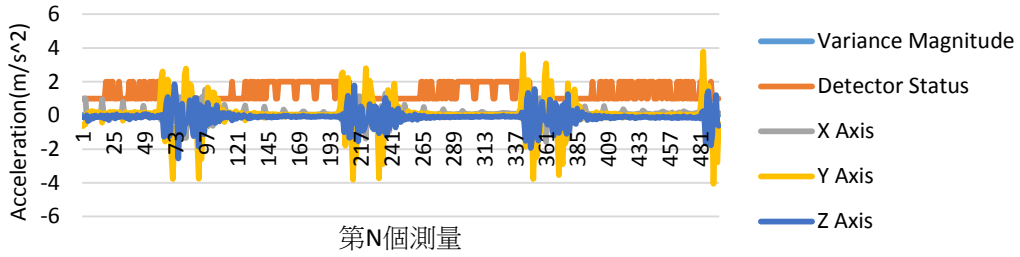
週期運動產生器產生向前 1.2cm，向後 1.2cm 的週期運動。在移動間個中會停止 1 秒鐘，此時裝置成靜止狀態。

本實驗測試將閥值設為 0.0004 到 0.005 的狀況。其以 0.0001 的間隔分別測試。此實驗的輸出為三軸加速度數值、觀察窗中數列變異數震幅及靜置感應器狀態。



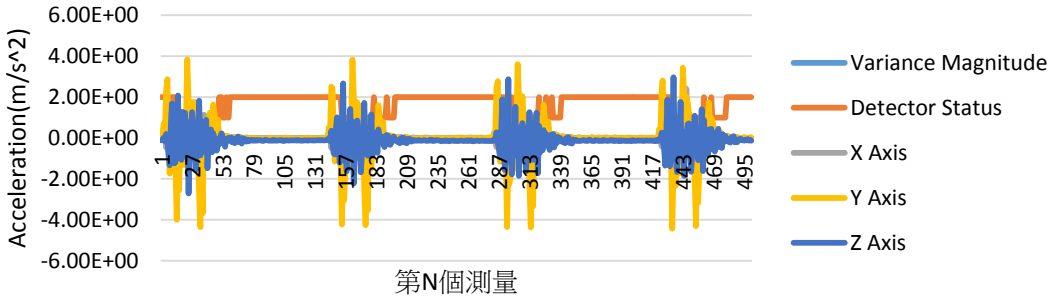
(圖十九) 實驗設置

當閥值設為 0.0004 的輸出圖:



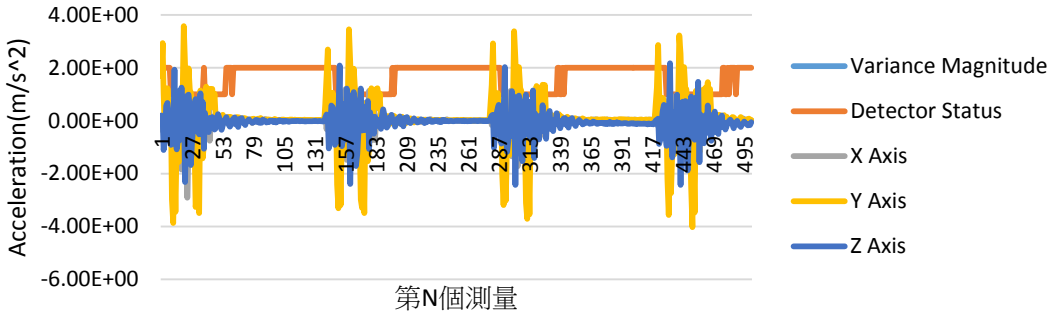
(圖二十) 靜置感應器閥值=0.0004

(圖二十)顯示如果三軸變異數大小小於 0.0004, 靜置感應器輸出 2; 如果三軸變異數大小大於 0.0004， 靜置感應器輸出 1。當靜置感應器閥值設為 0.0004，感應器顯得太過敏感。舉另一個極端的例子將靜置感應器閥值設為 0.005，輸出以下(圖二十四)圖形:



