

# 2017 年臺灣國際科學展覽會 優勝作品專輯

作品編號 190016

參展科別 電腦科學與資訊工程科

作品名稱 NoDistort: Drawing Distortion Recovery  
System for Shaky Screens

得獎獎項 大會獎：四等獎

就讀學校 臺北市立建國高級中學

指導教師 李文禮、陳炳宇

作者姓名 蘇柏穎

關鍵字 Distortion Recovery、Motion Estimation、  
User Interface

## 作者簡介



大家好，我是蘇柏穎。目前就讀於台北市建國中學普通班三年級，在高一上時加入了專題研究社，經過推選後，在高二時成為第三屆社長。

國一時我加入了數理資優班，我的啟蒙導師劉睿荷老師開了發明課及認識研究等課程，還邀請了許多國內創意發明界的大師現場示範，其大神風采使我大為震撼，也不小心開啟了我內心深處某種連自己也很陌生的開關。

進入建國中學後，充斥在專題研究社學術與創意瀟灑的氛圍中，我對專題研究的熱情與感動也日益深厚。在指導老師李文禮以及專題研究社創社社長暨現任社團指導老師的指導與信念加持下，我一步步建築起這個對於我甚至是於現代人都意義重大的專題研究。研究的過程中，雖然經常遭遇阻礙，無論是能力上或是精神上甚至是體力上，我總是想著專研的社訓告訴我：只要相信，就會成真。

我真的很感謝一路來家人、師長、學長、朋友們以及眾多社員犧牲許多時間教導我、陪伴我，讓我能將一個個問題解決，使目標一步步落實，還提供了我許多發人深省的建議。未來這把專研烈火將在我心中持續燃燒著，因為專題研究不只是一種信念，更是一種態度。

## 摘要

在這個智慧型裝置普及的時代，觸控螢幕已經深入人們的生活。隨著手寫輸入法、滑行輸入法等筆跡輸入系統的完善，其也成為人們與裝置互動最重要的工具，卻也常因為搖晃造成輸入的筆跡變形。為了改善人們在不穩定環境下的輸入體驗，本研究提出了“筆跡變形修正系統”。通過結合內置感應器的資訊，系統估計裝置的運動軌跡，並以提出的屏幕-手指互動模型恢復失真的筆跡。研究中使用了一種不需要額外儀器的校準方法校正感應器誤差，再套用本研究所開發之變形修正演算法，成功在生活中恢復失真筆跡並證明了其有效性及必要性。經過數代的改進與實驗參數優化，修正效果已大幅提升。

## Abstract

In the era of smart mobile technology, touchscreen-enabled devices have gone deep into modern lives. As handwriting recognition technology evolves, input methods such as traditional character recognition or Swype-like keyboards has become the most important tool for humans to interact with mobile devices. But more often than not, people suffer from handwriting distortion while drawing under unstable settings. To improve writing experiences on mobile devices, the study presents NoDistort, which stands for "Drawing Distortion Recovery System for Shaky Screens". By combining data from internal motion sensors, the system estimates the displacement and posture of the device. Then the system corrects the distorted drawings by the proposed drawing correction algorithm. The study also designed an equipment-free self-calibration pipeline which aimed to calibrate sensors easily. After several updates and optimizations, the algorithm had gained huge progress. In real life situations, the system had successfully corrected distorted handwritings and had shown its effectiveness and usefulness.

## 壹、研究動機

觸控螢幕的方便性使行動裝置的普及率迅速增長。如今觸控裝置除了能繪畫、做手寫筆記，甚至還能夠進行手寫辨識。觸控螢幕的輕薄化也使觸控螢幕降臨在現代生活的每一個角落，如先進工廠機具的操作螢幕、智慧汽車的主控台、行動裝置甚至是戰鬥飛機的儀表板。但所有裝置都存在著一個共同的待解決問題：在不穩定的環境中，筆跡會因為搖晃而導致筆跡變形。

根據 Do. 等人於 2011 年所做的調查<sup>[1]</sup>，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。而在這種劇烈晃動的環境中(如公車上)，人們輸入的筆跡會因晃動而變形扭曲。

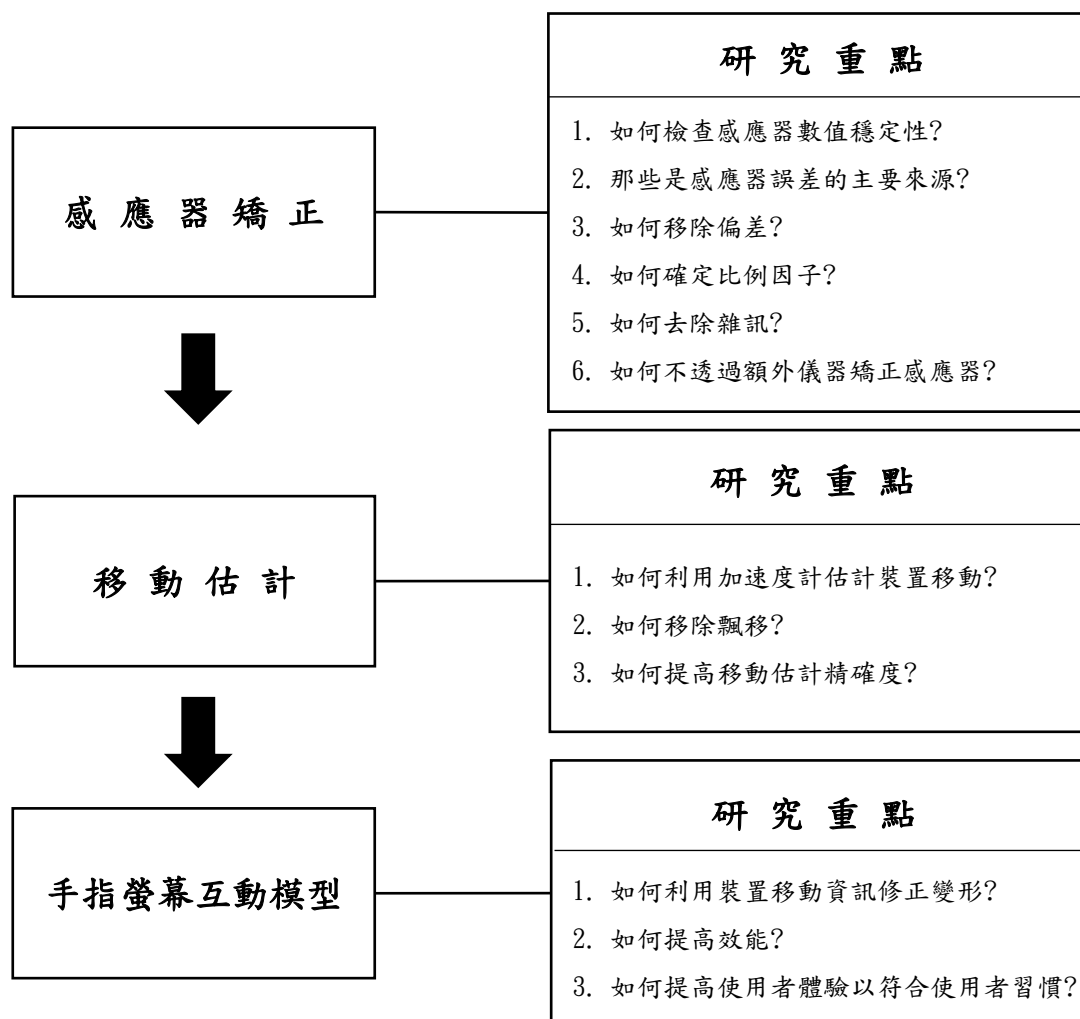
(表 1)、收集時間內各地方使用軟體的頻率 (單位：平均事件數/小時)<sup>[2]</sup>

地點 軟體	住家	公司	朋友家	朋友公司	餐廳	運動	交通工具	假日	逛街購物	休閒	其他	平均值
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

為了解決這個長期存在的問題，與提高觸控裝置的手寫體驗，本研究提出全新的演算法和解決方案，解決裝置晃動造成的筆跡扭曲。

## 貳、研究目的





本研究的目的是提出一個全新的演算法，修正因裝置晃動所導致的變形，稱為筆跡變形修正系統。為了使本系統能夠廣泛應用於各種類型的觸控裝置上，本系統主要利用常見於智慧型裝置的內建加速度計、磁力計與陀螺儀，為了取得精確的感應器數據，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，建立一個變形修正演算法，修正因為晃動變形的筆跡，以下(圖 1)為筆跡變形修正系統研究流程與重點。



(圖 1)、筆跡變形修正系統流程圖

## 參、研究設備及器材

(表 2)、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

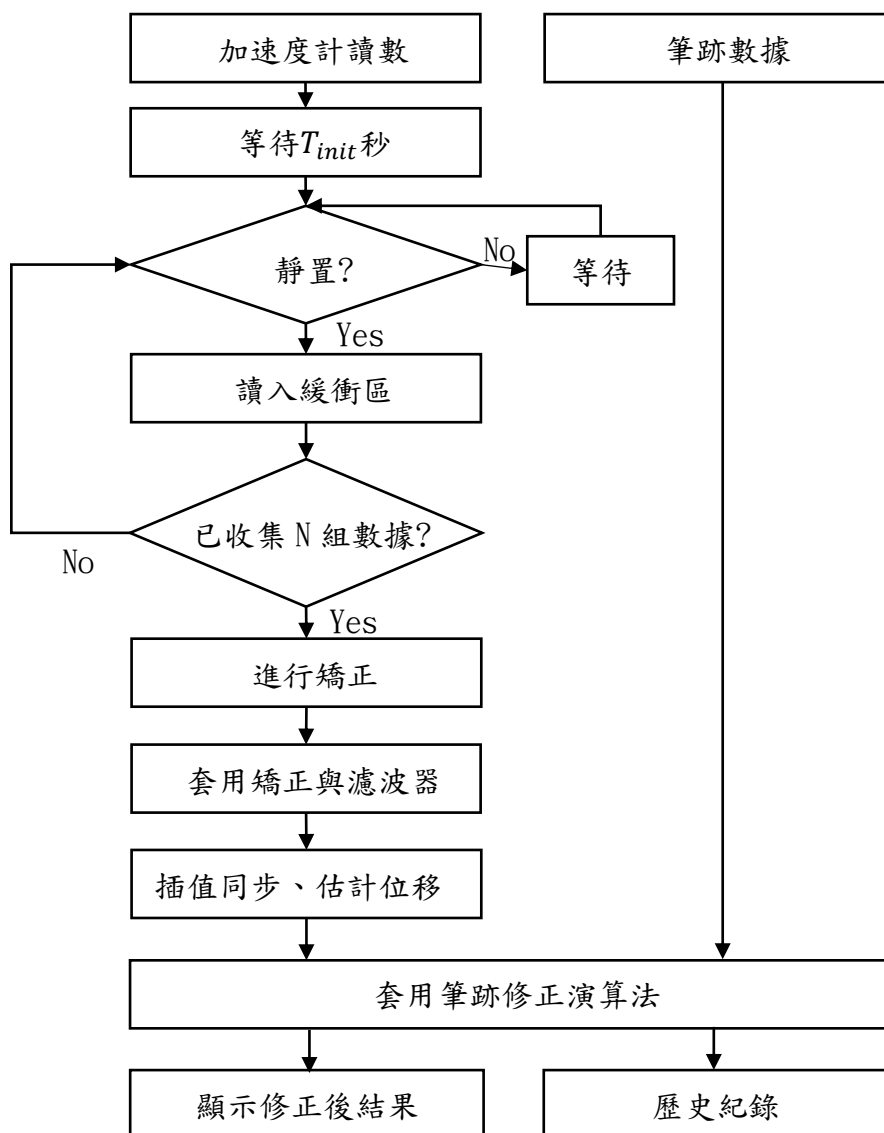
(表 3)、研究軟體一覽

軟 體	圖 片	版 本
Matlab		R2016b
Android Studio		v2.1
Arduino Software (IDE)		V2.0

## 肆、研究過程或方法

### 一、NoDistort 運作架構圖：

(圖 2)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，變形的筆跡透過筆跡修正演算法得到修正。



(圖 2)、筆跡變形修正系統

## 二、Allan variance:

為了知道何時開始獲取感應器資料，這裡使用了 Allan variance 技術<sup>[3]</sup>進行分析。Allan variance 用於觀察震盪系統的雜訊特性和穩定性。這裡用來觀察加速度計訊號雜訊的特性。本系統中 Allan variance 透過以下過程計算：

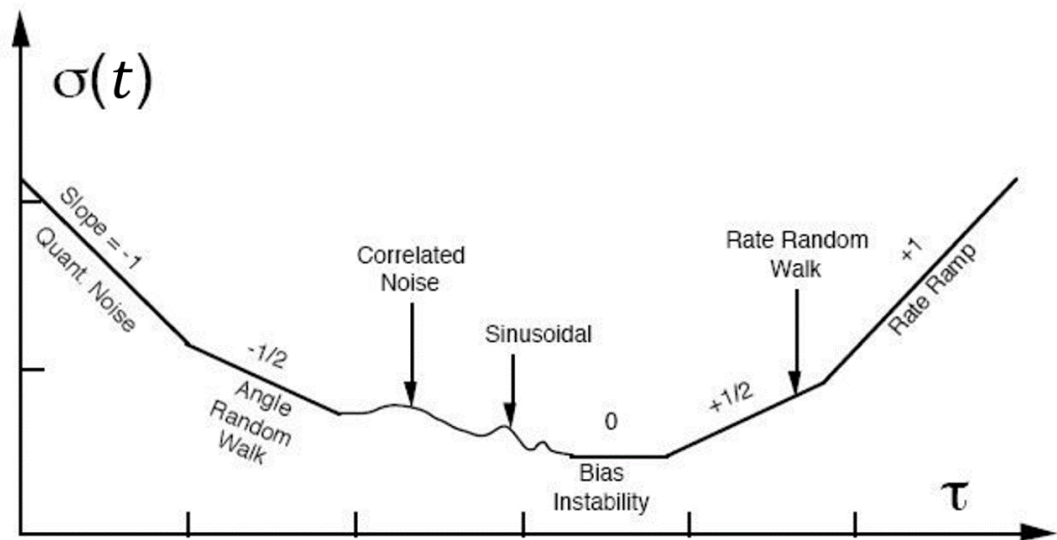
- (一)、從裝置的感應器獲得的 3 軸的數據。
- (二)、將數據讀入環形緩衝區。
- (三)、從緩衝區讀入 100 個採樣數據，並以時間長  $t=1s$  將其分為小段。
- (四)、對於每個長度  $t$  的小段，取每一段的平均  $a(t_1)$ ， $a(t_2)$ ， $a(t_3) \cdots a(t_n)$ 。
- (五)、計算 Allan variance:

$$AVAR(t) = \frac{1}{2(n-1)} \sum_{i=1}^n (a(t_{i+1}) - a(t_i))^2 \quad (\text{式 1})$$

下式用來得到基本雜訊的特徵，Allan 偏差的定義如下：

$$\sigma(t) = \sqrt{AVAR(t)} \quad (\text{式 2})$$

以 Allan 偏差檢查感應器的初始化階段：



(圖 3)、感應器初始化的不同階段

由(圖 3)可知，當三個軸的 Allan 偏差的斜率為 0 時，感應器已經進入穩定階段，是開始接收穩定感測器數據的時間點。以本研究的測試情況，等待 35 秒後即可開始獲得穩定的感應器資料。



### 三、移動平均濾波器：

本研究之測試裝置的感應器資料以 100Hz 的速率更新，100Hz 的更新速率已足夠套用移動平均濾波器的同時，仍然保留細節。測試結果顯示設置移動平均窗口為 $\frac{1}{4}$ 秒可以有很好的濾除雜訊效果。(式 3)顯示一個時間範圍內 $\Delta t$  的平均值，N是觀察窗口內的採樣數。

$$SMA(t) = (a_{t-\frac{\Delta t}{2}} + \dots + a_{t+\frac{\Delta t}{2}})/N \quad (\text{式 3})$$

### 四、加速度計的矯正：

以下表格為加速度計主要的誤差來源：

#### (一)、偏差：

感應器偏差是在加速度計的輸出信號中持續造成誤差的一個常數。在理想情況下 $b^a$ 是一個 3x1 零向量，偏差向量可以表示為：

$$b^a = \begin{bmatrix} b_x^a \\ b_y^a \\ b_z^a \end{bmatrix} \quad (\text{式 4})$$

#### (二)、比例因子：

在理想情況下比例因子 $K^a$ 是單位矩陣，比例矩陣可以表示為：

$$K^a = \begin{bmatrix} s_x^a & 0 & 0 \\ 0 & s_y^a & 0 \\ 0 & 0 & s_z^a \end{bmatrix} \quad (\text{式 5})$$

#### (三)、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：

$$a^o = K^a(a^s + b^a + v^a) \quad (\text{式 6})$$

$a^o$  : 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 偏差常數矩陣， $v^a$  : 3x1 感應器雜訊矩陣。

(四)、定義加速度計 cost function:

誤差模型中，有 6 個參數需要被確定。參數向量可以表示為:

$$\lambda^a = [s_x^a, s_y^a, s_z^a, b_x^a, b_y^a, b_z^a] \quad (\text{式 7})$$

實際值可通過以下公式計算:

$$a^o = h(a^s, \lambda^a) = K^a(a^s + b^a) \quad (\text{式 8})$$

由於靜態時加速度計所測量的加速度值的大小必須等於重力，因此(式 9)中定義 cost function 表示重力的大小  $\|g\|$  與測量值的差作為誤差大小。

$$L(\lambda^a) = \sum_{k=1}^n (\|g\|^2 - \|h(a_k^s, \lambda^a)\|^2)^2 \quad (\text{式 9})$$

$n$  為採樣窗口中的所做的測量數。

---

### Algorithm 1 感應器校正

---

Require:  $T_{\text{init}}$ (wait before start acquiring data),  $a^s$  (Calibrated Data), **datasetNum** (number of cycles to collect data)

---

```
1: Wait for  $T_{\text{init}}$  seconds
2: for  $i = 0$ : datasetNum
3:   if static == true
4:      $\text{dataset}_i \leftarrow$  average of array of  $a^s$ 
5:   else
6:     retry
7:   end
8: end
9: Params  $\leftarrow$  optimize using dataset and Levenberg – Marquardt algorithm
10:  $a^o \leftarrow$  calibrate  $a^s$  using Params
```

---

## 五、Euler Integration:

Euler Integration 為從加速度計算速度和位置最直接的方法。以下為 Euler Integration 運作流程:

(一)、物體的運動由(式 10)與(式 11)表示:

$$\frac{dx}{dt} = v \quad (\text{式 10})$$

$$\frac{dv}{dt} = a \quad (\text{式 11})$$

已知的初始位置和速度：

$$x(0) = x_0 \quad (\text{式 12})$$

$$v(0) = v_0 \quad (\text{式 13})$$

在本系統中，每當使用者下筆 $x(0)$ 就會被重置，當裝置被偵測靜置時，位移的估計就會以初速度 $v(0) = 0$ 開始。當 $\Delta t$ 足夠小時( $\Delta t$ 感應器採樣間隔)：

$$dt \approx \Delta t \quad (\text{式 14})$$

位置對時間的微分為速度：

$$\frac{dx}{dt} \approx \frac{x(t+\Delta t) - x(t)}{\Delta t} \quad (\text{式 15})$$

速度對時間的微分為加速度：

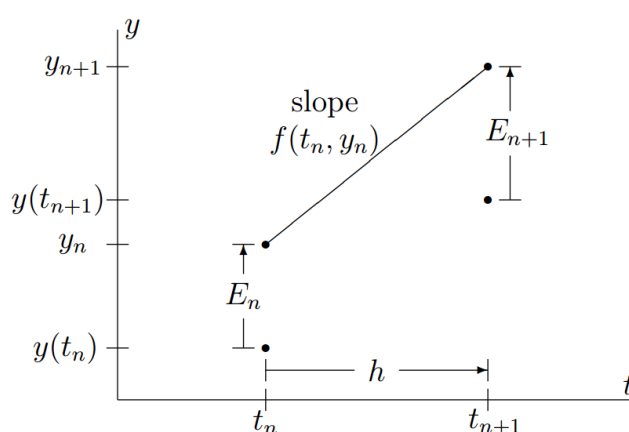
$$\frac{dv}{dt} \approx \frac{v(t+\Delta t) - v(t)}{\Delta t} \quad (\text{式 16})$$

由(式 17)與(式 18)可知從位置與加速度計算速度：

$$x(t + \Delta t) \approx x(t) + v(t)\Delta t \quad (\text{式 17})$$

$$v(t + \Delta t) \approx v(t) + a(t)\Delta t \quad (\text{式 18})$$

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



(圖 4)、一次歐拉積分過程

下式給出由 Euler Integration 於 $y_0$ 估計 $y_1$ 之值：

$$y_1 = y_0 + hf(t_0, y_0) \quad (\text{式 19})$$

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

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

$$y(t_0 + h) = y(t_0) + hy'(t_0) + \frac{1}{2}h^2y''(t_0) + O(h^3) \quad (\text{式 20})$$

此時將  $y(t_0 + h)$  減去 Euler Method 的估計值  $y_0$  得到局部截尾誤差，由 (式 21) 所示：

$$\text{LTE} = y(t_0 + h) - y_1 = \frac{1}{2}h^2y''(t_0) + O(h^3) \quad (\text{式 21})$$

由上式可知，當  $h$  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),  $\mathbf{a}$  (measured acceleration),  $\mathbf{dt}$  (time elapsed since previous measurement),  $\mathbf{SD}$  (static detector status)

---

```

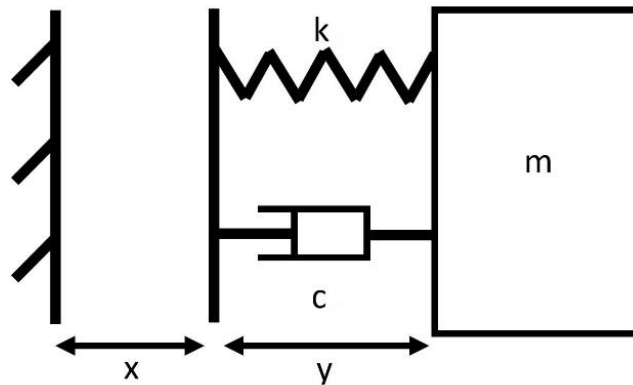
1:  $\mathbf{v_p} += \mathbf{a} * \mathbf{dt}$ 
2:  $\mathbf{p_p} += \mathbf{v_p} * \mathbf{dt}$ 
3: if  $\mathbf{Static} == \text{true}$ 
4:   set  $\mathbf{v_p}$  and  $\mathbf{a}$  zero
5: end

```

---

## 六、Mass-Spring-Damper Model:

Mass-Spring-Damper Model 是個由彈簧及阻尼構成的物理模型，其靈感來自 Mass-Spring-Damper System。這樣的系統與螢幕在手上搖晃時的行為非常相似<sup>[5]</sup>，此模型將螢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系統受力，其中與螢幕平行的兩軸的彈簧、阻尼是相互獨立的，(圖 5)顯示系統在一維的情況。



(圖 5)、一維 Spring-mass-damper physical model

系統的行為依賴以下阻尼比：

$$\zeta = \frac{c}{2\sqrt{km}} \quad (\text{式 22})$$

當  $\zeta < 1$ ，系統呈現欠阻尼狀態，因此不斷來回震盪；當  $\zeta > 1$ ，系統呈現過阻尼狀態，這會使系統花費太多時間回到平衡狀態；當設  $\zeta = 1$ ，系統呈現臨界阻尼狀態，這是最佳的情況，也是此方法的設定。此時  $c = 2\sqrt{km}$ ， $\zeta = 1$ 。臨界阻尼系統方程式可表示為：

$$\ddot{y} + 2\sqrt{km}\dot{y} + ky = -A(t) \quad (\text{式 23})$$

透過將  $-A(t)$  與系統的 Impulse Response  $H(t)$  計算卷積可以得到位移：

$$Y(t) = H(t) * -A(t) \quad (\text{式 24})$$

其中臨界阻尼狀態的 Mass-Spring-Damper System 的 Impulse Response  $H(t)$  為：

$$H(t) = te^{-t\sqrt{k}} \quad (\text{式 25})$$

### Algorithm 3 Mass-Spring-Damper Model

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

```

1: for i = 0: bufferSize
2:   t ← (1 -  $\frac{i}{\text{bufferSize}}$ ) * sampleDelay;
3:   impulseResponse ←  $te^{-t\sqrt{k}}$ 
4:   sum += (mBuffer[i] * impulseResponse); //convolve buffer with impulseResponse
5: end
6: return sum

```

## 七、Runge-Kutta Integration:

一階 Euler Integration 在加速度對時間二次積分後會產生很大的偏差，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。Runge-Kutta Integration 的過程需要下列方程式：

$$y_{n+1} = y_n + \frac{\Delta t}{6} (k_1 + 2k_2 + 2k_3 + k_4) \quad (\text{式 26})$$

其中

$$\frac{dy}{dt} = f \quad (\text{式 27})$$

$$k_1 = f(t_n, y_n) \quad (\text{式 28})$$

$$k_2 = f(t_n + \frac{\Delta t}{2}, y_n + \frac{\Delta t}{2} k_1) \quad (\text{式 29})$$

$$k_3 = f(t_n + \frac{\Delta t}{2}, y_n + \frac{\Delta t}{2} k_2) \quad (\text{式 30})$$

$$k_4 = f(t_n + \Delta t, y_n + \Delta t k_3) \quad (\text{式 31})$$

y可以表示為位置或是速度。RK4 方法的局部截尾誤差(Local truncation error, LTE) 為 $O(h^5)$ ，因此當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),  $\mathbf{a}$  (measured acceleration),  $\mathbf{dt}$  (time elapsed since previous measurement),  $\mathbf{SD}$  (static detector status)

