國立交通大學

電控工程研究所

碩士論文

結合無線感測網路之機器人召喚系統設計

A Call-to-Service Design for Mobile Robots Using Wireless Sensor Network

研 究 生：洪上畯

指導教授：宋開泰 博士

中華民國 一百零一 年 五 月

結合無線感測網路之機器人召喚系統設計

A Call-to-Service Design for Mobile Robots Using Wireless Sensor Network

研 究 生：洪上畯　　Student: Shang-Chun Hung

指導教授：宋開泰 博士　　Advisor: Dr. Kai-Tai Song

國 立 交 通 大 學

電 控 工 程 研 究 所

碩 士 論 文

A Thesis

Submitted to Institute of Electrical Control Engineering

College of Electrical and Computer Engineering

National Chiao Tung University

in Partial Fulfillment of the Requirements

for the Degree of Master

in

Electrical Control Engineering

May, 2012

Hsinchu, Taiwan, Republic of China

中華民國一百零一年五月

結合無線感測網路之機器人召喚系統設計

學生:洪上畯　　指導教授:宋開泰 博士

國立交通大學電控工程研究所

摘要

本論文主要目的為設計一套基於無線感測網路之機器人召喚系統。當使用者召喚機器人時,機器人首先會透過 Zigbee 無線感測網路模組 CC2431 定位引擎來估測使用者之座標,並且將使用者之座標當作自主導航系統的目標位置,讓機器人朝向使用者前進。當機器人接近使用者時,透過 Kinect 感測器所提供的影像來偵測使用者人形與人臉,並且計算出使用者在影像中的位置資訊,接著透過影像追蹤控制器,讓機器人修正自身方向來朝向使用者前進,並且在設定的安全距離下,停在使用者面前。本論文所設計的機器人自我定位系統,整合里程計與 CC2431 定位引擎定位資訊,可以改善機器人因為移動過久、運作時間過長而產生的定位誤差,同時亦降低因無線感測網路室內定位之不確定性,使機器人可進行長距離之導航,找到使用者。經由實驗驗證,本論文所設計的機器人自我定位系統,在機器人移動 60 公尺後,機器人自身定位誤差平均為 68 公分。本論文所設計的機器人召喚系統,機器人能順利找到使用者,並且停在使用者面前與使用者互動。

A Call-to-Service Design for Mobile Robots Using Wireless Sensor Network

Student: Shang-Chun Hung Advisor: Dr. Kai-Tai Song

Institute of Electrical Control Engineering

National Chiao Tung University

Abstract

In this thesis, a call-to-service system for mobile robots is developed based on a wireless sensor network. In this design, a ZigBee-based location-aware system is responsible for estimating the position of user who calls the robot for service. The autonomous navigation system takes the location information as goal position and guides the robot moving toward to the user upon calling. While the accuracy of the ZigBee-based location-aware system is limited, the robot uses Kinect to detect user's body and face to find the exact position of the user when the robot approaches the user. A visual tracking controller guides the robot to move toward to the user upon calling, and track the user in front of him/her by a set distance. For navigation control, we propose a self-localization system of a mobile robot by fusing the ZigBee-based location-aware system and odometer to improve the localization accuracy. Experimental results show that the average localization error is 68 cm in a 60 m travel. Experimental results also verified the effectiveness of finding the user upon calling at various distances and locations.

誌謝

謹向我的指導教授宋開泰博士致上感謝之意，感謝他三年來在專業上的指導，以他豐富的學識與經驗，配合理論的應用，使得本論文可以順利完成。

感謝博士班學長姐孟儒、嘉豪、格豪、巧敏、信毅以及允智的指導與建議，感謝與我共同奮鬥的同學建宏、士晟、碩成的相互鼓勵及提攜，以及學弟家昌、章宏、Carlos 在生活上帶來的樂趣。

最後，特別感謝我的爺爺、奶奶、爸爸、媽媽、姑媽、叔叔、大姊、二姊，由於他們的辛苦栽培，使我能順利求學至今，而完成這篇論文。在此以這篇論文獻給我摯愛的家人們。

目錄

圖例

表例

第一章 緒論

1.1 研究動機

近幾年來，機器人的科技發展日新月異，逐漸從傳統的工業應用走向家庭服務，對於幫忙家庭整潔、家庭保全、家庭看護或是娛樂家人，都已經是機器人可以完成的工作內容。但是實際應用上，使用者需要機器人時，還是得主動走向機器人面前來操作使用，這樣是非常麻煩的。況且面齡高齡化、少子化的社會趨勢，服務型機器人將會成為未來長者的助手，不能被召喚過來的機器人，對於需要被看護或是行動不便的長者而言是非常不實用的。所以服務型機器人必須從原本的被動等待，進步到能主動貼近需要服務的人。比如說當家人口渴時，可以召喚機器人過來，然後下達幫忙拿水的命令，讓機器人去幫忙拿水回來。又比如去逛大賣場時，消費者能在門口召喚機器人來迎接，協助幫忙購物，而不用消費者大費周章地找尋手推車，費時費力。延伸到長者照護方面，在長者有突發狀況時，機器人也能即時移動到長者面前給於照護，如[1]中提到當長者不小心跌倒時，長者所攜帶的無線感測裝置會自動發送緊急訊號，當機器人偵測到緊急訊號就會以簡訊通知其他家人，並且馬上前往長者所在位置，將影像畫面傳至家人的 PDA 上，讓家人第一時間了解狀況，把握住黃金時刻。機器人能夠隨傳隨到是很實用的功能，可以讓服務性機器人的功能更具完整性，所以本論文希望能夠設計一套機器人召喚系統，當使用者有需求時，能夠召喚機器人過來服務。

1.2 相關研究回顧

關於設計召喚機器人的方法，最直接聯想到的就是透過聲音來召喚機器人，讓機器人過來服務，所以有些研究就是專注於聲音這塊領域。[2]提出一套透過辨識名子來召喚機器人的方法，如圖 1-1 所示，機器人會先處於等待狀態，只要有聲音訊號被偵測到，就會進入名字辨識階段，然後當辨識名字正確，機器人會透過麥克風陣列，使用 Multiple Signal Classification (MUSIC)演算法來估算出聲源位置，然後前往使用者。或者是進入登記名子階段，讓更多不同使用者來登記

