基於智慧購物車之排隊辨識

吳政瑋 任毅 Muhammad Alfiansyah 高新惟 辛承緯 曾煜棋
國立交通大學資訊科學與工程研究所
Email: {renyi, yctseng}@cs.nctu.edu.tw

摘 要

排隊辨識(Queuing Recognition)是近年來一門新興的研究議題，其主要目的乃利用感測裝置以自動化的方式辨識人類排隊的行為。然而，先前的研究大多以智慧型手機為主要的感測裝置，在大賣場的情境中有使用不便及效率不佳等問題。有鑑於此，本研究考慮大賣場中的排隊情境，率先提出一套以智慧購物車為基礎的排隊辨識架構，其主要包括以下四個核心模組：(1)短程動作辨識、(2)排隊狀態辨識、(3) 所屬隊伍辨識、(4)排隊等候時間估計。本研究實作上述各模組之雛型並模擬真實排隊情境對其進行實驗評估。實驗顯示本研究所提出的架構及方法具備相當優良的辨識結果。

關鍵字：群體行為辨識、排隊行為辨識、機器學習、智慧購物車、行動運算

一、緒論

排隊行為(Queuing Behavior)是人類生活中常見的一種群體行為(Group Activity)，這種行為經常出現在超市及大賣場的情境中。在超市及大賣場中，人們經常花費大量的時間於排隊等候上，並想知道排在哪一行隊伍能節省最多的時間。當有多行隊伍時，知道每行隊伍的資訊及等候時間能夠同時裨益業者及顧客。業者能利用此資訊管理顧客排隊的狀況，並依據排隊的狀況配置適當的人力資源及商品數量，進而降低業者的成本並提高獲利。不僅如此，此資訊還能幫助顧客選擇等候時間較短的隊伍，進而減少顧客耗費於排隊等候的時間。

現今有一些研究[1, 5, 11, 12, 13, 15, 19]提出以攝影機或智慧型手機辨識人類排隊的行為。然而，這些方法並未針對超市及大賣場的排隊情境而設計，因此有辨識效率不佳及使用不便的問題。基於攝影機的方法其主要缺點為建置成本較昂貴。在人群較密集的超市或大賣場中，攝影機亦可能因人群遮蔽問題而導致其辨識率不佳。有些研究[1, 13, 11, 19]則提出以智慧型手機內的感測器推論使用者排隊時所需的等候時間。然而，那些方法[1, 13, 11, 19]需要使用者將手機以固定角度擺放在固定位置，使用者還需安裝專屬的 APP 應用程式。不斷地以手機執行專屬的 APP 應用程式會造成使用者的手機耗電量過高，甚至降低手機內其它應用程式的執行效率，在實務的使用上相當地不便利。

* 本研究接受科技部編號：MOST 105-2221-E-009-101-MY3 研究計畫經費補助

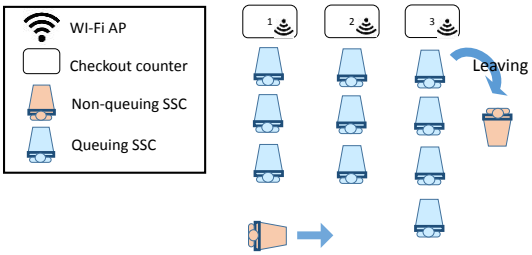

圖 1 本研究所考慮的排隊情境示意圖

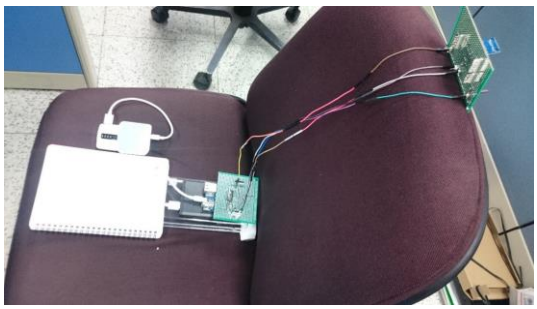

圖 2 以可移動式的椅子模擬裝有 Arduino Yun 及 MPU6050 之智慧購物車雛型

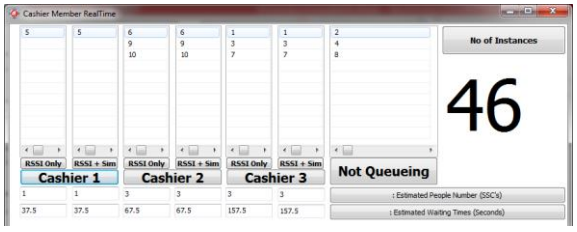

圖 3 屏幕上即時顯示我們系統的判斷結果

有鑑於此，於本研究中，我們提出一套適用於超市及大賣場情境的排隊辨識架構。我們以 Arduino Yun 為基礎設計一套可即時辨識排隊行為及估計等候時間的智慧購物車雛型。我們以本研究所開發的智慧購物車為基礎，提出以下四項核心模組。(1)短程動作辨識：其利用智慧購物車上的加速度及陀螺儀辨識智慧購物車的短程移動行為；(2)排隊狀態辨識：其利用智慧購物車連續發生的短程動作推論該智慧購物車是否處於排隊狀態；(3) 所屬隊伍辨識：從智慧購物車收到的 Wi-Fi AP 訊號資料萃取特徵並以分群演算法辨識哪些智慧購物車屬於哪一行隊伍；(4)排隊等候時間估計：其利用分群結果辨識不同隊伍的購物車數量並估計排隊所需的等候時間。

