使用深度影像及FEMD改良方法之手指偵測

王國華　高振傑

輔仁大學資訊工程系所

khwang@csie.fju.edu.tw　zerox99@csie.fju.edu.tw

摘　要

本論文針對部分手勢識別系統(Part-Based Hand Gesture Recognition System)中所使用之FEMD(Finger-Earth Mover's Distance)手指偵測方法進行改良，原本之手勢辨識系統是以彩度影像作為原始圖像輸入，並利用固定閾值分解(Threshing Decomposition)方法去除FEMD線圖中的手掌雜訊。本論文提出之改良方法分成兩方面，首先，將輸入圖形由彩度影像更換成深度影像，此舉在進行背景切割與膚色偵測不全時，將有效降低手部圖像所產生的雜訊；另一方面，我們以掌心圓相對距離閾值分解取代固定閾值分解，此閾值分解方法在以不同人的手掌及不同手勢作辨識時，可即時對應最佳閾值並降低因手掌改變所產生之雜訊。根據實驗結果，我們所提出之改良方法可以大幅降低FEMD線圖雜訊，辨識率可從92.9%提升到98.2%，而平均辨識時間不超過34毫秒。

關鍵字：手勢辨識、深度影像、手指偵測、閾值分解

Finger Detection by Renewed FEMD Method using Kinect Depth Camera Image

王國華　高振傑

輔仁大學資訊工程系所

khwang@csie.fju.edu.tw　zerox99@csie.fju.edu.tw

Abstract

In this paper, we improve the Finger-Earth Mover Distance (FEMD) gesture recognition method used in「Part-Based Hand Gesture Recognition System」. This recognition system uses RGB image as the input image and fixed threshold for the threshing decomposition. The proposed improvements consist of two aspects. Firstly, the hand gesture images are captured with the depth image rather than RGB image, which can effectively reduce the noise on hand image while doing the background subtraction and skin color detection failure. Moreover, for the Threshing Decomposition, we replace the fixed threshold by the relative distance threshold to the circle of the palm. It can immediately compute the corresponding optimal threshold for the recognition of different hand gestures with respect to various types of palms. By our experimental result, our proposed method can increase the recognition rate from 92.9% to 98.2% and the average recognition time is less than 34 ms.

關鍵字：Depth Image, Finger Detection, FEMD, Threshing Decomposition

1. 序論

近年來，虛擬實境(Virtual Reality, VR)與擴增實境(Augmented Reality, AR)的應用崛起，相關的研究很多，其中包含人機互動介面(Human-Computer Interaction, HCI)，此研究的目的為加強人與機器之間的自然交流，人機互動有很多種方式，而人與機器間之介面對於互動便利性影響很大。由於科技持續的進步，人機互動的方式逐漸偏向自然用戶介面(Natural-User Interface, NUI)[1]。

對於使用者而言，採用手勢與機器互動的方式較為直觀自然，且靈活性也相對地高，因此目前的體感偵測技術多偏向於手勢偵測與辨識。一般而言，獲取手勢最有效的方式是以電子手套或其他接觸型感測器[2]獲取輸入資訊，但這些裝置通常都很昂貴，而且容易阻礙手勢的運動以及需要複雜的校準與設定，所以目前的手勢辨識技術仍以視覺分析方式為主。

針對部分手勢辨識系統[3]有兩方面的問題點：

(1) **視覺分析手勢辨識其前置作業繁雜，難以運用於即時系統中**。

首先，RGB 彩度影像容易受光線與環境因素干擾，在抓取手部影像時易造成空洞雜訊，進而增加影像處理前置作業的負擔。除此之外，以論文[6]中所提出的手部擷取技術為例，傳統的視覺分析手勢辨識方法[4][5]需要大量的前置作業，故難以運用於即時系統中。而且得出的手部處理區域也可能會包含手臂的部分，在傳統以直方圖(histogram)或提出形狀對照為主的識別方法[8][9]上，勢必需要切割手臂和手掌[7]，並旋轉其方向及計算相對距離，才能夠得出手勢辨識的結果。

(2) **固定閾值分解方法於系統中，無法即時對應到不同的手勢及不同人的手掌**。

其次，論文[3]中所提出的 FEMD(Finger-Earth Mover's Distance)手指分析演算法，其以閾值分解(Threshing Decomposition)方法去除 FEMD 線圖中手掌雜訊，但因其閾值為固定值，造成此方法無法即時對應到不同的手勢及不同人的手掌。

針對上述問題提出兩方面的改良，第一將原始輸入圖像從彩度影像換成深度影像，大幅降低手部擷取的流程，加速效能的提升。第二改寫原本部分手勢辨識系統[3]中取閾值的方式，降低雜訊的產生率，從原本固定值閾值取法，改由每一次取平均距離閾值，或是每一次進行掌心圓相對距離的計算方式，此舉讓辨識成功率向上增加，但運算時間也會有些許增加。希望藉由此方式能夠對加速人機互動手勢辨識運算流程，在將來使手勢辨識能夠更普遍運用於各式即時人機互動系統上。

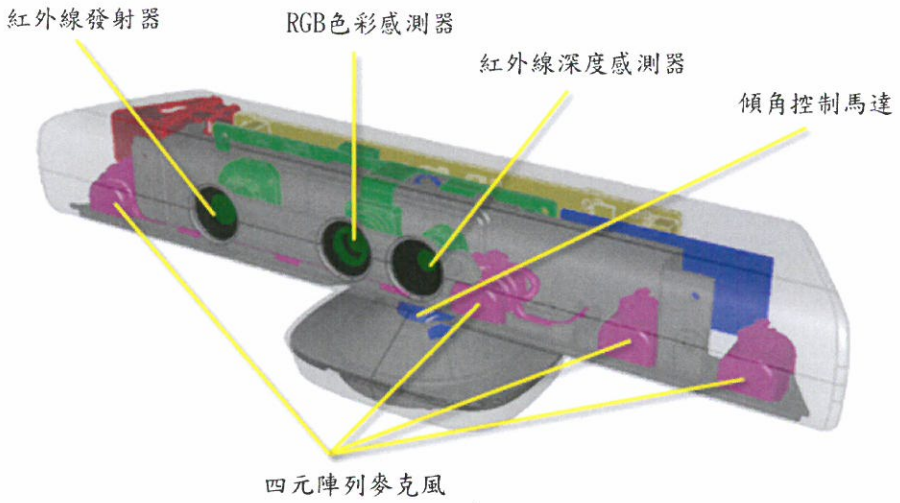
本論文其餘部分內容如下：第二節介紹與本論文相關的背景知識，包括所使用的裝置、開發工具、演算法及研究想法，及本論文所參考的部份手勢辨識系統[3]；第三節將介紹本論文所提出手勢識別系統流程及閾值計算；第四節將詳細介紹實驗的流程，並詳加敘述實驗結果與討論分析；第五節則總結本論文的成果及未來可改進之研究工作。

2. 相關背景知識

本節介紹手勢辨識系統相關研究，將依序說明 Kinect 裝置、手勢辨識相關研究、手指分析演算法及部分手識辨識系統。

2.1 Kinect 介紹

Kinect[10][11]為微軟所開發，一開始作為 Xbox360 與 Xbox one 上的操作設備，藉由感測玩家身體的動作與聲音，帶給玩家「不需使用控制器的遊戲娛樂體驗」。圖一是 Kinect 的元件示意圖，Kinect 感應器是一台整合式感應器，擁有三個鏡頭，中間的鏡頭是 RGB 彩色攝影機，左右兩邊鏡頭分別為紅外線發射器與深度感應器，內部則有陣列式麥克風，由多組麥克風同時收音，可以相互比對並消除雜音。