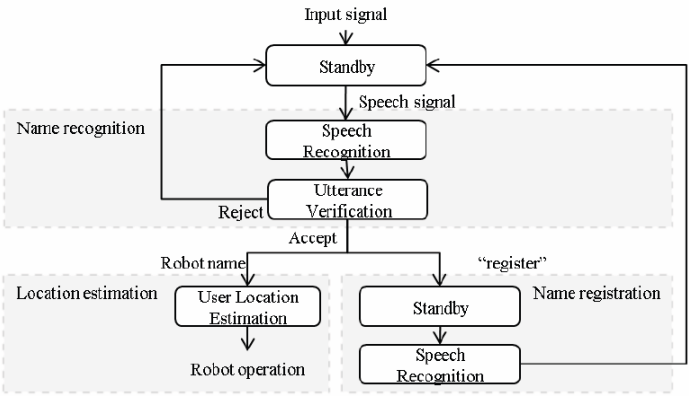

圖 1-1 透過聲音來召喚機器人的流程圖[2]

使用。[3]提出結合聲音與影像的方式來召喚機器人，機器人會先處於等待狀態，只要有聲音訊號被偵測到，就進入名字辨識階段，只要名字辨識正確，機器人會利用 Generalized Cross Correlation with Phase Transform (GCC-PHAT)方法與事先建立好的資料庫來估測出聲源位置，然後前往使用者，移動途中再透過人臉偵測來彌補聲音定位的誤差，使得機器人可以更靠近使用者。

以上兩種都是透過聲音來召喚機器人，只要名字辨識成功，機器人就會定位使用者的聲源，估算出相對距離與方向，然後前往使用者。此方法最大的好處就是使用自然的影音介面，可以讓使用者不需要攜帶額外的裝置就能實現召喚功能，並且透過不同的名字可以讓不同的使用者使用，或是由不同的語音內容來執行不同的任務。但是透過聲音來召喚機器人，必定會受限於距離與環境的影響，當使用者距離機器人太遠，或是環境障礙物太多，必定會影響聲音定位之結果，並且也無法在多個房間或長距離情況下的環境實現召喚機器人之功能。

而有別於聲音的方法，就是要透過其他人員偵測感測器來實現，[4]提出使用 Pyroeletric Infrared (PIR)感測器來進行室內定位，如圖 1-2 所示，將多顆 PIR 感測器安置於天花板上，當使用者進入 PIR 感測器的感測範圍內時，偵測到使用者的 PIR 感測器就會有訊號輸出，此訊號輸出會送至電腦端來進行資料整合，再透過演算法來估算使用者的所在位置。[5]提出結合 PIR 感測器與無線感測網路的方式來召喚機器人，如圖 1-3 所示，將 PIR 感測器與 Zigbee 無線感測網路模

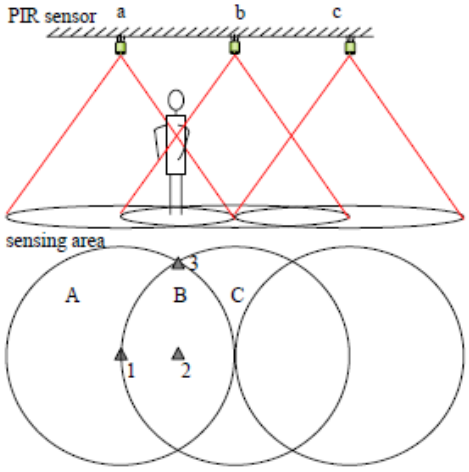

圖 1-2 PIR 感測器的定位示意圖[4]

圖 1-3 裝置與配置圖[5]

組結合為一體，採取格狀拓撲分佈來安裝於天花板上。讓 PIR 感測器來偵測使用者，將其訊號輸出透過 Zigbee 無線感測網路傳送到機器人，機器人再透過演算法來估算出使用者的所在位置。如此當使用者召喚時，機器人就會移動到使用者的所在位置。

以上兩種都是透過 PIR 感測器來定位使用者，實驗結果顯示的定位誤差不會超過 1.2 公尺，顯示出機器人能在使用者附近停下，讓使用者可以觸及使用。但是在多人使用的情況下，機器人就無法判別應該定位哪一個使用者。而且此方法也會受到環境的影響，只能在無障礙物的環境下使用，並且定位誤差再小，也無法保證機器人能準確的移動到使用者面前，如此就會帶來使用者使用上的困擾。

由以上的相關研究可知，要完成召喚系統這項設計，前提就是要有一套定位系統(Location-Aware System)，讓機器人知道使用者的所在位置，機器人才能透過導航來前往使用者身邊。

透過無線感測網路來進行室內定位[6][7]，是近年來熱門的研究議題，應用於機器人身上，讓機器人可以自我定位(Self-Localization)，也可以定位使用者的所在位置。目前無線感測網路室內定位的方法大致可歸類成以下四種，收訊時間法、收訊時間比較法、收訊角度法、接收訊號強度指標法。

收訊時間法(Time of Arrival, ToA)[8]，如圖 1-4 所示，已知參考點 A(x_1, y_1)、B(x_2, y_2)、C(x_3, y_3)的座標位置，再透過同步計時電路測量出目標點(x_m, y_m)與各參考點的接收訊號時間，然後透過數學模型求出彼此的距離 D_1、D_2、D_3，接著透過 Least Square 演算法來估算出目標點的位置座標。

圖 1-5 顯示收訊時間比較法(Time Difference of Arrival, TDoA)之功能原理[8][9]，目標點傳送訊號給三個以上的參考點，透過同步計時電路測量出目標點(x_m, y_m)與各參考點的接收訊號時間，然後透過數學模型求出彼此的距離 D_1、D_2、D_3，再利用雙曲線的特性，兩焦點至曲線上的距離差值是定值，來繪製兩組雙曲線 $D_3 - D_2$、$D_3 - D_1$，最後透過 Least Square 演算法來估算出兩組雙曲線之交點，即為目標點的所在位置。

收訊角度法(Angle of Arrival, AoA)[10][11]，則是利用具有方向性天線(Directional Antenna)或天線陣列(Antenna Array)來計算出各自天線與目標點之夾角，然後各自天線朝著目標點畫一直線，則兩條以上的直線之交點就可以視為目標點之位置，所以透過數學關係就能估算出目標點的所在位置，如圖 1-6 所示。

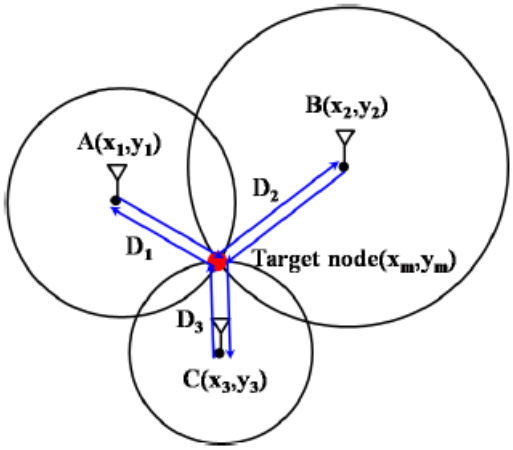

圖 1-4 ToA 位置估測[8]