我們實作上述各模組並設計一系列實驗評估所提出之模組之執行效率。實驗的結果顯示，在使用多類別邏輯斯迴歸(Multi-class Logistic Regression)分類器的情況下，短程動作辨識模組與排隊狀態辨識模組的正確率分別高達 76% 及 94%；所屬隊伍辨識模組則可達到約 70% 的正確率。在排隊等候時間估計模組方面，其有 80% 的估計誤差低於 111 秒。

本論文之後續章節安排如下。第二節回顧與排隊辨識相關的研究。第三節介紹本研究所提出的架構及方法。第四節以實驗評估我們所提出的方法。第五節為結論，第六節提供系統之影片展示網址。

二、相關研究

此章節將介紹與排隊辨識相關的研究，包括：(1) 基於監控攝影機的方法與(2)基於智能手機的方法。以下分別敘述各類方法之特色及其優缺點。

2.1 基於監控攝影機的方法

此類型的方法[5][12][15]其主要概念乃利用監控攝影機及影像處理技術偵測人群移動的方向。然而，此類型的方法經常需要架設多台攝影機於不同位置，因此其佈署成本較昂貴。此外，其辨識的結果也容易受到環境因素的影響(如：光線變化、攝影機擺設位置和物件被遮蔽等)。因此，此類型的方法在多行平行隊伍及人群密集的情境下(如：超市或大賣場的排隊情境中)，其辨識效率較為不佳。

2.2 基於智慧型手機的方法

LineKing[1]是一套基於 Cloudsourcing 的排隊時間估計系統，其利用手機內的 Wi-Fi 訊號強弱及 GPS(Global Positioning System)定位系統估計排隊所需等候的時間。Wang 等學者[13]則提出以手機收到的 Wi-Fi 訊號變化偵測排隊者的排隊狀態，如：排隊等候中、於櫃台接受服務、離開櫃台等行為。然而，上述兩種方法[1][13]僅考慮單行隊伍的情境，並未考慮多行隊伍的情境。Li 等學者[12]率先考慮多行隊伍的情境並提出 QueueSense 方法。QueueSense 利用手機中的羅盤及加速度計感測使用者的排隊特徵，再以協同式方法(Collaborative Approach)與鄰近的手機交換所感測到的資料，並利用這些資料辨識使用者是否正在排隊以及排於哪行隊伍中。由於 QueueSense 需與相鄰的手機交換資料，其可能引發傳輸成本過高、高耗電量及延遲過久等問題。有鑒於協同式方法的缺點，Okoshi 等學者[11]率先提出一套名為 QueueVadis 的非協同式(Non-collaborative Approach)的排隊者辨識方法。QueueVadis 的主要特點乃其僅以個人的排隊特徵辨識使用者是否正在排隊，而毋需與鄰近的手機交換資料。由於 QueueVadis 利用羅盤的角度資訊來區分兩兩排隊者是否屬於相同隊伍，在多行平行隊伍的情境下，不同隊伍朝向櫃台的角度差異較小，QueueVadis 的正確率可能會大幅受到影響。

三、所提出的架構

於本章節中，我們先介紹本研究所考慮的排隊情境，再詳細描述所提出的架構及方法。

3.1 本研究所考慮的排隊情境

圖 1 顯示本研究所考慮的排隊情境。於該情境下，智慧購物車的狀態可分為排隊(Queuing) 及非排隊(Non-queuing)兩類。在一個區域內，共有 M ($M \geq 1$) 個櫃台及 N ($0 \leq N \leq M$)行面向櫃檯的隊伍，每一行隊伍近似於直線形狀。櫃檯之間互相平行且面向同一方向。每個櫃台都安裝著一個 Wi-Fi AP (Wi-Fi Access Point)。於 3.3 及 3.4 節中，我們將會介紹如何利用 Wi-Fi AP 的 RSSi (Received Signal Strength Indicator)訊號強度於我們的方法中。

3.2 智慧購物車雛型設計及伺服器設置

本研究所提出的架構為主從式架構(Client-Server Architecture)。智慧購物車為系統的 Client 端，車上安裝 Arduino Yun 及 MPU6050 晶片。MPU6050 內嵌著 3 軸加速度計及 3 軸陀螺儀。我們以包含四個輪子的椅子模擬超市中的購物車以節省成本。在智慧購物車上所收集到的即時資訊，將會定期傳送至 Server 端進行處理。經過運算後，Server 端的屏幕上將會即時顯示哪些購物車排在哪行隊伍中，以及每行隊伍的預估等候時間(如圖 3)。