圖一： Kinect 內部元件示意圖

2.2 手勢辨識相關研究

本節介紹在實作手勢辨識時，所需要的函式庫及演算法。包括以 EmguCV 作為系統中影像處理的函式庫、4Y-Model 的生物特徵條件辨識手指方法及 K 曲率法則的概念。

(1) **EmguCV**

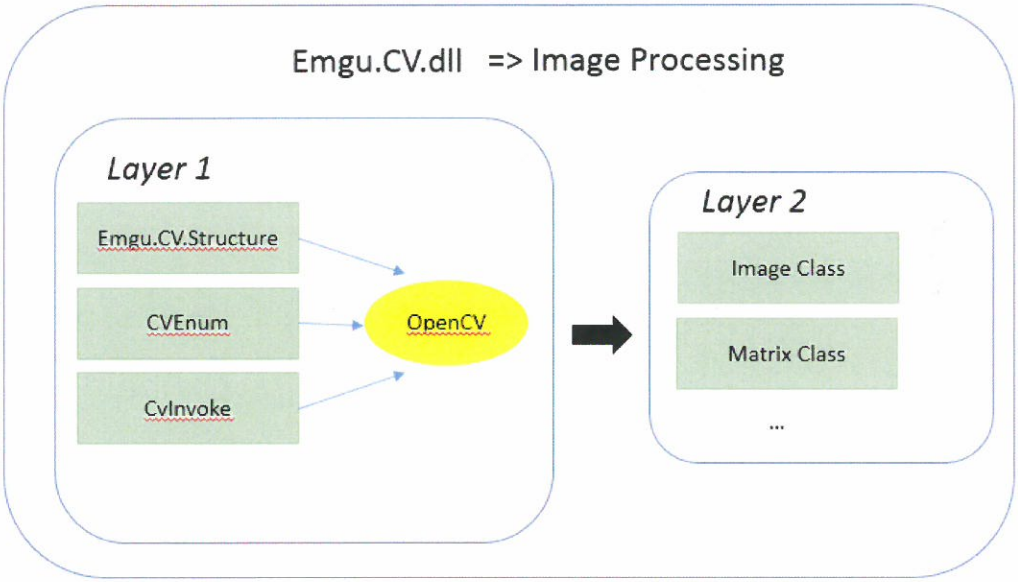
EmguCV 是包裝 OpenCV 的跨平台.Net 影像處理函式庫[12]，讓 Visual Studio、Xamarin Studio、Unity 編譯器使用 OpenCV 的內建函式，其架構可參照圖二。由圖二中可以得知 EmguCV 有兩層封裝，第一層是基本層，直接對應 OpenCV 包括函數與結構。第二層是擴充層，包括與.NET 庫的一些混合類別，使 EmguCV 有著適用於.NET 的優勢。

(2) **4Y-Model**

4Y-Model[13]是由所 Ahmed Al Marouf 提出，講述的是一種手指辨識的演算流程及辨識方式，這種手指識別演算法是基於幾何計算和一般人類手部形狀的生物特徵。

(3) **K 曲率法則(K-Curvature)**

K 曲率法則[14]主要是用在找尋指尖與指縫的位置，方式為使用一連續的手指邊緣座標點陣列，找尋序列上座標點前後 K 間隔的兩點座標求出其向量，以內角公式由計算兩個向量中間夾角的角度。



圖二： EmguCV 架構圖

2.3 Finger-Earth Mover's Distance (FEMD 手指分析演算法)

FEMD[3]手指分析的演算法源自 Earth Mover's Distance (EMD)方法[15]，EMD 常用於基於內容為主的圖像檢索與辨識[16]。FEMD 方法可以有效改善 EMD 用於手勢偵測的缺點，EMD 無法藉由局部特徵(Local Features)來正確辨識每根手指，且大量的局部特徵容易減慢圖像的對照，以及 EMD 局部對照的方式在許多情況下是不合理的，仍容易造成對照的混淆出現錯誤。而 FEMD 為了解決上述問題則是採用的整體特徵(Global Features)為群集做運算與比對，並對手掌的空洞雜訊(Empty Hole)做出修正已降低其對照需求。

圖三為 FEMD 實作示意圖，FEMD 的運作方式為先從影像中切割出手掌，得到圖三(a)原始手部圖像，其中圖三(a)紅點為掌心，綠點為手掌邊緣起始點，將紅點至綠點的角度設為初始角度，然後依據手掌掌心到手掌邊緣的角度與距離畫出如圖三(b)的線圖。

2.4 部分手勢識別系統(Part-Based Hand Gesture Recognition System)

本節介紹部分手勢辨別系統(Part-Based Hand Gesture Recognition System)[3]，其流程主要包含手部偵測與手勢識別兩個模組。

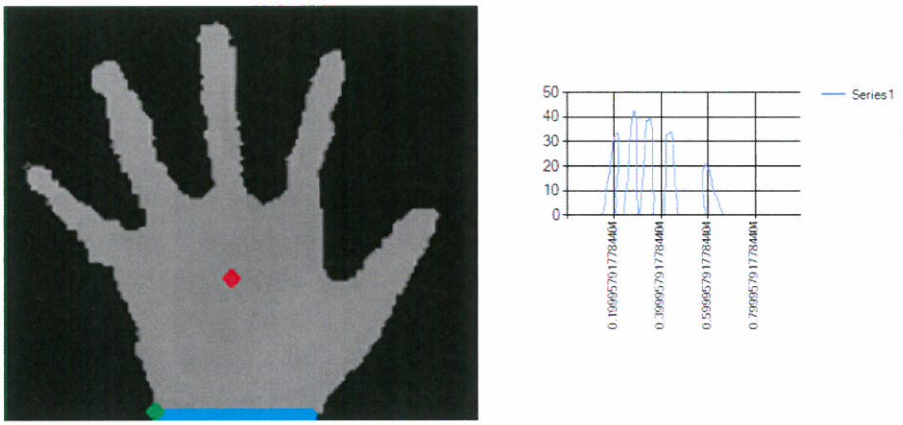
(1) **手部偵測**

部分手勢辨識系統使用 Kinect 作為影像資料的輸入裝置，並為了區別手部與手臂的部分，此系統要求在使用者手腕繫上一條黑色的色帶，以便在影像處理時可以快速將手臂與手掌

的影像切割。手部偵測分成兩步驟，第一步驟是抓取手部位置，開啟 Kinect 中的對人體追蹤功能，藉由一定深度間隔的距離中取得大略的手部位置。第二步是為了更準確的細分手部形狀，利用隨機抽樣一致演算法(Random Sample Consensus, RANSAC)偵測黑色色帶的位置，再檢測手型之後，以 FEMD 紀錄輪廓邊緣點和中心點之間的相對距離，得出 FEMD 的線圖。FEMD 線圖可以參考圖三(b)，圖中橫軸表示輪廓邊緣點，並且相對於中心點的開始點，通過 360°之間的角度。縱軸表示該輪廓的頂點和中心點，由最大內切圓的半徑為標準的歐幾里德距離。