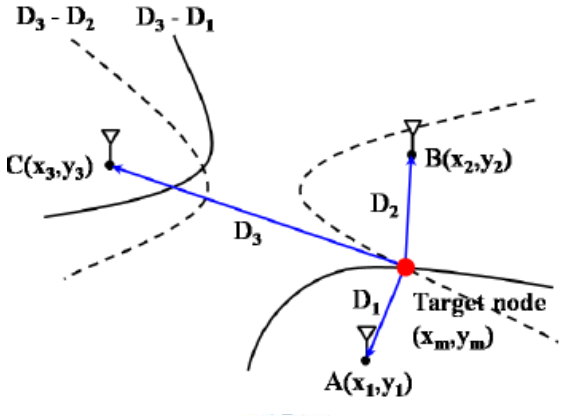

圖 1-5 TDoA 位置估測[8]

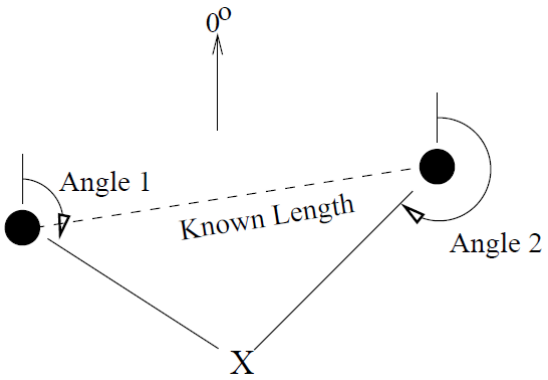

圖 1-6 AoA 位置估測[11]

接收訊號強度指標法(Received Signal Strength Indicator, RSSI)，是透過收集環境中的訊號強度來進行定位。因為不同的位置所接收到的訊號強度一定有所不同，所以可以透過比較訊號強度之差異來估測出目標點位置，基於這個特性，就有許多種使用訊號強度之定位方法被發展出來，其中最常見的是樣式比對法[12-14]，如圖 1-7 所示。圖 1-7 (a)為訓練階段，首先在環境中佈署許多個存取點(Access Point, AP)，然後讓使用者(Mobile User, MU)站在參考點(Reference Point, RP)上，收集來自多個存取點之訊號強度，並且將所收集的訊號強度儲存到資料庫裡，完成第一次訓練。然後使用者移至另一個參考點，重複以上步驟，直到所有的參考點都被訓練過，訓練階段就算完成。圖 1-7 (b)為位置估測，當使用者站在已訓練過的參考點區域，就可以將目前所在位置收集到的訊號強度與之前建立好的資料庫進行比對，透過各種演算法來估測出使用者之位置。

基於收訊時間法、收訊時間比較法與收訊角度法都需要特殊且精確的裝置，對於成本考量上都是一大負擔，而且在室內環境下，容易受到多重路徑(Multi-path)之影響，造成收訊時間法、收訊時間比較法與收訊角度法之定位準確性下降。相對而言，訊號強度法不需要特殊的裝置，實現起來也簡單許多。但是只要環境太大，訓練階段所花費的時間就越冗長，而且使用無線感測網路來進行室內定位，必定會有誤差，導致機器人就不能準確的移動到使用者面前，如此就會帶來使用者使用上的困擾。

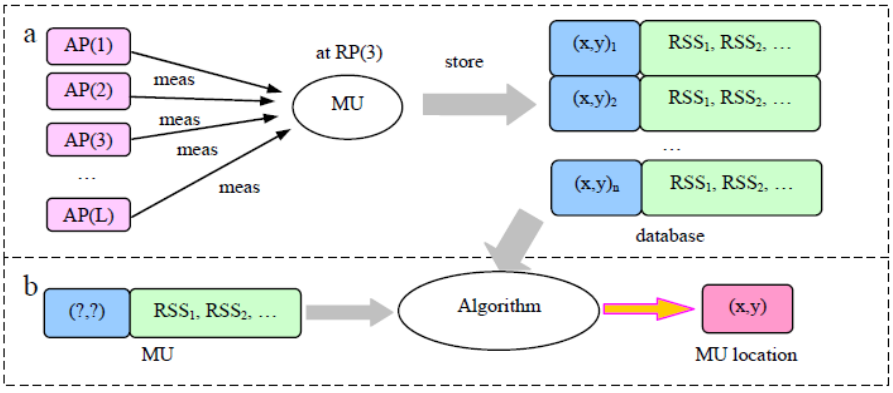

圖 1-7 樣式比對法流程圖[14]

由文獻調查可知，以上的定位方法都有各自的優點、也有各自的缺點。聲音定位會受限於聲音傳遞之距離無法太遠，導致只能局限在一個區域內使用，對於實現長距離的召喚系統是無法使用的。使用 PIR 感測器來進行定位，也只能侷限在無障礙物的環境下，對於實際應用上，會產生許多問題。而使用無線感測網路來進行室內定位，則是會遇到定位誤差之影響，導致機器人無法準確的移動到使用者面前。

1.3 問題描述

本論文想要實現一個機器人召喚系統，當使用者有需求時，機器人可以不受環境距離、障礙物等影響，順利找到使用者，並且移動到使用者面前。實現機器人召喚系統之前提，必須先有一套良好的定位系統，才可以讓機器人知道需要服務的使用者在哪裡，所以在定位系統選用上，本論文採用無線感測網路來進行室內定位，讓機器人可以定位使用者之位置並且不受環境距離之影響。但是使用無線感測網路來進行室內定位，必定會受到定位誤差之影響，導致機器人無法準確的移動到使用者面前，所以本論文想透過影像資訊來彌補這項缺點，讓無線感測網路先定位使用者之位置，然後當機器人接近使用者時，再透過影像追蹤使用者之方式，讓機器人準確的移動到使用者面前。

另一方面，機器人需要自我定位系統，一般採用里程計(Odometer)來推算出機器人之位置與朝向角，此方法在短距離內尚稱準確，但是當機器人移動較遠的距離後，其準確度會因為兩輪打滑造成的誤差、不平的路面造成的誤差、輪胎的磨損及載重不均造成輪胎半徑的改變等因素影響，造成嚴重的定位誤差。而無線感測網路室內定位之結果會隨機跳動，但是其結果為絕對座標並且跳動為一個有限範圍，所以本論文希望融合里程計與無線感測網路室內定位之特性，讓機器人定位誤差不會隨著里程計之誤差累積而發散，同時也降低無線感測網路室內定位之不確定性，使得機器人在長時間、長距離下的移動後，還能保持自我定位之準確性。