3.3 短程動作辨識模組

排隊是一種先來先服務的行為。在一行排隊隊伍中，若有智慧購物車(Smart Shopping Carts；簡稱 SSC) 向前移動或離開隊伍，則在其後面的 SSC 也會在一小段延遲後往前移動。相較於非排隊狀態的 SSC，在排隊中的 SSC 會較有規律地交錯著往前移動及停下來等候的行為模式。相反地，不在排隊狀態的 SSC 較不會有這種行為模式。

基於此特性，我們利用 SSC 上的加速度計及陀螺儀收集感測資料以推論每 θ ($\theta \geq 0$) 秒內 SSC 正處於哪種移動事件，藉此得知該 SSC 在 ω ($\omega \geq \theta$) 秒內循序移動的行為。在實作中，我們將 θ 及 ω 分別設定為 3 秒及 30 秒，並考慮以下三種 SSC 的移動行為辨識：{靜止}、{向前移動}、{其它}。我們利用 MPU6050 內的加速度計及陀螺儀擷取三軸加速度及三軸角加速度串流資料。在資料收集的過程中，若取樣頻率(Sampling Rate)為 δ Hz，則感測器每秒會對每一個軸向取樣 δ 筆感測資料。

本研究採用不重疊的滑動視窗方法切割感測資料。我們將滑動視窗的長度設定為 θ，以 θ 秒為單位切割所收集到的串流感測資料。經過切割處理後所得的資料我們稱其為資料片段(Data Segment)。對於每一個資料片段 S，我們對其擷取低階特徵。令 $A = \langle a_1, a_2, \ldots, a_n \rangle$ 為 S 中任意一軸的時序資料，其中 a_i 為 A 中第 i ($1 \leq i \leq n$) 次取樣的結果。表 1 顯示我們對時序資料 A 取的特徵及特徵計算公式。

表 1 對同種感測器每一軸的感測資料擷取的特徵

特徵名稱	符號表式	計算公式
Mean	$\mu(A)$	$\frac{1}{n}\sum_{i=1}^{n} a_i$
Variance	$Var(A)$	$\frac{1}{n}\sum_{i=1}^{n}(a_i-\mu(A))^2$
Energy	$Eng(A)$	$\frac{1}{n}\sum_{i=1}^{n} a_i^2$
Average Absolute Difference	$AAD(A)$	$\frac{1}{n-1}\sum_{i=2}^{n}\lvert a_{i-1}-a_i\rvert$
Skewness	$Skew(A)$	$\frac{\frac{1}{n}\sum_{i=1}^{n}(a_i-\mu(A))^3}{\left[\frac{1}{n}\sum_{i=1}^{n}(a_i-\mu(A))^2\right]^{3/2}}$
Kurtosis	$Kurt(A)$	$\frac{\frac{1}{n}\sum_{i=1}^{n}(a_i-\mu(A))^4}{\left[\frac{1}{n}\sum_{i=1}^{n}(a_i-\mu(A))^2\right]^{2}}$

表 2 對同感測器的任兩軸感測資料擷取的特徵

特徵名稱	符號表式	計算公式
Correlation	$Corr(A, B)$	$\frac{1}{n}\sum_{i=1}^{n}(a_i-\mu(A)(b_i-\mu(B))$
Covariance	$Cov(A, B)$	$\frac{Cov(A,B)}{\sigma_A\sigma_B}$

表 3 對同種感測器的三軸感測資料擷取的特徵

特徵名稱	符號表式	計算公式
Mean Magnitude	$Mmg(A, B, C)$	$\frac{1}{n}\sum_{i=1}^{n}\sqrt{a_i^2+b_i^2+c_i^2}$

S	*S*	*S*	*M*	*O*	*S*	*S*	*M*	*M*	*O*

圖 4 購物車的循序移動序列示意圖

令 $A = \langle a_1, a_2, \ldots, a_n \rangle$ 及 $B = \langle b_1, b_2, \ldots, b_n \rangle$ 為 *S* 中同感測器任兩軸的時序資料，表 2 顯示我們對 *A* 及 *B* 取的特徵及特徵計算公式。令 $A = \langle a_1, a_2, \ldots, a_n \rangle$、$B = \langle b_1, b_2, \ldots, b_n \rangle$ 及 $C = \langle c_1, c_2, \ldots, c_n \rangle$ 為 *S* 中同感測器相異三軸的時序資料。表 3 顯示我們對 *A*、*B*、*C* 取的特徵及特徵計算公式。