(2) **手勢識別**

部分手勢辨識系統[3]中使用兩種方式來做為手勢識別的方法。第一種為模板比對，利用所輸出的 FEMD 線圖與本身建立的模板做比對，將差異度最小的模板視為所辨識出的手勢形狀。第二種為利用所出的 FEMD 線圖進行手指的偵測，以手指的位置與其相對的距離做出比較。而手指辨識的方法，將原本提出的最小近凸分解演算法(Minimum near-convex decomposition algorithm)做出修正，改為閾值分解法(Threshing Decomposition)。原因在於雖然近凸分解算法能夠準確地檢測手指部分，其運算過程是非常複雜地運作，並不能運用於實時系統之中。閾值分解法的原理在於不必準確計算決定 FEMD 線圖切割手指的位置，而是直接訂定一個標準閾值，只要 FEMD 線圖中高於閾值的部份都視為手指，藉以此種方式可以快速的檢測手指，提升運算速度與減少運算所需的流程。



(a)原始手部圖　　(b)FEMD 線圖

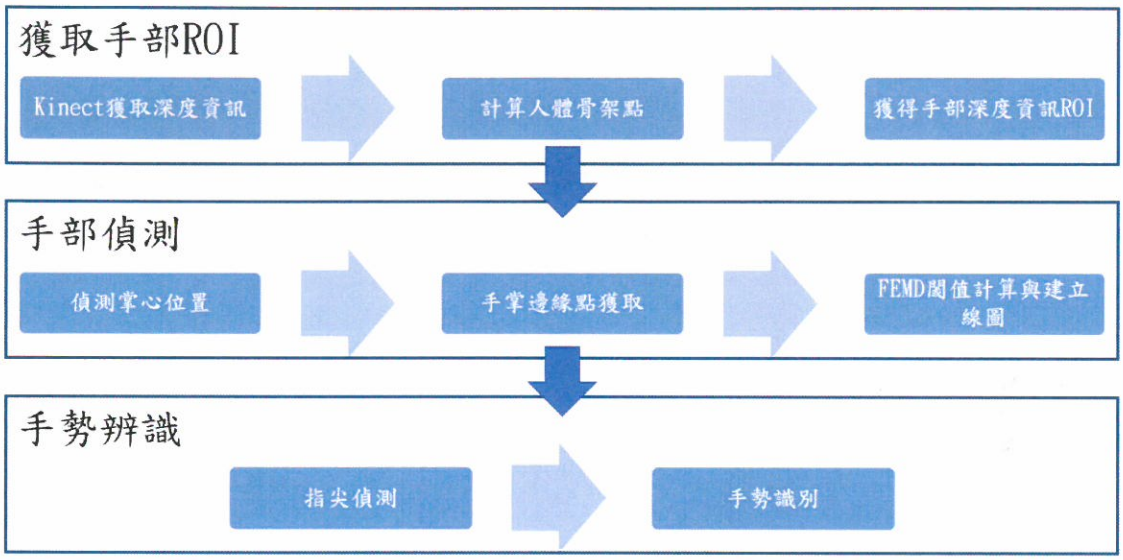
圖三： FEMD 實作示意圖

3. 手勢辨識系統

本節將介紹本論文所建置之手勢辨識系統架構與系統運作方法，內容包括如何擷取原始輸入的手部圖像、改良之 FEMD 手指偵測演算法及手勢辨識方法。

3.1 系統流程

圖四為系統執行的流程圖，主要分成三大步驟：擷取手部 ROI、手部偵測與手勢辨識。



圖四： 系統執行流程圖

3.2 擷取手部圖像

本論文提出之手勢辨識系統中只需要深度圖像的分析資料，相對於部分手勢辨識統[3]之 RGB 影像更為簡單。不過在擷取畫面內只可以有一個人的影像，不然容易造成系統誤判。

(1) **啟動深度影像感測器**

開啟 Kinect 的深度串流功能，利用 Kinect 得到深度串流影像。本系統只需擷取人物的上半身的影像資料，所以人物與鏡頭的距離會非常靠近，為了不讓深度影像失真與誤算深度距離，在啟動深度串流功能時，必須開啟深度串流的 Near Mode 設定。Near Mode 可以將深度距離有效範圍定在 400~3000 毫米，讓系統在近距離的上半身深度偵測時，不會變成 TooNearDepth 或 UnknowDepth 的數值，以至於在進行 FEMD 手指偵測時無法讀出數值。

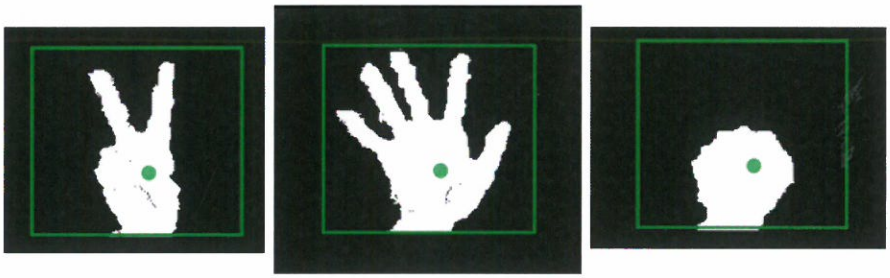
(2) **計算人體骨架點**

當 Kinect 要取得人物的骨架時，則必須開啟 Kinect 骨架串流(SkeletonStream)功能，並設定 Seated Mode。設定 Seated Mode 的理由為系統需求僅需要抓取人物的上半身。因為原先 Default 設定的演算方式，無法在深度影像開啟 Near Mode 情況下計算骨架，Near Mode 的設定會造成骨架串流無法正確的獲取人物的骨架關節點，所以在此必須設定骨架串流的近距離骨架辨識模式(TrackingInNearRange)。而且藉由近距離(Near Mode)模式搭配坐姿模式，可以增加骨架追蹤的精確度。

(3) **抓取手部深度 ROI**

本系統 ROI 的獲取方法為藉由內含玩家編號的深度影像，利用骨架串流的演算法分別找出第一位人物上半身 10 個關節點位置以得到關節 JointType 為 HandLeft 的座標點(HLx, HLy)。這裡 HandLeft 座標點定為中心，以長 145*寬 135 pixel 的長方形作為深度影像手部

ROI 範圍，這裡將獲取的深度影像 ROI 範圍定名為 HLROI。圖五觀察各種手勢下的 HLROI 與 HandLeft 的相對關係示意圖，圖五各圖中綠點代表 HandLeft 的座標點，而綠色方框則代表興趣區域的 HLROI。此處 HLROI 先以 WriteableBitmap 資料型態儲存，WriteableBitmap 大小為 145*135，本論文定名為 bitmap。而 HLROI 裡面的內容為以深度影像 ROI 範圍對照，其深度值大於 400 釐米以及小於 1000 釐米之深度值以白色(RGB 三值皆為 0xFF)數值存入 bitmap 之中，而其他像素則以黑色儲存 (RGB 三值皆為 0)儲存。



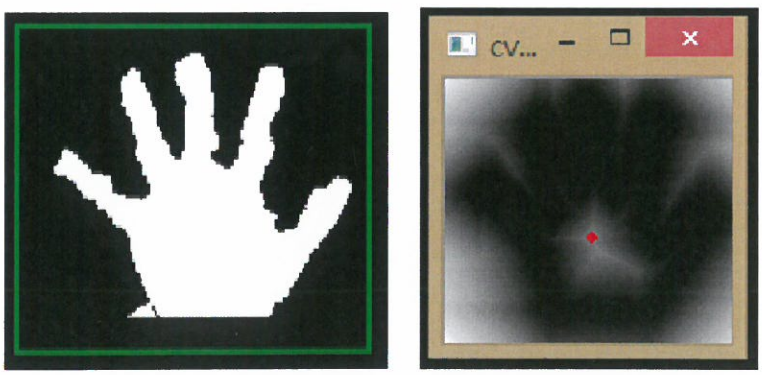
圖五： 不同手勢 ROI 圖像

3.3 FEMD 手指偵測

本系統中 FEMD 手指偵測的演算法使用 EmguCV 電腦視覺函式庫作相關的影像處理。在此處理過程中，必須先使用 EmguCV 的影像轉換函式轉換成 Image 型態。此轉換過影像定名為 sourceImage，以利接下來 FEMD 運算。