---

```

1: //RK4 divides the integration process into 4 steps , 4 positions.
2:  $\mathbf{V}_1$ (velocity at position  $\mathbf{X}_1$ )  $\leftarrow \mathbf{v}_p + \mathbf{0} * \mathbf{0}$            //the 1st point at  $t = 0$ 
3:  $\mathbf{V}_2$ (velocity at position  $\mathbf{X}_2$ )  $\leftarrow \mathbf{v}_p + \mathbf{V}_1 * \mathbf{dt} * 0.5$    //the 2nd point at  $t = 0.5 * \mathbf{dt}$ 
4:  $\mathbf{V}_3$ (velocity at position  $\mathbf{X}_3$ )  $\leftarrow \mathbf{v}_p + \mathbf{V}_2 * \mathbf{dt} * 0.5$    //the 3st point at  $t = 0.5 * \mathbf{dt}$ 
5:  $\mathbf{V}_4$ (velocity at position  $\mathbf{X}_4$ )  $\leftarrow \mathbf{v}_p + \mathbf{V}_3 * \mathbf{dt}$          //the 4th point at  $t = \mathbf{dt}$ 

6:  $\mathbf{V}_{all} \leftarrow \frac{1}{6}(\mathbf{V}_1 + 2\mathbf{V}_2 + 2\mathbf{V}_3 + \mathbf{V}_4)$ 

7:  $\mathbf{p}_p += \mathbf{V}_{all} * \mathbf{dt};$ 
8:  $\mathbf{v}_p += \mathbf{a} * \mathbf{dt};$ 
9: return  $\mathbf{p}_p$ 

```

---

## 八、Exponential smoothing 濾波器：

### (一)、低通濾波器：

此處使用一階指數平滑公式作低通濾波器，達到保留低頻信號的效果，(式 32)是一階指數平滑化公式。 $X_t$ 為輸入數據， $S_t$ 為輸出數據：

$$S_t = \alpha \cdot X_t + (1 - \alpha) \cdot X_{t-1} \quad (\text{式 32})$$

### (二)、高通濾波器：

可將原始信號減去低頻部分便得到高通濾波的效果：

$$S_t = X_t - (\alpha \cdot X_t + (1 - \alpha) \cdot X_{t-1}) \quad (\text{式 } 33)$$

$X_t$ 為輸入數據， $S_t$ 為輸出數據，平滑常數 $\alpha$ 皆透過實驗以確定最佳值。

## 九、卡爾曼濾波器：

卡爾曼濾波器也稱線性二次估計(Linear Quadratic Estimation, LQE)是一個利用過去的測量(包含統計雜訊及其他不精確因素等等)估計未知變量的方法，卡爾曼濾波器所得數值通常比測量值還要精準，因此被本系統用作裝置位移的預測。此外，還有一個優點是由於其屬於遞歸濾波器，因此不需要紀錄歷史資料。

## 十、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  $t$  的變異大小透過以下計算：

$$\zeta(t) = \sqrt{[VAR_{t_w}(a_x^t)]^2 + [VAR_{t_w}(a_y^t)]^2 + [VAR_{t_w}(a_z^t)]^2} \quad (\text{式 } 34)$$

$\zeta(t)$ 為變異幅度。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。閾值則由實驗來確定。

---

### Algorithm 5 靜置感應器

---

**Require:** W (3-axis data observed in a rectangular window of  $t_w$  seconds at time  $t$ ), T (static time required to trigger static detector), threshold

---

```

1: for  $t = t_0 : t_n$ 
2:   calculate the variance of data in the windows:  $var_x(t), var_y(t), var_z(t)$ 
3:   if  $\sqrt{[var_x(t)]^2 + [var_y(t)]^2 + [var_z(t)]^2} > threshold$ 
4:     static_start_time  $\leftarrow t_n$ 
5:   end
6:   if  $t_n - t_0 > 1 \text{ sec}$ 
7:     staticdetector.trigger()
8:   end
9: end

```

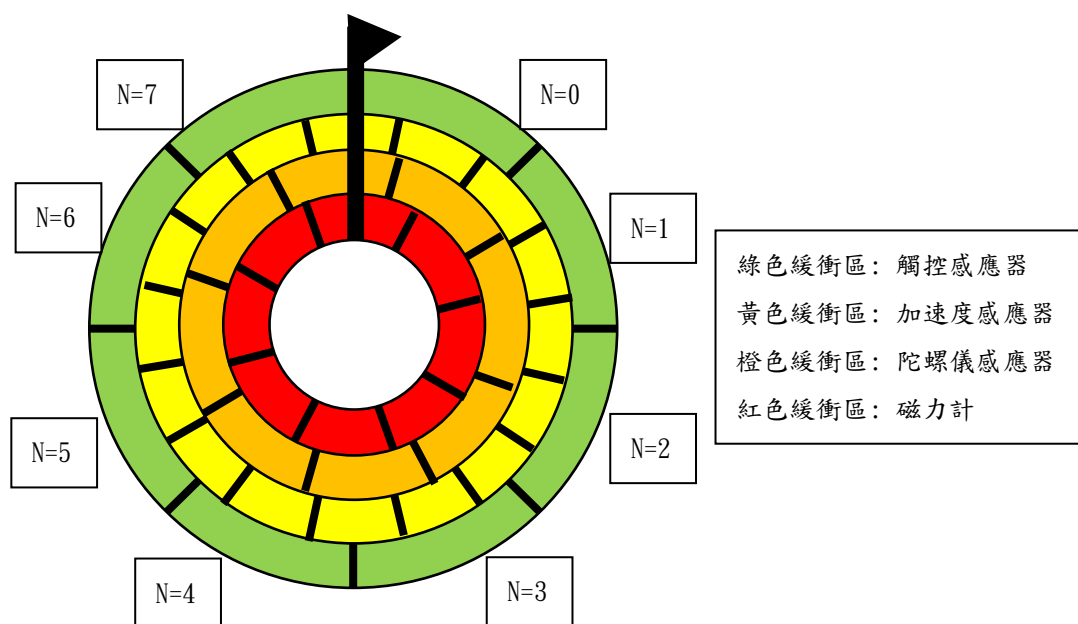
---

## 十一、感應器時間戳同步：

由於裝置的不同感應器有著各自的更新速率及更新時間，因此為了使系統運算時各感應器數據的時間戳同步，本系統編寫共同環形緩衝區(Ring Buffer)統一處理數據並以樣條插植法，使所有感應器能依需求進行時間戳的對齊。

### (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，會剔除舊數據以維持緩衝區大小不變，共同緩衝區除了使數據的抽取更加方便，也更容易的進行插值或套用窗函數(Window Function)，共同環形緩衝區運作之示意圖如(圖六)。

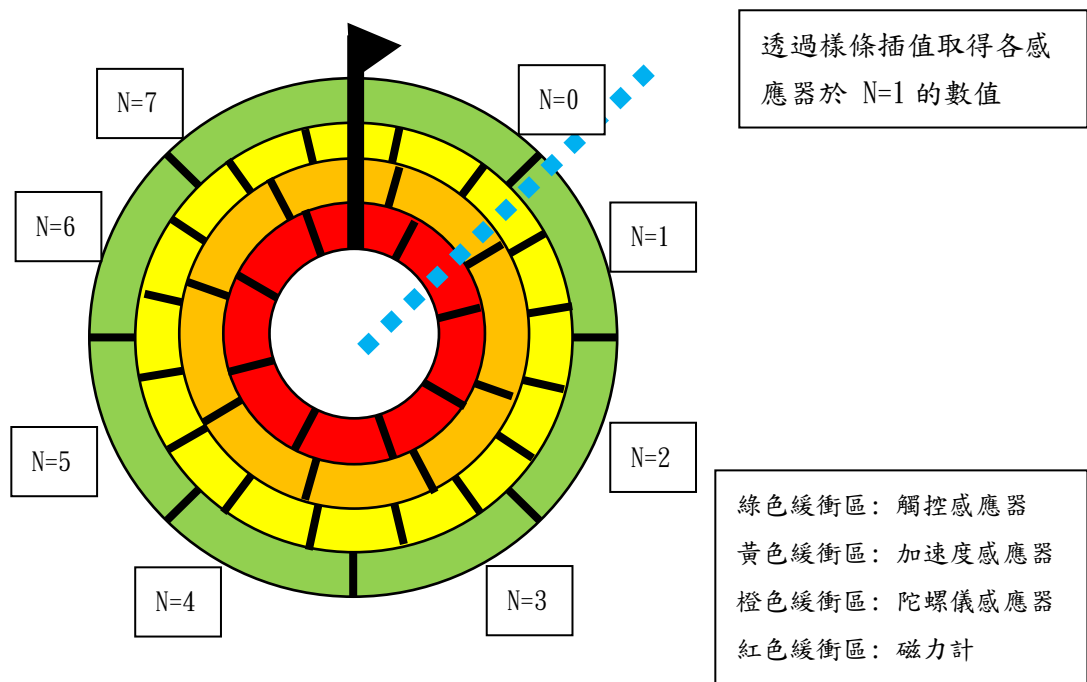


(圖 6)、共同環形緩衝區

### (二)、樣條插植：

移動估計與進行變形修正的過程中需要進行各感應器數據時間戳的對齊。這些感應器包含加速度計、陀螺儀、磁力計及觸控螢幕。此對齊的過程需要利用共同環形緩衝區與樣條插值方法實現，本系統在時間戳同步的過程中採用的插值法為三次樣條插值(Cubic Spline Interpolation)，此插值法的優點是不會發生抖動(龍格現象)。





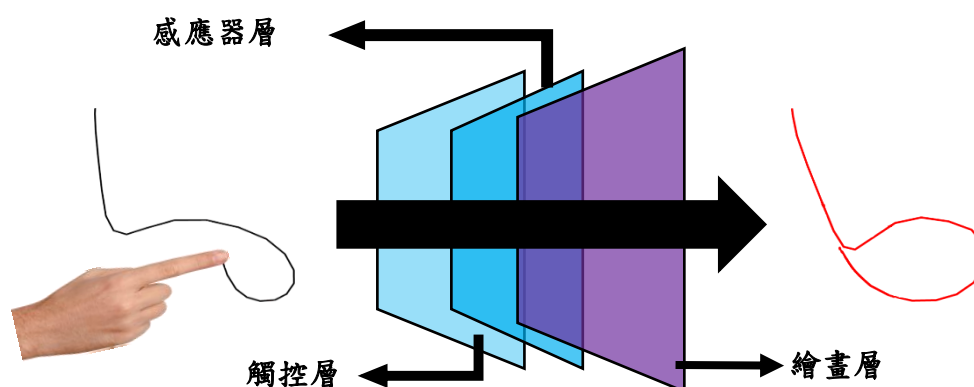
(圖 7)、環形緩衝區中進行樣條插值

## 十二、異常偵測：

本研究所開發的系統聚焦於裝置小範圍移動的常見狀況，因此將部分特殊狀態視為使用者的意識行為。以下情況被視為特殊狀態：

- |         |    |                 |
|---------|----|-----------------|
| (一)、加速度 | >  | $4\text{m/s}^2$ |
| (二)、速度  | >  | $1\text{m/s}$   |
| (三)、位移量 | >  | $4\text{cm}$    |
| (四)、加速度 | 同號 | 超過 1 秒          |

### 十三、手指-螢幕互動模型：



(圖 8)、手指-螢幕互動模型

(圖 8)顯示本研究提出的手指-螢幕互動模型中包含三部分：

#### (一)、繪畫層：

繪畫層是固定於世界座標上的一張畫布，而使用者意圖在這張畫布上寫下筆跡，是本系統變形修正後的最終目標。這張畫布在空間中的位置及角度在下筆的第一瞬間被固定，並且在同一筆畫中不會有任何改變。

#### (二)、感應器層：

感應器層是連結繪畫層及觸控層之間的橋樑。本系統利用裝置內建的加速度計、陀螺儀以及磁力計計算感應器層平面與世界座標之間的相對位置與角度，感應器層包含在現代智慧型裝置普遍內置的感應器。這些感應器皆是微機電(MEMS)製程，而建立於感應層的移動估計系統屬於 strapdown system，因此感應器層與觸控層始終保持相互平行固定。

#### (三)、觸控層：

觸控層是本系統與手指直接物理接觸的媒介，它的記錄包括觸控位置及觸控壓力大小的資訊，此層與感應器層也始終保持相互平行固定。

#### 十四、筆跡修正演算法(第一代):

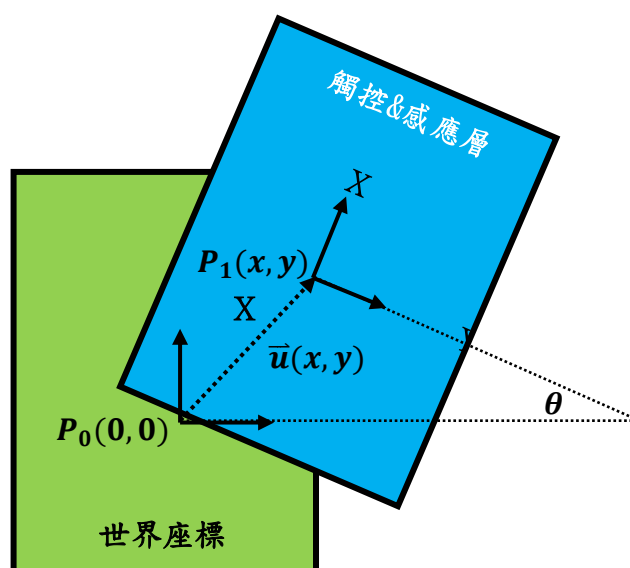
第一代的手指-螢幕互動模型中，所有層都相互平行於地面。

##### (一)、第一代考慮的手指-螢幕互動模型:

繪畫層: 使用者意圖繪畫的平面，被固定平行於地表。

感應器層: 裝有加速度計、陀螺儀及磁力計。平行於地表，會隨著裝置平移旋轉但局限於2維平面上。

觸控層: 用來取得觸控感應點數據，且始終與感應器層相連。



(圖 9)、第一代手指-螢幕互動模型

##### (二)、將觸控層映射到繪畫層:

- 1、設定下筆當下，所有層座標軸平行，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 $\theta$ 。
- 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)表示。

(1)、設使用者於繪畫層意圖的筆跡位置向量：

$$\vec{P} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (\text{式 35})$$

(2)、設 $T'$ 轉到繪畫層座標系之向量，其中可以知道：

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

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

$$\vec{T'} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \vec{T} \quad (\text{式 37})$$

即可得到：

$$\vec{P} = \vec{S} + \vec{T'} = \vec{S} + \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \vec{T} = \begin{bmatrix} X_s \\ Y_s \\ 1 \end{bmatrix} + \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_t \\ Y_t \\ 1 \end{bmatrix} = \begin{bmatrix} X_s + X_t \cos(\theta) - Y_t \sin(\theta) \\ Y_s + Y_t \cos(\theta) + X_t \sin(\theta) \\ 1 \end{bmatrix} \quad (\text{式 38})$$

6、將繪畫層筆跡顯示在螢幕上。

## 十五、筆跡修正演算法(第二代)：

在第二代的更新中，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，使使用者在移動載具上也能使用本系統，去除了非意料之外的加速度數值。第二代也解除了三層(觸控層、感應器層以及繪畫層)相互平行的假設，完整地利用三軸加速度資訊對筆跡進行三維空間的線性變換。最後，第二代具有同步修正功能，使系統效能大幅提升，讓使用者能在書寫的同時能及時查看修正成果，並使筆跡持續與手指位置相連，使用者的感受更加簡單實用。

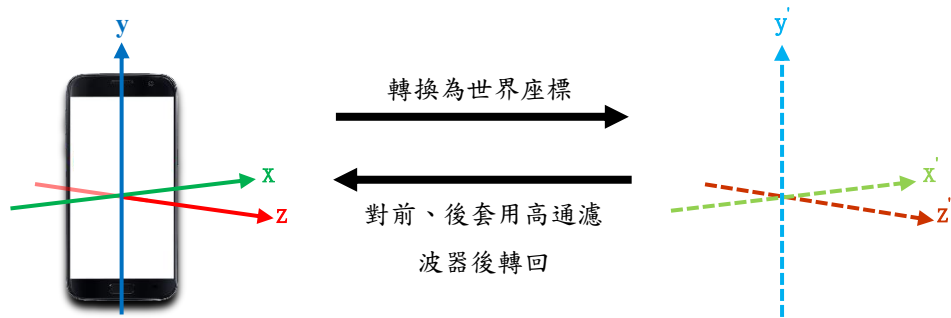
(一)、第二代的手指-螢幕互動模型：

繪畫層：使用者意圖繪畫的層，由下筆瞬間被確定，始終固定。

感應器層：裝有加速度計、陀螺儀及磁力計。初始時原點重疊、座標軸平行，隨裝置平移旋轉。

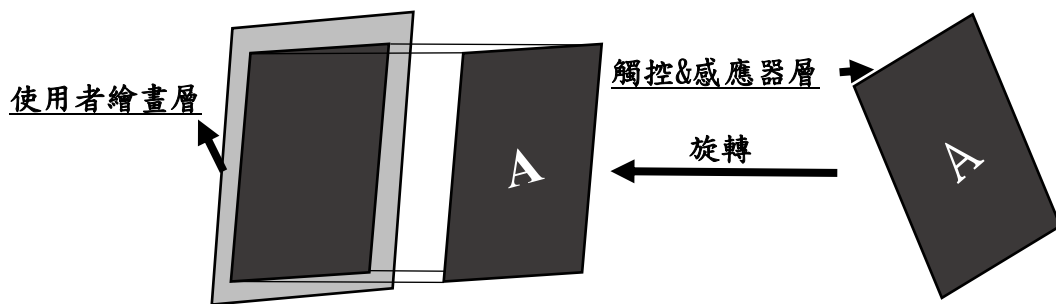
觸控層：用以取得觸控感應點，始終與感應器層相連。

(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：



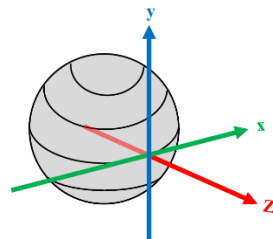
(圖 10)、對移動載具加速度方向相同的分量(前、後)套用高通濾波器

(三)、將觸控層映射到繪畫層：



(圖 11)、進行三維觸控點的線性變換

- 1、下筆當下所有層座標軸平行共用原點(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、為了特定情形下環架鎖定(Gimbal lock)的發生，這裡的所有旋轉皆以四元數(Quaternion)操作，下筆瞬間所有層的絕對方向  $q_0$ ，下一觸控點絕對方向  $q_1$ 。
- 5、這裡 Quaternion 中的旋轉向量座標如下定義：



(圖 12)、世界座標的定義(X 軸:Y 與 Z 軸之外積方向( $Y \times Z$ ), 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：

$$\vec{P} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \quad (\text{式 39})$$

(2)、設 $T'$ 轉到繪畫層座標系之向量，其中：

$$\vec{P} = \vec{S} + \vec{T'} \quad (\text{式 40})$$

(3)、設定初始及下一觸控點之 Quaternion：

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

$$q_1 = q_1[0] + q_1[1]i + q_1[2]j + q_1[3]k \quad (\text{式 42})$$

(4)、取兩四元數之差：

$$q_{01} = q_1 q_0^{-1} = q_1 \frac{q_0^*}{\|q_0\|^2} \quad (\text{式 43})$$

(5)、因為從 Android Rotation Vector 取出之四元數的範數皆為 1，故：

$$q_{01} = q_1 q_0^* \quad (\text{式 44})$$

其中：

$$q_0^* = q_0[0] - q_0[1]i - q_0[2]j - q_0[3]k \quad (\text{式 45})$$

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

$$\vec{T'} = q_{01} \vec{T} q_{01}^{-1} \quad (\text{式 46})$$

用旋轉矩陣的形式表示：

$$\vec{T'} = \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} (1 - 2q_{01}[2]^2 - 2q_{01}[3]^2) & 2(q_{01}[1]q_{01}[2] + q_{01}[0]q_{01}[3]) & 2(q_{01}[1]q_{01}[3] - q_{01}[0]q_{01}[2]) \\ 2(q_{01}[1]q_{01}[2] - q_{01}[0]q_{01}[3]) & (1 - 2q_{01}[1]^2 - 2q_{01}[3]^2) & 2(q_{01}[2]q_{01}[3] + q_{01}[0]q_{01}[1]) \\ 2(q_{01}[1]q_{01}[3] + q_{01}[0]q_{01}[2]) & 2(q_{01}[2]q_{01}[3] - q_{01}[0]q_{01}[1]) & (1 - 2q_{01}[1]^2 - 2q_{01}[2]^2) \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \quad (\text{式 47})$$

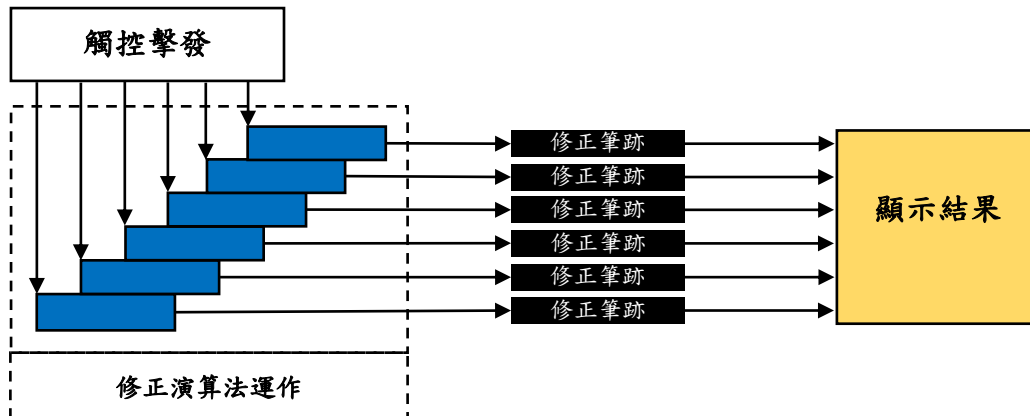
(7)、最後得到：

$$\vec{P} = \begin{bmatrix} X_s \\ Y_s \\ 0 \end{bmatrix} + \begin{bmatrix} (1 - 2q_{01}[2]^2 - 2q_{01}[3]^2) & 2(q_{01}[1]q_{01}[2] + q_{01}[0]q_{01}[3]) & 2(q_{01}[1]q_{01}[3] - q_{01}[0]q_{01}[2]) \\ 2(q_{01}[1]q_{01}[2] - q_{01}[0]q_{01}[3]) & (1 - 2q_{01}[1]^2 - 2q_{01}[3]^2) & 2(q_{01}[2]q_{01}[3] + q_{01}[0]q_{01}[1]) \\ 2(q_{01}[1]q_{01}[3] + q_{01}[0]q_{01}[2]) & 2(q_{01}[2]q_{01}[3] - q_{01}[0]q_{01}[1]) & (1 - 2q_{01}[1]^2 - 2q_{01}[2]^2) \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \quad (\text{式 48})$$

(8)、將繪畫層筆跡顯示在螢幕上

(四)、改採用同步修正功能：

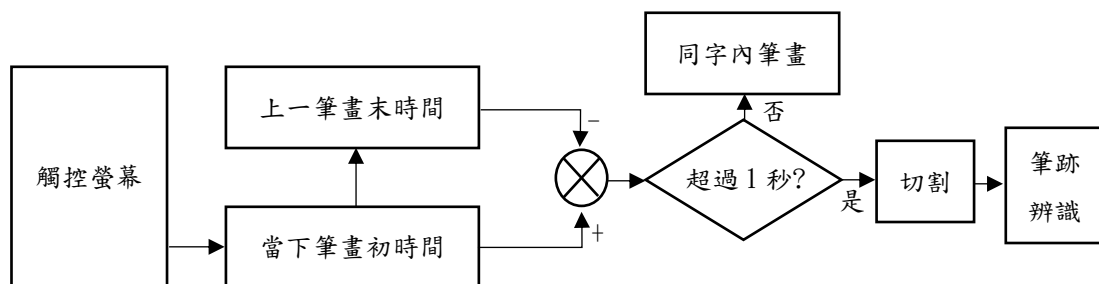
第二代利用多執行序處理使處理速度、介面反應速度皆大幅提升。



(圖 13)、多執行序處理

(五)、同個字內筆畫偵測：

本系統設計基於筆畫時間差機制，切割後群組屬於同個字之筆畫，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...)，。



(圖 14)、同字內筆畫偵測流程圖

(六)、使筆跡持續跟隨手指位置：

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連。此有助於讓使用者了解最新修正狀況並提升使用效果、且更加直覺易用。



(圖 15)、第一、二代顯示方式比較 (左:第一代、右:第二代)

## 十六、筆跡修正演算法(第三代):

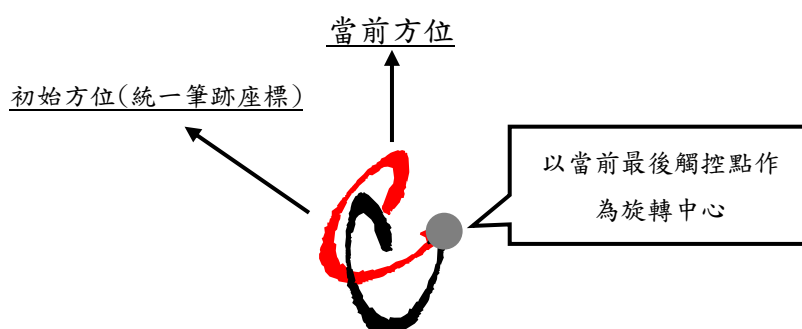
第三代的修正算法以第二代演算法為基礎，進行大量的性能優化，同時加入筆跡平滑化使修正結果更加美觀。