擷取低階特徵後，我們利用一個預先建立好的分類器來分類 SSC 在 θ 秒內的移動事件。此分類器的輸入為擷取低階特徵後的資料片段，其輸出為{靜止}、{向前移動}或{其它}。我們利用 Weka 內的分類方法建立此分類模型，並將模型建置於 SSC 的 Arduino Yun 中。

3.4 排隊狀態辨識模組

由於 SSC 在排隊時會有規律且斷斷續續地交錯著往前移動及停下來等候的動作。不處於排隊狀態的 SSC 的移動模式較不會具有上述特性。我們以重疊的滑動視窗方法收集 ω 秒內 SSC 連續的短程動作辨識結果，其又稱為 SSC 的循序移動序列 (Movement Sequence)，代表 SSC 在 ω 秒內循序移動的方式。並將視窗每次滑動的長度設定為 γ 秒。在我們的實作中，γ 設定為 15 秒。

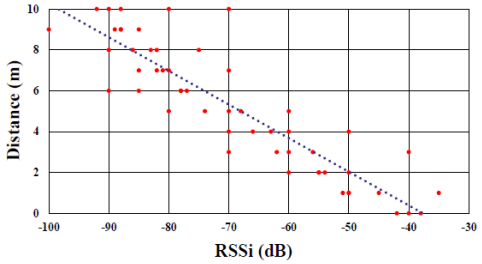

圖 5 對一個 Wi-Fi AP 訓練出來的迴歸模型

表 4 一台 SSC 的距離時序資料

	t_1	t_2	t_3	...	t_{v-1}	t_v
AP_1	10	20	25	...	40	50
AP_2	8	18	30	...	38	48
AP_3	5	15	28	...	35	45

舉例說明，圖 4 為一台 SSC 的循序移動序列，其中 *S* 代表{靜止}(Still)、*M* 代表{向前移動}(Moving Forward)、*O* 代表{其它} (Others)。令 *T* 為 *S*、*M* 或 *O*，對於每一個循序移動序列 *R*，我們對其擷取下列高階特徵，也就是循序移動特徵。

- ***Diff*(*R*)**：在 *R* 中，相鄰短程動作相異的總次數。在圖 2 中，*Diff*(*R*)為 5 次。
- ***Num*(*R*, *T*)**：在 *R* 中 *T* 發生的總次數。在圖 4 中，*S*、*M* 及 *O* 發生的總次數分別為 5、3、2。
- ***Min*(*R*, *T*)**：在 *R* 中，*T* 的最少連續發生次數。在圖 4 中，*S*、*M* 及 *O* 的最少連續發生次數分別為 *min*{3, 2} = 2、*min*{1, 2} = 1、*min*{1, 1} = 1。
- ***Max*(*R*, *T*)**：在 *R* 中，*T* 的最多連續發生次數。在圖 4 中，*S*、*M* 及 *O* 的最多連續發生次數分別為 *max*{3, 2} = 3、*max*{1, 2} = 2、*max*{1, 1} = 1。
- ***Avg*(*R*, *T*)**：在 *R* 中， *T* 的平均連續發生次數。在圖 4 中，*S*、*M* 及 *O* 的平均連續發生次數分別為 (3+2)/2 = 2.5、(1+2)/2 = 1.5、(1+1)/2 = 1。

除了擷取 SSC 的循序移動特徵外，我們還從 SSC 與櫃檯 Wi-Fi AP 的 RSSi 訊號資料中擷取利於分類的特徵，其處理過程如下所述。在離線階段，對於每一個在櫃台的 Wi-Fi AP，我們收集 SSC 在不同地點時與它的 RSSi 訊號強度及實際距離，並利用簡單線性迴歸方法[6]找出 RSSi 訊號強度與距離之間的對應關係，其主要目的乃利用 RSSi 訊號強度粗略地推估 SSC 與櫃檯之間的距離。圖 5 顯示一個習得的簡單線性迴歸方程式，若輸入的 RSSi 值為-50dB，則迴歸模型判斷 SSC 與該 Wi-Fi AP 的距離約為 2 公尺。在上線階段，對於每一個在櫃台上的 Wi-Fi AP，SSC 會以重疊的滑動視窗方法收集 ω 秒內 SSC 收到的連續 RSSi 訊號資料。該視窗每次滑動的長度也為 γ 秒。接著利用離線階段建立好的迴歸模型將視窗內擷取到的每一筆 RSSi 資料點轉換為距離。我們稱這個經過轉換後的資料為距離時序資料，如圖 4 所示。