在機器人導航系統設計上，採用[15]所提出的導航控制當作基礎，讓機器擁有原本隨著環境變化而自行調整之能力外，進一步整合影像追蹤與自我定位系統，使得機器人可以順利導航到使用者面前，完成本論文想要實現的機器人召喚系統。

1.4 章節說明

本論文共分為七章，第一章為描述相關研究背景，並說明研究動機與目的。第二章介紹本論文所使用的定位系統。第三章為機器人自我定位系統之設計。第四章說明如何透過影像來偵測使用者以及影像追蹤控制器之設計。第五章為整合影像追蹤控制器與自我定位系統之機器人導航系統設計。第六章為機器人自我定位、影像追蹤控制器以及機器人召喚系統之實驗結果。最後，第七章為結論與未來展望。

第二章 無線感測網路室內定位

本章節將介紹本論文所發展的定位系統(Location Aware System)，說明所採用的無線感測網路裝置如何進行室內定位，以及定位參數之設定，並且在環境中，如何佈署無線感測網路之節點。

2.1 CC2431定位引擎

本論文之定位系統採用 TI 公司的 Zigbee CC2431[16]為基礎裝置所組成，透過 Zigbee CC2431 晶片上的定位引擎(Location Engine)，可以在不添加任何定位裝置下，估測出 Zigbee 無線感測網路中某個節點的位置，其定位範圍可達 64×64 平方公尺，成本低廉、耗電量低、穩定性高，以及不錯的定位準確度，非常適合用於室內定位。而定位系統如圖 2-1 所示，分別由參考點(Reference Node, RN)、盲點(Blind Node, BN)和協調介面(Dongle)三者所組成。

參考點是一個已知自身位置的靜態節點，功用是將自身的位置訊息(x_n, y_n)透過 Zigbee 感測網路發送給盲點。而盲點是一個待估測的動態節點，會向參考點請求並接受參考點的位置訊息，並且間接獲得盲點與參考點之間的訊號強度值(Received Signal Strength Indication, RSSI)，等收集時間結束後，盲點會將收集到的參考點之位置訊息(x_n, y_n)與 RSSI 送到 CC2431 晶片上的定位引擎，透過定位引擎來計算出盲點之自身位置，並且將盲點自身位置透過 Zigbee 無線感測網路傳送到協調介面。最後，協調介面則是連接主控電腦(PC)的靜態節點，負責將收到的盲點位置藉由 RS232 傳送到主控電腦，同時也是將主控電腦所設定的參考點之自身座標或盲點之定位參數，透過 Zigbee 無線感測網路發送出去。

完整的定位系統工作流程如圖 2-2 所示，協調介面負責將主控電腦設定的參考點之自身座標(RN Configuration)與盲點之定位參數(BN Configuration)，透過 Zigbee 無線感測網路發送出去。盲點負責找出自身位置，首先向附近的參考點發送請求位置訊息(XY-RSSI Request)，只要參考點收到請求位置資訊，就會回傳自身位置資訊(XY-RSSI Response from RN)給盲點，等到收集完附近參考點之位置