### (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，而非如前兩代在每一次重繪事件中重新運算全畫面的筆跡修正結果。此全新的累加演算法主要內容如下:

#### 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 16)、將筆跡旋轉至當前方位

旋轉裝置時，系統持續保持筆跡與使用者相對位置正確，有利於實際使用時的直覺與方便性。在筆跡結束時，系統自動將筆跡轉正以利系統辨識。

#### 2、統一筆跡座標:

為了使同一筆畫中的所有觸控點座標統一，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，方向符合統一座標之要求，於下筆瞬間設定初始絕對方向 $q_0$  (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連接

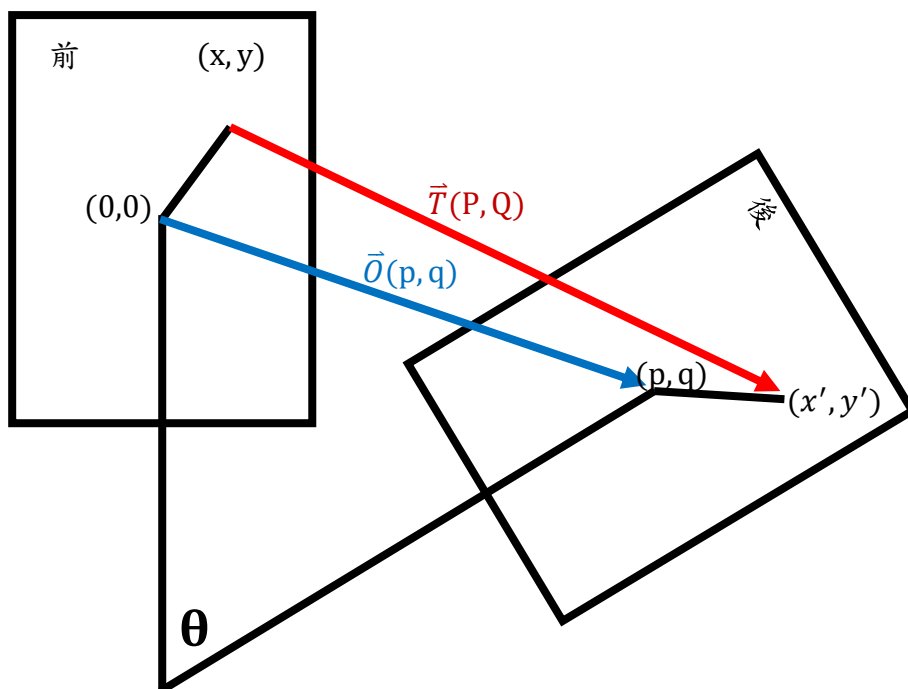


至上一筆跡點得到新筆跡座標。此修正的作用在於將因旋轉變成曲線的直線進行曲率修正、拉直。

### 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、陀螺儀及磁力計)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 $\theta$ 度，其旋轉矩陣 $R$ 。



(圖 17)、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 $(x',y')$ 。

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \vec{O} + R \begin{bmatrix} x \\ y \end{bmatrix} \quad (\text{式 49})$$

- (3)、整理上式得知正確位移 $\vec{T}$ 。

$$\vec{T} = \begin{bmatrix} P \\ Q \end{bmatrix} = \begin{bmatrix} x' \\ y' \end{bmatrix} - \begin{bmatrix} x \\ y \end{bmatrix} = \vec{O} + R \begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} x \\ y \end{bmatrix} \quad (\text{式 50})$$

### (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)平滑化，結果更加美觀。

## 十七、手寫字跡辨識：

本研究的手寫筆跡辨識部分，採用了開源的 Lipitoolkit 4.0，此 Lipitoolkit 由 hp lab 研發，美國麻省理工學院(Massachusetts Institute of Technology, MIT)授權。本系統使用其預設提供的英文字訓練資料，搭配 Android Native Interface 使其在 Android 裝置上執行。以下說明此工具包使用要點：

### (一)、輸入/輸出內容：

1、輸入的是一個筆畫陣列，其中筆畫定義為使用者一次輸入的連續觸控點，必須以輸入的時間順序排序。

2、輸出三個信心度由高至低排列的三個候選字。其中個別包含信心度數值。

### (二)、特性：

1、平均辨識一字母需要花費約 0.1 秒。

2、可以接受多筆畫字母的辨識(如 A、D、E、F...)。

3、工具包本身使用 C++ 語言編寫。

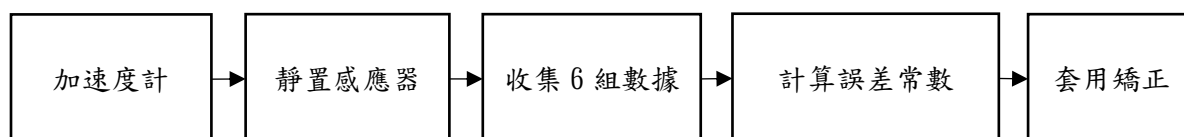
### (三)、使用限制：

1、一次只能夠辨識一個字母。

2、辨識集為英文 26 個字母及 10 個數字(可以透過所提供的工具擴展到其他語言)。

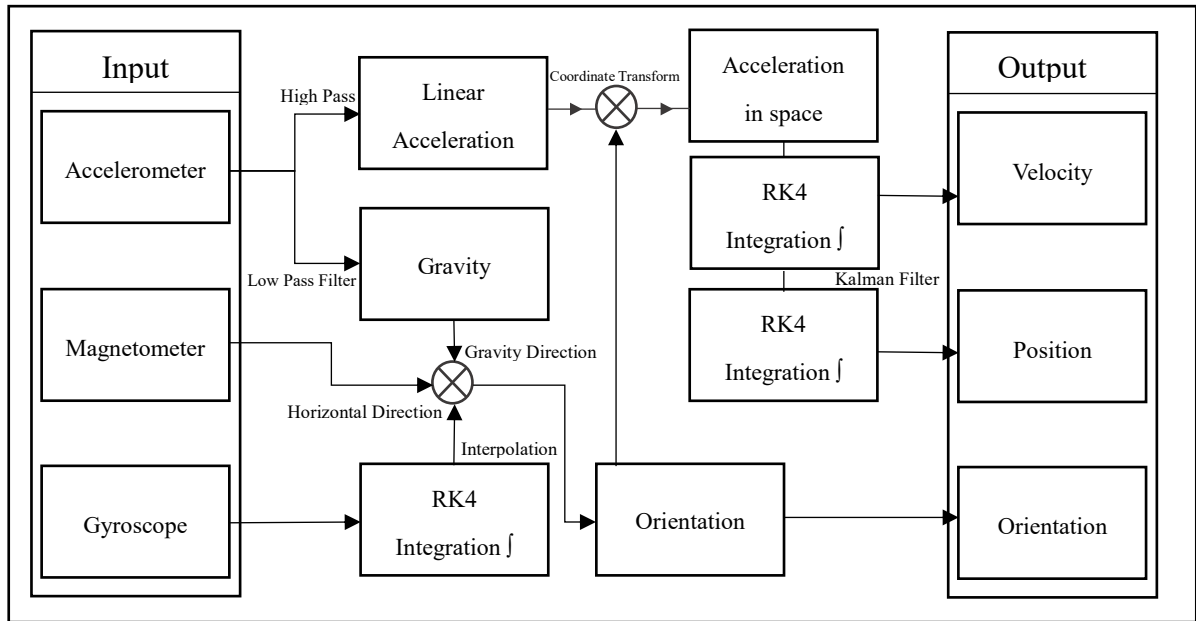
## 十八、完整流程：

### (一)、感應器矯正：



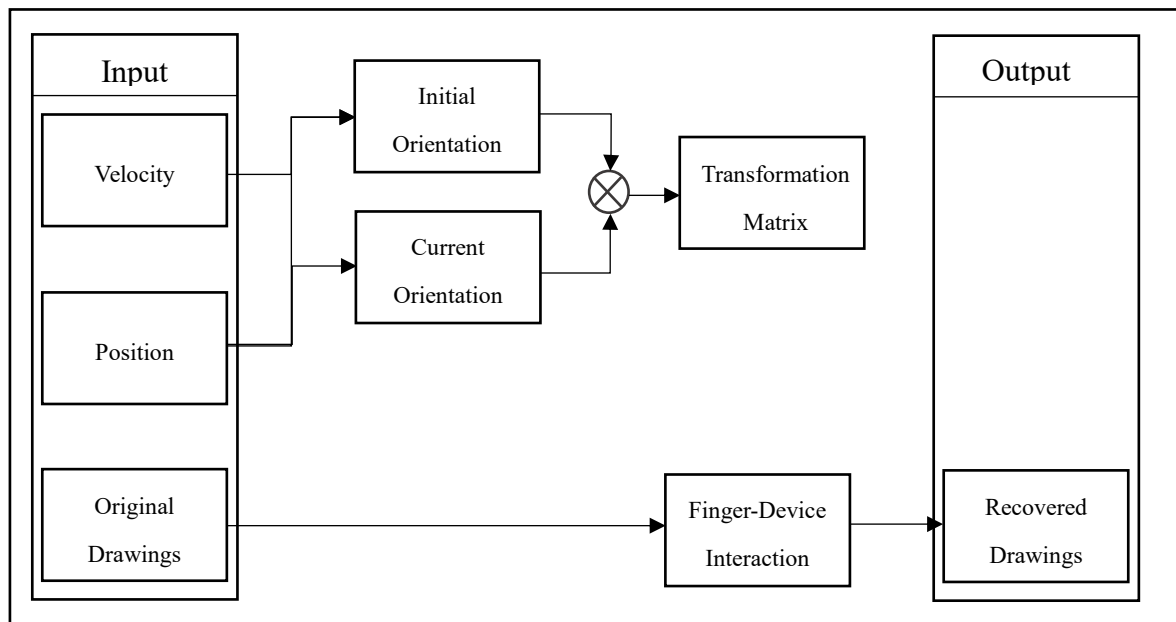
(圖 18)、感應器矯正流程圖

(二)、移動估計模組：



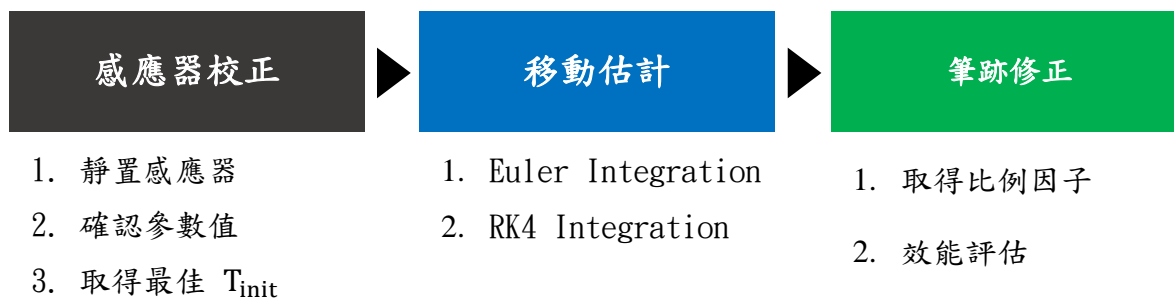
(圖 19)、移動估計模組流程圖

(三)、筆跡變形修正模組：



(圖 20)、筆跡變形修正模組流程圖

## 伍、研究結果及討論



(圖 21)、實驗流程

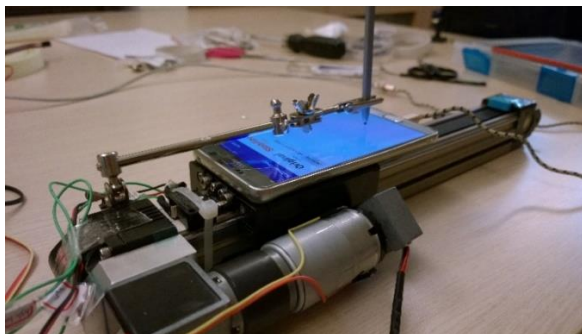
### 一、感應器校正：

#### (一)、調整靜置感應器閾值：

靜置感應器的觸發是由觀察窗內三軸加速度數值變異數數值是否小於一個給定的閾值決定。這個閾值若太高，靜置感應器不容易被觸發；相反的，若閾值太低，靜置感應器則太容易被觸發。本實驗透過嘗試不同的閾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：

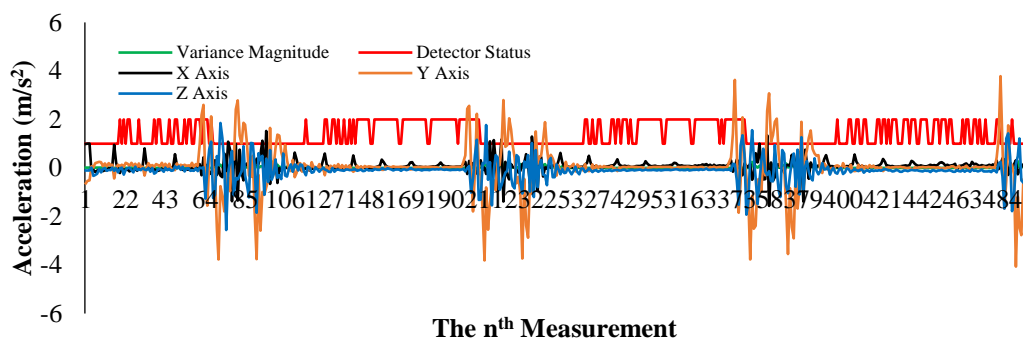
- 1、裝置被靜置於平台上。
- 2、裝置被週期運動產生器移動。

週期運動產生器產生向前 2cm，向後 2cm 的週期運動，在移動間隔中會停止 1 秒鐘，此時裝置成靜止狀態。本實驗測試將閾值設為 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。



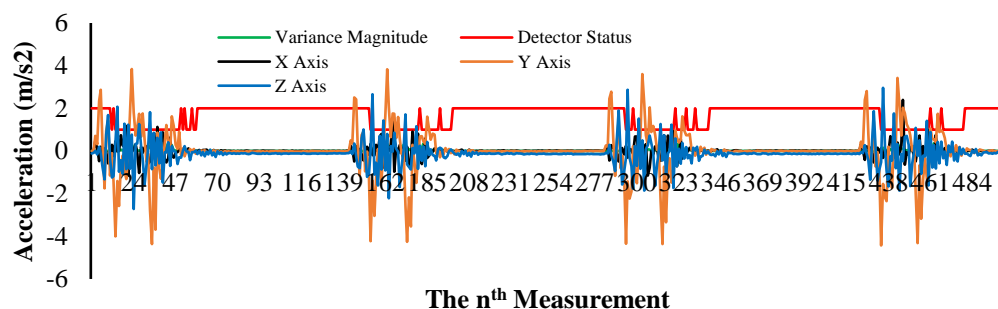
(圖 22)、實驗設置

當閾值設為 0.0004 的輸出圖：



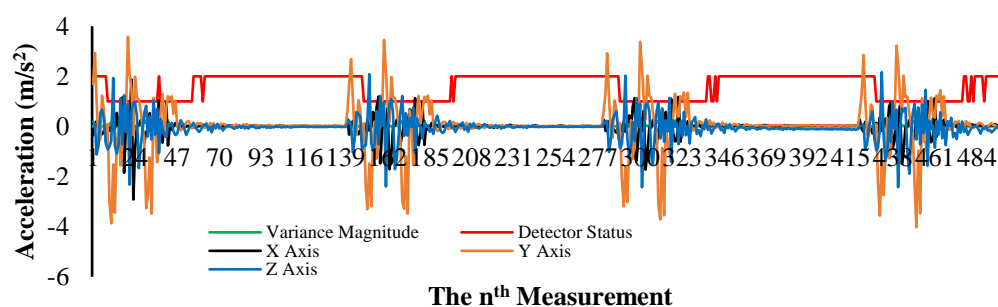
(圖 23)、靜置感應器閾值=0.0004

(圖 23)顯示如果三軸變異數小於 0.0004，靜置感應器輸出 2;如果三軸變異數大於 0.0004，靜置感應器輸出 1。當靜置感應器閾值設為 0.0004，感應器顯得太過敏感。舉另一個極端的例子，將靜置感應器閾值設為 0.0005，輸出以下(圖 24)圖形：



(圖 24)、靜置感應器閾值=0.0005

太高的靜置感應器閾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閾值，本實驗測試了 0.0004 到 0.0005，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閾值 0.0017 可以達到最佳的靜止/移動區分性能。



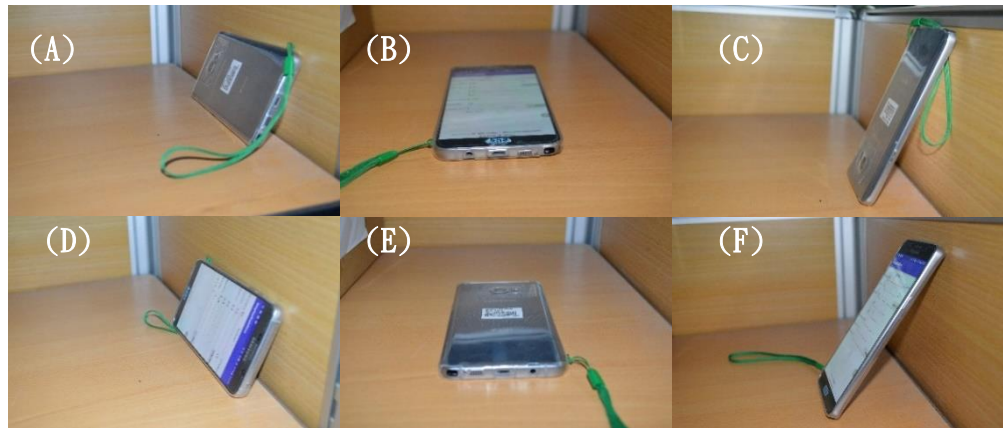
(圖 25)、靜置感應器閾值=0.0017

結果(圖 25)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。以收集 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 cost function，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 26)、將裝置擺放六種姿態

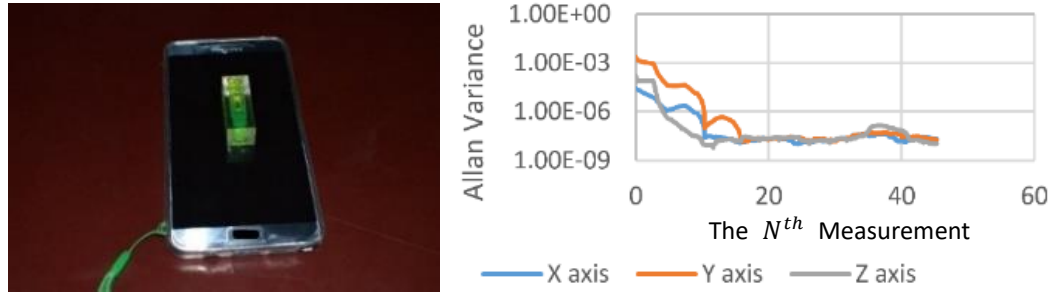
### 2、(表 4)顯示所得校正參數數值：

(表 4)、校正參數數值

Times	Bias_X	Bias_Y	Bias_Z	Scale_X	Scale_Y	Scale_Z
First	0	0	0	1	1	1
Second	0	0	0	0.9999999	1	1
Third	0	0	0	1	1	1
Forth	0	0	0	1	1	1
Fifth	0	0	0	1	0.9999999	1

### (三)、計算 Allan variance 確認 $T_{init}$ :

透過計算 Allan variance,  $T_{init}$  的值可以被確定。當三軸的 Allan 差逐漸縮小到極小值時, 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, 如(圖 27(A))。當程式啟動時, 系統開始以 0.25 秒的觀察窗計算 Allan variance。



(圖 27)、(A)實驗設置, (B) Allan variance 結果

由(圖 27(B))可知;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, 如(圖 28), 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, 使用的為(式 51)與(式 52):

$$\Delta x = -x_t * R_{tm} \quad (\text{式 51})$$

$$\Delta y = -y_t * R_{tm} \quad (\text{式 52})$$

其中  $R_{tm}$  是螢幕像素對應的長度,  $x_t$  是 x 軸上觸控筆在螢幕上的位移,  $y_t$  是 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 28)、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：

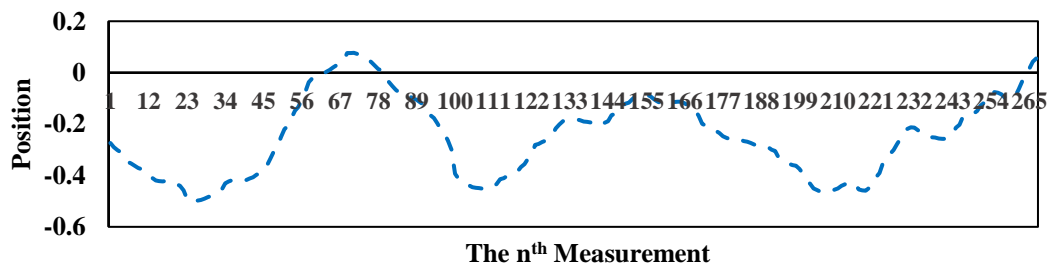
$$\text{Cosine similarity} = \cos\theta = \frac{A \cdot B}{\|A\| \|B\|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}} \quad (\text{式 } 53)$$

$A_i$  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，使用內建感應器重塑移動曲線形狀。

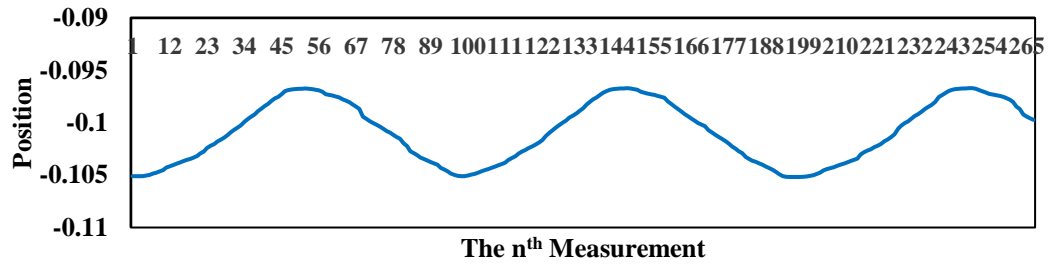
## (二)、使用 RK4 方法進行移動估計：

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。RK4 組態(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器重塑移動曲線形狀。

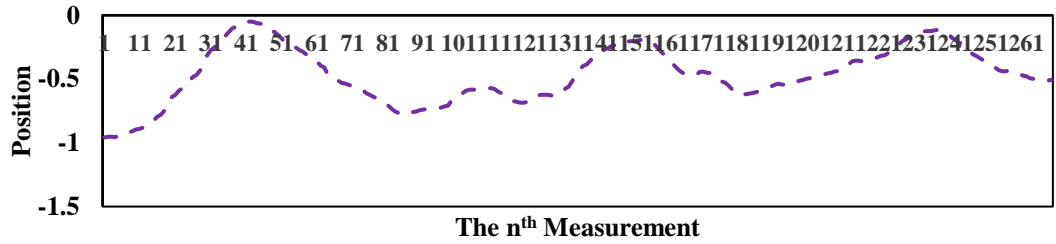


(圖 29)、使用 Euler Integration 進行移動估計

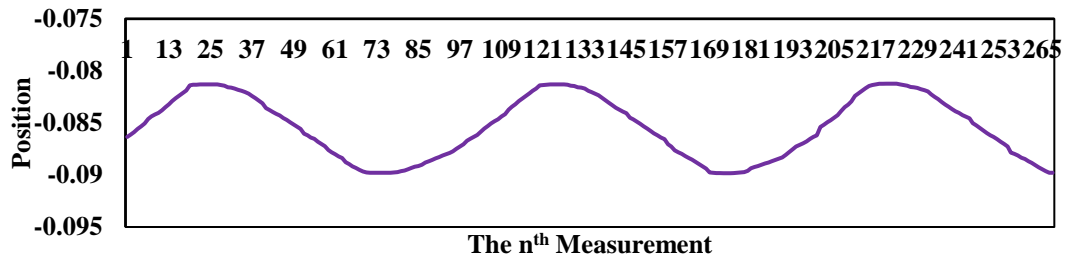




(圖 30)、Euler Integration 實驗中的實際位移



(圖 31)、使用 RK4 方法進行移動估計



(圖 32)、Euler Integration 實驗中的實際位移

### 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。

$$\text{error} = \sqrt{(\text{staX} - \text{ideX})^2 + (\text{staY} - \text{ideY})^2} \quad (\text{式 } 54)$$

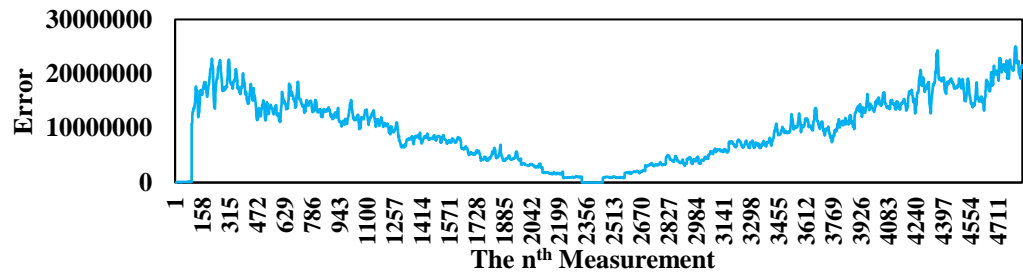
其中

$$\text{staX} = \text{stroX} - \text{posX} * \text{Multiplier} \quad (\text{式 } 55)$$