(1) **偵測掌心位置**

在進行 FEMD 的運算時，必須先取得手掌心的位置，圖六為掌心偵測的示意圖，掌心偵測的計算方法是以圖六(a)的原始圖像 sourceImage 呼叫 EmguCV 中的 cvNormalize(均一化濾波)函式，將其映射至[0,1]之間，可以得到圖六(b)NormalizeImage。再以 NormalizeImage 為輸入呼叫 EmguCV 中的 cvMinMaxLoc 函式，找出影像中的最大值(maxPoint)、最小值(minPoint)以及其位置，而 maxPoint 將會是掌心的位置，如圖六(b)中紅點位置所示。

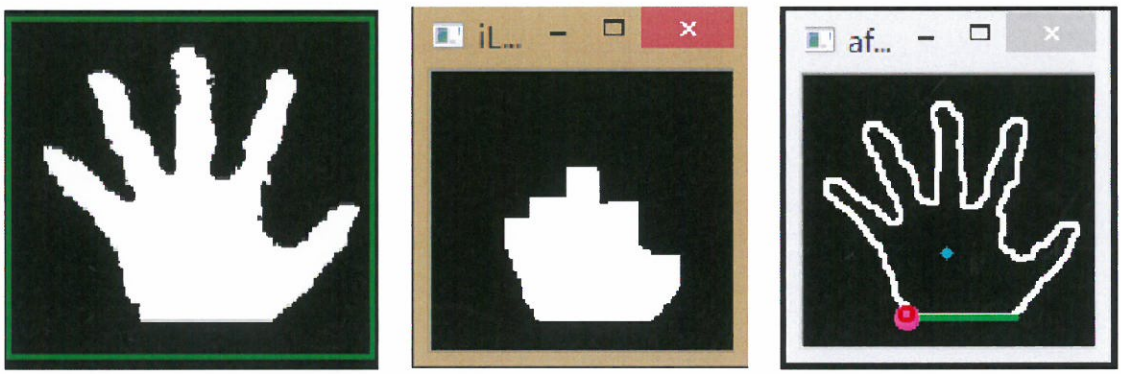


(a) sourceImage　　(b) NormalizeImage 成像

圖六： 掌心偵測示意圖

(2) **手掌邊緣點獲取**

本系統中手掌邊緣點之獲取有兩個重點，其一，要完整取得所有的手掌邊緣點，其二，界定邊緣的計算起始點。為了獲取初始邊緣點，本論文先將圖七(a)的 sourceImage 作數次的侵蝕(Erode)動作後再進行同等次數之擴張(Dilate)動作，出現的結果會如同圖七(b)圖像所示。這麼做的原因是因為本論文使用 EmguCV 中的 HoughLines 的函式抓取直線，若利用 sourceImage 直接呼叫 HoughLines，則有可能先抓取到手指的直線，而非本論文所要的手掌切線。但若先進行侵蝕，並將 HoughLines 中最小線段長度距離增加，則可以有效的降低錯誤的判斷，而利用所得到的 HoughLines 做起始點的獲取也非常簡單，因為 HoughLines 所得到的線段陣列，第一位也就是 HoughLine[0]，剛好就會是圖七(c)右邊圖像粉色圓圈的位置。而手掌邊緣的座標點的獲取，是呼叫 EmguCV 中的 FindContour 函式作處理，將獲得的座標點以 Contour<System.Drawing.Point>資料型態儲存，並命名為 contour。利用 contour 本身 ToArray()的函式就可以直接將 contour 轉換成 Point[]陣列，需要特別注意的是這裡的邊緣點並沒有按照順序排列，所以要進行 FEMD 運算時需要特別加以注意，下一節將會詳細說明如何使用 Point[]做計算。



(a) sourceImage 原始圖像　(b)侵蝕過後的成像　(c) HoughLines 處理完的結果圖

圖七： 手掌切線示意圖

(3) **FEMD 閾值計算**

本系統中 FEMD 線圖中閾值分解(Threshing Decomposition)的功能為快速去除線圖中雜訊的部分，而所謂雜訊指的是在 FEMD 手指偵測演算法經由閾值分解後，所殘留不為手指的範圍。以下將說明以三種閾值分解計算方式進行實驗，依照實驗的結果比較不同閾值分解方法對於辨識率之影響。

◎ **固定閾值分解**

固定閾值分解[3]是為了取代的原本手指偵測時所使用之最低近凸分解(Minimum Near-Convex Decomposition, MNCD)演算法[18]，所謂的最低近凸分解演算法是由近凸分解演算法(Near-Convex Decomposition)改良而來，此演算法可用於手指切割。雖然近凸分解演算法具有高度的準確性，但是相對來說，其計算複雜度較閾值分解方法高出許多，無法有效運用於即時系統之中。而在閾值分解方法，其利用固定的閾值對 FEMD 線圖進行切割，直接去除大部分或全部的雜訊，留下代表手指凸出的部份。

◎ **平均閾值分解**

平均閾值主要的概念來自於為了因應不同人與不同手勢所造成最佳閾值有所浮動，因此在每一次計算出所有邊緣對掌心的距離後，就必須重新做一次閾值的計算，加強其對於各種不同手勢所需最佳閾值的自動化，以便即時地對應對各種不同手勢與手掌形狀。平均閾值分解方法的公式為利用經由邊緣點 Point[]以及所得知 maxPoint(掌心)，利用 Point[]計算每一個邊緣點與 maxPoint(掌心)的距離並將其加總後，將其數字除以 Point[]的總數，即可以得到所有邊緣點與掌心距離的平均值，其平均值即為每一次 FEMD 閾值分解方法的平均閾值。平均閾值分解演算法請參考圖八。

```
Point[] : 所有邊緣座標點
Alldist : 加總的距離值
Athreshold : 平均閾值

double Alldist = 0;
foreach (double p in Point)
    Alldist += distance(p , maxPoint); //計算並加總每一個邊緣點至掌心的距離
double Athreshold; //平均閾值
```

圖八： 平均閾值計算演算法

◎ **掌心圓相對距離閾值分解**

本論文提出之掌心圓相對距離閾值分解方法。主要目的為改進固定閾值分解方法與平均閾值分解方法的缺點，此方法不僅可以計算閾值分解所需之閾值，也可以利用手掌與手指的相對關係有效解決平均閾值分解方法造成誤判的問題。掌心圓相對距離閾值分解方法的計算方式為先以得到的 maxPoint(掌心)為中心，計算出以 maxPoint(掌心)為中心的最大圓半徑 R。R 的計算方式是找出所有 Point[]與 maxPoint(掌心)的距離的