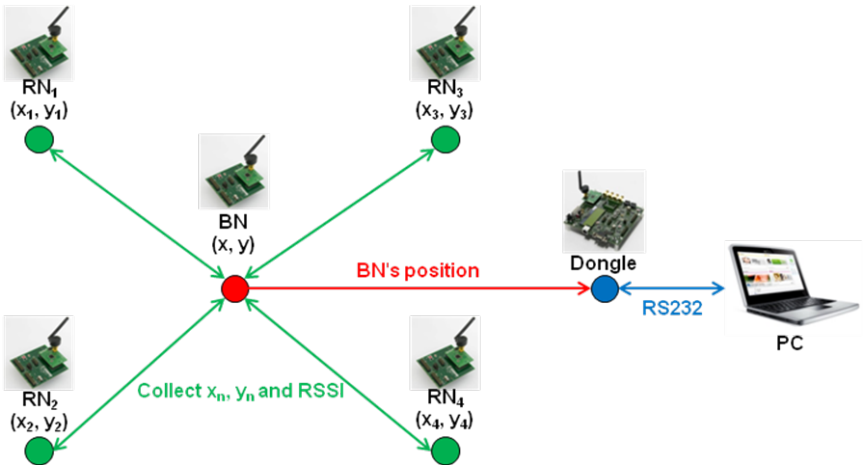

圖 2-1 定位系統架構圖

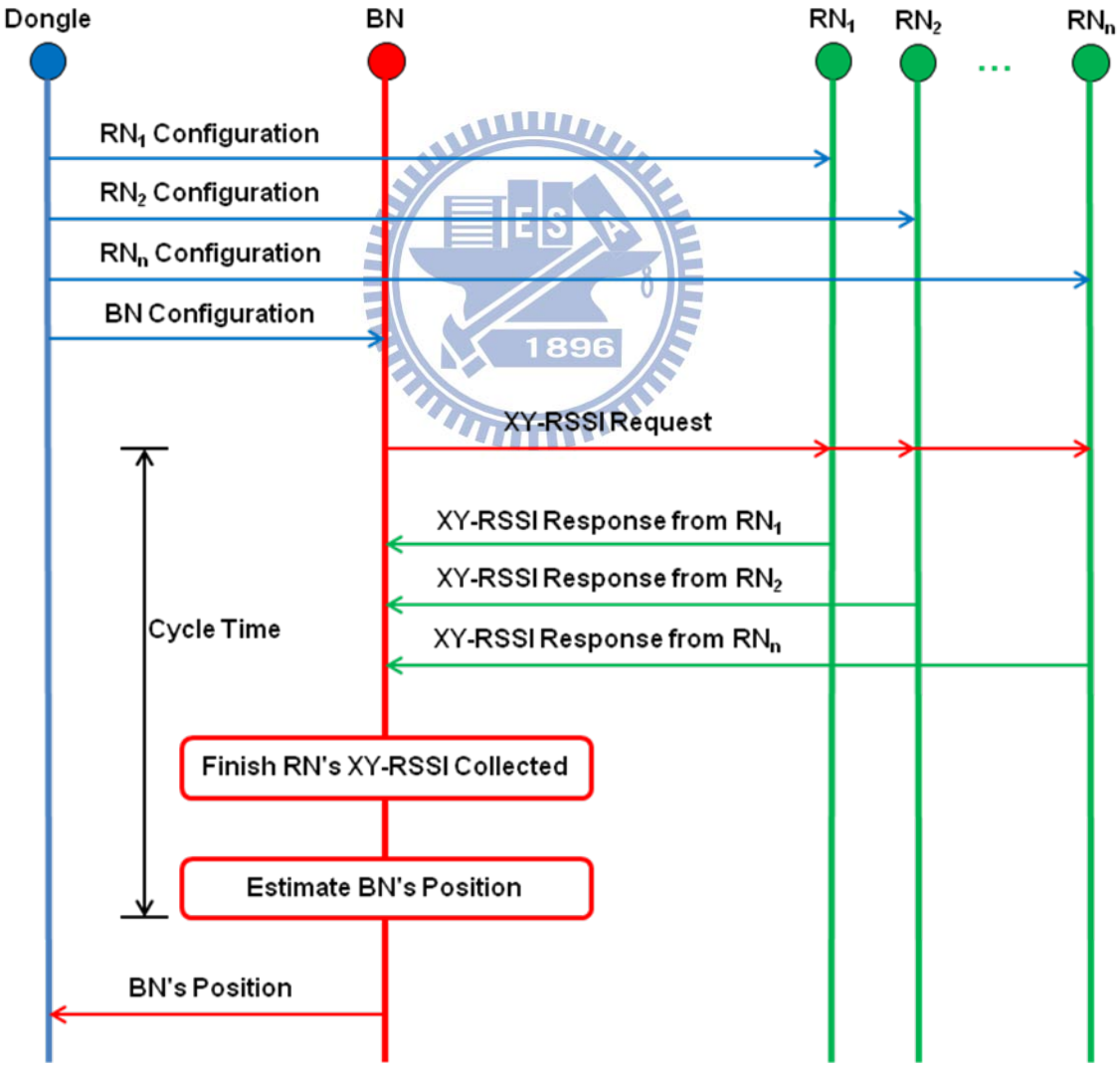

圖 2-2 定位系統工作示意圖

訊息(Finish RN's XY-RSSI Collected)，盲點會將收集到的參考點之位置訊息(x_n, y_n)與 RSSI 送到 CC2431 晶片上的定位引擎，透過定位引擎來計算出盲點之自身位置(Estimate BN's Position)，並且將盲點自身位置(BN's Position)透過 Zigbee 無線感測網路傳送到協調介面，而整個定位盲點之週期時間(Cycle Time)設定為 1 秒，週期時間從盲點發送請求位置訊息開始，直到盲點計算出自身位置後結束。

另一方面，由於參考點的自身位置已知，因此不需要定位引擎，這也表示參考點能使用 Zigbee CC2430[17]來取代，但是盲點一定要使用 Zigbee CC2431。

2.2 定位原理

CC2431 定位引擎是基於 RSSI 進行定位，每當盲點接收到參考點所發送的位置訊息，就能從位置訊息中讀取到盲點與參考點之間的 RSSI，然後將所收到的 RSSI 利用經驗模型轉化為彼此間的距離，如圖 2-3 所示，RSSI 會隨著傳送距離而衰減，當盲點與參考點之間的距離越近，接收到的 RSSI 越強，盲點與參考點之間的距離越遠，接收到的 RSSI 越弱，可以將此關係看成一條對數函數，如(2-1)所示：

$$RSSI = -(10 \cdot n \cdot \log_{10} d + A) \tag{2-1}$$

其中

RSSI 為接收訊號強度指標，單位 dBm

n 為訊號衰減係數，隨著盲點與參考點之間的距離增加，其 RSSI 衰減之速率，單位 dBm

d 為盲點與參考點之間的距離，單位 m

A 為盲點與參考點之間距離 1 公尺的 RSSI，單位 dBm

由(2-1)進行簡單的數學轉換，就能將 RSSI 換算成距離，如(2-2)所示：

$$d = 10^{\left(\frac{-A-RSSI}{10n}\right)} \tag{2-2}$$

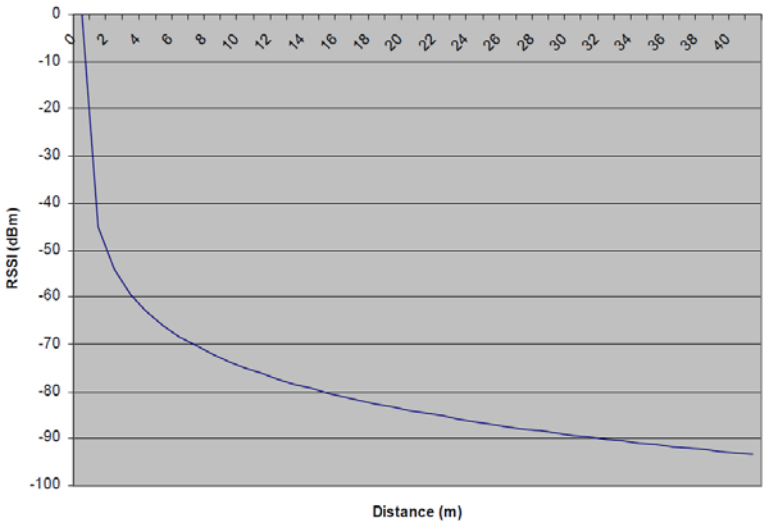

圖 2-3 訊號強度與距離之關係圖[16]

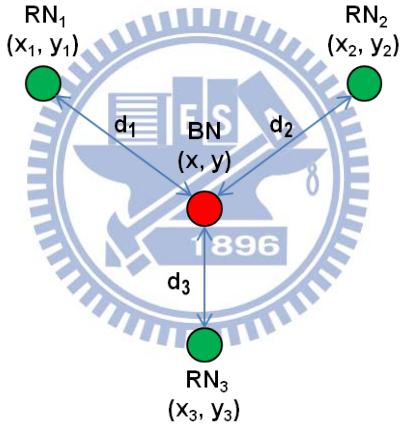

圖 2-4 盲點與參考點之間的距離關係圖

如圖 2-4 所示，當盲點獲得附近得參考點位置(x_n, y_n)與彼此間的距離 d_n 後，就能利用數學關係來求出盲點的所在位置(x, y)。在此，CC2431 定位引擎是採用[18]所提出的 Maximum Likelihood Estimation Method 來計算盲點的自身位置。當盲點收到來自 n 個參考點所傳來的位置訊息(x_1, y_1)、(x_2, y_2)、…、(x_n, y_n)以及透過 RSSI 所推算出盲點與參考點之間的距離 d_1、d_1、...、 d_n，利用兩點之間的距離公式，可以將盲點與 n 個參考點之間的位置關係表示為(2-3)所示：

$$\begin{cases} (x-x_1)^2+(y-y_1)^2=d_1 \\ \vdots \\ (x-x_n)^2+(y-y_n)^2=d_n \end{cases} \tag{2-3}$$