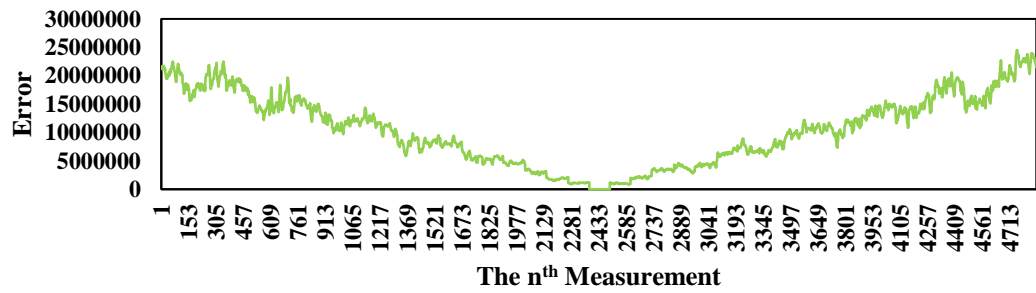
$$\text{staY} = \text{stroY} - \text{posY} * \text{Multiplier} \quad (\text{式 } 56)$$

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，由於在實驗觸控筆在空間中的位置被固定，因以  $ideX$  及  $ideY$  始終為 0。

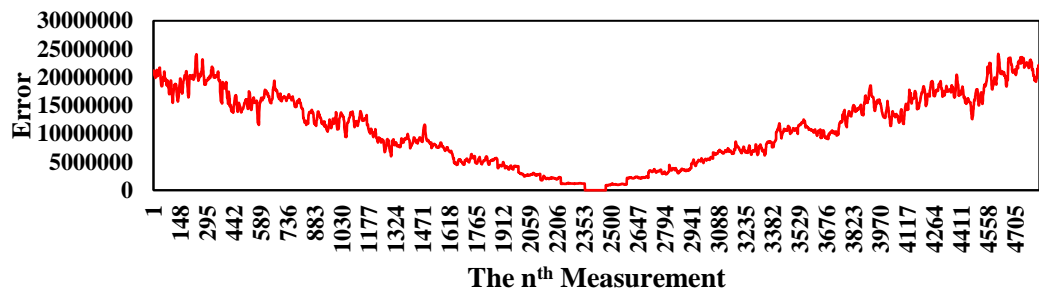
2、調整比例因子從 -1000 到 1000，間隔 50。尋找使誤差最小的比例因子：



(圖 33)、Spring-Mass-Damper 實驗結果

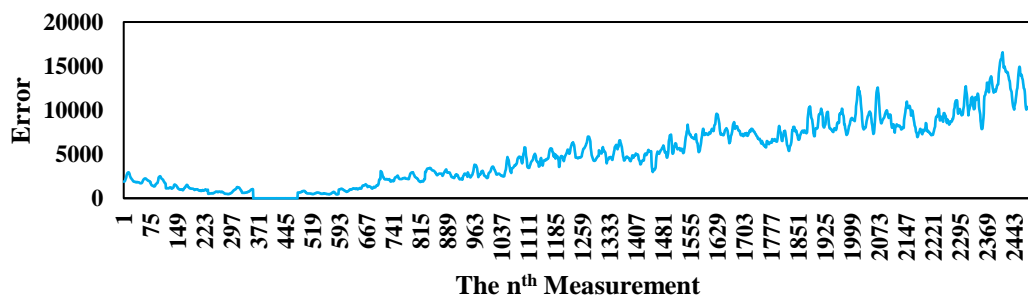


(圖 34)、Euler Integration 實驗結果

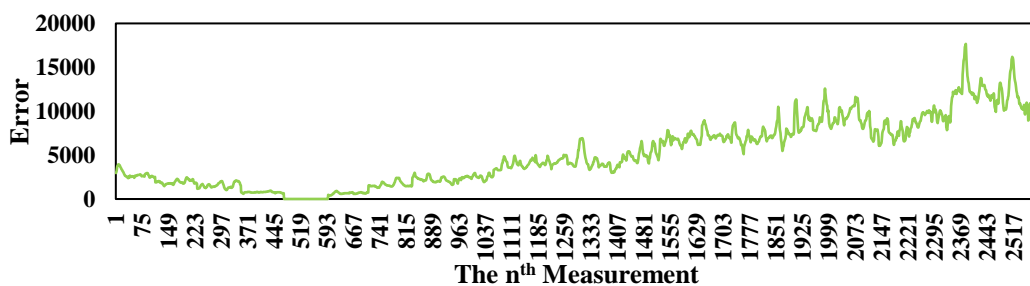


(圖 35)、RK4 Integration 實驗結果

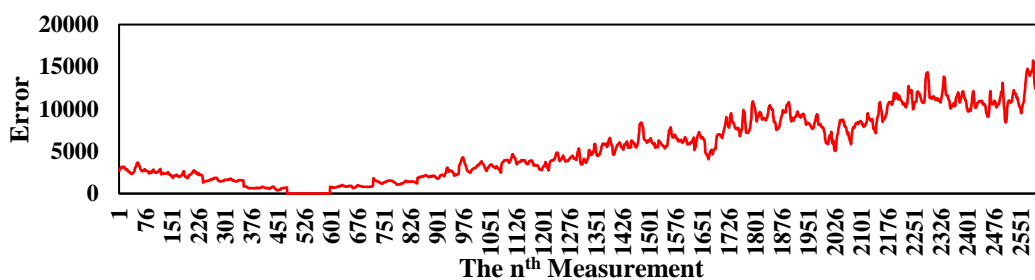
3、調整比例因子從 0.2 到 0.9，間隔 0.05。尋找使誤差最小的比例因子：



(圖 36)、Spring-Mass-Damper 實驗結果

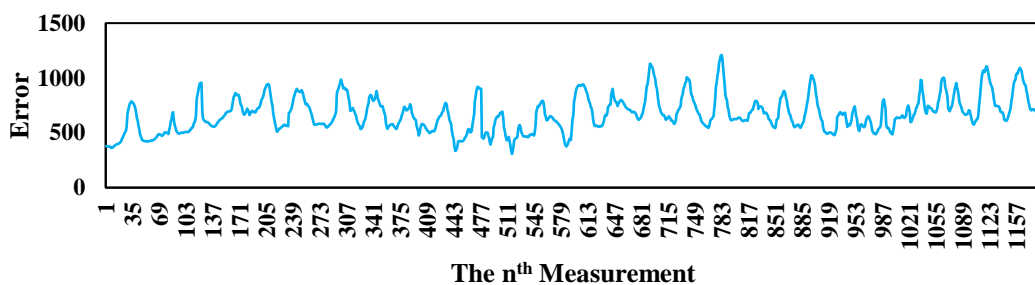


(圖 37)、Euler Integration 實驗結果

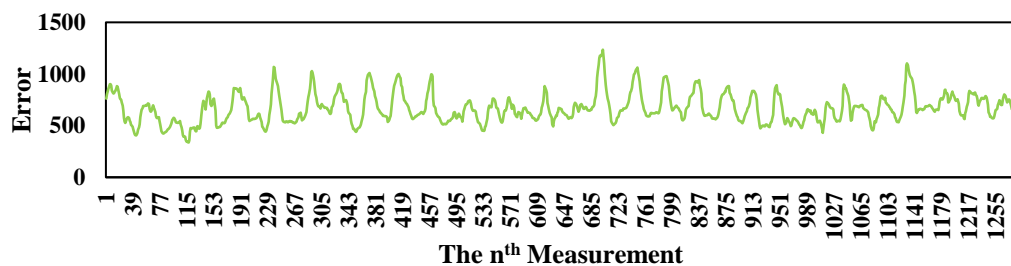


(圖 38)、RK4 Integration 實驗結果

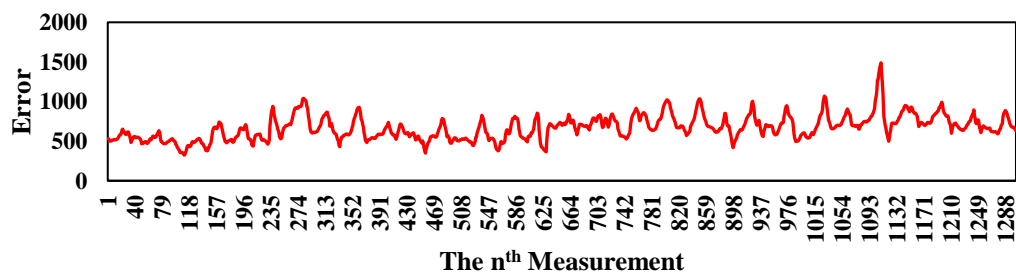
4、調整比例因子從 0.05 到 0.06，間隔 0.001。尋找使誤差最小的比例因子：



(圖 39)、Spring-Damp-Mass Model 實驗結果



(圖 40)、Euler Integration 實驗結果



(圖 41)、RK4 Integration 實驗結果

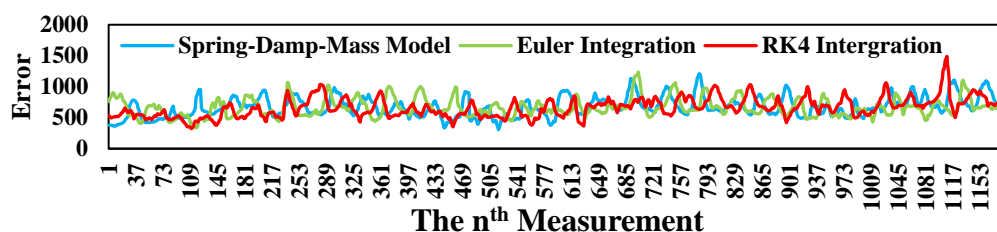
(二)、評估筆跡變形修正系統之效能：

1、比較修正後筆跡與正確筆跡之均方根誤差

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：

$$\text{error} = \sqrt{(\text{sta}X)^2 + (\text{sta}Y)^2} \quad (\text{式 } 57)$$

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 42)、誤差比較

誤差平均值以 RK4 Integration 最低，平均值為 503 像素，與原始筆跡平均誤差 1036 像素(與標準位置(0, 0)比較)，降低約 48.5%的誤差。

(三)、以字元辨識評估本系統修正效能：

為了衡量本系統於日常生活之應用價值，本實驗藉由統計修正前與修正後筆跡對字元辨識的成功率評估本系統修正效能。本實驗個別測試英文字母 a, b, c, d, e，分別於週期運動產生器±2cm 的隨機晃動測試各 100 次，黑色方格表示辨識正確，白色方格表示辨識錯誤。結果如下：

a	修正前		23%
	修正後		51%
b	修正前		11%
	修正後		45%
c	修正前		42%
	修正後		68%
d	修正前		9%
	修正後		31%
e	修正前		18%
	修正後		49%
平均增加28.2%			

(圖 43)、字元辨識修正效能

實驗結果顯示經本系統修正後筆跡辨識成功率平均提高 28.2%，為修正前的 2.34 倍。

## 陸、結論

本研究成功開發筆跡變形修正系統，於實際生活中修正因裝置晃動造成之筆跡變形。系統主要由三個步驟構成：首先，使用者進行感測器校正。接著，本研究提出了變形修正演算法，透過此演算法利用移動估計結果修正變形的筆跡。本研究透過探討不同移動估計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 44)可見經過修正後辨識成功率大幅提高。



(圖 44)、筆跡變形修正系統實際運作效果

實驗結果顯示本系統在 $\pm 2\text{cm}$ 的晃動範圍內，降低約 48.5% 的誤差。字元辨識測試結果顯示在 $\pm 2\text{cm}$ 的隨機晃動下，經本系統修正後筆跡辨識成功率提高 28.2%，為修正前的 2.34 倍。

本系統可套用於任何擁有基本感應器之觸控螢幕裝置上，在提高使用者書寫體驗有巨大的貢獻。本系統可以立即安裝到如智慧車輛、手機、平板等裝置上，對於字跡辨識的成功率與筆跡變形有立即性的顯著改善，因此希望本系統盡快應用於生活。未來也能深入嵌入各書寫軟體中，成為提升書寫體驗不可或缺的基本元件。

本研究將持續改善更新，目標如下：

- 一、透過感應器精確度的提升，擴大晃動消除適用範圍
- 二、改善演算法，使精確度提升、降低資源占用
- 三、編寫 API，使廣大的開發者受惠、進而提升所有觸控螢幕使用者的使用體驗
- 四、加入使用情境辨識功能，偵測使用場景以動態調整修正筆跡修正力度。

## 柒、參考文獻

- [1] Do, T. M. T., Blom, J., & Gatica-Perez, D. (2011). *Smartphone usage in the wild: a large-scale analysis of applications and context*. In Proceedings of the 13th international conference on multimodal interfaces (pp. 353-360). ACM.
- [2] KPCB. (2015). *INTERNET TRENDS 2015 - CODE CONFERENCE*. Retrieved from [www.KPCB.com](http://www.KPCB.com):<http://www.KPCB.com/Internet-trends>
- [3] Hou, H. (2005). *Modeling inertial sensors errors using Allan variance*. Library and Archives Canada/Bibliothèque et Archives Canada.
- [4] Menegatti, L. P. E., Tedaldi, D., & Pretto, A. (2013). *IMU calibration without mechanical equipment*.
- [5] Rahmati, A., Shepard, C., & Zhong, L. (2009). *NoShake: Content stabilization for shaking screens of mobile devices*. In Pervasive Computing and Communications, 2009. PerCom 2009. IEEE International Conference on (pp. 1-6). IEEE.
- [6] Barnard, L., Yi, J. S., Jacko, J. A., & Sears, A. (2005). *An empirical comparison of use-in-motion evaluation scenarios for mobile computing devices*. International Journal of Human-Computer Studies, 62(4), 487-520.
- [7] Mathie, M. J., Coster, A. C. F., Lovell, N. H., & Celler, B. G. (2003). *Detection of daily physical activities using a triaxial accelerometer*. Medical and Biological Engineering and Computing, 41(3), 296-301.
- [8] Goel, M., Findlater, L., & Wobbrock, J. (2012). *WalkType: using accelerometer data to accomodate situational impairments in mobile touch screen text entry*. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (pp. 2687-2696). ACM.

- [9] Syed, Z. F., Aggarwal, P., Goodall, C., Niu, X., & El-Sheimy, N. (2007). *A new multi-position calibration method for MEMS inertial navigation systems*. Measurement Science and Technology, 18(7), 1897.
- [10] Skog, I., & Händel, P. (2006). *Calibration of a MEMS inertial measurement unit*. In XVII IMEKO World Congress (pp. 1-6).
- [11] Shin, E. H., & El-Sheimy, N. (2002). *A new calibration method for strapdown inertial navigation systems*. Z. Vermess, 127, 1-10.
- [12] Ozyagcilar, T. (2012). *Calibrating an ecompass in the presence of hard and soft-iron interference*. Freescale Semiconductor Ltd.
- [13] Kok, M., & Schön, T. B. (2016). *Magnetometer calibration using inertial sensors*. IEEE Sensors Journal, 16(14), 5679-5689.
- [14] Welch, G., & Bishop, G. (1995). *An introduction to the Kalman filter*.
- [15] Ribeiro, M. I. (2004). *Kalman and extended kalman filters: Concept, derivation and properties*. Institute for Systems and Robotics, 43.
- [16] Jimenez, A. R., Seco, F., Prieto, C., & Guevara, J. (2009). *A comparison of pedestrian dead-reckoning algorithms using a low-cost MEMS IMU*. In Intelligent Signal Processing, 2009. WISP 2009. IEEE International Symposium on (pp. 37-42). IEEE.
- [17] Ayazi, F. (2011). *Multi-DOF inertial MEMS: From gaming to dead reckoning*. In Solid-State Sensors, Actuators and Microsystems Conference (TRANSDUCERS), 2011 16th International (pp. 2805-2808). IEEE.
- [18] Chen, J. H., Lee, S. C., & DeBra, D. B. (1994). *Gyroscope free strapdown inertial measurement unit by six linear accelerometers*. Journal of Guidance, Control, and Dynamics, 17(2), 286-290.
- [19] Popović, L. Z., Šekara, T. B., & Popović, M. B. (2010). *Adaptive band-pass filter (ABPF) for tremor extraction from inertial sensor data*. Computer methods and programs in biomedicine, 99(3), 298-305.

- [20] Hide, C., Moore, T., & Smith, M. (2003). *Adaptive Kalman filtering for low-cost INS/GPS*. The Journal of Navigation, 56(01), 143-152.
- [21] Barshan, B., & Durrant-Whyte, H. F. (1995). *Inertial navigation systems for mobile robots*. IEEE Transactions on Robotics and Automation, 11(3), 328-342.
- [22] Savage, P. G. (1998). *Strapdown inertial navigation integration algorithm design part 1: Attitude algorithms*. Journal of guidance, control, and dynamics, 21(1), 19-28.
- [23] Savage, P. G. (1998). *Strapdown inertial navigation integration algorithm design part 2: Velocity and position algorithms*. Journal of Guidance, Control, and Dynamics, 21(2), 208-221.



## Chapter 1 Introduction

The ease-of-use touch screens have led to the rapidly growing popularity of smart devices. In addition to drawing and making handwritten notes, touchscreen devices nowadays are even capable of doing handwriting recognition. Light and slim touch screens now permeate every aspect of our lives; such as control panels in advanced factory machines, smart cars, mobile devices or even on aircraft dashboards. However, there is a common issue that exists in these applications.

According to the survey was done by Do. et al. (2011) [1], highlighted in the yellow region shown in Table 1-1; people spend huge amounts of time sending SMS, browsing websites and sending emails on transportation using smartphones. Under such unstable environments (such as vehicle), people's handwriting input distorts due to undesired shaking.

Table 1-1: Collects the usage of apps in different places (Unit: average number of events occurred/hour) [2]

Software Location	SMS	Voice communications	Network	Multimedia	Clock	Camera	Email	Calendar	Voice Chat	Maps	Exercise Tracker	Multimedia Streaming
Home	0.41	0.18	0.05	0.06	0.07	0.02	0.09	0.02	0.06	0.01	0.03	0.06
Company	0.51	0.29	0.09	0.07	0.01	0.02	0.11	0.04	0.05	0.01	0.04	0.05
Friend's House	0.63	0.34	0.30	0.06	0.06	0.03	0.17	0.04	0.05	0.01	0.04	0
Friend's company	0.72	0.30	0.02	0.05	0.02	0.05	0	0.04	0.04	0.01	0	0
Restaurant	0.52	0.27	0.19	0.02	0.01	0.03	0.10	0.05	0	0.06	0	0
Sports	0.93	0.26	0.09	0.03	0.01	0.03	0.32	0.07	0.23	0.03	0.10	0.04
Vehicle	0.33	0.57	0.52	0.17	0.07	0.07	0.07	0.15	0.05	0.04	0.04	0.01
Holiday	0.09	0.14	0.43	0.22	0.01	0.81	0.06	0	0.35	0.07	0	0
Shopping	0.21	0.33	0.22	0.08	0.04	0.09	0	0.03	0.06	0.01	0	0
Leisure	0.15	0.51	0.29	0.29	0	0	0	0	0	0.07	0	0
Others	0.75	0.40	0.16	0.09	0.02	0.09	0.20	0.11	0.05	0.05	0.05	0.05
Average	0.48	0.24	0.08	0.07	0.05	0.03	0.03	0.04	0.06	0.01	0.03	0.05

To solve this long-standing problem of handwriting distortion in unstable environments, by using new algorithms, this research presents a novel solution for improving the handwriting experience of touch screen devices.

The purpose of this study is to present a new algorithm to recover distorted drawings due to device shaking, which was known as the NoDistort. To make the system widely applicable on various devices, the system uses an accelerometer, gyroscope, and magnetometer, which are common in smartest devices. To obtain precise sensor readings, sensor calibration must be done before use. The calibration process should be effective, fast and equipment-free. Then an accurate motion estimation method is used to estimate the actual displacement of the device. Finally, the Drawing Distortion Recovery Algorithm, which bases on the Finger-Device Interaction Model, to recover distorted drawings. The below shows issues facing during the development of NoDistort and a brief introduction of the development process (Figure 1-1).

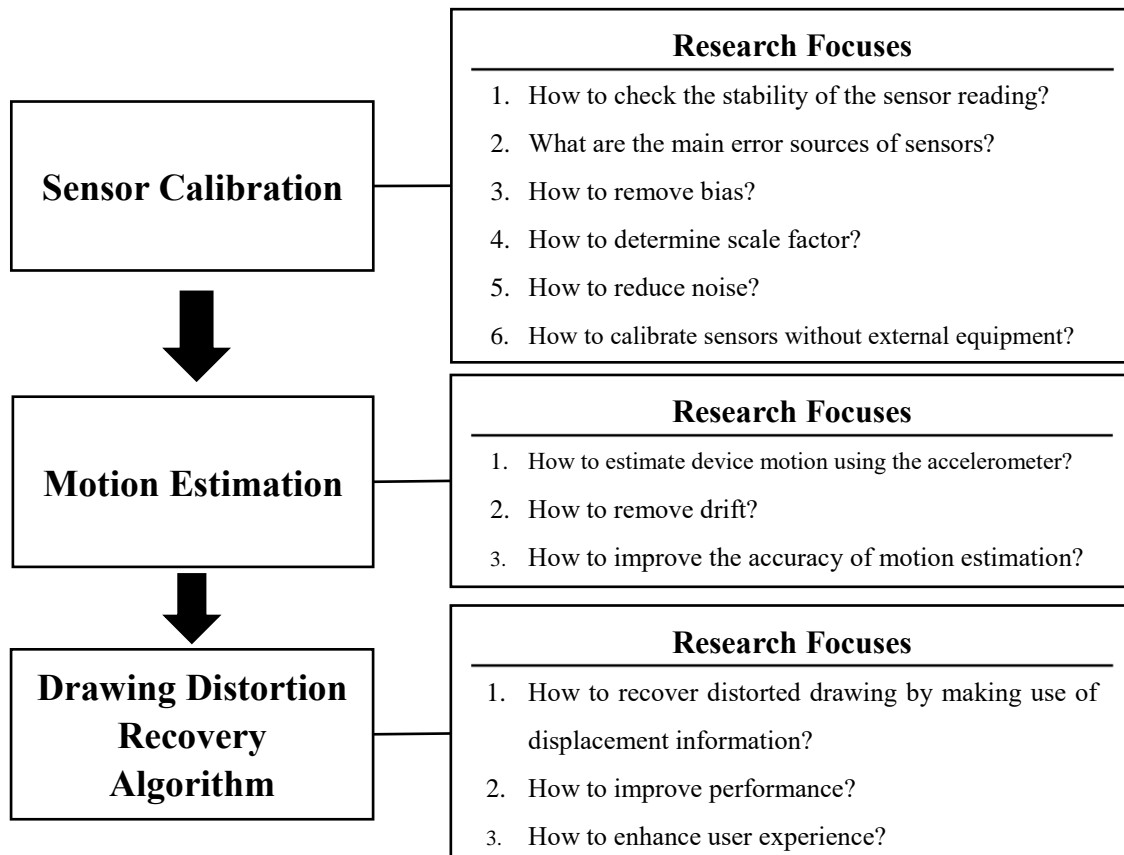


Figure 1-1: NoDistort development process

## 1.1 NoDistort System Flowchart

Figure 1-2 shows the process of the proposed NoDistort; In the initialization phase, the system waits for  $T_{init}$  seconds. Then the system prompts the user to rotate and place the device for N times to collect N sets of calibration data. Levenberg-Marquardt algorithm is used to obtain correct parameter values of Sensor Error Model. After sensor calibration, the system tried three different motion estimation algorithms as Euler Integration, RK4 integration and Spring-mass-damper model. Finally, the distorted drawings were recovered by the NoDistort.