(圖二十一) 靜置感應器閥值=0.005

太高的靜置感應器閥值被會導致感應器低靈敏度。有時候甚至在裝置移動時靜置感應器依然判定裝置靜置。為了尋找最佳的靜置感應器閥值，本實驗測試了 0.0004 到 0.005，以 0.0001 為間隔的所有數值。最終，實驗結果說明 設定靜置感應器閥值 0.0017 可以達到最佳的靜止/移動區分性能。



(圖二十二) 靜置感應器閥值=0.0017

結果(圖二十二)靜置感應器成功的在裝置靜止時輸出靜止狀態(數值為 2)。這樣長的靜止時段也足夠常讓系統收集校正所需資料。

(二) 確認感應誤差模型參數:

利用 Algorithm 1 可以確認感應器誤差模型中各參數數值。首先翻轉並靜置裝置 6 次，每次都經過一個資料收集循環。這會收集 6 組裝置在不同靜止姿勢的加速度數值組。 接著呼叫 Levenberg-Marquardt 演算法最小化誤差的能量函數，最後輸出並套用最適合的參數數值。

1. 收集 6 組裝置在不同靜止姿勢的加速度數值組

(圖二十三) 姿勢 1

(圖二十四) 姿勢 2

(圖二十五) 姿勢 3

(圖二十六) 姿勢 4

(圖二十七) 姿勢 5

(圖二十八) 姿勢 6

2.

3. (表四) 校正參數數值(表四)顯示所得校正參數數值

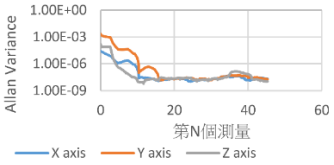
(表四) 校正參數數值

第 1 次	Bias_X	Bias_Y	Bias_Z	Scale_X	Scale_Y	Scale_Z
	0	0	0	1	1	1
第 2 次	Bias_X	Bias_Y	Bias_Z	Scale_X	Scale_Y	Scale_Z
	0	0	0	0.9999999	1	1
第 3 次	Bias_X	Bias_Y	Bias_Z	Scale_X	Scale_Y	Scale_Z
	0	0	0	1	1	1
第 4 次	Bias_X	Bias_Y	Bias_Z	Scale_X	Scale_Y	Scale_Z
	0	0	0	1	1	1
第 5 次	Bias_X	Bias_Y	Bias_Z	Scale_X	Scale_Y	Scale_Z
	0	0	0	1	0.9999999	1

(三) 計算 Allan variance 確認 T_{init}:

透過計算 Allan variance， T_{init} 的值可以被得知。而當三軸的 Allan 差逐漸縮小到最小值時，系統就可以開始收集數據，也就是T_{init} 的時候。在這個實驗中，手機以螢幕向上的姿勢被靜止放置在平坦的桌面上。當程式啟動時，系統開始以 0.25 秒的觀察窗計算 Allan variance。(圖二十九)為實驗設置。

(圖二十九) 平坦靜置



(圖三十) Allan variance

(圖三十)顯示 Allan variance 的輸出；在 20 秒的時，Allan variance 已經逐漸縮小，因此 T_{init}被設定為 20 秒，也就是當程式啟動時必須等待 20 秒才會開始收集感應器數據。

二、 移動估計:

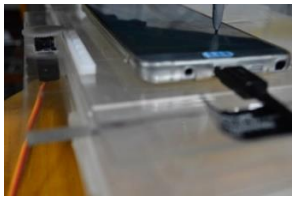
(一) 使用 Euler Integration 對加速度值積分做移動估計:

本實驗透過比較 Euler Integration 計算的位移和實際位移之間的相似形評估移動估計效能，週期運動產生器將裝置前後移動使裝置在空間中產生週期性運動，真實位移是利用一個固定在空間中的觸控筆與在觸控螢幕上畫下的軌跡計算而得，使用的是以下兩方程式:

$$\Delta x = -x_t * R_{tm} \quad \text{(式 45)}$$

$$\Delta y = -y_t * R_{tm} \quad \text{(式 46)}$$

其中R_{tm} =螢幕像素對應的長度，x_t=x 軸上觸控筆在螢幕上的位移，y_t= y 軸上觸控筆在螢幕上的位移，位置的初始值都被預先設定為 (0,0)，接著比對裝置估計所得移動與實際移動之間的差異。