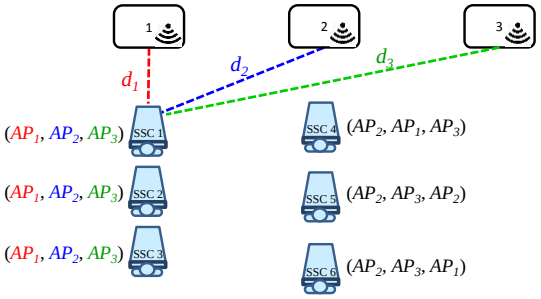

圖 6 相同隊伍的 SSC 會有較相似的 AP 順序特徵

輸入： (1) SSC *I* 的 ARF; (2) SSC *J* 的 ARF;
輸出： (1) *SSC I* 與 *SSC J* 的相似度;
演算法步驟: 01 **Initialize** *sim*(*I*, *J*) ← 1; 02 **For**(*K* **from** 1 **to** *M*) 03 { **If** (ARF[*I*, *K*] == ARF[*J*, *K*]) 04 { //do nothing; } 05 **else if** { *sim*(*I*, *J*) ← *sim*(*I*, *J*) – penalty p_K; } 06 } 07 **Output:** *sim*(*I*, *J*);

圖 7 以 AP 順序特徵為基礎的相似度計算演算法

在圖 4 中，$AP_i\,(1 \le i \le M)$表示第 i 個櫃台上的 Wi-Fi AP，$t_j\,(1 \le j \le v)$表示距離時序資料中的第 j 個時間點。距離時序資料記錄著一台 SSC 與不同 *AP* 在不同時間點的估計距離。若對每個時間點 $t_j\,(1 \le j \le v)$ 取出不同 AP 中最小的距離值，將形成另一條時序資料。我們稱此時序資料為最短距離時序資料，其代表 SSC 在不同時間點上與最接近的 AP 的估計距離。

接著，我們使用指數平滑化方法(Exponential Smoothing)對最短距離時序資料進行平滑化以凸顯趨勢變化並降低雜訊干擾。指數平滑化方法的計算方式如下：

$$m_0 = d_0,$$
$$m_j = \alpha \times d_j + (1-\alpha) \times m_{j-1}, \quad j > 0,$$

在上述公式中，$d_j\,(1 \le j \le v)$ 代表最短距離時序資料中第 j 筆資料；$m_j\,(1 \le j \le v)$ 代表 d_j 經過指數平滑化後的值；$\alpha\,(0 < \alpha < 1)$ 為平滑係數。經過內部實測，我們將 α 設定為 0.8。接著，我們對經過平滑化後的資料取平均值、變異數、FFT 能量(FFT Energy)、及 FFT 熵(FFT Entropy)等特徵值，又稱為 SSC-AP 距離特徵。

我們利用一個預先建立好的分類器來分類 SSC 在 ω 秒內的動作較屬於排隊狀態或是非排隊狀態。此分類器的輸入為上述的循序移動特徵及 SSC-AP 距離特徵，其輸出為{排隊狀態}或{非排隊狀態}。在實作上，我們利用 Weka 內的分類方法建立此分類模型，並將模型建置於 SSC 中的 Arduino Yun。

輸入： (1) *D*：包含 *n* 台 SSC 資料的資料集;
輸出： (1) 分群法所產生的階層式群聚樹(Dendrogram);
演算法步驟: 01 將 *D* 中的每一筆資料視為一群; 02 如果兩個群 C_a 及 C_b 有最小的距離 $dis(C_a, C_b)$，則合併 C_a 及 C_b 形成一群; 03 重複步驟 02 及 03，直到沒有群集可再合併; 04 **Output:** 分群法最後所產生的階層式群聚樹;

圖 8 凝聚式階層分群法之虛擬碼

3.5 所屬隊伍辨識模組

若 SSC 被排隊狀態辨識模組辨別為{排隊狀態}，系統會將該 SSC 的辨識碼(Identifier)及重疊滑動視窗內所收集到的距離時序資料傳送到 Server 端的所屬隊伍辨識模組進行集中處理。

Server 收到 SSC 的距離時序資料後，會對其每一列的數值取平均值，以作為 ω 秒內 SSC 與每個 AP 的平均距離。從觀察得知，若一台 SSC 排於隊伍 $K\,(1 \le K \le M)$，則它與 AP_K 的距離會小於它與其它 AP 的距離。以圖 6 為例，SSC 1 排於隊伍 1，它與 AP_1 的距離(d_1)會分別小於它與 AP_2 及 AP_3 的距離(d_2 及 d_3)。因此，距離 SSC 1 由近而遠的 AP 順序為 $\langle AP_1, AP_2, AP_3 \rangle$。對於任意一台 SSC *h*，我們稱這項資訊為 *h* 的 AP 順序特徵(AP Ranking Feature；簡稱 ARF)。我們以符號 ARF[*h*, *i*]表示，*h* 的 ARF 中第 *i* 個值 $(1 \le i \le M)$。

接著，我們以 ARF 為基礎計算兩兩 SSC 的相似度。我們以符號 *sim*(*I*, *J*)表示 SSC *I* 與 SSC *J* 之相似度。圖 7 顯示我們所提出的相似度計算演算法。該方法的輸入為 SSC *I* 及 SSC *J* 的 ARF，其輸出為 *sim*(*I*, *J*)。該演算法首先將 *sim*(*I*, *J*)初始化為 1，接著進行 *M* 次迭代(Iteration)。在第 $K\,(1 \le K \le M)$ 次的迭代中，若 ARF[*I*, *K*] 與 ARF[*J*, *K*]不相等，則 SSC *I* 與 SSC *J* 的相似度會減去一個使用者自訂的懲罰係數 p_K，以降低 SSC *I* 與 SSC *J* 的相似度。