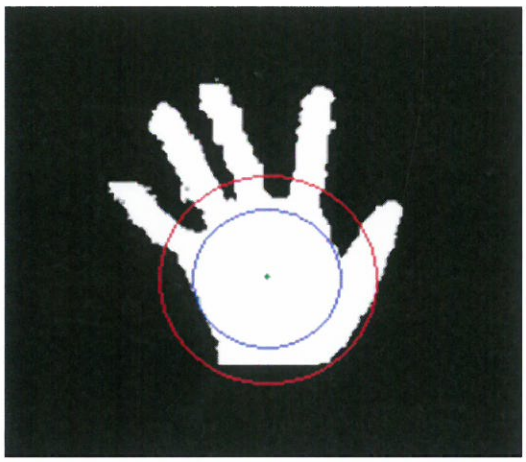
最小者，選擇最小的原因是這樣才可以找出不超過手掌邊緣的圓半徑。最大掌心圓半徑演算法可以參考圖九。

```
R: 最大掌心圓半徑
mindist : 暫存掌心圓半徑
Point[] : 所有邊緣座標點
maxPoint : 掌心座標點

int mindist;
mindist = distance(Point[0] , maxPoint); //最小距離先等於第一個點距離
for ( i=1 ; i<Point[].Length ; i++ )
    if(mindist< distance(Point[i] , maxPoint)) //任何距離值小於將其視為 mindist
        mindist = distance(Point[i] , maxPoint);
```

圖九： 計算最大掌心圓半徑演算法

使用掌心圓最大半徑 R 值，進一步計算所有 Point[]與 maxPoint(掌心)的距離值，將大於 1.5 倍 R 的距離值加總並做平均，其值即為掌心圓相對距離閾值分解方法的閾值，掌心圓相對距離示意圖可以參考圖十，圖中中間的綠點為掌心，藍色圈為最大掌心圓，而最外層的紅色圈為 1.5 倍掌心圓。將平均標準訂定為 1.5 倍以上是因為一般人整體手掌在 1.5 倍掌心圓覆蓋下，可以有效去除掉所有手掌掌部的部份，將手指作有效的切割。

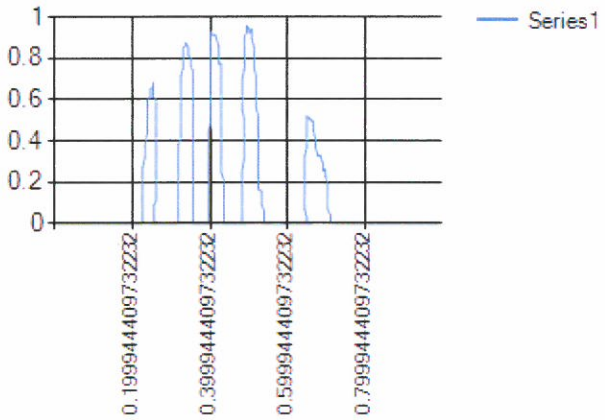


圖十： 掌心圓相對距離示意圖

(4) **FEMD 線圖**

因為系統計算所得 Point[]並沒有按照實際的順序儲存，而是以雜亂的方式記錄所有的點。所以 point[0]也不一定代表邊緣計算的起始點。所以在進行 FEMD 計算時，先將 HoughLine[0](起始點)和 maxPoint(掌心)向量設定為角度 0，需要依序將邊緣點 Point[]與 HoughLine[0](起始點)和 maxPoint(掌心)向量作相對角度換算，這邊可以利用內角公式作

cos 換算，再將 cos 換算成角度。然後將最後所得到的相對角度(tsAngle)與 maxPoint(掌心)到 Point[]的減去閾值的相對距離(tsDist)加入 List<double[]> list 中，最後就會得出依 tsAngle 排列的 list。系統中的線圖是以 Chart 類別繪製，將計算之後的 lsit 依序加入 Chart，角度(tsAngle)為 X 軸，距離(tsDist)為 Y 軸，但這裡的距離如同上述所言，指的是扣除閾值之後的相對距離值，也就是原本的相對距離減去運算之後的閾值(因為每次實驗的距離並非同一種單位)，此舉是為了能夠去掉大部份的雜訊，降低消除雜訊所需的效能負擔。最後所呈現的線圖將會如圖十一。



圖十一： 系統中 FEMD 利用 Chart 所產生的線圖

3.4 手勢辨識

本節將介紹本論文所使用的手勢辨識方法，主要先進行指尖的偵測，再以指間的相對位置作手勢辨識。

(1) 指尖偵測

本系統中指尖偵測的方法是將最後所產生的 FEMD 線圖，以 K 曲率演算法計算線圖的尖端[17]。因為在線圖中 Y 軸的距離皆已經扣除閾值，所以不會呈現出指縫，所以利用 K 曲率演算法只要找出小於特定角度的尖端，偵測出的尖端即可以代表指尖的位置及數目，不需要作傳統 K 曲率演算法的分析指尖與指縫之操作。此處以 K 曲率演算法計算所得小於 30 度角的角度為指尖，K 值設定為 20 是依據論文[14]的最佳指尖計算值。

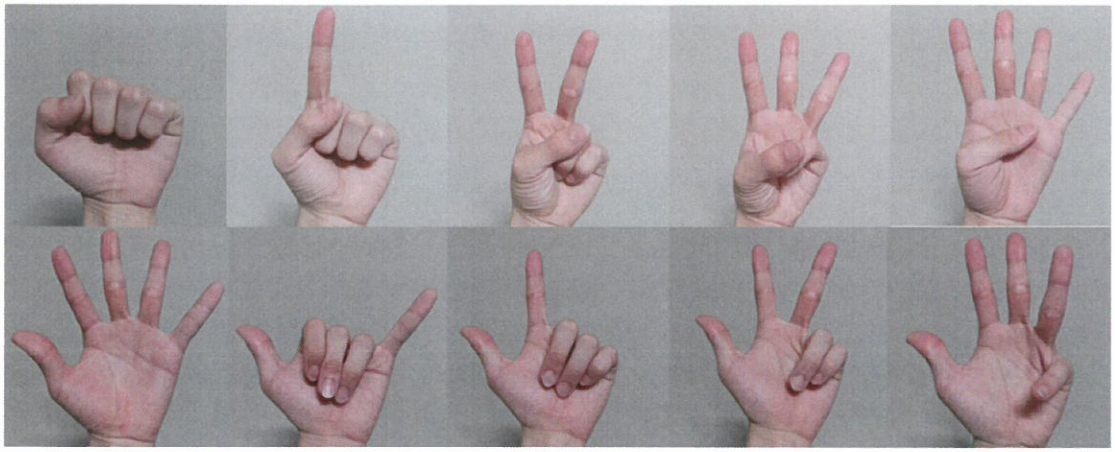
(2) 手指相對位置手勢辨識

本辨識系統使用 4Y-Model[12]相對位置觀念作手勢辨識，因為本論文實驗只有辨識左手的部位，所以手指與手掌的相對位置在手勢基準上，幾乎不會有所改變。手勢辨識的方法為，對照與手指數目相符合的指尖角度，利用手指與手掌相對位置不變的特點，即可正確判斷手勢。

(3) 十種手勢分類

本系統分析的手勢共有 10 種，如圖十二所示。指頭的數目則是有 6 種，分別為 0 根到 5 根。本系統辨識手勢的基準是依照指頭的數目作分類，再去比照指間的相對位置，其分類

則可以參照表一。



圖十二： 實驗中手勢 0~9 圖像示意圖

表一： 十種手勢分類表

指頭的數目	分類說明
0 根手指	若指頭數目為 0，則為手勢 0。
1 根手指	頭數目為 1，則對應為手勢 1。
2 根手指	指頭數目為 2 根的話情況會比較特別，因為可能為手勢 2、手勢 6 或手勢 7，分辨的基準事先以對應最左邊指頭(左手依照鏡頭影像，左邊指頭為小拇指)的位置。在指頭數為 2 根的情況下，有小拇指則為手勢 6，其餘兩種則是一樣以大拇指位置分辨。
3 根手指	指頭數目為 3 根的話，則可能為手勢 3 或手勢 8，辨識的方式以大拇指為基準。
4 根手指	指頭數目為 4 根的話，則可能為手勢 4 或手勢 9，辨識的方式以大拇指為基準。
5 根手指	若指頭的數目等於或大於 5，則將其視為手勢 5。