其中

x、 y 為盲點的自身位置，單位 m

x_n、y_n 為第 n 個參考點的位置訊息，單位 m

d_n 為盲點與第 n 個參考點之間的距離，單位 m

然後將(2-3)的每一項與第 n 項進行相減後展開，如(2-4)所示：

$$\begin{cases} x_1^2-x_n^2-2(x_1-x_n)x+y_1^2-y_n^2-2(y_1-y_n)y=d_1^2-d_n^2 \\ \vdots \\ x_{n-1}^2-x_n^2-2(x_{n-1}-x_n)x+y_{n-1}^2-y_n^2-2(y_{n-1}-y_n)y=d_{n-1}^2-d_n^2 \end{cases} \tag{2-4}$$

如此就能將(2-4)看成一個 AX=B 矩陣關係，如(2-5)~(2-7)所示：

$$A=\begin{bmatrix} 2(x_n-x_1) & 2(y_n-y_1) \\ \vdots & \vdots \\ 2(x_n-x_{n-1}) & 2(y_n-y_{n-1}) \end{bmatrix} \tag{2-5}$$

$$B=\begin{bmatrix} d_1^2-d_n^2-(x_1^2+y_1^2)+(x_n^2+y_n^2) \\ \vdots \\ d_{n-1}^2-d_n^2-(x_{n-1}^2+y_{n-1}^2)+(x_n^2+y_n^2) \end{bmatrix} \tag{2-6}$$

$$X=\begin{bmatrix} x \\ y \end{bmatrix} \tag{2-7}$$

透過最小平方近似解(Least Square Method)就能計算出盲點的自身位置，如(2-8)所示：

$$\overline{X}=(A^TA)^{-1}A^TB \tag{2-8}$$

其中

$\overline{X}=\begin{bmatrix} \overline{x} \\ \overline{y} \end{bmatrix}$ 為盲點的自身位置之近似解，單位 m

以上(2-1)~(2-8)均在 CC2431 定位引擎內部進行計算。CC2431 定位引擎會自動向附近參考點請求位置訊息(x_n, y_n)與 RSSI，然後經由盲點上的定位引擎來計算出盲點的自身位置 $\overline{X}$，最後盲點將自身位置 $\overline{X}$ 透過 Zigbee 無線感測網路傳送到協調介面，完成一次定位盲點之週期時間。當協調介面收到盲點的自身位置 $\overline{X}$ 後，就會將盲點的自身位置 $\overline{X}$ 透過 RS232 傳送到主控電腦。

2.3 定位參數設定

由(2-2)可知，要獲得距離 d，除了 RSSI 外，使用者必須根據所在環境來選擇定位參數 A 與 *n*，所以本論文設計以下兩個實驗來找出定位參數 A 與 *n*。

根據 A 的定義為盲點與參考點之間距離 1 公尺之 RSSI，本論文將盲點與參考點依每隔 45 度、相隔 1 公尺的方式來擺放，如圖 2-5 所示，讓盲點收集來自八個方向之參考點所傳送的 RSSI，其結果如表 2-1 所示，結果平均 RSSI 為-45 dBm。

根據 *n* 的定義為盲點與參考點之間的距離增加，其 RSSI 衰減之速率，本論文採取由(2-1)來反求 *n*，如(2-9)所示：

$$n = \frac{-A - RSSI}{10 \cdot \log d} \tag{2-9}$$

首先收集盲點與參考點相距 1 公尺的 RSSI，當作定位參數 A 來使用，再讓盲點與參考點在相同直線下、每隔 1 公尺，直到相隔 7 公尺方式來擺放，如圖 2-6 所示，然後盲點收集來自相同方向之參考點所傳送的 RSSI，透過(2-9)來求出 *n*，其結果如表 2-2 所示，其平均 *n* 為 3.179 dBm。

因為 CC2431 定位引擎並非直接採用定位參數 *n* 值，而是透過查表 2-3 來獲得整數索引值(*n_index*)，才能寫入到 CC2431 定位引擎，所以平均 *n* 為 3.179 dBm，透過查表所得之 *n_index* 為 14。最後將 A = -45 與 *n_index* = 14 透過協調介面傳送到盲點，完成定位參數之設定。

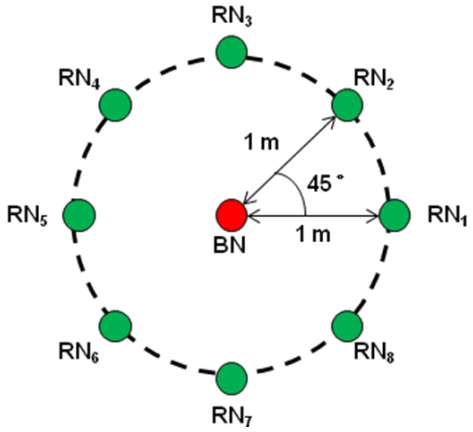

圖 2-5 計算定位參數 A 之裝置擺設

表 2-1 計算定位參數 A 之實驗結果

Angle (degree)	0°	45°	90°	135°	180°	225°	270°	315°
RSSI (dBm)	-40	-46	-47	-46	-42	-44	-46	-49
Average RSSI (dBm)	-45							

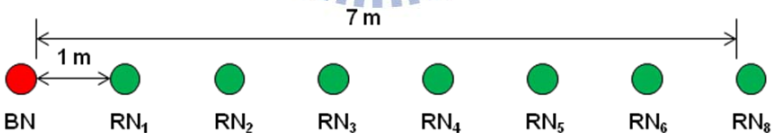

圖 2-6 計算定位參數 *n* 之裝置擺設

表 2-2 計算定位參數 *n* 之實驗結果

Meter (m)	1	2	3	4	5	6	7
RSSI (dBm)	-47	-56	-56	-68	-69	-81	-74
n (dBm)		2.990	1.886	3.488	3.148	4.369	3.195
Average *n* (dBm)	3.179						

表 2-3 *n* 與 *n_index* 之對應圖[16]

n index	*n*	*n index*	*n*
0	1.000	16	3.375
1	1.250	17	3.500
2	1.500	18	3.625
3	1.750	19	3.750
4	1.875	20	3.875
5	2.000	21	4.000
6	2.125	22	4.125
7	2.250	23	4.250
8	2.375	24	4.375
9	2.500	25	4.500
10	2.625	26	4.625
11	2.750	27	5.000
12	2.875	28	5.500
13	3.000	29	6.000
14	3.125	30	7.000
15	3.250	31	8.000

2.4 無線感測網路節點佈置

CC2431 定位引擎透過附近參考點的 RSSI 來計算盲點的所在位置。由於室內的天花板、地面和牆壁會吸收 RSSI，為了確保 CC2431 定位引擎的性能，最佳的方案是讓所有節點的高度相同，並且遠離天花板、地面和牆壁，所以為了盡可能合乎要求，本論文讓參考點放置在高度 0.45 公尺的小圓凳上，而盲點放置在使用者、機器人的肩部和腰部之間，以降低天花板、地板吸收 RSSI 之情況。