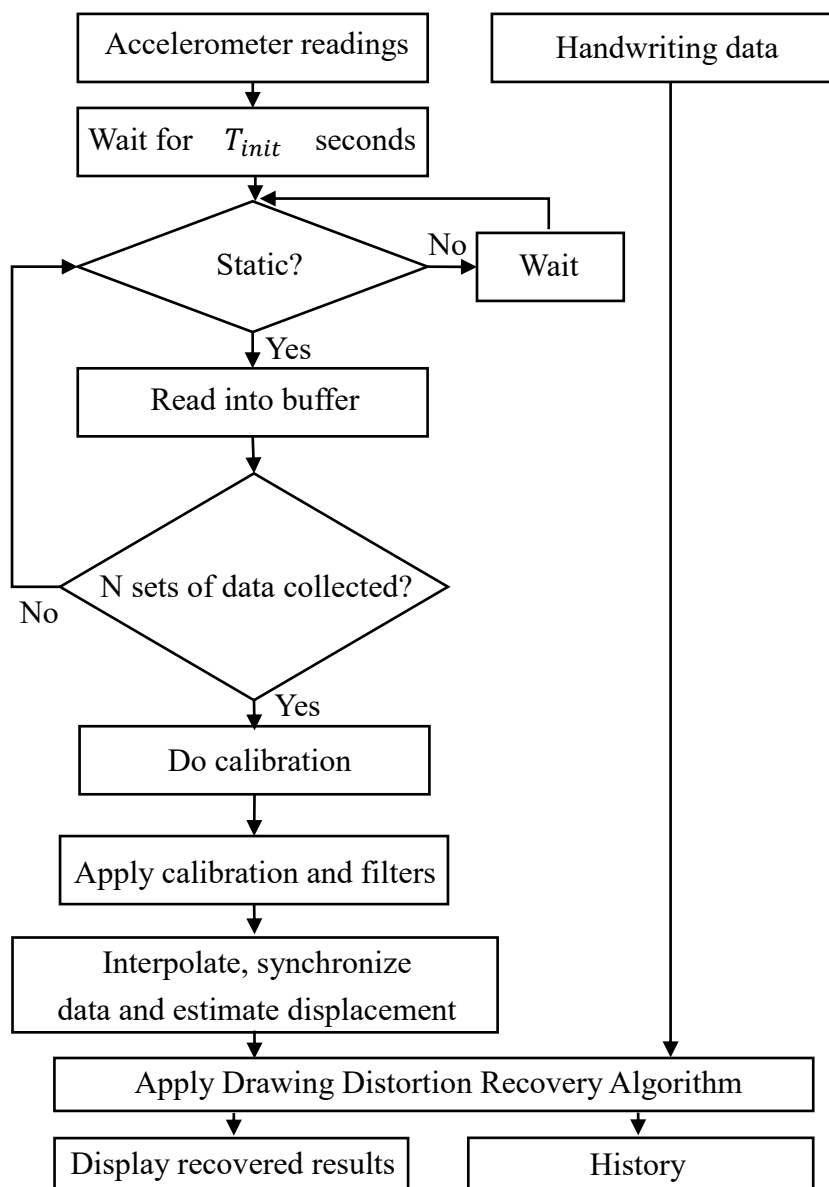


Figure 1-2: NoDistort flowchart

## 1.2 Allan variance

To know when to start fetching the sensor data, here the system uses the Allan variance technique [3] for analysis. Allan variance is used for observing noise characteristics and stability of a clock system. Here it is used to observe noise characteristics of the accelerometer. In the system, Allan variance is calculated through the following procedure:

1. 3-axis data obtained from the sensor.
2. Load data into the ring buffer.
3. Read 100 samples from the buffer and divide them into small pieces of duration  $t=1s$ .
4. For every piece of duration  $t$ , take the average of each  $(a(t_1) , a(t_2) , a(t_3) \dots a(t_n))$
5. To calculate Allan variance:

$$AVAR(t) = \frac{1}{2(n-1)} \sum_{i=1}^n (a(t_{i+1}) - a(t_i))^2 \quad (1-1)$$

6. The formula below is used to observe the intrinsic characteristics of noise; Allan Deviation is defined as follows:

$$\sigma(t) = \sqrt{AVAR(t)} \quad (1-2)$$

Allan deviation is used to check the sensor initialization phase:

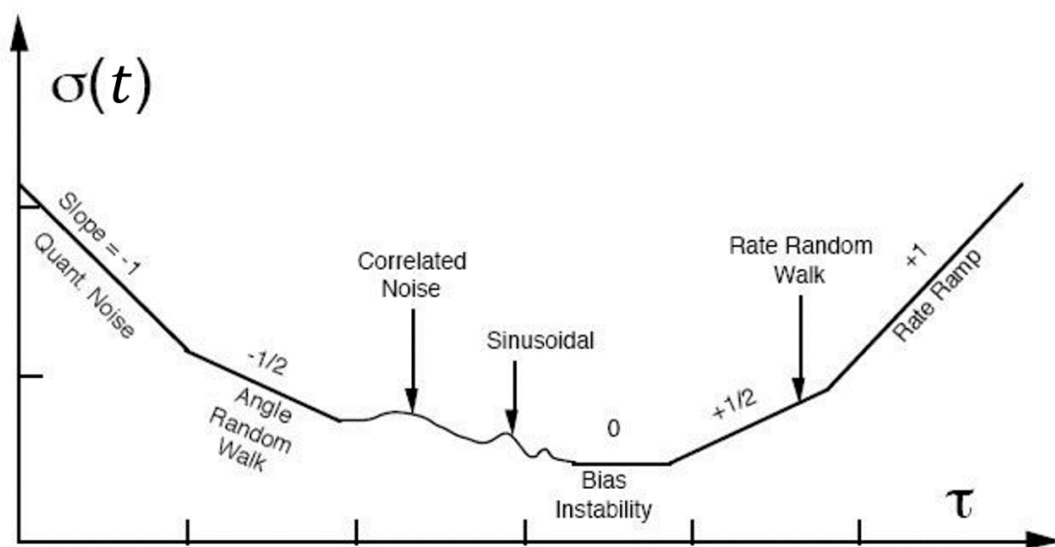


Figure 1-3: Different stages in sensor initialization [3]

When the slope of the Allan deviation of the three axes is equal to 0, the sensor has entered a stable stage; the system begins receiving stable sensor data, shown in Figure 1-3. This study concludes that after 35 seconds, the sensor starts to receive stable data.

### 1.3 Moving Average Filter

In this study, the sensor of the device updates at the rate of 100Hz. The test shows that an average window with a time span of 1/4 second can effectively reduce noise. Eq. (1-3), shows the average of  $\Delta t$ ,  $N$  indicates the number of samples in the observation windows.

$$\text{SMA}(t) = (a_{t-\frac{\Delta t}{2}} + \dots + a_{t+\frac{\Delta t}{2}})/N \quad (1-3)$$

### 1.4 Accelerometer Calibration

The form below shows main error sources of accelerometer:

#### 1. Bias

Sensor bias is a constant continuously exists in the accelerometer output signal. Under an ideal situation,  $b^a$  is a  $3 \times 1$  matrix, a bias matrix can be expressed as:

$$b^a = \begin{bmatrix} b_x^a \\ b_y^a \\ b_z^a \end{bmatrix} \quad (1-4)$$

#### 2. Scale Factor

In an ideal situation,  $K^a$  is an identity matrix, it can be expressed as:

$$K^a = \begin{bmatrix} s_x^a & 0 & 0 \\ 0 & s_y^a & 0 \\ 0 & 0 & s_z^a \end{bmatrix} \quad (1-5)$$

#### 3. Sensor Error Model

In the sensor error model, noise should also be considered. Therefore, the complete sensor error model can be expressed as:

$$a^o = K^a(a^s + b^a + v^a) \quad (1-6)$$

where  $a^o$  is  $3 \times 1$  true acceleration value,  $K^a$  is  $3 \times 3$  scale matrix,  $a^s$  is  $3 \times 1$  input matrix,  $b^a$  is  $3 \times 1$  bias matrix and  $v^a$  is  $3 \times 1$  noise matrix.

#### 4. Define accelerometer cost function

In Error model, the study determines the bias and the scale factor. There are 6 parameters need to be determined. Parameter vector can be expressed as:

$$\lambda^a = [s_x^a, s_y^a, s_z^a, b_x^a, b_y^a, b_z^a] \quad (1-7)$$

True values can be calculated by the following formula:

$$a^o = h(a^s, \lambda^a) = K^a(a^s + b^a) \quad (1-8)$$

Since the accelerometer magnitude value when device remains static must be equal to the gravity constant  $g$ , Eq. (1-9) defines a cost function which represents the error between gravity constant  $\|g\|$  and the measured values.

$$L(\lambda^a) = \sum_{k=1}^N (\|g\|^2 - \|h(a_k^s, \lambda^a)\|^2)^2 \quad (1-9)$$

$N$  represents the amount of measurements in the observation windows.

---

#### Algorithm 1 Sensor Calibration

---

Require:  $T_{init}$  (wait before start acquiring data),  $a^s$  (Calibrated Data), datasetNum (number of cycles to collect data)

---

```

1: Wait for  $T_{init}$  seconds
2: for  $i = 0$ : datasetNum
3:   if static == true
4:     dataseti ← average of array of  $a^s$ 
5:   else
6:     retry
7:   end
8: end
9: Params ← optimize using dataset and Levenberg – Marquardt algorithm
10:  $a^0 \leftarrow$  calibrate  $a^s$  using Params

```

---

## 1.5 Euler Integration

Euler Integration is the most direct method to compute velocity and position from acceleration. The following shows a Euler Integration processes:

1. Eq. (1-10) and Eq. (1-11) show the motion of the device:

$$\frac{dx}{dt} = v \quad (1-10)$$

$$\frac{dv}{dt} = a \quad (1-11)$$

2. Known initial position and velocity:

$$x(0) = x_0 \quad (1-12)$$

$$v(0) = v_0 \quad (1-13)$$

3. In the system, each time when a user writes,  $x(0)$  resets. When the device is detected static, motion estimation starts at a velocity of 0.

4. When  $\Delta t$  ( $\Delta t$  is the interval between each sample) is small enough:

$$dt \approx \Delta t \quad (1-14)$$

5. The differential of position over time is velocity.

$$\frac{dx}{dt} \approx \frac{x(t+\Delta t) - x(t)}{\Delta t} \quad (1-15)$$

6. The differential of velocity over time is acceleration:

$$\frac{dv}{dt} \approx \frac{v(t+\Delta t) - v(t)}{\Delta t} \quad (1-16)$$

7. Velocity and position can be calculated from the following equations:

$$x(t + \Delta t) \approx x(t) + v(t)\Delta t \quad (1-17)$$

$$v(t + \Delta t) \approx v(t) + a(t)\Delta t \quad (1-18)$$

Local truncation error (LTE) of Euler Integration:

1. The following shows estimating value of  $y_1$  from  $y_0$  using Euler Integration:

$$y_1 = y_0 + hf(t_0, y_0) \quad (1-19)$$

here  $f$  is the derivative of position over time  $h = t_{n+1} - t_n$ .

2. Set  $y_1 = y(t_0 + h)$  and expand  $y_1$  using Taylor Series.

$$y(t_0 + h) = y(t_0) + hy'(t_0) + \frac{1}{2}h^2y''(t_0) + O(h^3) \quad (1-20)$$

3. Then subtract true value  $y(t_0 + h)$  by the Euler Method estimated value  $y_0$  and get LTE.

$$\text{LTE} = y(t_0 + h) - y_1 = \frac{1}{2}h^2y''(t_0) + O(h^3) \quad (1-21)$$

4. The above shows that when the step size  $h$  is small, local truncation error of Euler Integration is proportional to  $h^2$ .

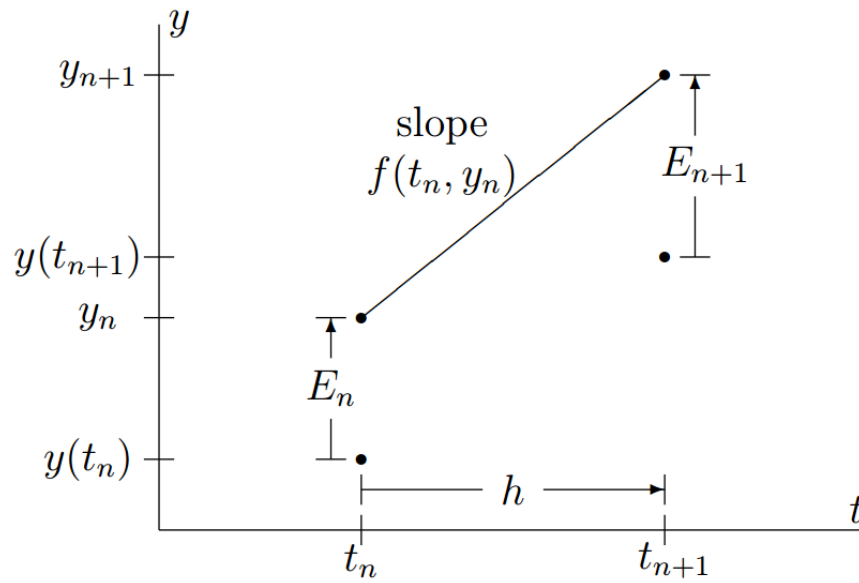


Figure 1-4: A Euler integration process

---

### Algorithm 2 Euler Integration

---

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

---

```

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

```

---



## 1.6 Mass-Spring-Damper Model

Mass-Spring-Damper Model is physical model consisting of a spring and a damper, which was inspired by the Mass-Spring-Damper System. Such a system behaves very similarly to a device shaking caused by hands [5]. The model sees a touch screen as an object suspending in space, and it is connected to a spring and a damp. By applying force, the screen of the device acts as the Mass-Spring-Damper System. Figure 1-5 shows a one-dimensional Mass-Spring-Damper System.

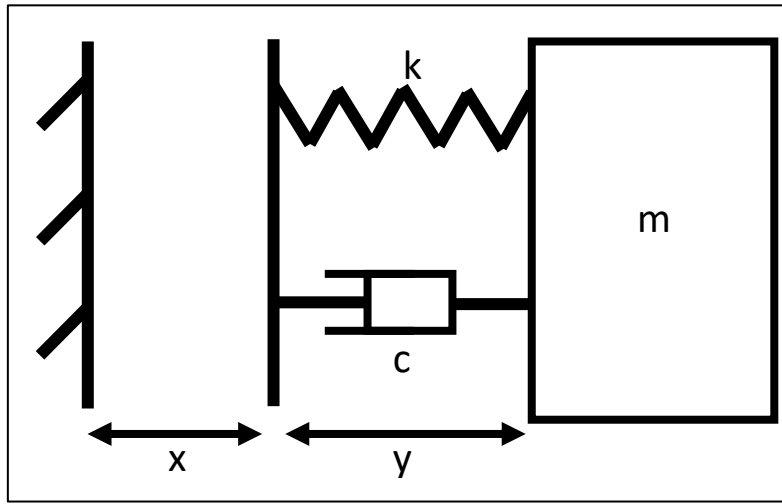


Figure 1-5: One-Dimensional Mass-Spring-Damper System

1. The behavior of the system depends on the following damping ratio.

$$\zeta = \frac{c}{2\sqrt{km}} \quad (1-22)$$

2. When  $\zeta < 1$ , the system is underdamped, it continuously oscillates back and forth. When  $\zeta > 1$ , the system is overdamped, which causes the system to spend too much time to return to a state of equilibrium. When setting  $\zeta = 1$ , the system turns into a critical damping system. It is the best situation and was used in the experiment. Setting  $c = 2\sqrt{km}$ ,  $\zeta = 1$ , the critical damped system equation can be described as:

$$\ddot{y} + 2\sqrt{km}\dot{y} + ky = -A(t) \quad (1-23)$$

3. By computing the convolution of  $-A(t)$  and the Impulse Response of the system  $H(t)$ , the displacement was then calculated.

$$y(t) = H(t) * -A(t) \quad (1-24)$$

4. Impulse Response of critical damping System  $H(t)$  was given:

$$H(t) = te^{-t\sqrt{k}} \quad (1-25)$$

---

### Algorithm 3 Mass-Spring-Damper Model

---

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

---

```

1: for i = 0: bufferSize
2:   t ← (1 -  $\frac{i}{\text{bufferSize}}$ ) * sampleDelay;
3:   impulseResponse ←  $te^{-t\sqrt{k}}$ 
4:   sum += (mBuffer[i] * impulseResponse); //convolve buffer
   with impulseResponse
5: end
6: return sum

```

---

## 1.7 Runge-Kutta Integration

First-order Euler Integration of acceleration over time will result in significant bias, whereas Runge-Kutta Integration has a higher Taylor precision. Among all Runge-Kutta method, RK4 is the most commonly used. The process requires the following four Runge-Kutta Integration equations:

$$y_{n+1} = y_n + \frac{\Delta t}{6}(k_1 + 2k_2 + 2k_3 + k_4) \quad (1-26)$$

where

$$\frac{dy}{dt} = f \quad (1-27)$$

$$k_1 = f(t_n, y_n) \quad (1-28)$$

$$k_2 = f(t_n + \frac{\Delta t}{2}, y_n + \frac{\Delta t}{2}k_1) \quad (1-29)$$

$$k_3 = f(t_n + \frac{\Delta t}{2}, y_n + \frac{\Delta t}{2}k_2) \quad (1-30)$$

$$k_4 = f(t_n + \frac{\Delta t}{2}, y_n + \frac{\Delta t}{2} k_3) \quad (1-31)$$

y can be position or velocity

The Local truncation error (LTE) of RK4 method is  $O(h^5)$ , thus when h is small.

---

#### Algorithm 4 RK4 Integration

---

**Require:**  $\mathbf{v_p}$  (previous velocity),  $\mathbf{p_p}$  (previous position),  $\mathbf{a}$  (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  $X_1$ )  $\leftarrow v_p + 0 * 0$            //the 1st point at  $t = 0$ 
3:  $V_2$ (velocity at position  $X_2$ )  $\leftarrow v_p + V_1 * dt * 0.5$  //the 2nd point at  $t = 0.5 * dt$ 
4:  $V_3$ (velocity at position  $X_3$ )  $\leftarrow v_p + V_2 * dt * 0.5$  //the 3st point at  $t = 0.5 * dt$ 
5:  $V_4$ (velocity at position  $X_4$ )  $\leftarrow v_p + V_3 * dt$        //the 4th point at  $t = dt$ 

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

7:  $p_p += V_{all} * dt$ ;
8:  $v_p += a * dt$ ;
9: return  $p_p$ 

```

---

## 1.8 Filter Base on Exponential smoothing

### 1. Low Pass Filter:

The first-order exponential smoothing formula was used here as a low-pass filter to filter out high-frequency signal, Eq. (1-32) is a formula of first-order exponential smoothing.

$X_t$  is input data,  $S_t$  is output data

$$S_t = \alpha \cdot X_t + (1 - \alpha) \cdot X_{t-1} \quad (1-32)$$

### 2. High pass filter:

Subtracting the original signal by the low-frequency proportion results in high pass filtering effects.  $X_t$  is input data,  $S_t$  is output data

$$S_t = X_t - (\alpha \cdot X_t + (1 - \alpha) \cdot X_{t-1}) \quad (1-33)$$

where the smoothing constant  $\alpha$  above was determined by the experiment.

## 1.9 Kalman Filter

Kalman filter is also known as a linear quadratic estimator Linear Quadratic Estimation (LQE), which is a method using the measure of the past (including statistical noise and other uncertain factors, and so on) to estimate unknown variables. Kalman filter values are often more accurate than the measured values. Thus, it is often used to predict motion. Since Kalman filter is a kind of recursive filter, there is no need to record historical data.

## 1.10 Static Sensor

Because calibration data hugely affect the accuracy of calibration, distinguishing whether the device is static or not is critical. Given time interval  $t_w$ , variance of sensor reading was calculated by the following

$$\text{Variance Magnitude } \zeta(t) = \sqrt{[VAR_{t_w}(a_x^t)]^2 + [VAR_{t_w}(a_y^t)]^2 + [VAR_{t_w}(a_z^t)]^2} \quad (1-34)$$

By checking if  $\zeta(t)$  is higher or lower than the static detector threshold, the detector determines whether the device is placed static or not. The threshold of the static detector was determined by experiment.

---

### Algorithm 5 Static Detector

---

**Require:** W (3-axis data observed in a rectangular window of  $t_w$  seconds at time t), T (static time required to trigger static detector), threshold

---