由於在本研究的實驗中，我們只有考慮三個櫃台的情境進行實驗，所以僅需設定 p_1、p_2 及 p_3。經由我們內部的實驗測得，當 p_1、p_2 及 p_3 分別設定為 0.6、0.2 及 0.2 時會有不錯的結果。

計算完兩兩 SSC 的相似度後，我們可得到一個相似度矩陣。所屬隊伍辨識模組會將此相似度矩陣輸入至分群演算法中，將具有相似排隊特徵的 SSC 歸納在同一群，以辨識出哪些 SSC 排在相同的隊伍中。本研究以凝聚式階層分群法(Agglomerative Hierarchical Clustering)[6]並搭配 Average Linkage 的方式計算群與群的距離。圖 8 為凝聚式階層分群法之虛擬碼。令 C_a 及 C_b 為兩個不同的群，$|C_a|$ 及 $|C_b|$ 分別代表 C_a 及 C_b 群內的資料筆數，則 Average Linkage 的距離計算方式為 C_a 中各點至 C_b 中各點的平均距離，也就是：

$$dis(C_a, C_b) = \frac{1}{|C_a||C_b|}\sum_{I=1}^{|C_a|}\sum_{J=1}^{|C_b|}(1 - sim(I, J)) \text{ 。}$$

表 4 短程動作辨識模組之訓練及測試資料資訊

	靜止	向前移動	其它	總筆數
訓練資料	1,054	500	792	2,346
測試資料	873	511	740	2,124

表 5 短程動作辨識模組之訓練及測試資料資訊

	排隊狀態	非排隊狀態	總筆數
訓練資料	79	331	410
測試資料	76	334	410

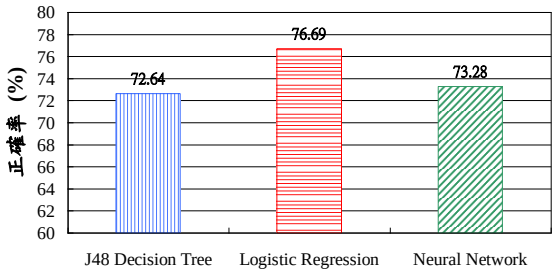

圖 9 短程動作辨識模組的正確率

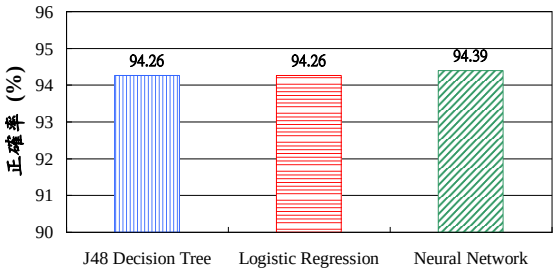

圖 10 排隊狀態辨識模組的正確率

經過凝聚式階層分群法處理後，我們可得到一個階層式群聚樹[6]。利用適當的距離切割門檻值對階層式群聚樹進行切割，我們可找出數個群集。適當的距離切割門檻值可從觀察歷史資料推得。得到數個群集後，對於每一個群集，我們分析該群中 SSC 的 ARF，，並從中挑選個數最多的值作為該群的標籤，也就是該群所屬的隊伍編號。

3.6 排隊等候時間估計模組

令 $Q^* = \langle Q_1, Q_2, ..., Q_K \rangle$為所屬隊伍辨識模組的分群結果，其中 Q_i $(1 \le i \le K)$ 為第 i 個群集。令 $Q = \langle q_1, q_2, ..., q_m \rangle$為 Q^*中的任意群集，其包含 m $(m \ge 1)$個被辨識為排隊狀態的 SSC 且 q_j $(1 \le j \le m)$ 表示隊伍中的第 j 台 SSC。則此模組估計 Q 的隊伍長度為$|Q| = m$，且估計的排隊等候時間為

$$\sum_{j=1}^{|Q|=m} ItemNumber(q_j) \times \beta \ ,$$

其中，$ItemNumber(q_j)$為 Q 中第 j 台 SSC 內的商品數量；β 為櫃台人員對一個商品掃描條碼的平均時間，其可從觀察歷史資料推得。雖然本研究目前尚未提出偵測 SSC 內商品個數的方法，但在未來研究中，我們預計整合商品個數偵測技術於我們所提出的 SSC 架構中。若能有效地偵測 SSC 中的商品個數，在直覺上，我們所提出的排隊等候估計方法其正確率會大幅優於先今最佳的方法(如：[11][12])。