4. 實驗結果

本節介紹我們的實驗平台與三種實驗設計及其實驗結果。

4.1 實驗平台與實驗說明

我們的實驗硬體平台所使用之 CPU 為 Intel® care™ i7 950 @3.07GHz 四核心八執行緒處理器，搭配三通道 2GB 記憶體 3 條共 6GB，顯示卡則是 NVIDIA GeForce GTX660。軟體開發環境為 Windows8 x64 作業系統，所搭配的 SDK 為 Kinect for Windows V1.8。

三個實驗設計說明如下：

◎ 實驗一：實驗目的為比較固定閾值、平均閾值及掌心圓相對距離閾值三種分解方法的辨識

率與速度；

◎ 實驗二：將手掌掌型分成三類，瞭解三種分解方法是否會因為不同人手掌形狀而影響辨識率。

◎ 實驗三：以基準手掌掌型為主，了解手掌大小是否會影響手勢結果。

4.2 實驗一： 三種閾值計算之辨識率比較

比較不同閾值計算方法對辨識率的影響，三種閾值計算方法分別為固定閾值(Fthresh)、平均閾值(Athresh)與掌心圓相對距離閾值分解方法(Cthresh)，每個手勢各執行 100 次辨識，比較其辨識率及所需計算時間，瞭解其之間的差異。

表二固定閾值的實驗結果，其縱軸數字 **0~9** 欄位代表原始影像所擺出的手勢，而橫軸數字 **0~9** 欄位代表手勢辨識結果，**辨識率**欄位(倒數第二列)顯示對應手勢的正確辨識率，**平均運算時間**欄位代表不同手勢的辨識運算時間，時間單位為毫秒(ms)。以表二之手勢 0 為例，在 100 次辨識中，有 94 次正確辨識為手勢 0，1 次誤判為手勢 5 及 5 次誤判為手勢 6。表二顯示固定閾值之手勢辨識率範圍為 0.86 到 1.00，平均運算時間為 22.02 ms 到 32.03 ms。

表三與表四分別顯示平均閾值及掌心圓相對距離閾值的實驗結果，表三及表四分別顯示，表三顯示平均閾值之辨識率範圍為 0.88 到 1.00，平均運算時間為 32.97 ms 到 34.54 ms。表四顯示掌心圓相對距離閾值之辨識率範圍為 0.91 到 1.00，平均運算時間為 31.66 ms 到 36.46 ms。

表二： 固定閾值之辨識結果

手勢 / 辨識結果	0	1	2	3	4	5	6	7	8	9
0	94	0	0	0	0	0	0	0	0	0
1	0	96	0	0	0	0	0	0	0	0
2	0	0	88	0	0	0	0	5	2	0
3	0	2	2	99	0	0	8	4	7	0
4	0	0	1	0	98	0	3	0	0	6
5	1	0	0	0	1	100	0	0	0	2
6	5	1	0	0	0	0	89	0	0	0
7	0	1	9	0	0	0	0	86	0	0
8	0	0	0	0	0	0	0	5	88	1
9	0	0	0	1	1	0	0	0	3	91
辨識率	0.94	0.96	0.88	0.99	0.98	1.00	0.89	0.86	0.88	0.91
平均運算時間(ms)	25.56	25.27	24.61	22.06	29.89	29.97	26.59	22.02	25.87	32.03

表三： 平均閾值之辨識實驗結果

手勢 / 辨識結果	0	1	2	3	4	5	6	7	8	9
0	88	0	0	0	0	0	0	0	0	0
1	0	94	0	0	0	0	0	0	0	0
2	1	0	93	0	0	0	0	3	1	0
3	3	2	2	96	0	0	2	0	6	0
4	1	0	1	0	98	0	1	0	0	6
5	1	0	0	0	1	100	0	0	0	2
6	6	2	0	0	0	0	97	0	0	0
7	0	2	4	0	0	0	0	96	0	0
8	0	0	0	1	0	0	0	1	91	1
9	0	0	0	3	1	0	0	0	2	91
辨識率	0.88	0.94	0.93	0.96	0.98	1.00	0.97	0.96	0.91	0.91
平均運算時間(ms)	33.5	33.26	34.54	32.97	33.12	33.11	33.76	34.2	33.48	33.89

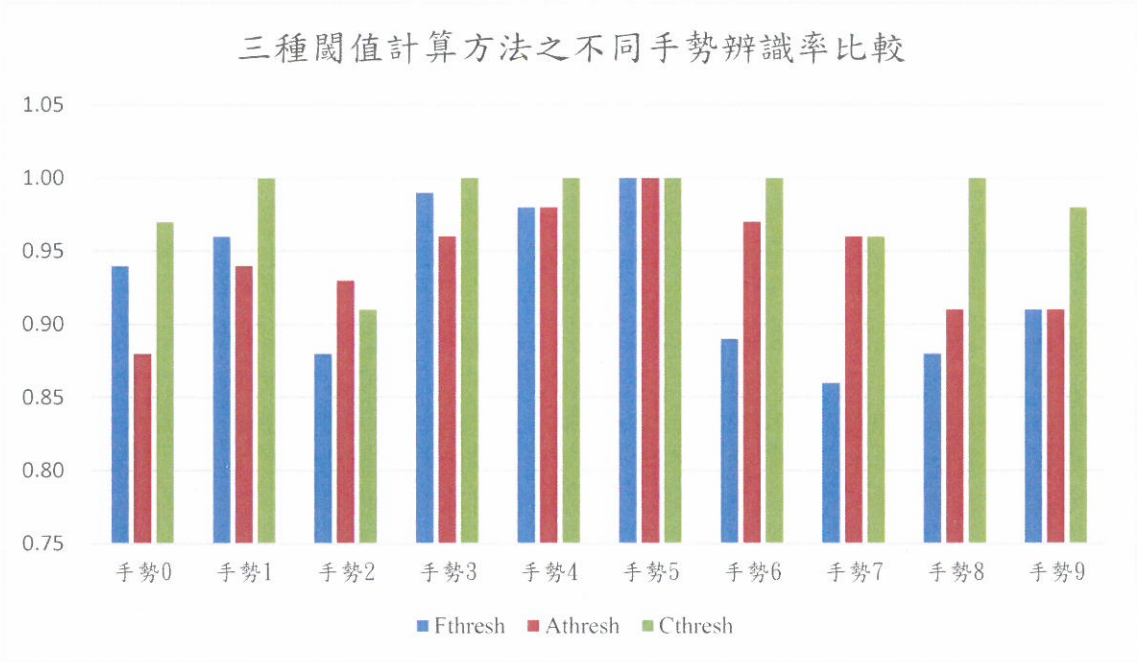
表四： 掌心圓相對距離閾值之辨識實驗結果

手勢 / 辨識結果	0	1	2	3	4	5	6	7	8	9
0	97	0	0	0	0	0	0	0	0	0
1	3	100	2	0	0	0	0	2	0	0
2	0	0	94	0	0	0	0	2	0	0
3	0	0	0	100	0	0	0	0	0	0
4	0	0	0	0	100	0	0	0	0	2
5	0	0	0	0	0	100	0	0	0	0
6	0	0	0	0	0	0	100	0	0	0
7	0	0	4	0	0	0	0	96	0	0
8	0	0	0	0	0	0	0	0	100	0
9	0	0	0	0	0	0	0	0	0	98
辨識率	0.97	1.00	0.91	1.00	1.00	1.00	1.00	0.96	1.00	0.98
平均運算時間(ms)	33.69	31.66	36.46	33.23	31.78	33.41	32.46	38.09	34.98	32.52