```
1: for  $t = t_0 : t_n$ 
2:   calculate the variance of data in the windows:  $var_x(t), var_y(t), var_z(t)$ 
3:   if  $\sqrt{[VAR_x(t)]^2 + [VAR_y(t)]^2 + [VAR_z(t)]^2} > threshold$ 
4:     static_start_time  $\leftarrow t_n$ 
5:   end
6:   if  $t_n - t_0 > 1 \text{ sec}$ 
7:     staticdetector.trigger()
8:   end
9: end
```

---

## 1.11 Sensor Timestamps Synchronization

Since different sensors have their update rates, NoDistort utilizes the shared ring buffer and cubic spline interpolation to synchronize and accept on-demand sensor data input.

### 1. Shared ring buffer:

Create a shared ring buffer, then load all of the incoming data into the buffer, where  $N$  represents the sensor data in the buffer as an array of data. Meanwhile, the buffer throws out the old data while the new data load into the buffer zone to maintain the buffer size unchanged. The Shared Ring Buffer not only make data extraction easier, but it also makes it simpler to interpolate or apply the window functions. Figure 1-6 shows Common ring buffer operation diagram.

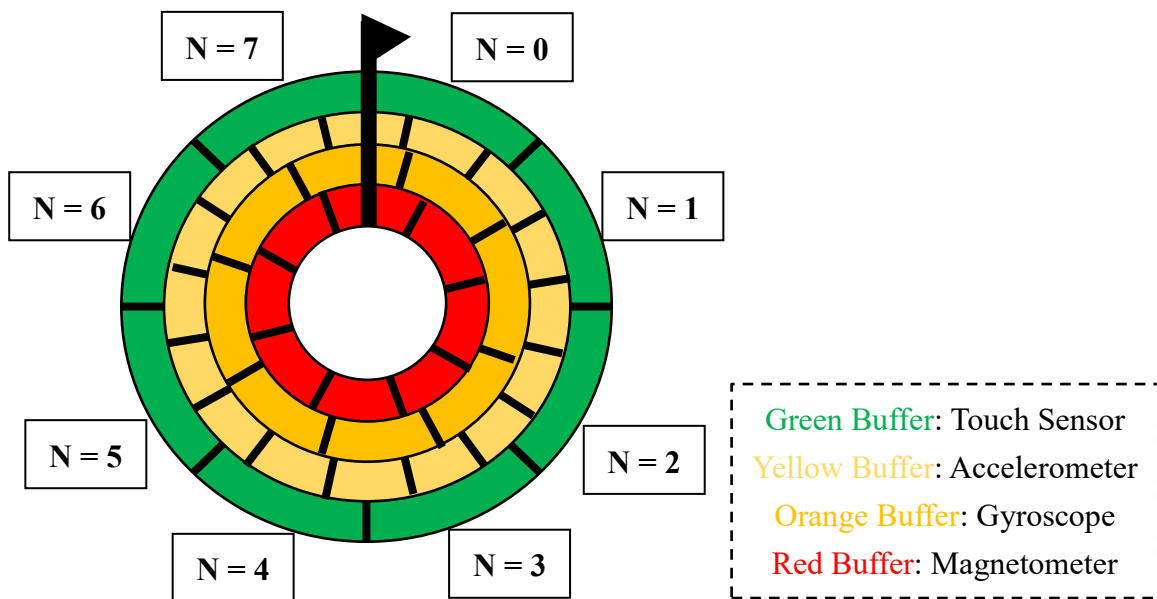


Figure 1-6: Common ring buffer

### 2. Spline interpolation:

In the process of motion estimation and distortion recovery, the system has to synchronize the sensors. These sensors include accelerometers, gyroscope, magnetometer and a touch screen. This alignment process requires the use of a shared ring buffer with spline interpolation methods. In the system timestamps synchronization process, cubic spline

interpolation is used. One of the benefits of this interpolation method is it does not produce dithering as Runge's phenomenon.

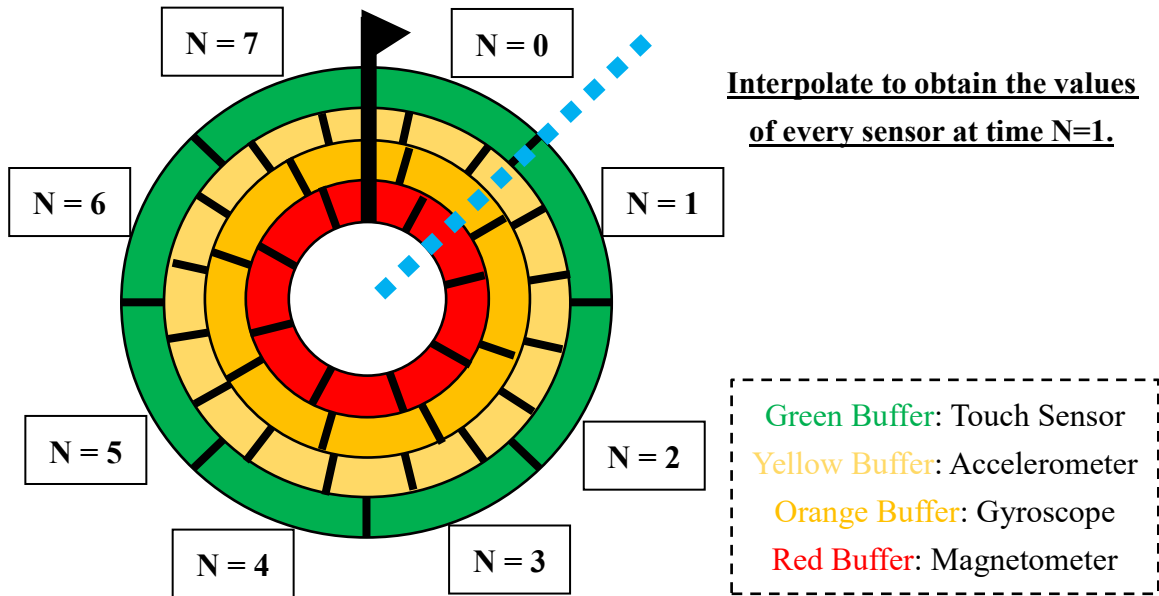


Figure 1-7: Do spline interpolation in the shared ring buffer

## 1.12 Status Detection

NoDistort focuses on common situations which the device moves in a limited range.

Therefore, some special situations are seen as user's awareness behavior. The following cases are considered an exception situation:

1. The value of acceleration is larger than  $4 \text{ m/s}^2$ .
2. The value of velocity is larger than  $1 \text{ m/s}$ .
3. The length of displacement is larger than  $4 \text{ cm}$ .
4. The acceleration value remains positive or negative more than 1 second.

### 1.13 Finger-Device Interaction Model

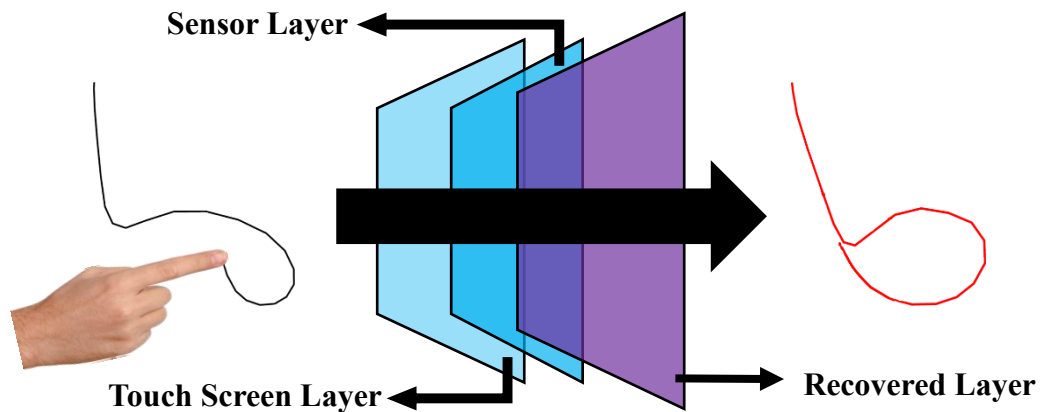


Figure 1-8: NoDistort is based on 3 layers model

Figure 1-8 Showed finger-screen interaction model. It consists of 3 parts:

#### 1. Drawing Layer:

Drawing Layer is an imagined layer fixed in the space (in world coordinates). Users intend to draw on this virtual canvas, which is the ultimate goal NoDistort aimed to recover distorted drawing. Drawing Layer is fixed in space at the moment user starts drawing, it does not change until the user pulls up his finger.

#### 2. Sensor Layer:

Sensor Layer is a bridge link between the Drawing and Touch layer. NoDistort utilizes a built-in accelerometer, gyroscope, and magnetometer to calculate the inclination angle and relative position between the world coordinates and the Sensor Layer. These sensors are Micro-Electro-Mechanical Systems. Since sensors in the device are fixed and have no movable parts, it is a strap-down system. Thus, the Sensor Layer is always parallel to the Touch Layer.

#### 3. Touch Layer:

Touch Layer is a medium finger physically contacts to. It records information including touch position and touch pressure (the research device can record touch location and touch pressure). It is parallel to the Sensor Layer.

## Chapter 2 NoDistort v1.0

In the v1.0 finger-device interaction model, every of the layers is parallel to each other.

### 2.1 The v1.0 Finger-Device Interaction Model

#### 1. Drawing Layer:

The layer user intended to draw on; it is fixed in space.

#### 2. Sensor Layer:

Consists of an accelerometer, gyroscope, and magnetometer. Sensor Layer is parallel to earth surface, and it would move and rotate with the device but limited on a 2-dimensional platform.

#### 3. Touch Layer:

Touch Layer is used to get touch positions, and it is always connected to the Sensor Layer.

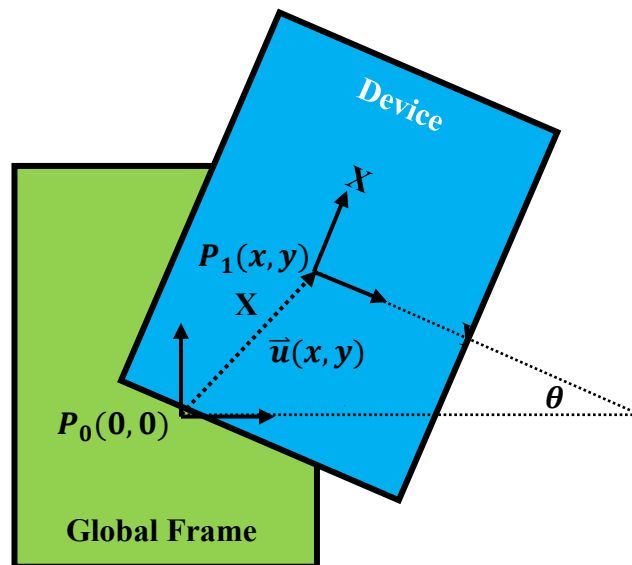


Figure 2-1: The v1.0 finger-device interaction model



## 2.2 Mapping Touch Layer to Drawing Layer

1. At the time the user hands down, align all layers. Align origins of all layers (0,0).
2. Set the Touch Layer moves  $(X_s, Y_s)$  regarded to its origin. Set  $\vec{S}$  the position vector of the origin of the Touch Layer on the Drawing Layer.
3. Set a touch point  $(X_t, Y_t)$ , its position vector can be described as  $\vec{T}$ .
4. Set the Touch Layer rotates counterclockwise (+) around the center of the screen by  $\theta$ .
5. Assumed between sampling points from current to the next, Sensor Layer did a linear transform consists of translation and rotation relative to Touch Layer. For convenience, the below uses homogeneous coordinates.

- (1) Set position vector of the user-intended stroke on the Drawing Layer:

$$\vec{P} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (2-1)$$

- (2) Set  $T'$  the position vector when  $\vec{P}$  is converted to the Drawing Layer:

$$\vec{P} = \vec{S} + \vec{T'} \quad (2-2)$$

- (3) To know  $\vec{P}$ , first convert the touch position vector  $\vec{T}$  to the Drawing Layer, which is

$$\vec{T'} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \vec{T} \quad (2-3)$$

the recovered position can be obtained:

$$\vec{P} = \vec{S} + \vec{T'} = \vec{S} + \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \vec{T} \quad (2-4)$$

$$\vec{T} = \begin{bmatrix} X_s \\ Y_s \\ 1 \end{bmatrix} + \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_t \\ Y_t \\ 1 \end{bmatrix} = \begin{bmatrix} X_s + X_t \cos(\theta) - Y_t \sin(\theta) \\ Y_s + Y_t \cos(\theta) + X_t \sin(\theta) \\ 1 \end{bmatrix} \quad (2-5)$$

6. Then display the recovered drawings on the screen.

## Chapter 3 NoDistort v2.0

In the v2.0 update, the NoDistort greatly improved its performance and effect. First of all, the v2.0 recovery algorithm removes significant motion caused by moving vehicles by applying a high-pass filter. This new feature ensures that the algorithm won't be affected by intended motion. The v2.0 recovery algorithm also removed the assumption that the three layers are always parallel to one another. It allows the system make the 3-axis motion sensors to do 3-dimensional linear transformations. Finally, the v2.0 recovery algorithm was re-written to make it a real-time recovery system. This allows for users to view recovery results simultaneously while they draw. Furthermore, NoDistort v2.0 sticks the result with the finger to make the overall experience better.

### 3.1 The v2.0 Finger-Device Interaction Model

1. Drawing Layer is an imagined layer fixed in the space (in world coordinate). Users intends to draw on this virtual canvas, which is the ultimate goal NoDistort aimed to recover distorted drawing to.
2. Sensor Layer: Consists of an accelerometer, gyroscope, and magnetometer. Initially, the layers are aligned, it transforms and rotates with the device.
3. Touch Layer: used to get touch points always connected to the sensor layer.
4. Fingers: v2.0 recovery algorithm do not consider unexpected finger movements.
5. Apply high-pass filter to the device's acceleration direction:

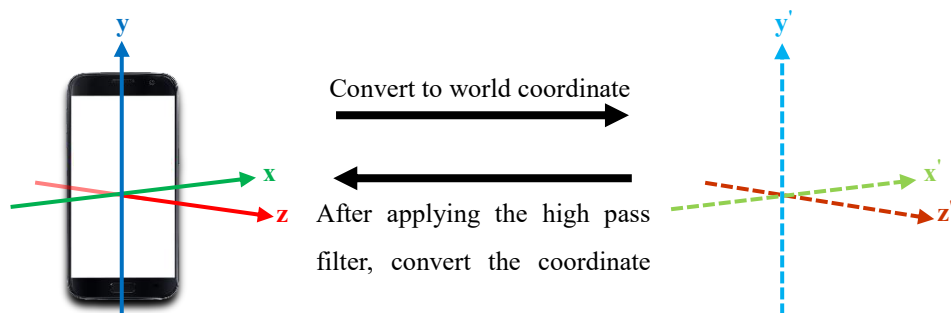


Figure 3-1: Apply high-pass filter to the device's acceleration direction

### 3.2 Mapping Touch Layer to Drawing Layer

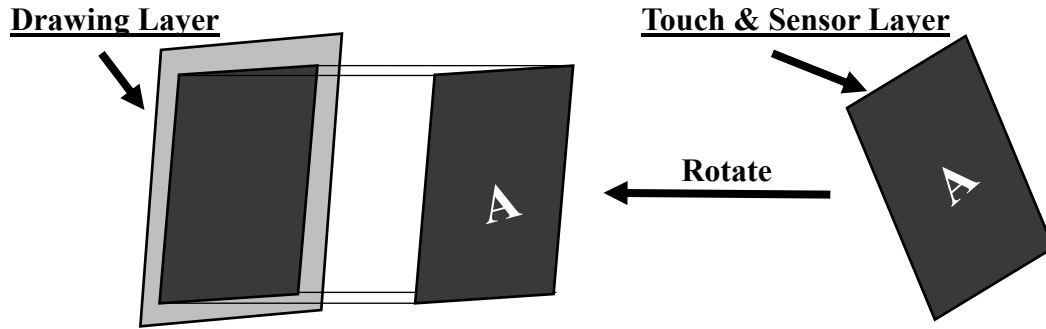


Figure 3-2: Do three-dimensional linear transform to touch points

1. At the moment user hands down, all of the layers were aligned and shared the same origin  $(0,0)$ .
2. Since people do not draw vertically, the Z-axis drawing is ignored. Set the Touch Layer moves  $(X_s, Y_s)$  with regard to the origin, and set  $\vec{S}$  the position vector of the Touch Layer relative to the Drawing Layer.
3. Set a touch position  $(X_t, Y_t)$  on the Touch Layer, its position vector on the layer is  $\vec{T}$ .
4. To avoid Gimbal lock happen, here all calculations of rotations were expressed using quaternion. Set the absolute orientation when the user hands down  $q_0$ , and the absolute orientation of next touch point  $q_1$ .
5. Here the rotation vector quaternion is defined as the following:

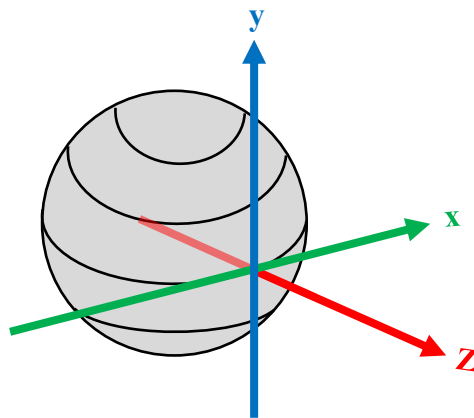


Figure 3-3: Definition of world coordinate. X axis: the direction of Y and Z axis ( $Y \times Z$ ), Y axis: positive at north direction, Z axis: positive at the direction pointing to the sky.

6. Set the origin of the Touch Layer moves  $(X_s, Y_s)$  and rotates  $q_{01}$  regarded to the Drawing Layer.

(1) Set position vector of the user-intended stroke on the Drawing Layer:

$$\vec{P} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \quad (3-1)$$

(2) Set  $T'$  the position vector when  $\vec{P}$  is converted to the Drawing Layer:

$$\vec{P} = \vec{S} + \vec{T}' \quad (3-2)$$

(3) Set initial and Quaternion of the next touch point:

$$q_0 = q_0[0] + q_0[1]i + q_0[2]j + q_0[3]k \quad (3-3)$$

$$q_1 = q_1[0] + q_1[1]i + q_1[2]j + q_1[3]k \quad (3-4)$$

(4) Take the quaternion difference:

$$q_{01} = q_1 q_0^{-1} = q_1 \frac{q_0^*}{\|q_0\|^2} \quad (3-5)$$

(5) Since the norm of the Rotation Vector is 1,

$$q_{01} = q_1 q_0^* \quad (3-6)$$

where

$$q_0^* = q_0[0] - q_0[1]i - q_0[2]j - q_0[3]k \quad (3-7)$$

(6) Then after  $q_{01}$  rotation  $\vec{T}$  it becomes  $\vec{T}'$ :

$$\vec{T}' = q_{01} \vec{T} q_{01}^{-1} \quad (3-8)$$

expressed using the rotation matrix:

$$\vec{T}' = \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} (1 - 2q_{01}[2]^2 - 2q_{01}[3]^2) & 2(q_{01}[1]q_{01}[2] + q_{01}[0]q_{01}[3]) & 2(q_{01}[1]q_{01}[3] - q_{01}[0]q_{01}[2]) \\ 2(q_{01}[1]q_{01}[2] - q_{01}[0]q_{01}[3]) & (1 - 2q_{01}[1]^2 - 2q_{01}[3]^2) & 2(q_{01}[2]q_{01}[3] + q_{01}[0]q_{01}[1]) \\ 2(q_{01}[1]q_{01}[3] + q_{01}[0]q_{01}[2]) & 2(q_{01}[2]q_{01}[3] - q_{01}[0]q_{01}[1]) & (1 - 2q_{01}[1]^2 - 2q_{01}[2]^2) \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \quad (3-9)$$

(7) Finally:

$$\vec{P} = \begin{bmatrix} X_s \\ Y_s \\ 0 \end{bmatrix} + \begin{bmatrix} (1 - 2q_{01}[2]^2 - 2q_{01}[3]^2) & 2(q_{01}[1]q_{01}[2] + q_{01}[0]q_{01}[3]) & 2(q_{01}[1]q_{01}[3] - q_{01}[0]q_{01}[2]) \\ 2(q_{01}[1]q_{01}[2] - q_{01}[0]q_{01}[3]) & (1 - 2q_{01}[1]^2 - 2q_{01}[3]^2) & 2(q_{01}[2]q_{01}[3] + q_{01}[0]q_{01}[1]) \\ 2(q_{01}[1]q_{01}[3] + q_{01}[0]q_{01}[2]) & 2(q_{01}[2]q_{01}[3] - q_{01}[0]q_{01}[1]) & (1 - 2q_{01}[1]^2 - 2q_{01}[2]^2) \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \quad (3-10)$$

(8) Then display the recovered drawings on the screen.

### 3.3 Re-Written into a Real-Time Recovery Process

NoDistort v2.0 make use of multithreading, which made interface speed increased significantly.

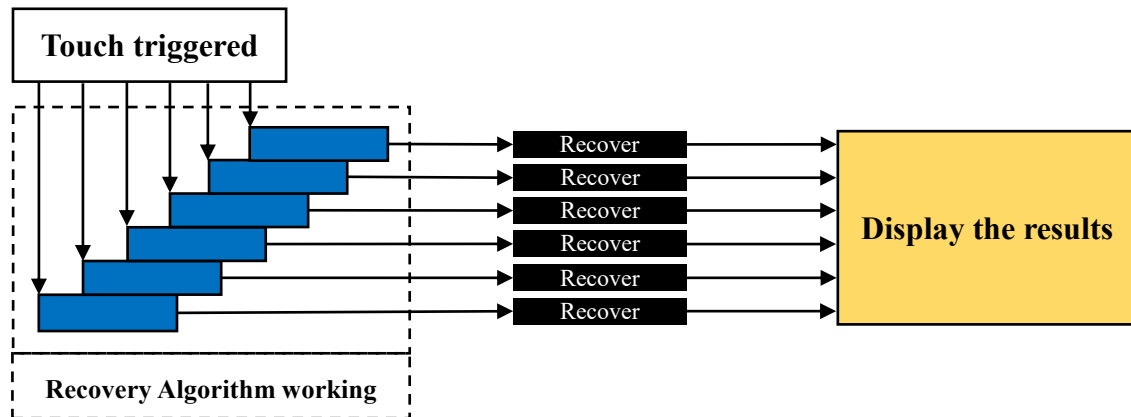


Figure 3-4: Multithreading

### 3.4 Detecting if Strokes Belongs to the Same Character

The study designs a stroke split/merge mechanism to detecting if strokes belong to the same character. This feature makes NoDistort being able to recognize multi-stroke characters (such as A, B, J...).

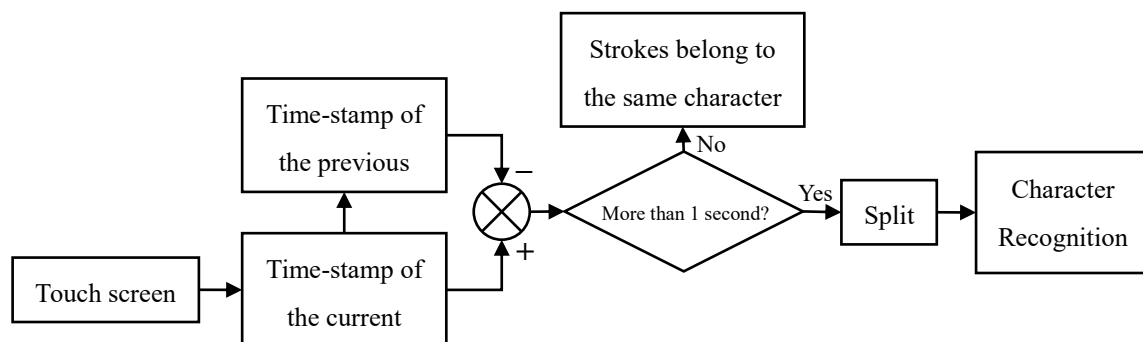


Figure 3-5: Same Character detecting flow chart

### 3.5 Let the Recovered Results Continuously Follow User's Finger Position

The new algorithm connects the end of the recovered drawings to user's finger. This new feature helps users to view the latest recovered results and make the user experience way more intuitive.

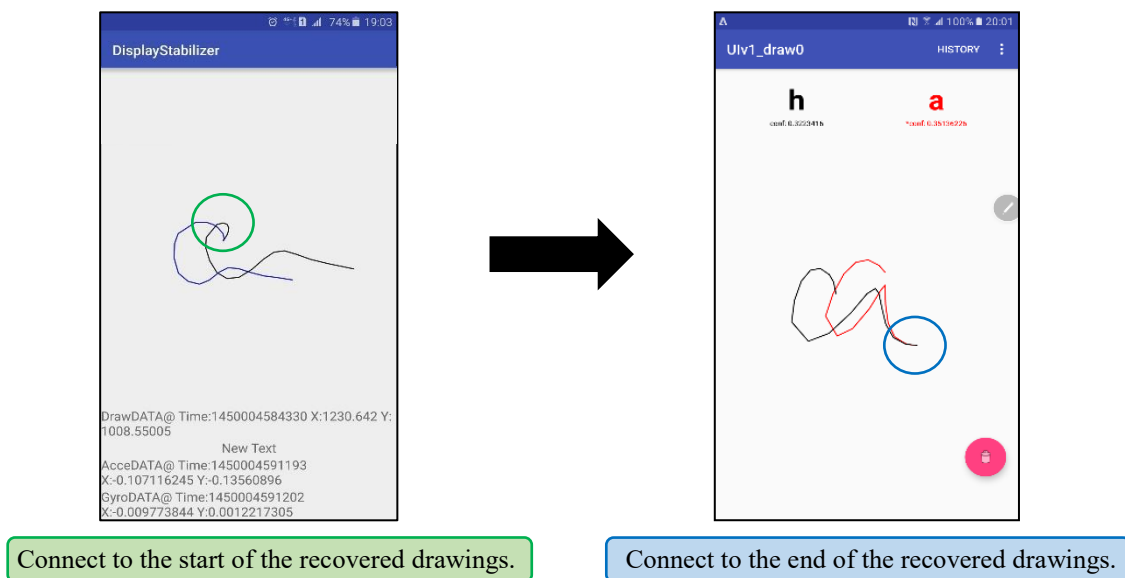


Figure 3-6: NoDistort v1.0 & v2.0 display style comparison (Left is v1.0, Right is v2.0).

## Chapter 4 NoDistort v3.0

The v3.0 recovery algorithm is based on the v2.0 version and it was greatly improved by extensive performance optimization. Drawing smoothing was also added as a new feature to make the recovered drawings even more beautiful.