(圖三十一) 週期運動產生器

本實驗應用了餘弦相似性評估裝置估計所得移動與實際移動之間的相似性，並尋找最合適的濾波器參數。

餘弦相似性是簡單有效的相似性比較方法。以下給出兩線餘弦相似性之定義:

$$\text{Cosine similarity} = \cos\theta = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^1}\sqrt{\sum_{i=1}^{n} B_i^1}} \quad \text{(式 47)}$$

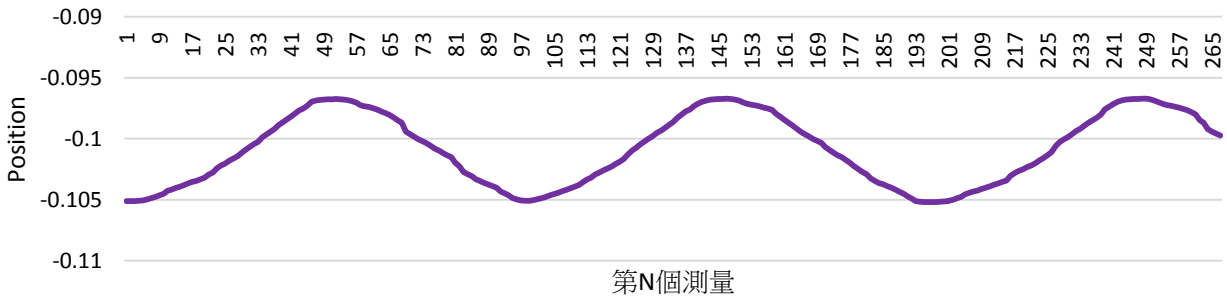
A_i 與B_i 分別為 A 與B 上的一小段向量，實驗發現經兩相似性評估方法得出相同的使用 Euler Integration 方法最佳參數組合: (低通濾波器 alpha=0.8；高通濾波器 alpha=0.7)之後若使用 Euler Integration 方法估計移動則皆使用此組參數。

此參數組合之餘弦相似性最高，最接近原移動曲線，雖然比例因子在此時未被確定，由圖及相似性可知 Euler Integration 法已可成功地進行移動估計，使用內建感應起重塑移動曲線形狀。

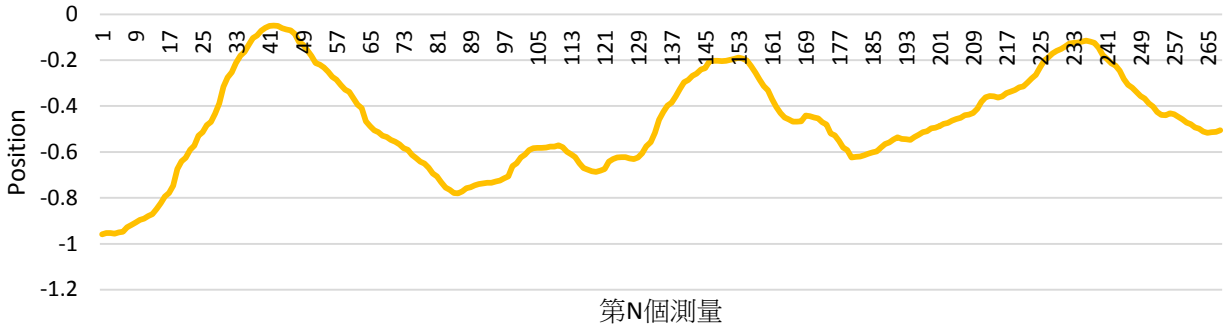
(二) 使用 RK4 方法進行移動估計:

本實驗透過相同的方法尋找適合的最佳濾波器參數，只差在使用 RK4 積分方法。組態(RK4，低通濾波器 alpha = 0.9，高通濾波器 alpha = 0.7)為最接近原移動曲線之參數組合。同樣的，雖然比例因子在此時未被確定，由數據與相似性可知使用 Rk4 積分方法可成功地進行移動估計，而且比 Euler Integration 更加準確地用內建感應起重塑移動曲線形狀。