(1) **綜合辨識結果**

圖十三為三種閾值計算對於不同手勢之辨識比較結果，從圖中可發現在 10 個手勢辨識過

程中，掌心圓相對距離方法有 6 種手勢(手勢 1、3、4、5、6、8)之辨識率達到 1.00，而固定閾值及平均閾值都只有一種手勢(手勢 5)辨識率達到 1.00。另一方面，掌心圓相對距離閾值分解方法的辨識率比另外兩種方法的辨識率高。另外，我們也觀察到固定閾值方法很容易因為雜訊而誤判為多一根手指的情況，而平均閾值的方法也有上述情況發生，但發生率已經大幅下降。而掌心圓相對距離閾值的方法只有在手勢 2 與 7 有類似情況，其他手勢不會有因雜訊而產生誤判的結果在辨識速度方面之比較，則是固定閾值分解方法中較其他兩種方式快，原因為閾值為系統給定的固定值，不需多做計算。而平均閾值與掌心圓相對距離閾值之運算時間相差不到。比較不同手勢之辨識時間，發現指頭數目為 2 根的手勢(2、6、7)所需的時間較多，原因是辨識所需的步驟較多，需先辨識是否有兩根手指，再判斷大拇指的有無，若沒有大拇指則可以判斷為手勢 2，若有大拇指則再作手勢 6 與手勢 7 的判斷。



圖十三： 三種閾值計算方法之不同手勢辨識率比較

(2) **分析與討論**

表五為不同手勢使用三種閾值計算辨識結果，我們發現固定閾值之辨識率最差，但其辨識時間最少。即時性雖然較佳，但容易因不同手勢、手掌大小及手型不同，對辨識結果造成不良的影響，例如當閾值設定過小時，容易造成 FEMD 線圖中產生雜訊，進而造成辨識率下降。而 Athresh 因為每次都會重新計算閾值，所以辨識時間會增加，但可降低 Fthresh 容易受不同手勢影響辨識率之缺點。Cthresh 可以同時改善 Athresh 與 Fthresh 的缺點，因其使用圓相對距離，可以有效降低手指被切除的情況發生，針對不同手勢及手掌閾值也會有所因應作調整改變，讓辨識率可提升到 0.982。

表五： 三種方法的時間與辨識率結果表

		手勢 0	手勢 1	手勢 2	手勢 3	手勢 4	手勢 5	手勢 6	手勢 7	手勢 8	手勢 9	平均	比率
辨識率	Fthresh	0.94	0.96	0.88	0.99	0.98	1.00	0.89	0.86	0.88	0.91	0.929	1.00
	Athresh	0.88	0.94	0.93	0.96	0.98	1.00	0.97	0.96	0.91	0.91	0.944	1.02
	Cthresh	0.97	1.00	0.91	1.00	1.00	1.00	1.00	0.96	1.00	0.98	0.982	1.06
計算時間 (ms)	Fthresh	25.56	25.27	24.61	22.06	29.89	29.97	26.59	22.02	25.87	32.03	26.387	1.00
	Athresh	33.5	33.26	34.54	32.97	33.12	33.11	33.76	34.2	33.48	33.89	33.583	1.27
	Cthresh	33.69	31.66	36.46	33.23	31.78	33.41	32.46	38.09	34.98	32.52	33.828	1.28

4.3 實驗二： 不同手掌分類辨識率比較

本實驗中會比較不同人的手掌型狀對於三種閾值計算分解方法之辨識結果。本次實驗有三位受測者皆為 18 到 22 歲的學生，其中一位為女性，其他兩位為男性。我們將其手掌分成三種類型：基準手掌、手指細長、手掌偏大手指較短，分類方式如下方說明：

◎ **基準手掌**：手掌比例依照掌心圓半徑與中指長度約莫 1:2 的長度。

◎ **手掌偏大手指較短**：手掌比例依掌心圓半徑與中指長度小於 1:1.8 的長度。

◎ **手指細長**：手掌比例依掌心圓半徑與中指長度大於 1:2.2 的長度。

實驗中三種掌形的受測者會使用 Fthresh 方法、Athresh 方法與 Cthresh 方法作手掌辨識，三種方法都會針對每種手勢作 20 次的辨識，並觀察其 FEMD 線圖是否能夠準確的去除雜訊或不會將手指位置誤判成雜訊去除。

(1) 基準手掌辨識率比較

表六為基準手掌三種閾值分解方法的辨識結果，不論是在 Fthresh 方法、Athresh 方法或者是 Cthresh 方法，其辨識率皆非常地高。實驗一的樣本手掌也是基準手掌，所以表五與實驗一的結果數值幾乎相同，但因為樣本數的減少導致辨識率有些許誤差。

表六： 基準手掌各方法的辨識率比較

	0	1	2	3	4	5	6	7	8	9	辨識率(平均)
Fthresh	0.95	1.00	0.80	1.00	0.95	1.00	0.90	0.90	0.90	1.00	0.94
Athresh	0.95	0.95	0.90	1.00	1.00	1.00	0.95	1.00	0.85	1.00	0.96
Cthresh	0.95	1.00	0.95	1.00	1.00	1.00	1.00	0.95	1.00	0.95	0.98

(2) 手掌偏大手指偏短手掌辨識率比較

表七為三種閾值分解方法的辨識結果，我們發現在使用 Fthresh 方法時，因為雜訊造成無法正確地辨識手勢，使用 Athresh 方式才將此情形改善，而 Cthresh 則是近一步提高手勢的辨識率，不過在手勢 3 與手勢 9 會指頭差點被切除掉的情況。

表七： 手掌偏大手指偏短手掌各方法的辨識率比較

	0	1	2	3	4	5	6	7	8	9	辨識率(平均)
Fthresh	-	-	-	-	-	-	-	-	-	-	0.00
Athresh	0.95	1.00	1.00	0.95	0.95	0.90	0.85	1.00	0.85	0.85	0.93
Cthresh	1.00	1.00	0.95	1.00	0.95	0.95	1.00	0.95	0.85	0.85	0.95

-: 代表該手勢無法辨識

(3) **手指細長手掌辨識率比較**

表八為三種閾值分解方法的辨識結果，我們發現手指細長手掌類型在使用 Fthresh 與 Athresh 方法時，比基準手掌類型的辨識率高，原因在於手指較長的情況下，Fthresh 方法比較少出現手指被切割掉的情況，Athresh 方法是因為手指細長使得整體閾值變大，比較不容易產生雜訊而導致辨識率下降。但 Athresh 方法中，於手勢 0 容易產生雜訊，原因在於手指長的人在握拳時映照出的影像偏長形，不像一般基準手掌握拳時偏圓形，此情形會造成 FEMD 線圖兩端出現雜訊，進而影響辨識結果。而在 Cthresh 方法，也沒有出現雜訊殘留及出現手指遭到切除的情況。