一般來說，要獲得一個可靠的定位座標至少需要三個參考點，如果參考點太少，會因為環境因素或其他訊號互相干擾，造成錯誤的RSSI值而影響定位結果。此外，如果盲點跑出參考點的範圍外，也會造成定位結果與實際位置相差過大，所以為了防止此現象，普遍都建議將參考點擺放在房間的各個角落，讓整個房間涵蓋在參考點的範圍內，所以每間房間需要佈置至少四個以上的參考點。但在實際定位下，會發現在房間中央處的盲點，因為距離角落的參考點過遠，導致收到的RSSI值微弱，以至於盲點的定位結果與實際位置相差過大。所以本論文設計一種參考點的佈置方法，如圖2-6所示，場地為實驗室與外面走廊，在每間房間或區域佈置至少四個以上的參考點(綠色圓點)，然後呈現交叉擺放，讓每間房間或區域內RSSI分佈平均，其定位結果的準確度也會穩定許多。最後，由圖2-6之參

考點佈置來進行定位實驗，讓盲點(紅色圓點)分別放置在已知位置座標(0.75, 1)、(0.75, 2)、…、(5, 6)，共26個已知位置上來進行位置估測，然後再將定位結果與已知位置進行比較。總共進行八次不同的時段、相同的實驗，平均八次結果如表2-4所示，平均定位誤差為1.325公尺，最大定位誤差會達到3.287公尺，而標準差為0.078。顯示此定位系統可以讓機器人定位使用者，其定位誤差不大，足以讓攝影機可以擷取到使用者，並且透過影像來追蹤使用者。

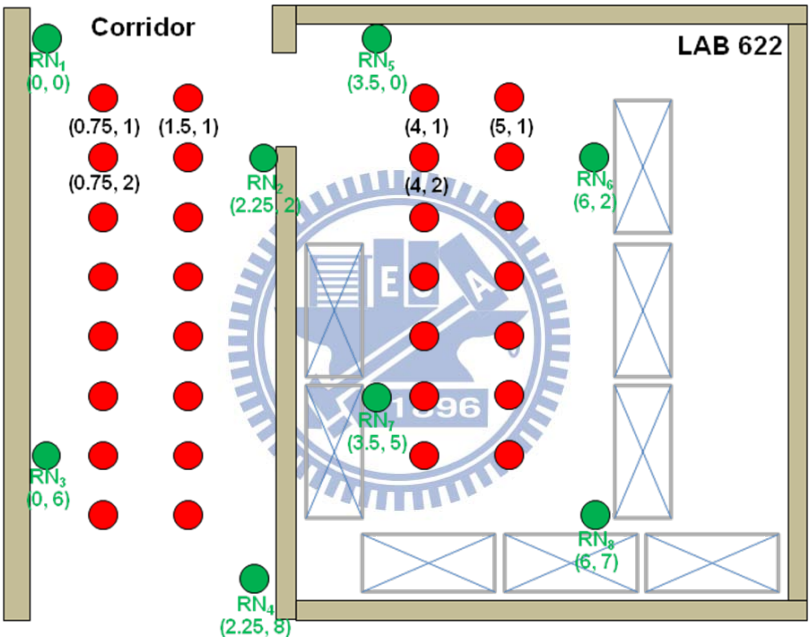

圖2-6 定位實驗環境圖

表2-4 定位實驗結果

	X-axis	Y-axis	Position
Mean error (m)	0.645	0.991	1.325
Maximum error (m)	1.803	3.059	3.287
Minimum error (m)	0.019	0.034	0.259
Standard deviation	0.090	0.090	0.078

2.5 結論與討論

要實現機器人召喚系統之前提，是先有一套良好的定位系統，才可以讓機器人知道需要服務的使用者在哪裡。所以本論文採用 CC2431 定位引擎來進行室內定位，只要使用者持有盲點，機器人就能估測出使用者的所在位置，雖然定位結果仍有 1.3 公尺的平均定位誤差，但是足以讓攝影機可以擷取到使用者，並且透過影像來追蹤使用者。

另一方面，機器人的自我定位系統，一般採用里程計(Odometer)來推算出機器人之位置與朝向角，此方法在短距離內尚稱準確，但是當機器人移動較遠的距離或是運作時間過長，其準確度就會因為里程計之誤差累積而發散。雖然定位系統之定位結果會隨機跳動，呈現一個有限範圍的定位誤差，但其定位結果是相對於世界座標之數值，沒有累積誤差之問題。所以本論文將在下一個章節討論，如何結合這兩種的定位系統，設計出一個能長距離、長時間移動而不失定位準確度的機器人自我定位系統。

第三章 機器人自我定位系統設計

本章節將介紹如何將第二章所發展的定位系統，應用於機器人的自我定位系統，讓里程計與 CC2431 定位引擎之優點互相結合，設計出一個可以長距離、長時間移動而不失定位準確度的機器人自我定位系統。

3.1 基於里程計之機器人自我定位

基於軸編碼器之里程計(Odometer)是機器人常用的自我定位方法，可以估測出機器人的姿態資訊(X, Y, θ)，其中(X, Y)是機器人在世界座標的位置，θ 為機器人的朝向角。里程計的定位方法是，先讓機器人啟動時設定好自身的初始姿態，當機器人移動時，再利用馬達軸編碼器(Encoder)所累積的脈波數(Pulse)來推算行駛距離以及改變的角度，將此與初始姿態相加，即可知道機器人目前的姿態。輪式機器人的運動模型如圖 3-1 所示，里程計的計算方式如(3-1)~(3-5)所示[15]：

$$d\theta(\mathrm{t}) = \frac{1}{\mathrm{E}}(ds_r(\mathrm{t}) - ds_l(\mathrm{t})) \tag{3-1}$$

$$ds(\mathrm{t}) = \frac{1}{2}(ds_r(\mathrm{t}) - ds_l(\mathrm{t})) \tag{3-2}$$

$$\mathrm{X}(\mathrm{t}) = \mathrm{X}(\mathrm{t}-1) + ds(\mathrm{t}) \times \cos\left(\frac{\theta(\mathrm{t}) + \theta(\mathrm{t}-1)}{2}\right) \tag{3-3}$$

$$\mathrm{Y}(\mathrm{t}) = \mathrm{Y}(\mathrm{t}-1) + ds(\mathrm{t}) \times \sin\left(\frac{\theta(\mathrm{t}) + \theta(\mathrm{t}-1)}{2}\right) \tag{3-4}$$