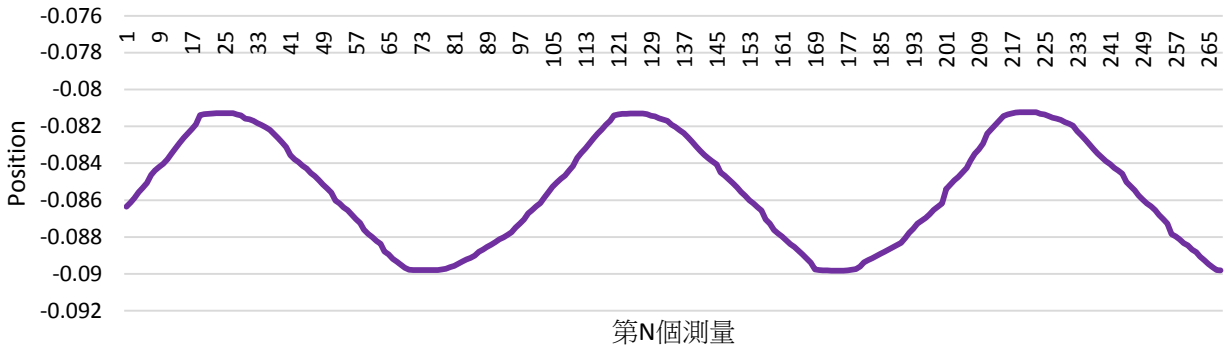
(圖三十二) 使用 Euler Integration 進行移動估計



(圖三十三) 實際位置



(圖三十四) 使用 RK4 方法進行移動估計



(圖三十五) 實際位置

三、 筆跡修正:

(一) 取得比例因子:

為了使修正結果更加準確，本實驗目的在於確定修正結果與理想筆跡之間的比例因子。本實驗循下列步驟進行:

1. 定義誤差

這裡的誤差被定義為修正結果與理想筆跡位置點的距離。

$$\text{error} = \sqrt{(staX - ideX)^2 + (staY - ideY)^2} \quad \text{(式 48)}$$

其中

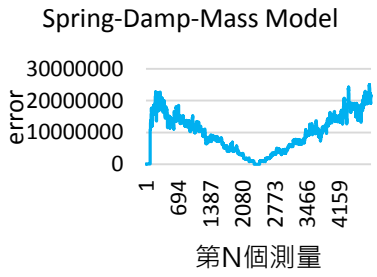
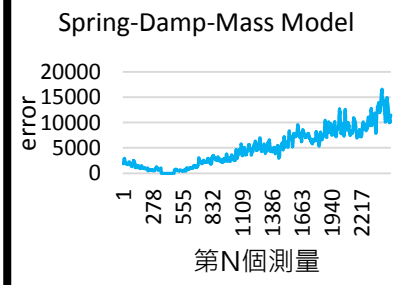
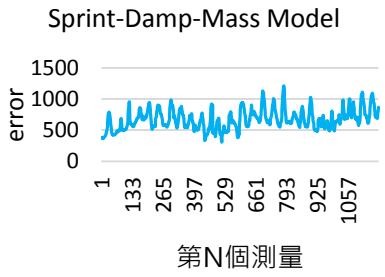
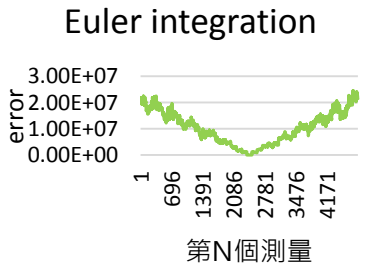
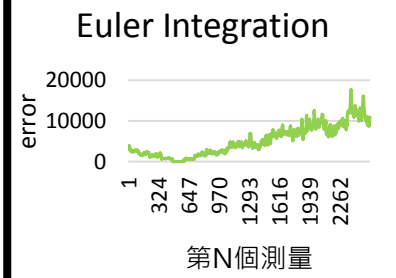
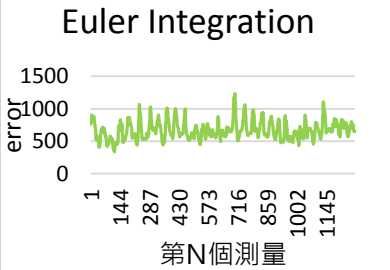
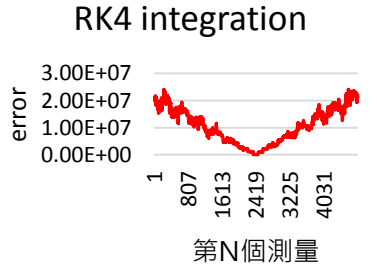
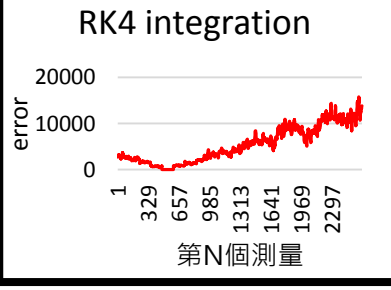
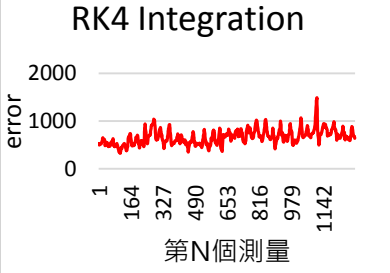
$$\text{staX} = \text{stroX} - \text{posX} * \text{Mutiplier} \quad \text{(式 49)}$$