### 4.1 Performance Optimization

The huge performance improvement of NoDistort v3.0 was the result of rewriting the algorithm in an incremental procedure which accumulates previous results rather than recomputing the full canvas at each redraw event. It avoids cumulative errors through setting the conservativeness of recovery strength.

### 4.2 Drawing Rotation

When the device rotates, the drawing rotates in the opposite direction accordingly to offset Rotary shaking. The absolute direction of the device,  $q_0$  (In the unified global coordinate), is set when the user starts drawing. Set the absolute direction of the device at the moment  $q_1$  (In the unified global coordinate). The system take the difference between the two quaternions and rotate the drawings to the user's view. Then display the drawings on the x-y plane.

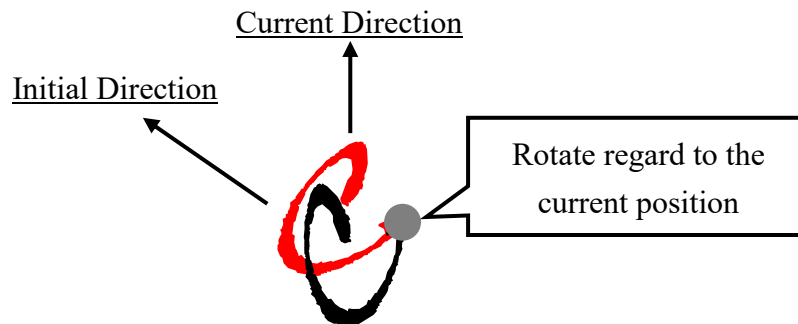


Figure 4-1: Rotate the drawings to the user's view

When the user rotates the device, the system continuously aligns the drawings to make it stay in the correct position relative to the user. It makes drawing on the device a lot easier. After the user pulls up his hand, NoDistort automatically rotates the drawings back in the exact orientation to facilitate handwriting recognition.

### 4.3 Uniform Drawing Coordinates

In order to unify touch point coordinates, NoDistort set all strokes toward a unified direction. To make the changes of touch positions meet the requirements of the unified touch coordinates, set the absolute direction of the device when the user starts drawing  $q_0$ . Set the absolute direction of the device at the moment  $q_0$ . The system take the difference between the two quaternions, revert the delta touch vector and append it to the previous touch position to obtain the new recovered position. The purpose of this amendment is to do curve line curvature correction and straightening.

### 4.4 Correct Estimation Errors Caused by the Relative Distance Between Built-in Sensors and Screens

Since there is a distance between built-in sensors (accelerometer, gyroscope, and magnetometer) and touch position, the following amendment is used to fix this problem.

1. Set the touch position  $(x, y)$ , rotated touch position  $(x', y')$ , estimated displacement by acceleration  $\vec{O}(p, q)$ , the actual displacement of touch point in space  $\vec{T}(P, Q)$ , the device rotated clockwise for  $\theta$ , and its rotation matrix  $R$ .

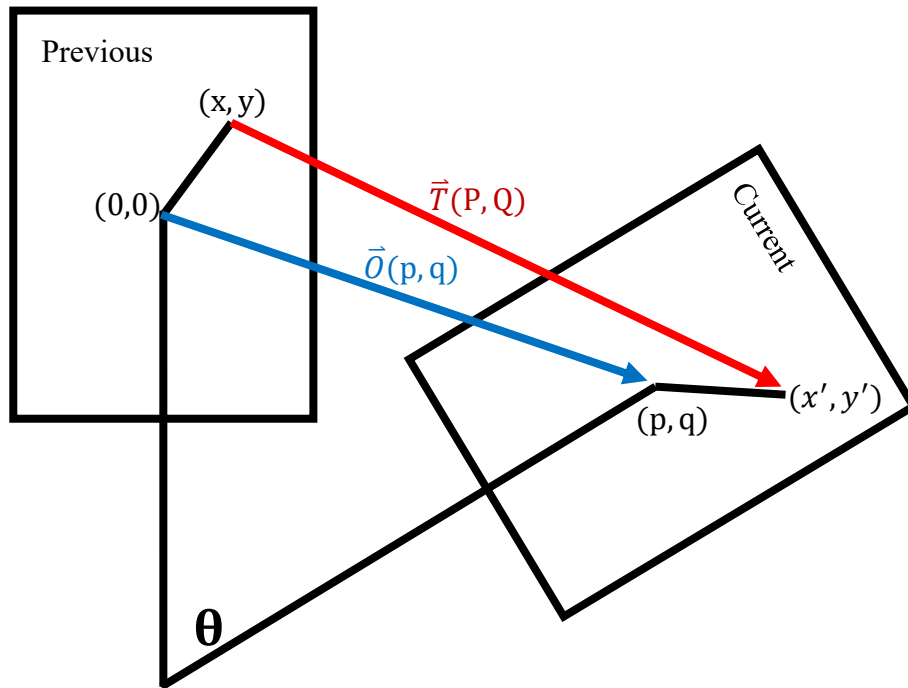


Figure 4-2: Calculate the actual displacement of touch points in space



2. Express the rotated touch point  $(x', y')$  using measured displacement  $\vec{O}$ .

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \vec{O} + R \begin{bmatrix} x \\ y \end{bmatrix} \quad (4-1)$$

3. Rewriting the above equation and then the correct displacement  $\vec{T}$  was obtained.

$$\vec{T} = \begin{bmatrix} P \\ Q \end{bmatrix} = \begin{bmatrix} x' \\ y' \end{bmatrix} - \begin{bmatrix} x \\ y \end{bmatrix} = \vec{O} + R \begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} x \\ y \end{bmatrix} \quad (4-2)$$

## 4.5 Handwriting Recognition

The study implemented the open-source Lipitoolkit 4.0 for character recognition. Lipitoolkit was developed by HP Laboratories, using (Massachusetts Institute of Technology, MIT) license.

The study uses toolkit's default training data and Android Native Interface to make it run on the Android device. The following are instructions of using this Kit:

### 1. Input/output:

- (1) Input is an array of strokes, which stroke is defined as continuous touch points users draw at a time, and must be entered in chronological order.
- (2) The outputs are three candidates ranked by its confidence from high to low, including degree of confidence.

### 2. Properties:

- (1) Identification a letter will take about 0.1 second on average.
- (2) It can accept multiple stroke letters (such as a, d, e, F ...).
- (3) Lipitoolkit itself is written in C++ language.

### 3. Use restrictions:

- (1) Can recognition only one letter at a time.
- (2) Characters for identification are the 26 alphabets (though it can expand recognition set through training samples using its official tools provided).

## 4.6 Full Procedure

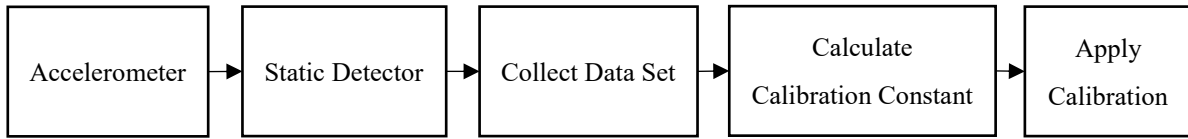


Figure 4-3: Sensor calibration flowchart

Figure 4-3 shows the flow of sensor calibration. First, NoDistort collects data from the accelerometer while the user place the device in six position. In the process, the Static Detector is used to distinguish if the device stays in the same place or have moved elsewhere. Second, NoDistort calculats the calibration constants in the Sensor Error Model by optimizing the cost function. Finally, the calibration is applied to the system for further use.

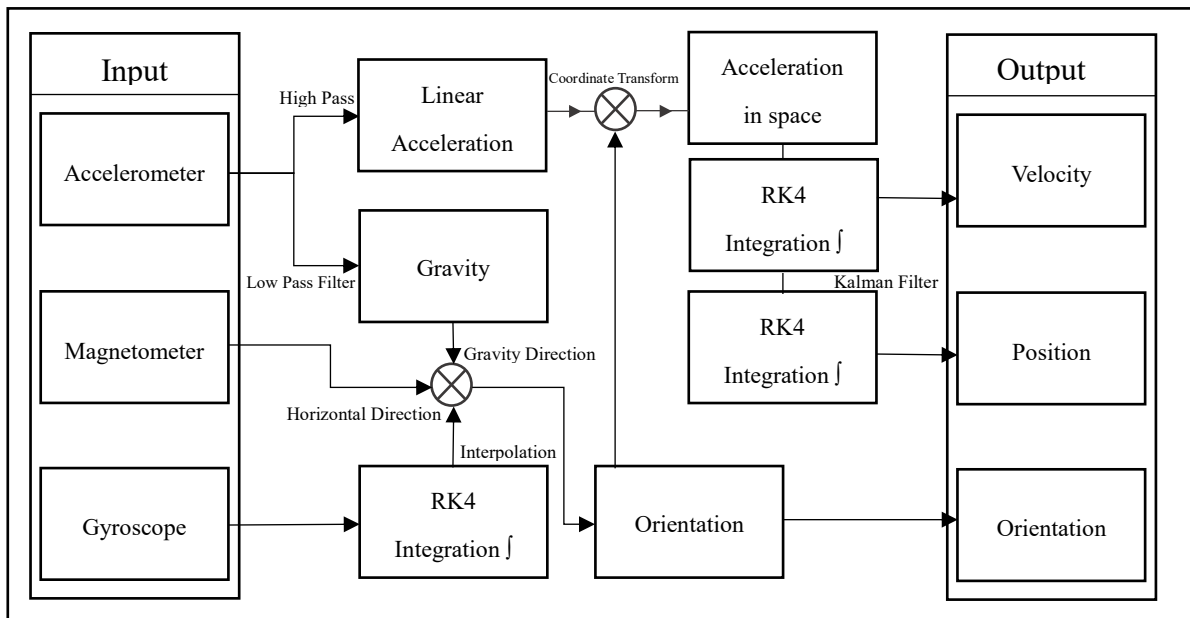


Figure 4-4: Motion estimation module flowchart

Motion Estimation (Figure 4-4) combines information from the accelerometer, gyroscope and magnetometer to compute the velocity, position, and orientation of the device. Through sensor fusion of gyroscope data and magnetometer data, the direction of magnetic north is determined. The gravity direction and north direction helps to determine the orientation of the device and through the cross product of the two, to construct a 3D coordinate axis system. Using data from the accelerometer, linear acceleration and gravity can be isolated by applying band

pass filters. Lastly, using RK4 integration and the constructed 3D axis, displacement and velocity of the device is then calculated from linear acceleration.

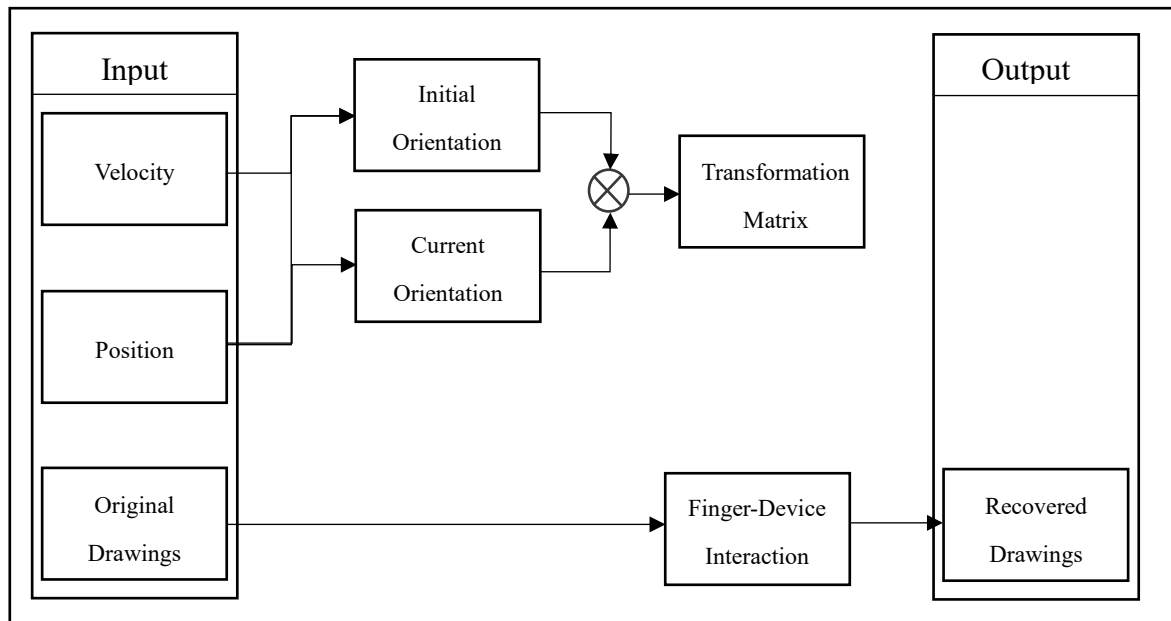


Figure 4-5: Drawing distortion Recovery Algorithm module flowchart

Given the actual position of the device in space and the distorted drawings received from the device's touch screen, my study proposes a Drawing Distortion Recovery Algorithm (Figure 4-5) to recover the drawings. It is based on the 3-Layer Model, and it maps the original drawings on the Touch Layer to the Drawing Layer.

## Chapter 5 Results and Discussion

The research includes experiments to obtain parameters and validate the performance of NoDistort. The experiment was done in an order of Sensor Calibration, Motion Estimation and Drawing Distortion Recovery Algorithm. Figure 5-1 shows the three phases of NoDistort and the parameters needed to be determined of each.

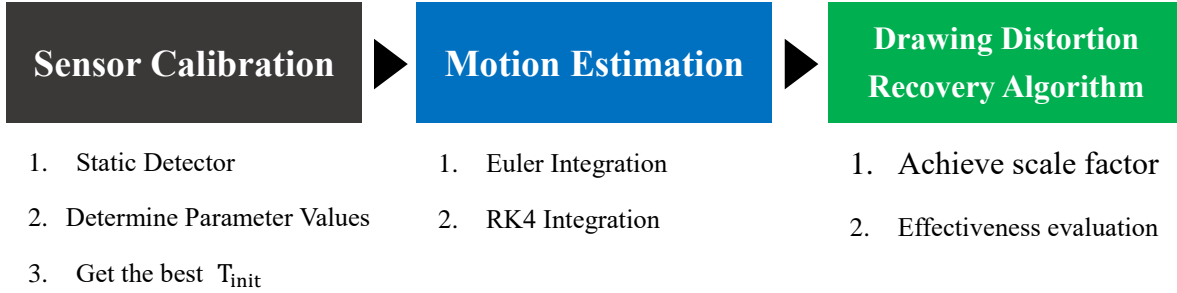


Figure 5-1: Experimental procedure

### 5.1 Sensor Calibration

1. Adjust Static Detector threshold:

Whether the static detector triggers or not depends on if the variation value in an observation windows is larger than the given threshold. If the threshold is too high, the observation windows is larger than the given threshold. If the threshold is too high, the Static Detector is less likely to be triggered; on the other hand, if the threshold is too low, the Static Detector is triggered too easily. This experiment tried different threshold size, trying to get a sensitivity and accuracy of the balance. Periodic motion generator used in this experiment will make the device produces two different types of motion; First, device is placed on a platform remaining static. Second, the device is moved by the Periodic motion generator.



Figure 5-2: Experiment Settings

Periodic motion generator produces 2cm forward, 2cm backward periodic motion. Within the generator stops 1 second, device enters a static state.

This experiment sets the threshold from 0.0004 to 0.005. Tested by an increment of 0.0001 each step. The experiment outputs are 3-axis acceleration values, the variance magnitude of the array in the observation windows and the status of Static Detector. Output of the Static Detector when the threshold was set to 0.0004 (Figure 5-3).

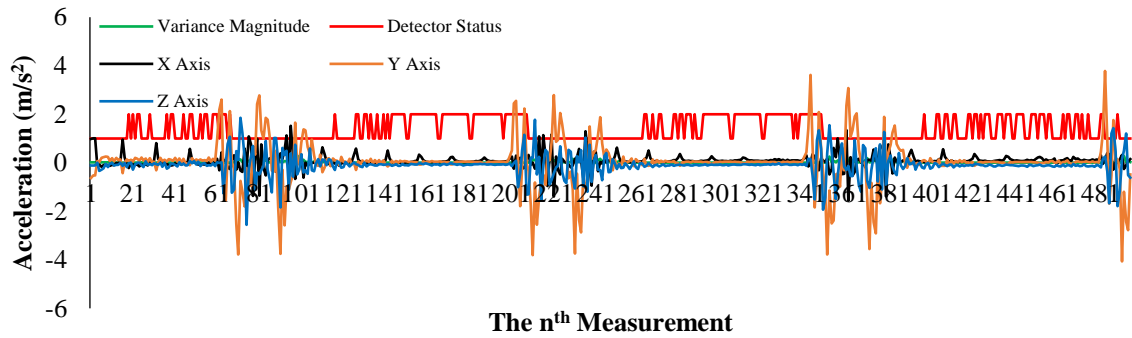


Figure 5-3: Static Detector threshold = 0.0004

The variation of three-axis less than 0.0004, static sensor outputs 2; If the variation of three-axis greater than 0.0004, static sensor output. When the static sensor threshold set to 0.0004, the sensor is too sensitive. Another extreme example of static sensor threshold is setting it 0.0005 (Figure 5-4).

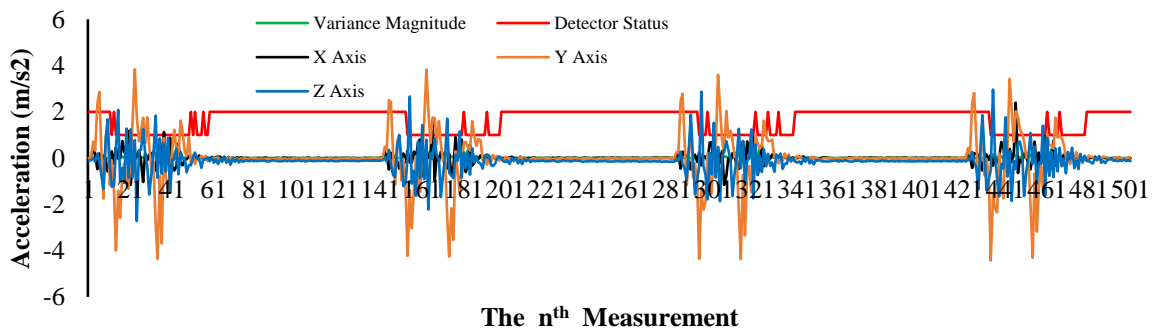


Figure 5-4: Static Detector threshold =0.0005

Setting the threshold is too high would result in low sensitivity of Static Detector. Sometimes even when the device is moving, Static Detector reports it static. In order to find the best static sensor threshold, the experiment tested values from 0.0004 to 0.0005,

every step increases by 0.0001. Finally, the experiment result below showed that the threshold of 0.0017 is the optimum (Figure 5-5).

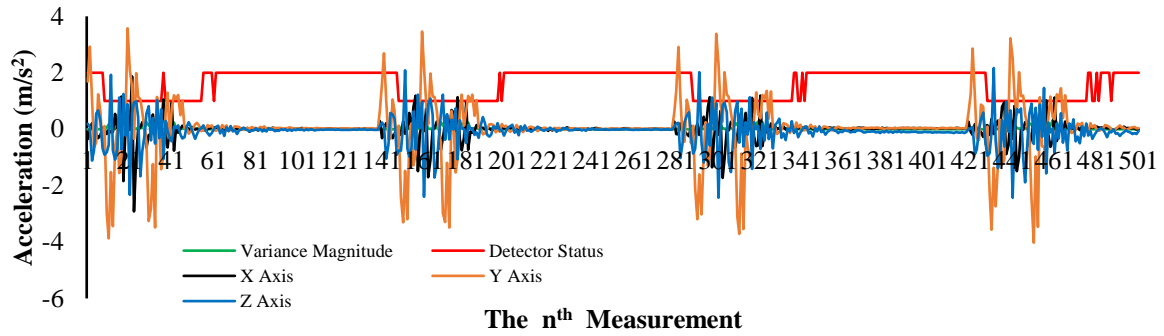


Figure 5-5: Static Detector threshold = 0.00017

The Static Detector accurately reported the device static (2 indicates the Static Detector was triggered) when the device was actually placed static.

## 2. Determine parameters of Sensor Error Model:

Using Algorithm 1 can determine parameter values in the Sensor Error Model. First flip and then place the device static 6 times. This will collect 6 sets of acceleration values of the device when placed in 6 different postures (Figure 5-6). Then the system calls Levenberg-Marquardt algorithm to minimize the cost function. Finally, it outputs and applies the most appropriate parameter values (Table 5-1).

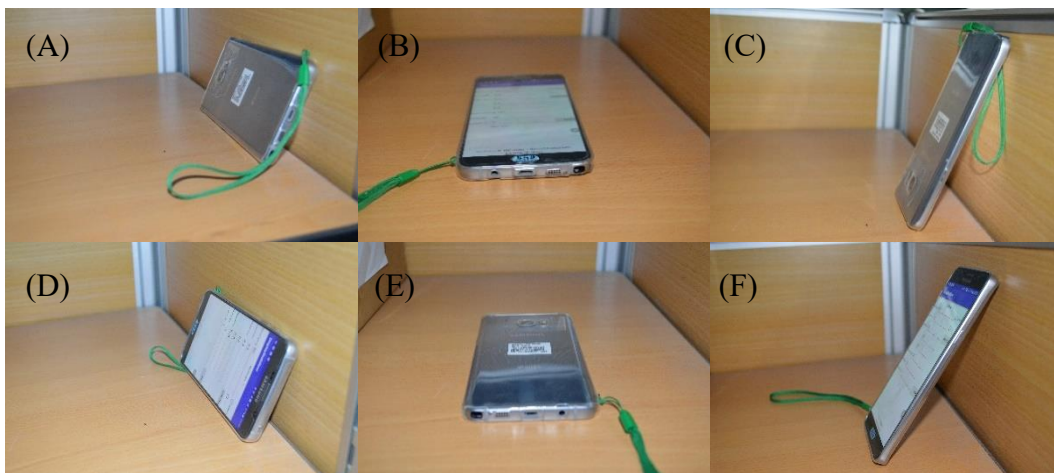


Figure 5-6: Performing Sensor Calibration for 6 different postures (A) to (F)

Table 5-1: Calibration parameter values

Times	Bias_X	Bias_Y	Bias_Z	Scale_X	Scale_Y	Scale_Z
First	0	0	0	1	1	1
Second	0	0	0	0.9999999	1	1
Third	0	0	0	1	1	1
Forth	0	0	0	1	1	1
Fifth	0	0	0	1	0.9999999	1

### 3. Calculate Allan variance $T_{init}$ :

By calculating Allan variance, the value of  $T_{init}$  can be determined. While the Allan deviation of the three-axis gradually reduced to the minimum value, the system can start to collect data at  $T_{init}$ . In this experiment, the phone is placed screen up on the flat desktop. When the program starts (Figure 5-7 (A)), the system starts at a 0.25-second observation window to Calculating Allan variance (Figure 5-7 (A)). At the moment of 20 seconds, the value of Allan Variance has minified. Thus, the time when NoDistort starts to fetch sensor data  $T_{init}$  was set to 20 seconds.

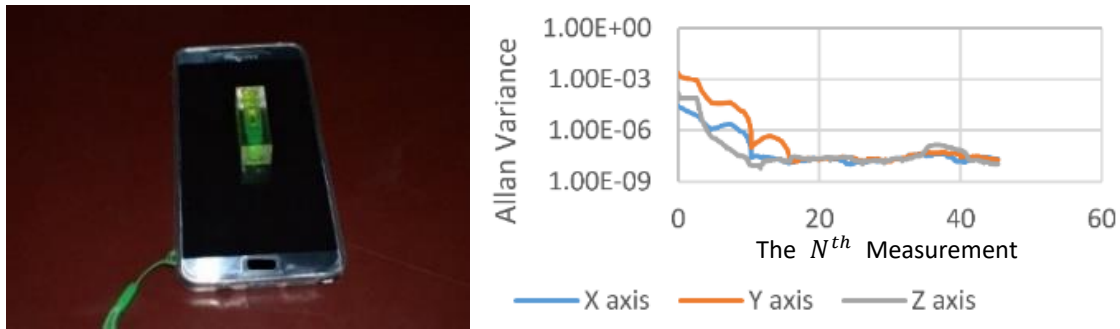


Figure 5-7: (A) Placing on flat surface, (B) the results of Allan variance

## 5.2 Motion Estimation

### 1. Motion estimation by integrating acceleration values using Euler method:

This experiment compared motion estimation performance by comparing the Euler Integration estimated motion and the actual displacement. The periodic motion generator

(Figure 5-8) moves the device back and forth periodically. The actual displacement was obtained by a stylus fixed in space and continuously draw on the device as it moves:

$$\Delta x = -x_t * R_{tm} \quad (5-1)$$

$$\Delta y = -y_t * R_{tm} \quad (5-2)$$

Where  $R_{tm}$  is the ratio of screen pixel to meters;  $x_t$  is the displacement of stylus on x axis;  $y_t$  is the displacement of stylus on y axis. Initial position was set to (0,0) every time. Then compare the estimated motion to the actual move.

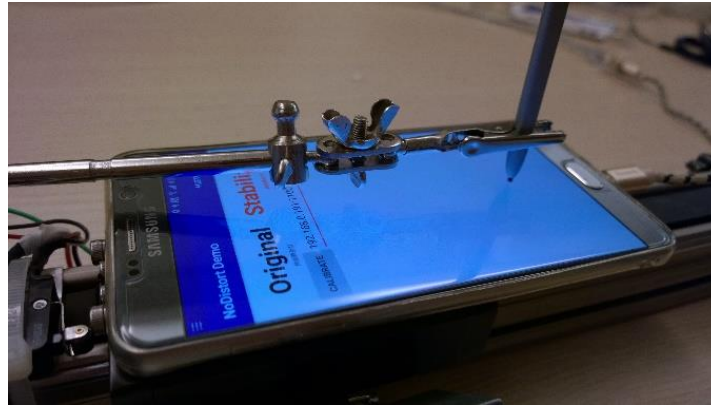


Figure 5-8: Periodic Motion Generator

The study makes use of cosine similarity technique to evaluate the similarity between the estimated curve and the exact curve to find the best suit filter parameters. Cosine similarity is a simple and effective method of measuring the similarity between two lines. Cosine similarity of the two lines is defined as Eq. (5-3).