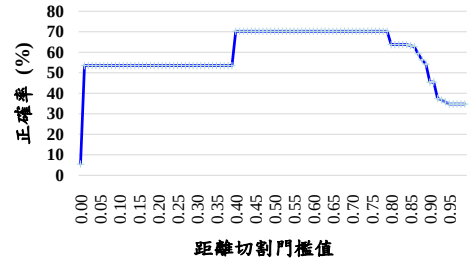

圖 11 所屬隊伍辨識模組的正確率

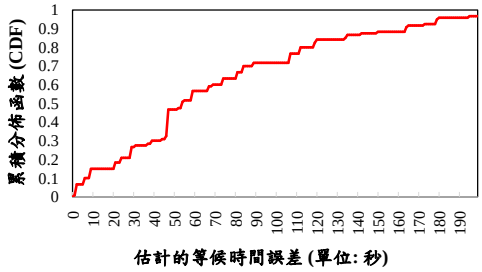

圖 12 排隊等候時間估計模組之錯誤率累積分佈

四、實驗結果

於此章節，我們以實驗評估所提出的各模組之執行效率。

4.1 短程動作辨識模組之效率評估

於此實驗中，我們評估短程動作辨識模組的執行效率。我們收集的訓練資料及測試資料其資訊如表 4 所示。感測資料的取樣頻率皆為 20Hz。短程動作辨識模組可採用不同的分類器進行辨識。於實驗中，我們考慮三種常見但不同類型的分類器，期分別為 J48 決策樹(J48 Decision Tree)、多類別邏輯斯迴歸(Multi-class Logistic Regression)、及三層式倒傳遞神經網路(Three-layers Back-propagation Neural Network)。我們以知名資料探勘工具 Weka [15] 及其內部預設的參數建構各分類模型。圖 9 顯示短程動作辨識模組使用不同分類器時的正確率(Accuracy)[6]。如圖 9 所示，短程動作辨識模組具備相當不錯的正確率。各分類器的正確率皆高於 70%，且多類別邏輯斯迴歸的正確率高達 76%，為後續的排隊辨識方法奠定良好的基礎。

4.2 排隊狀態辨識模組之效率評估

接著，我們評估排隊狀態辨識模組的效率。於此實驗中，我們使用 10 張可移動式的椅子模擬 10 台在賣場中排隊及隨意移動的 SSC，並考慮置有 3 個 Wi-Fi AP 的櫃台與一行隊伍。我們將十台 SSC 分成兩個群組。第一個群組共有 5 台 SSC，這些 SSC 都面向櫃台 2 並排成一行隊伍，以模擬 SSC 排隊實的情況。我們將剩餘的 5 台 SSC 分在第二個群組，並隨意地移動它們，以模擬顧客在賣場隨意移動的行為。表 5 顯示我們所收集的訓練資料及測試資料資訊。圖 10 顯示排隊狀態辨識模組使用 J48 決策樹、多類別邏輯斯迴歸及三層式倒傳遞神經網路時的正確率。如圖 10 所示，此模組的正確率高達 94%以上。

表 6 所屬隊伍辨識模組的混亂矩陣

	被模組分在相同群中	被模組分在不同群中
實際情況為同隊伍	*TP*	*FN*
實際情況為不同隊伍	*FP*	*TN*

4.3 所屬隊伍辨識模組之效率評估

於此實驗中，我們評估所屬隊伍辨識模組的效率。我們使用 10 張可移動式的椅子模擬 10 台在賣場中排隊的 SSC，並考慮置有 3 個 Wi-Fi AP 的櫃台與 3 行隊伍。每個櫃台配置著一個 LED 燈及一個服務時間隨機產生器。服務時間隨機產生器會隨機產生一個介於 15 到 90 秒的服務時間，以代表 SSC 停留在櫃台接受服務的時間。當服務時間倒數為 0 時，該櫃檯的 LED 燈會閃爍，屆時我們會移動在該櫃台的 SSC，使其隨機重新加入某一行隊伍中進行排隊。每一行隊伍的 SSC 個數則控制在 2 到 4 台之間。基於此模組在實測的結果，我們可以建構一個如表 6 的混亂矩陣，並定義此模組的正確率為 $(TP + TN)/(TP + FP + FN + TN)$。圖 11 顯示所屬隊伍辨識模組在不同距離切割門檻值時的正確率。如圖 11 所示，此模組在距離切割門檻值介於 0.40 到 0.75 時，其正確率高達 70%左右。