$$\text{staY} = \text{stroY} - \text{posY} * \text{Multiplier} \quad \text{(式 50)}$$

staX: 修正後位置點 X 座標，staY: 修正後位置點 Y 座標，ideX: 理想位置點 X 座標，ideY: 理想位置點 Y 座標，由於在實驗中觸控筆在空間中的位置被固定在一點，因$ideX$及$ideY$始終為 0。

2. 調整比例因子從-1000 到 1000 ，間隔 50。尋找使誤差最小的比例因子:

(表五) 尋找使誤差最小的比例因子

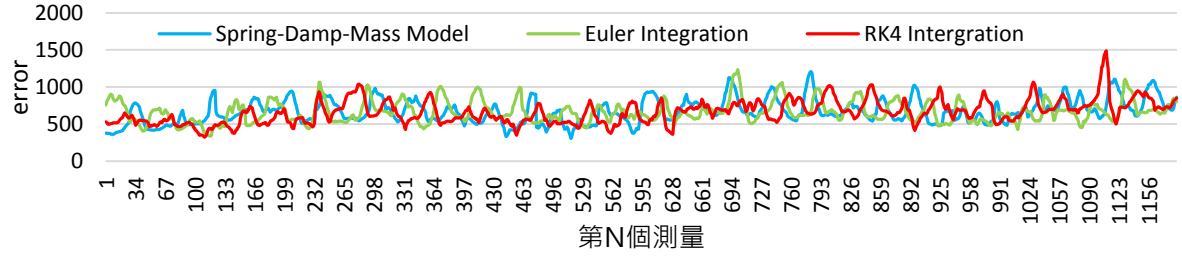
-1000 到 1000 (間隔 50)	0.2 到 0.9 (間隔 0.05)	0.05 到 0.06 (間隔 0.001)
		
		
		

(二) 評估筆跡變形修正系統之效能

將觸控筆固定於一點，理想座標必為(0,0)。因此筆跡變形修正系統之效能可由以下方程式評估:

$$\text{error} = \sqrt{(staX)^2 + (staY)^2} \qquad \text{(式 51)}$$

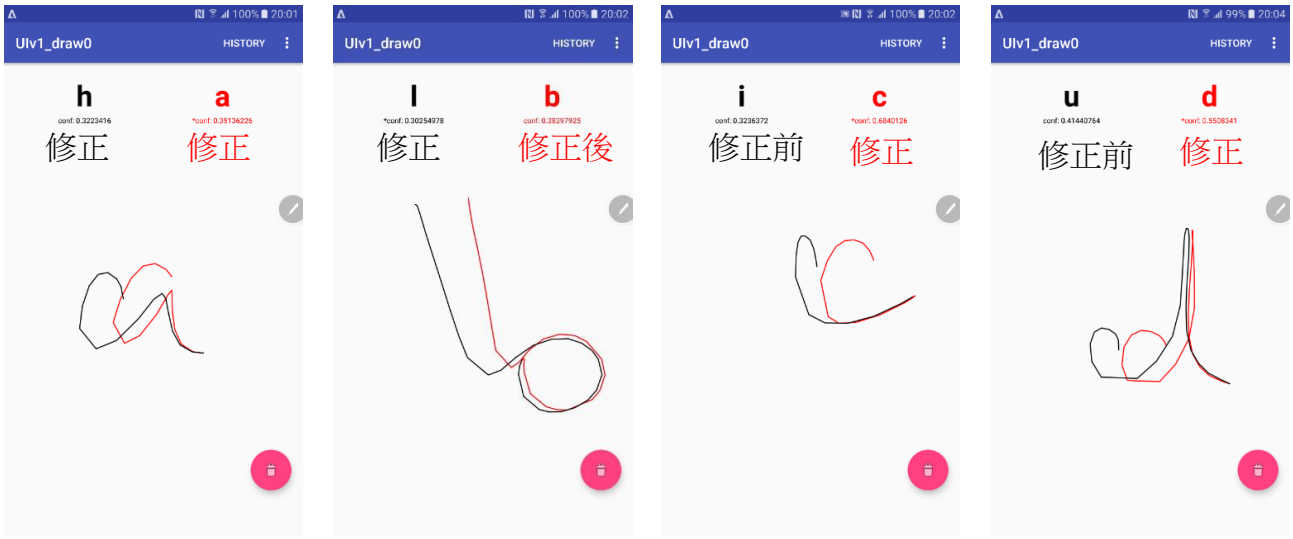
以下顯示使用不同移動估計算法所得之筆跡變形修正系統之效能，誤差越小越佳



(圖三十六) 誤差比較

陸、 結論

本研究提出了筆跡變形修正系統，目的在於修正具觸控螢幕裝置之筆跡變形。筆跡變形常出現在身處於不穩定環境的使用者欲再裝置上書寫時。此系統由三個主要步驟構成:首先，使用者等待由 Allan variance 決定的特定秒數後進行感測器校正。透過翻傳並靜置裝置數次，系統取得足夠數據後進行校正。接著，本研究提出了變形矯正算法，說明了裝置本身的移動、手指與觸控螢幕互動以及整體在空間中的關係。透過此模型可利用移動估計修正變形的筆跡。最後，透過探討不同利用加速度值算位移的方法及濾波器參數組態尋找到最佳移動估計方法。以下顯示筆跡變形修正系統實際使用效果。黑色為原始筆跡，紅色為修正後筆跡，書寫過程中裝置皆由右往左移動。由(圖 38)可見經過修正後辨識成功率大幅提高。



*裝置移動方向: ←　　原始筆跡:黑色　　修正後筆跡:紅色

(圖三十七) 筆跡變形修正系統實際運作效果

柒、 參考資料及其他

[1] KPCB. (2015 年 5 月 27 日). INTERNET TRENDS 2015 - CODE CONFERENCE. 擷取自 www.kpcb.com: http://www.kpcb.com/internet-trends

[2] Trinh Minh Tri Do , Jan Blom, & Daniel Gatica-Perez. (2011). Smartphone usage in the wild: a large-scale analysis of applications and context.

[3] Haiying Hou 2004 Modeling Inertial Sensors Errors Using Allan Variance

[4] David Tedaldi, Prof. Emanuele Menegatti , & Ing. Alberto Pretto. (2013 年 9 月 23 日). IMU calibration without mechanical equipment. Page 18.

[5] Ahmad Rahmati, Clayton Shepard, & Lin Zhong . (2009). NoShake: Content Stabilization for Shaking Screens of Mobile Devices. Page 1-6.