$$\text{Cosine similarity} = \cos\theta = \frac{A \cdot B}{\|A\| \|B\|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}} \quad (5-3)$$

$A_i$  and  $B_i$  are vectors on A and B respectively. According to the experiment, the study uses cosine similarity to evaluate and optimize to achieve the best parameter combination for the Euler Integration method. As the result, the alpha value of the low pass filter is 0.8 and the alpha value of the high pass filter is 0.7.

This parameter combination generates the highest cosine similarity and is the closest to the original curve when using Euler Integration as the motion estimation method. Here Euler Integration successfully estimated device motion and reshape the movement curve.



## 2. Motion estimation using RK4 :

This experiment searches for the optimum filter parameters using the same way, but with RK4 method. Configuration (RK4,  $\alpha = 0.9$  low-pass filter, high-pass filter  $\alpha = 0.7$ ) is the parameter which makes the estimated curve closest to the original curve. Similarly, though the scale factor were not determined yet, by comparing the similarity the experiment concluded that Rk4 method can be successfully used to estimate motion, and it was more accurate than Euler Integration.

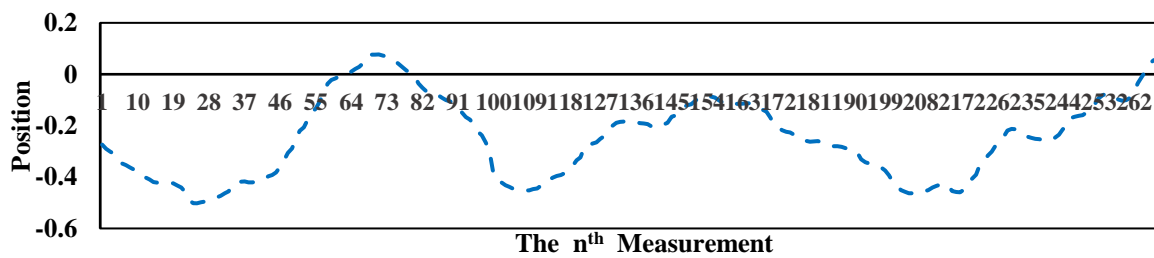


Figure 5-9: Motion estimation using Euler Integration

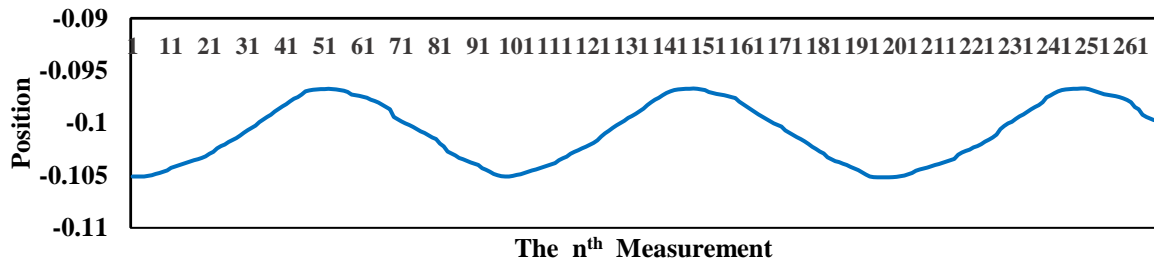


Figure 5-10: Actual movement of the Euler Integration experiment

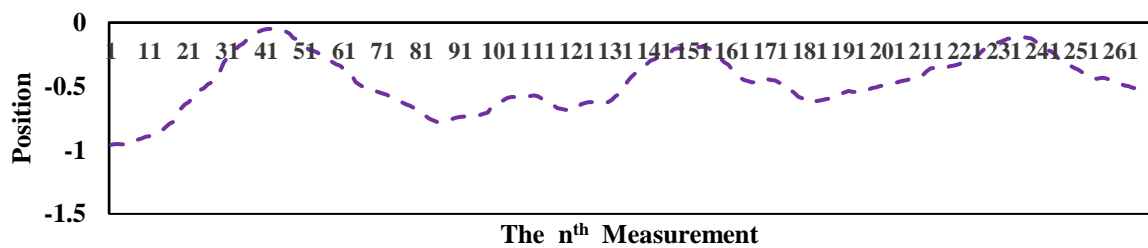


Figure 5-11: Motion estimation using RK4

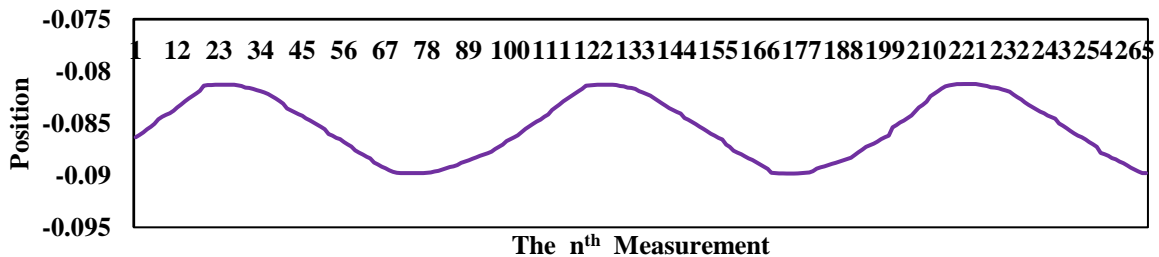


Figure 5-12: Actual movement of the RK4 Integration experiment

### 5.3 Drawing Distortion Recovery Algorithm

#### 1. Achieve scale factor :

In order to make the result more accurate, this study aimed to determine the scaling factor between the recovered and ideal one. The experiment followed the following steps.

##### (1) Define errors

Here error is defined as the distance of the ideal position and the recovered position.

$$\text{error} = \sqrt{(\text{staX} - \text{ideX})^2 + (\text{staY} - \text{ideY})^2} \quad (5-4)$$

where

$$\text{staX} = \text{stroX} - \text{posX} * \text{Multiplier} \quad (5-5)$$

$$\text{staY} = \text{stroY} - \text{posY} * \text{Multiplier} \quad (5-6)$$

*staX* is Recovered X position, *staY* is Recovered Y position, *ideX* is Ideal X position and *ideY* is Ideal Y position. Since the stylus was fixed in space throughout the experiment, *ideX* and *ideY* should remain 0.

##### (2) Adjust the scale factor from -1000 to 1000, with intervals of 50. Seek the scale factor to minimize the error:

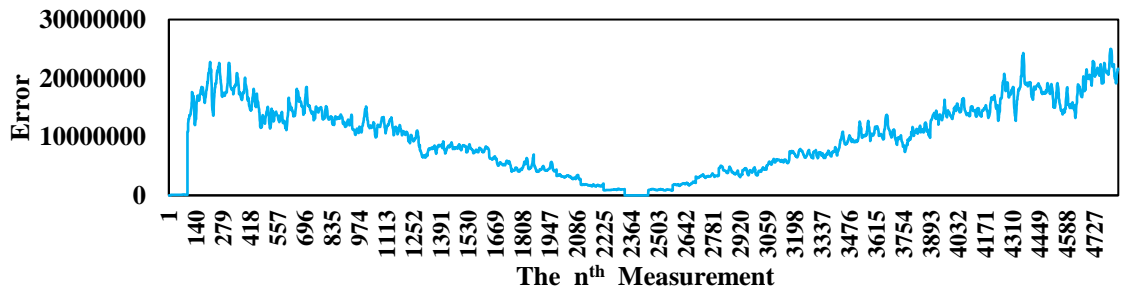


Figure 5-13: Spring-Damp-Mass Model Results

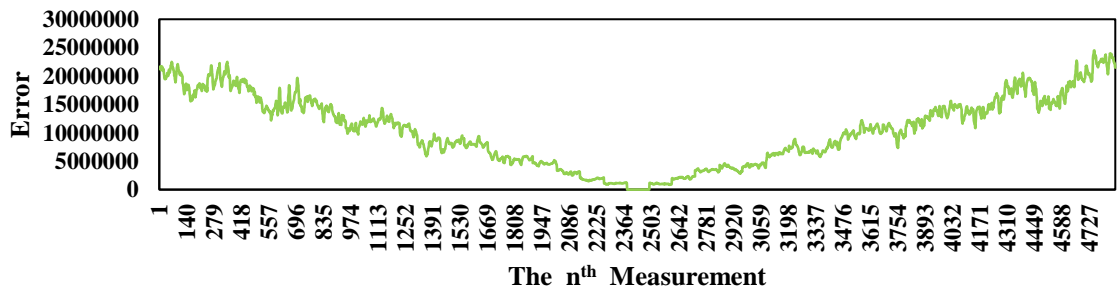


Figure 5-14: Euler Integration Results

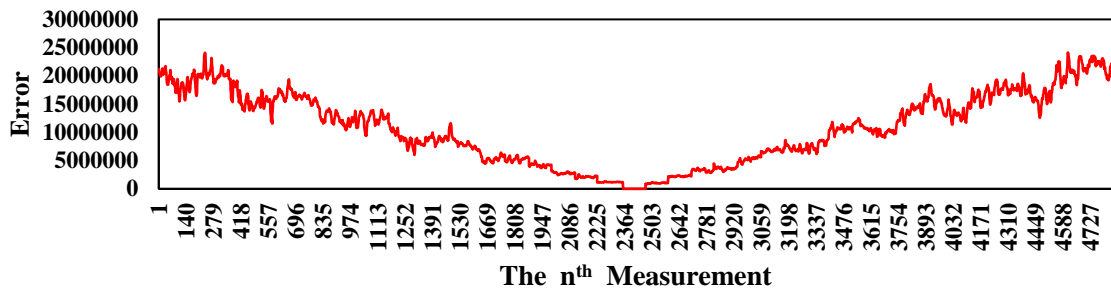


Figure 5-15: RK4 Integration Results

- (3) Adjust the scale factor from 0.2 to 0.9, with intervals of 0.05. Seek the scale factor to minimize the error:

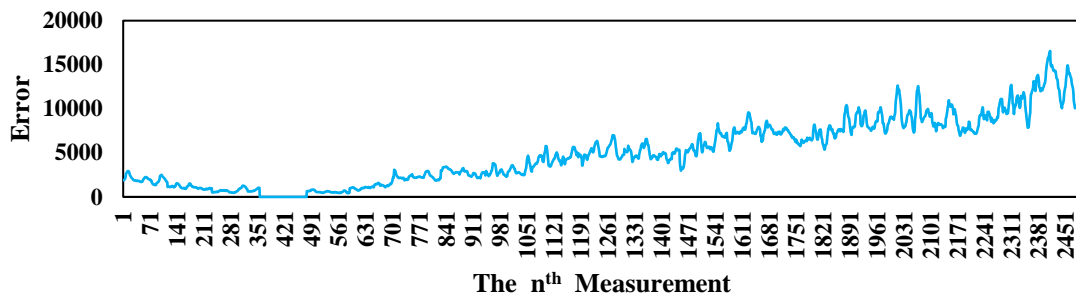


Figure 5-16: Spring-Damp-Mass Model Results

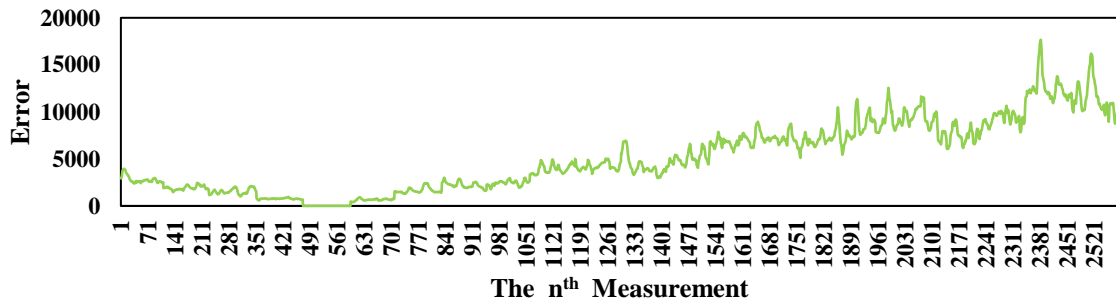


Figure 5-17: Euler Integration Results

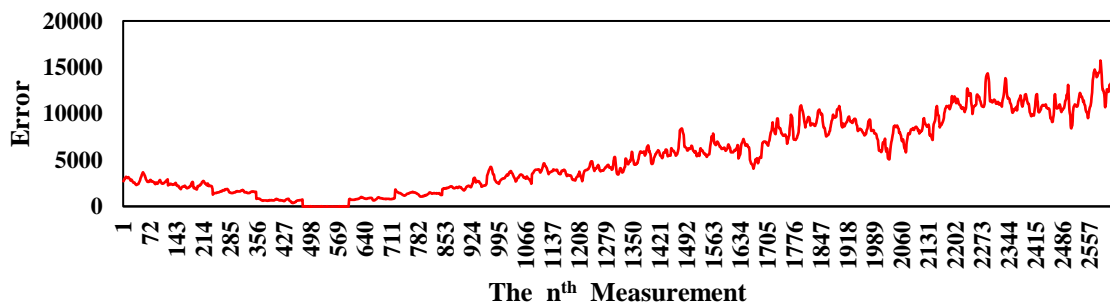


Figure 5-18: RK4 Integration Results

- (4) Adjust the scale factor from 0.05 to 0.06, with intervals of 0.001. Seek the scale factor to minimize the error:

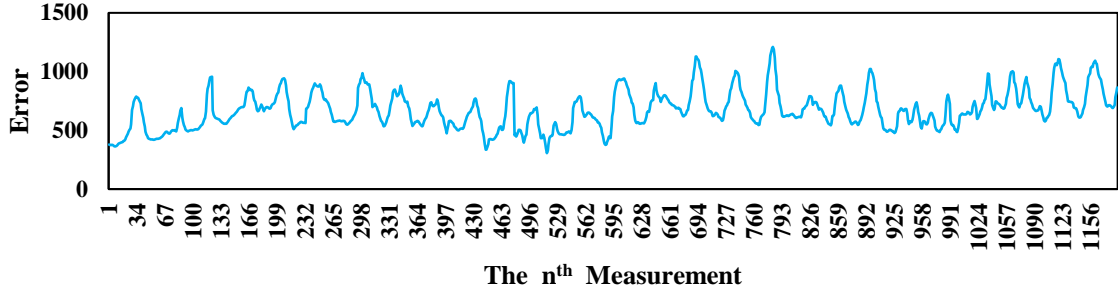


Figure 5-19: Spring-Damp-Mass Model Results

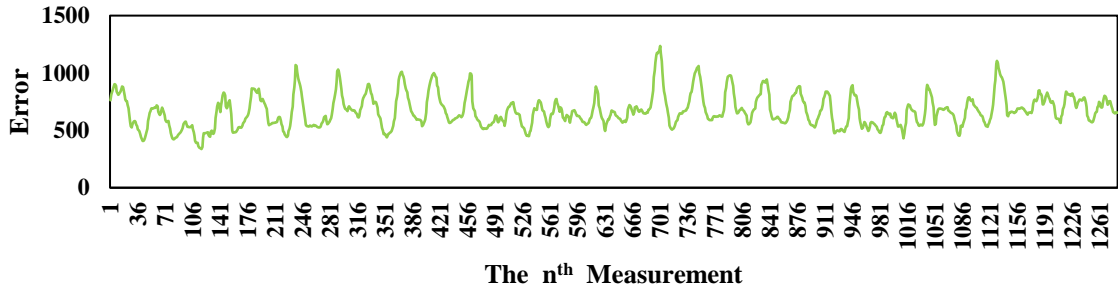


Figure 5-20: Euler Integration Results

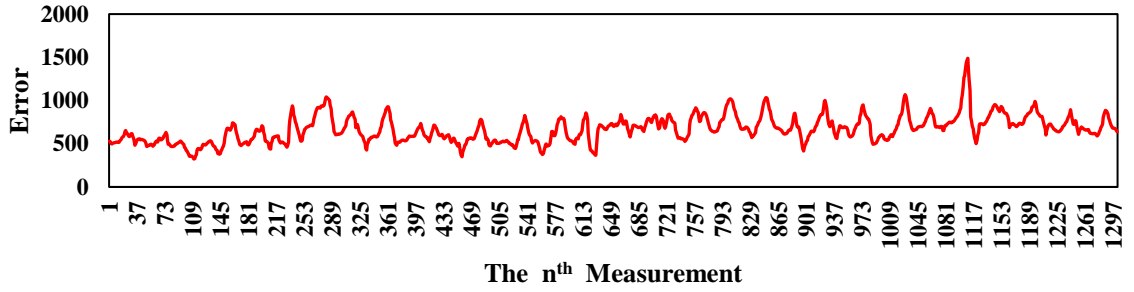


Figure 5-21: RK4 Integration Results

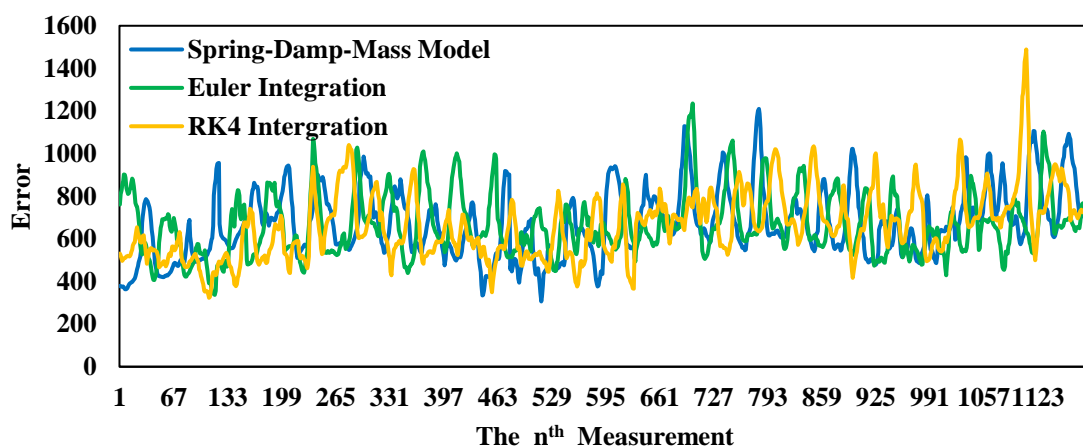
## 2. Evaluating effectiveness of NoDistort:

- (1) Compare the root mean square error between the recovered stroke and the ideal stroke.

Since the stylus is fixed at one point, ideal coordinates equals (0,0). Therefore, the performance of NoDistort can be assessed by the following procedure:

$$\text{error} = \sqrt{(\text{sta}X)^2 + (\text{sta}Y)^2} \quad (5-7)$$

The below shows the performance differences between different motion estimation methods. The smaller the error, the better performance it is.



The lowest error appears when using RK4 Integration, which is on an average of 503 pixels. Compared to error without recovery, which is 1036 pixels (standard location (0,0)), NoDistort reduced About 48.5% errors.

(2) Evaluate NoDistort performance by character recognition.

In order to measure the value of NoDistort in everyday life, this experiment evaluates the effectiveness of the NoDistort by comparing the character recognition success rate between NoDistort enabled and disabled.

Figure 5-23 shows the experiment tested five letters “a”, “b”, “c”, “d”, “e” individually. Every of which was tested through periodic motion generator generating random shaking within  $\pm 2\text{cm}$  for 100 times.

[illegible]

Figure 5-23: Evaluating the effectiveness of NoDistort by character recognition. (The black blocks indicate success, and the white blocks indicates failure.)

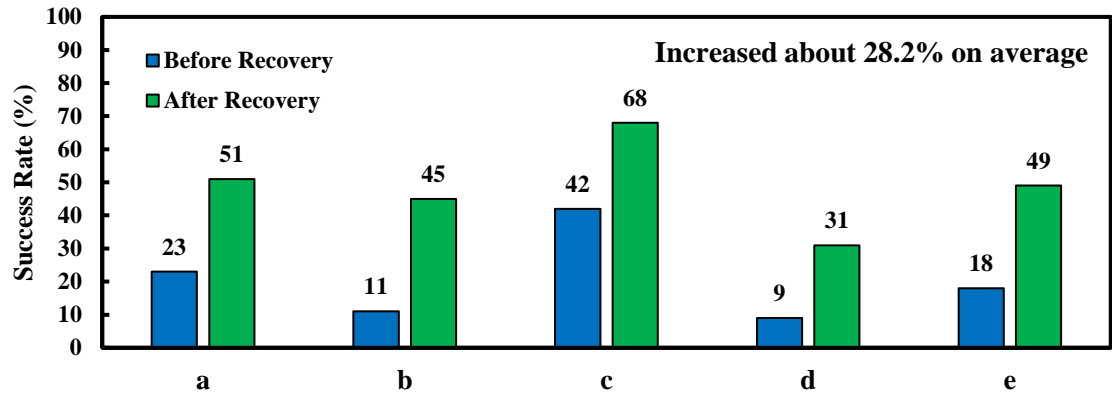


Figure 5-24: NoDistort increased the success rate of character recognition

Figure 5-24 shows after recovery of NoDistort, the success rate of character recognition raised about 28.2% on average, which was 2.34 times better than before recovery.

## Chapter 6 Conclusions

This study successfully developed NoDistort, and recovered distorted drawings in real life. NoDistort consists of three main steps: firstly, users do sensor calibration. Then the study proposed the Drawing Distortion Recovery Algorithm and recover distorted drawings using the motion estimation results. This study seeks for optimum motion estimation method by discussing different integration methods and parameter combinations. The performance of NoDistort was evaluated by calculating RMS error and character recognition. The experimental results show as below:

- (1) The system reduced about 48.5% errors within  $\pm 2$ cm shaking. (Figure 5-22)
- (2) The character recognition test results showed that within random shake of  $\pm 2$ cm, the success rate of recognition raised 28.2%, which was 2.34 times better than before recovery. (According to Figure 5-24)

NoDistort can be applied to any touch screen devices which has basic built-in sensors and have made great contributions to improving the user experience of writing. This system can be installed immediately into such as smart cars, cell phones, tablets and other devices. For handwriting recognition, NoDistort can significantly improve the success rate of the handwriting on touch screen devices. NoDistort can also be deeply integrated into character recognition systems and become an essential component to improving the writing experience.

This research will continue to improve and update. The following are the objectives:

- (1) Through increasing sensor accuracy, expand the availability of NoDistort.
- (2) Improved algorithm to improve accuracy and reduce resource consumption.
- (3) Write API, so that the majority of developers will benefit from NoDistort.
- (4) Add scenario recognition ability to dynamically adjust NoDistort recover strength.

A real life test of NoDistort shows as below. Black as the original handwriting, and red as the handwriting recovered by NoDistort. In the process, the device moves from right to left. The recognition success rate increase significantly after recovery by NoDistort.

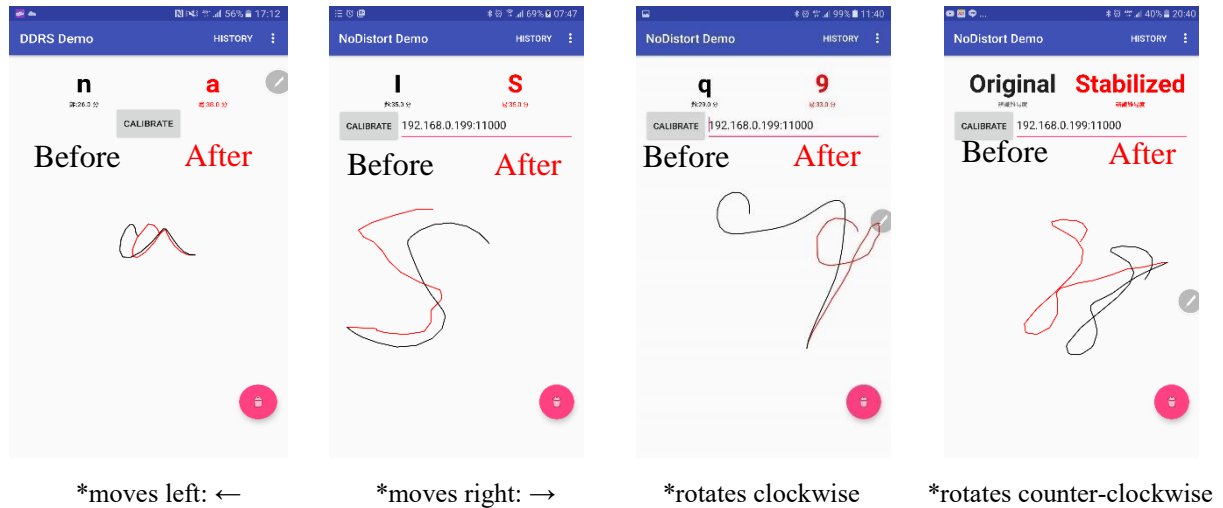


Figure 6-1: NoDistort Demonstration



## Chapter 7 References and Appendix

- [1] Do, T. M. T., Blom, J., & Gatica-Perez, D. (2011). *Smartphone usage in the wild: a large-scale analysis of applications and context*. In Proceedings of the 13th international conference on multimodal interfaces (pp. 353-360). ACM.
- [2] KPCB. (2015). *INTERNET TRENDS 2015 - CODE CONFERENCE*. Retrieved from [www.KPCB.com](http://www.KPCB.com):<http://www.KPCB.com/Internet-trends>
- [3] Hou, H. (2005). *Modeling inertial sensors errors using Allan variance*. Library and Archives Canada Bibliothèque et Archives Canada.
- [4] Menegatti, L. P. E., Tedaldi, D., & Pretto, A. (2013). *IMU calibration without mechanical equipment*.
- [5] Rahmati, A., Shepard, C., & Zhong, L. (2009). *NoShake: Content stabilization for shaking screens of mobile devices*. In Pervasive Computing and Communications, 2009. PerCom 2009. IEEE International Conference on (pp. 1-6). IEEE.
- [6] Barnard, L., Yi, J. S., Jacko, J. A., & Sears, A. (2005). *An empirical comparison of use-in-motion evaluation scenarios for mobile computing devices*. International Journal of Human-Computer Studies, 62(4), 487-520.
- [7] Mathie, M. J., Coster, A. C. F., Lovell, N. H., & Celler, B. G. (2003). *Detection of daily physical activities using a triaxial accelerometer*. Medical and Biological Engineering and Computing, 41(3), 296-301.
- [8] Goel, M., Findlater, L., & Wobbrock, J. (2012). *WalkType: using accelerometer data to accomodate situational impairments in mobile touch screen text entry*. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (pp. 2687-2696). ACM.

- [9] Syed, Z. F., Aggarwal, P., Goodall, C., Niu, X., & El-Sheimy, N. (2007). *A new multi-position calibration method for MEMS inertial navigation systems*. Measurement Science and Technology, 18(7), 1897.
- [10] Skog, I., & Händel, P. (2006). *Calibration of a MEMS inertial measurement unit*. In XVII IMEKO World Congress (pp. 1-6).
- [11] Shin, E. H., & El-Sheimy, N. (2002). *A new calibration method for strapdown inertial navigation systems*. Z. Vermess, 127, 1-10.
- [12] Ozyagcilar, T. (2012). *Calibrating an ecompass in the presence of hard and soft-iron interference*. Freescale Semiconductor Ltd.
- [13] Kok, M., & Schön, T. B. (2016). *Magnetometer calibration using inertial sensors*. IEEE Sensors Journal, 16(14), 5679-5689.
- [14] Welch, G., & Bishop, G. (1995). *An introduction to the Kalman filter*.
- [15] Ribeiro, M. I. (2004). *Kalman and extended kalman filters: Concept, derivation and properties*. Institute for Systems and Robotics, 43.
- [16] Jimenez, A. R., Seco, F., Prieto, C., & Guevara, J. (2009). *A comparison of pedestrian dead-reckoning algorithms using a low-cost MEMS IMU*. In Intelligent Signal Processing, 2009. WISP 2009. IEEE International Symposium on (pp. 37-42). IEEE.
- [17] Ayazi, F. (2011). *Multi-DOF inertial MEMS: From gaming to dead reckoning*. In Solid-State Sensors, Actuators and Microsystems Conference (TRANSDUCERS), 2011 16th International (pp. 2805-2808). IEEE.
- [18] Chen, J. H., Lee, S. C., & DeBra, D. B. (1994). *Gyroscope free strapdown inertial measurement unit by six linear accelerometers*. Journal of Guidance, Control, and Dynamics, 17(2), 286-290.
- [19] Popović, L. Z., Šekara, T. B., & Popović, M. B. (2010). *Adaptive band-pass filter (ABPF) for tremor extraction from inertial sensor data*. Computer methods and programs in biomedicine, 99(3), 298-305.

- [20] Hide, C., Moore, T., & Smith, M. (2003). *Adaptive Kalman filtering for low-cost INS/GPS*. The Journal of Navigation, 56(01), 143-152.
- [21] Barshan, B., & Durrant-Whyte, H. F. (1995). *Inertial navigation systems for mobile robots*. IEEE Transactions on Robotics and Automation, 11(3), 328-342.
- [22] Savage, P. G. (1998). *Strapdown inertial navigation integration algorithm design part 1: Attitude algorithms*. Journal of guidance, control, and dynamics, 21(1), 19-28.
- [23] Savage, P. G. (1998). *Strapdown inertial navigation integration algorithm design part 2: Velocity and position algorithms*. Journal of Guidance, Control, and Dynamics, 21(2), 208-221.

## 【評語】 190016

此作品開發手機系統上，校正手寫時，因移動所造成自行無法辨識的問題，此題目很有趣。此作品的實驗完整，非常用心，目前的作品限制在輸入的手部移動，只有持手機那手移動。建議未來可讓使用情境再更有彈性。

Intel 特別獎評語：

1. 很好穿的應用，很實際的解決現實生活中的問題。
2. 同樣的演算法也可應用到中文及其他語言的輸入法。
3. 建議可以直接找輸入法業者，將此演算法放到輸入法中。
4. 書寫的精準度也可加強。