表八： 手掌偏大手指偏短手掌各方法的辨識率比較

	0	1	2	3	4	5	6	7	8	9	辨識率(平均)
Fthresh	0.90	0.95	0.95	0.95	1.00	1.00	1.00	0.90	0.95	0.95	0.96
Athresh	0.95	1.00	1.00	0.95	0.90	1.00	0.95	1.00	1.00	1.00	0.98
Cthresh	1.00	1.00	0.95	1.00	1.00	1.00	1.00	1.00	0.95	1.00	0.99

4.4 實驗三： 基準手掌之不同手掌大小辨識率比較

本實驗有三位受測者，其手掌皆屬於基準手掌類型，我們將其手掌大小分為基準適中、基準偏大與基準偏小。每位受測者都會針對每種手勢作20次的辨識，並觀察不同閾值分解方法是否能夠準確的去除雜訊或將手指位置誤判成雜訊去除。表九為本實驗的辨識率結果。

由表中可以得知Cthresh方法在三位受測者當中都有最高的辨識率，其次為Athresh方法，最低的則是Fthresh方法。而三種閾值分解方法再不同大小手掌實皆會造成誤差，以Fthresh方法為例，Fthresh方法在基準偏大手掌手勢0時，因在於實驗中Fthresh方法設定的閾值對於該手掌來說太小，所以容易產生雜訊造成辨識率降低，反之在基準偏小手掌時，則會因為手指短的原故造成被切除的情況。而Athresh方法則是在基準偏大手掌的大部分手勢中，整體的辨識率有有下降，反而在在基準偏小手掌不會出現此問題。最後，Cthresh方法在三位受測者的辨識率都無太大不同，所以得出Cthresh在不同手掌大小的情況下，對辨識率影響較小。

表九： 基準掌形之不同手掌大小的辨識率比較

		0	1	2	3	4	5	6	7	8	9	辨識率(平均)
基準適中手掌	Fthresh	0.95	1.00	0.80	1.00	0.95	1.00	0.90	0.90	0.90	1.00	0.94
	Athresh	0.95	0.95	0.90	1.00	1.00	1.00	0.95	1.00	0.85	1.00	0.96
	Cthresh	0.95	1.00	0.95	1.00	1.00	1.00	1.00	0.95	1.00	0.95	0.98
基準偏大手掌	Fthresh	0.95	0.90	0.90	0.90	1.00	0.90	0.95	0.95	0.85	0.95	0.93
	Athresh	0.90	1.00	0.95	0.95	0.95	0.95	0.90	0.80	0.95	0.95	0.93
	Cthresh	0.90	1.00	0.90	1.00	0.95	1.00	0.95	1.00	0.95	0.95	0.96
基準偏小手掌	Fthresh	1.00	1.00	0.95	0.90	0.90	0.80	0.80	0.95	0.85	0.85	0.90
	Athresh	0.95	0.95	0.85	0.95	1.00	1.00	0.95	0.90	0.95	0.95	0.95
	Cthresh	0.90	1.00	0.95	0.95	1.00	0.95	1.00	1.00	0.95	0.85	0.96

5. 結 論

本論文提出以 Kinect 感測器為主建立手勢辨識系統，此系統使用之來源影像為深度影像而非 RGB 影像，配合 Kinect 內建的骨架偵測技術，使得原本手勢辨識所需的影像前置作業可大幅減少，也利用改良的 FEMD 方法，判斷手指位置，使得手勢辨識變得更加的精確及更有效率。藉由提出平均閾值及掌心圓相對距離閾值改良方法，可降低雜訊出現的情況，使其辨識率可進一步提升，從原本文獻[3]中的 92.9%提升至 98.2%，雖然在時間上增加為 1.28 倍，但因平均運算時間也不超過 34 毫秒並不會影響到即時性。在未來展望上，希望藉由改變手勢，降低在手勢 2 與手勢 7 誤判的機率以提高辨識率。

而本論文之後的研究重點將會放在如何將這次所建立的手勢辨識系統運用於 AR 以及 VR 的相關應用上，思考如何利用此套系統做出對生活中便利的相關應用，畢竟在不太遙遠的未來，眼鏡將最有潛力取代眾多螢幕，成為重要的互動裝置。利用 AR 眼鏡將每個裝置、家具甚至牆面變成電子裝置，利用 AR 眼鏡進行新型互動，而在此之中實時的手勢辨識將會是技術是否能發展的重大角色之一。

參考文獻

[1] Douglas Gantenbein, "Natural User Interfaces," TechFest Focus, 2011.

[2] Ji-Hwan Kim, "3-D hand motion tracking and gesture recognition using a data glove," IEEE International Symposium on Industrial Electronics, pp. 1013–1018, Jul. 2009.

[3] Zhou Ren, Junsong Yuan, Jingjing Meng and Zhengyou Zhang, "Robust Part-Based Hand Gesture Recognition Using Kinect Sensor," in IEEE Transactions on Multimedia, Vol. 15, pp. 1110 – 1120, 1520-9210, Aug. 2013.

[4] Yung-Fa Huang, Tan-Hsu Tan and Hua-Jui Yang, "A study of hand gesture recognition with wireless channel modeling by using wearable devices," Machine Learning and Cybernetics, pp. 484-487, 2015.

[5] Kertesz. C, "Physiotherapy Exercises Recognition Based on RGB-D Human Skeleton Models," Modelling Symposium, pp. 21-29, 2013.

[6] Yuh-Rau Wang, Wei-Hung Lin and Ling Yang, "A novel real time hand detection based on skin-color," IEEE International Symposium on Consumer Electronics, pp. 141-142, 2013.

[7] 張宸銘，「應用視訊之自動化手勢軌跡追蹤系統」，中原大學，2009 年。

[8] S. Belongie, J. Malik and J. Puzicha, "Shape matching and object recognition usingshape contexts," IEEE Trans. Pattern Anal. Mach. International, pp. 509-522, 2002.

[9] H. Ling and D. W. Jacobs, "Shape classification using the inner-distance," IEEE Trans. Pattern Anal.Mach. Intell, pp. 286-299, 2007.

[10] https://zh.wikipedia.org/wiki/Kinect

[11] 王森，「Kinect 體感程式設計入門」，ISBN：9789862765845，八月，2012。

[12] Ahmed Al Marouf, Shaumic Shondipon, Md. Kamrul Hasan and Hasan Mahmud, "4Y model A novel approach for dinger identification using Kinect," IEEE 2nd International Conference on Recent Trends in Information Systems, pp. 183-188, 2015.

[13] http://www.emgu.com/wiki/index.php/Main_Page

[14] 張財榮、陳定弘與蘇彥彰，「使用 K 曲率法則用於手語手勢辨識」，離島資訊技術應用論文集，第 47-51 頁，2014 年。

[15] Y. Rubner, C. Tomasi and L. J. Guibas, "The earth mover's distanceas a metric for image retrieval," Int. J. Comput.Vision, pp. 99-121, 2000.

[16] Yang Qing, "The research of pattern recognition of gear pump based on EMD and KPCA-SVM," Engineering Design and Manufacturing Informatization, pp. 1-4, 2011.

[17] M. Zabri Abu Bakar, Rosdiyana Samad, Dwi Pebrianti; Mahfuzah Mustafa and Nor Rul Hasma Abdullah, "Finger Application Using K-Curvature Method and Kinect Sensor in Real-Time," International Symposium on Technology Management and Emerging Technologies, pp. 218-222, Aug. 2015.

[18] Z. Ren, J. Yuan, C. Li and W. Liu, "Minimum near-convex decomposition for robust shape representation," Proc. IEEE International Conference Computer Vision, pp. 303-310, 2011.