4.4 排隊等候時間估計模組之效率評估

於此實驗中，我們評估排隊等候時間估計模組的效率。在 4.3 節的實驗中，我們同時也收集了 SSC 真實的排隊等候時間及此模組預估的排隊等候時間。令 *GV* 為一台 SSC 在一次分群後的實際等候時間，*EV* 為該 SSC 在該次分群後的估計等候時間，則等候時間的誤差定義為|GV − EV|。計算出每一台 SSC 在每次分群後的等候時間誤差後，我們將這些誤差值由小到大排序，並畫出其累積分佈函數圖(Cumulative Distribution Function；CDF)，其結果如圖 12 所示。從圖 12 中，我們可看出約有 80%的誤差小於 111 秒。造成誤差較大的原因是因為我們的模組假設每台 SSC 在櫃台接受服務的時間為常數，其尚未考慮 SSC 中的商品個數。但在實驗中，我們以服務時間隨機產生器對 SSC 產生隨機的服務時間，因而導致誤差較大。在未來的研究工作中，我們會對其進行調整及改善。

五、結論

在本論文中，我們提出第一套以智慧購物車為基礎的排隊辨識系統，其主要包括四項重要的貢獻：短程動作辨識、排隊狀態辨識、所屬隊伍辨識、排隊等候時間估計模組。我們實作上述各模組，並以實驗評估所提出的架構及方法。在本研究的實驗中，短程動作辨識模組的正確率約為 76%；排隊狀態辨識模組的正確率可達 94%；所屬隊伍辨識模組的正確率約為 70%。在排隊等候時間估計模組方面，有 80% 的估計誤差低於 111 秒。

六、系統展示

以下網址提供本研究所開發的系統展示影片：
https://drive.google.com/open?id=0B60iBQt-1W8kQU9jcDdnZmZFaTQ

參考文獻

[1] F. Bulut, M. Demirbas, and H. Ferhatosmanoglu., "LineKing: coffee shop wait-time monitoring using smartphones," IEEE Transactions on Mobile Computing, vol. 14, no. 10, pp. 2045-2058, 2015.

[2] M. Chang, N. Krahnstoever, S. Lim, and T. Yu., "Group level activity recognition in crowded environments across multiple cameras," in Proc. Of Advanced Video and Signal Based Surveillance, pp. 56-63, 2010.

[3] B. Chun and P. Maniatis., "Augmented smartphone applications through clone cloud execution," in Proc. Of the 15th Workshop on Hot Topics in Operating Systems, vol. 9 , pp. 8-11, 2009.

[4] H. Falaki, R. Mahajan, S. Kandula, D. Lymberopoulos, R. Govindan, and D. Estrin., "Diversity in smartphone usage," in Proc. Of the 8th International Conference on Mobile Systems, Applications, and Services, pp. 179-194, 2010.

[5] T. Huynh, M. Fritz, and B. Schiele. "Discover of activity patterns using topic models," in Proc. Of the 2008 Ubiquitous Computing, pp. 10-19, 2008.

[6] J. Han, M. Kamber, and J. Pei. "Data mining: concepts and techniques, 3rd edition," Morgan Kaufmann, 2011.

[7] T. Joshua, D. Vin, and L. John. "A global geometric framework for nonlinear dimensionality reduction," Science, vol. 290, no. 5500, pp. 2319-2323, 2000.

[8] H. Lu, W. Pan, N. D. Lane, T. Choudhury, and A. T. Campbell, "Soundsense: scalable sound sensing for people-centric applications on mobile phones," in Proc. Of the 7th International Conference on Mobile Systems, pp. 165-178, 2009.

[9] B. Leibe, E. Seemann, and B. Schiele, "Pedestrian detection in crowded scenes," in Proc. Of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 1, pp. 878-885, 2005.

[10] E. Miluzzo, N. D. Lane, K. Fodor, R. Peterson, H. Lu, M. Musolesi, S. B. Eisenman, X. Zheng, and A. T. Campbell, "Sensing meets mobile social networks: the design, implementation and evaluation of the cenceme application," in Proc. Of the 6th ACM Conference on Embedded Network Sensor System, pp. 337-350, 2008.

[11] T. Okoshi, Y. Lu, C. Vig, Y. Lee, R. Balan, and A. Misra. "QueueVadis: Queuing analytics using smartphones," in Proc. Of the 14th International Conference on Information Processing in Sensor Networks, pp. 214-225, 2015.

[12] L. Qiang, Q. Han, X. Chengand, and L. Sun. "Collaborative recognition of queuing behavior on mobile phones," IEEE Transactions on Mobile Computing, vol. 15, no. 1, pp. 60-73, 2016.

[13] Y. Wang, J. Yang, Y. Chen, H. Liu, M. Gruteser, and P. Martin. "Tracking human queues using single-point signal monitoring," in Proc. Of the 12th Annual International Conference on Mobile Systems, Applications and Services, pp. 42-54 2014.

[14] Y. Wang, J. Wang, and X. Zhang. "Qtime: a queuing-time notification system based on participatory sensing data," in Proc. Of the 37th Annual Computer Software and Applications Conference, pp.770-777, 2013.

[15] Weka 3: data mining software in Java, Available at: http://www.cs.waikato.ac.nz/ml/weka/

[16] LIBSVM: A Library for Support Vector Machines, Available at: https://www.csie.ntu.edu.tw/~cjlin/libsvm/