$$\theta(\mathrm{t}) = \theta(\mathrm{t}-1) + d\theta(\mathrm{t}) \tag{3-5}$$

其中

$d\theta$(t)為機器人在 t-1 至 t 時間內的朝向角差值

ds(t)為機器人在 t-1 至 t 時間內的移動距離

ds_r(t)、ds_l(t)為機器人右輪、左輪在 t-1 至 t 時間內的移動距離

E 為機器人的兩輪之間的距離

X(t)、Y(t)、θ(t)為機器人在 t 時間的姿態資訊

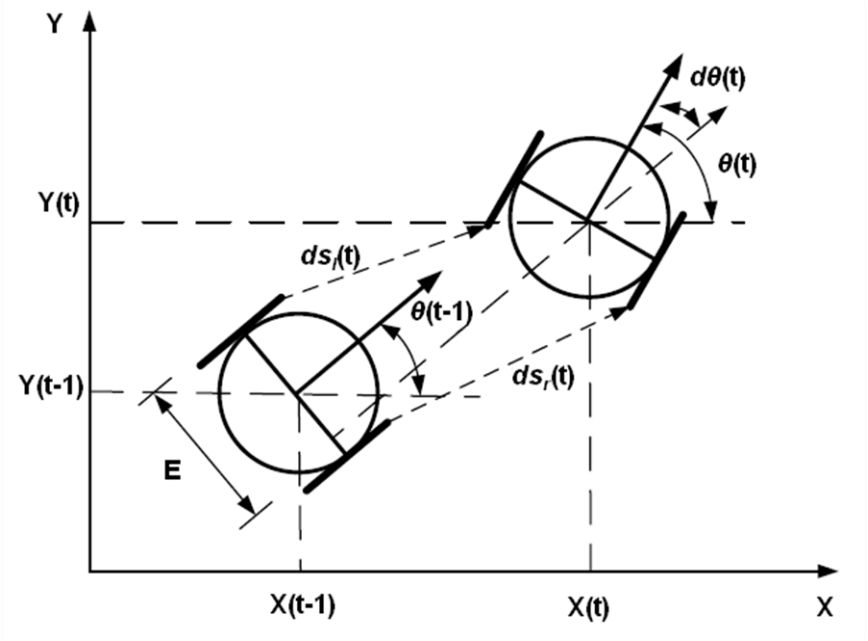

圖 3-1 輪式機器人的運動模型

此自我定位方法在短距離內尚還算準確，但是當機器人移動較遠的距離後，其準確度會因為兩輪打滑造成的誤差、不平的路面造成的誤差、輪胎的磨損及載重不均造成輪胎半徑的改變等因素影響，造成里程計累積嚴重的誤差，進而影響到機器人的姿態資訊。

3.2 融合里程計與 RSSI 之機器人自我定位設計

CC2431 定位引擎是基於 RSSI 來進行定位，由於 RSSI 容易受到環境影響，造成定位結果隨機跳動，呈現一個有限範圍的定位誤差。但其定位結果是相對於世界座標之數值，沒有累積誤差的問題，所以本論文結合兩種定位系統之優點來設計一套新的定位系統，當機器人移動距離不長時，其里程計累積誤差還在可容忍範圍內，讓機器人的定位系統由里程計主導，而當機器人移動距離過長時，其里程計累積誤差已經過大，而 CC2431 定位引擎的定位誤差仍為有限範圍內，且具有絕對座標之優點，所以讓機器人的定位系統由 CC2431 定位引擎主導，來修正機器人的自身位置。如此，結合這兩種誤差性質不同的定位系統，互相彌補彼此的不足之處以降低機器人之定位誤差，讓機器人能在長距離的移動下，仍不會

迷失自身的位置，順利找到召喚者。

因為 RSSI 會受到環境影響，導致估測物的位置會隨機跳動，難以建立物理模型。而機器人的定位系統何時由里程計或是 CC2431 定位引擎來主導，也是難以確定，如果單獨只依靠其中一種定位系統時，可能會因為一個錯誤的資訊，而導致機器人定位錯誤。所以對於這種不明確的資料處理上，本論文採用模糊邏輯控制來解決。

模糊邏輯控制是一種模仿人類思考、處理所有不明確的物理模型之數學控制。與傳統控制不同的是，模糊邏輯控制並非單純將邏輯二分化，而是可以依靠人類的直覺或是操作者的經驗，將原本的只有 0 或 1 的邏輯判斷，依照人類的主觀判斷，給予 0 到 1 之間的數字來表示資料的歸屬度(Membership)，然後透過模糊集合理論將人類的經驗與直覺表達成語言規則(IF…THEN…)，讓電腦來執行這些模糊規則。簡言之，模糊邏輯控制具有下列特點：

1. 用語言規則來描述系統變數間的關係。
2. 不需要知道控制對象的數學模型。
3. 模糊邏輯控制對於參數的變化具有很強的適應性。

基於以上特點，本論文採用模糊邏輯控制來設計機器人定位系統，讓里程計與 CC2431 定位引擎進行融合。

在 CC2431 定位引擎方面，雖然具有可信度的絕對座標與有限範圍定位誤差，但是對於不穩定的定位結果，會導致機器人的定位誤差變大，所以必須將 CC2431 定位引擎的穩定度列入考慮。當連續相鄰的兩個時間內，所估測出來的定位結果相差太多，就表示在這個時刻 CC2431 定位引擎的穩定度不是很好，那麼 CC2431 定位引擎的權重應該變小；相反的，當連續相鄰的兩個時間內，所估測出來的定位結果很接近，就表示在這個時刻 CC2431 定位引擎的穩定度不錯，那麼 CC2431 定位引擎的權重應該變大。現今之 CC2431 定位引擎為每 1 秒讀取一次定位結果。

在里程計方面，雖然在短距離內擁有不錯的定位準確性，但是卻有累積誤差

的問題，會導致機器人移動過遠後，機器人的定位誤差變大，所以必須將機器人移動距離列入考慮。當機器人開始移動時，此時里程計累積的誤差還不大，里程計的權重應該變大；相反的，當機器人移動距離過長時，里程計累積的誤差變大，里程計的權重應該變小。根據實驗發現，現今的里程計在機器人移動到 6 公尺左右時，里程計的累積誤差會開始變大。

所以本論文所設計的機器人自我定位系統如圖 3-2 所示，輸入為 CC2431 定位引擎的穩定度 D_{Rz} 和機器人移動距離 D_{Ro}，輸出為機器人的世界座標位置(X_R, Y_R)。當機器人移動距離超過 6 公尺時，CC2431 定位引擎會開始介入修正機器人

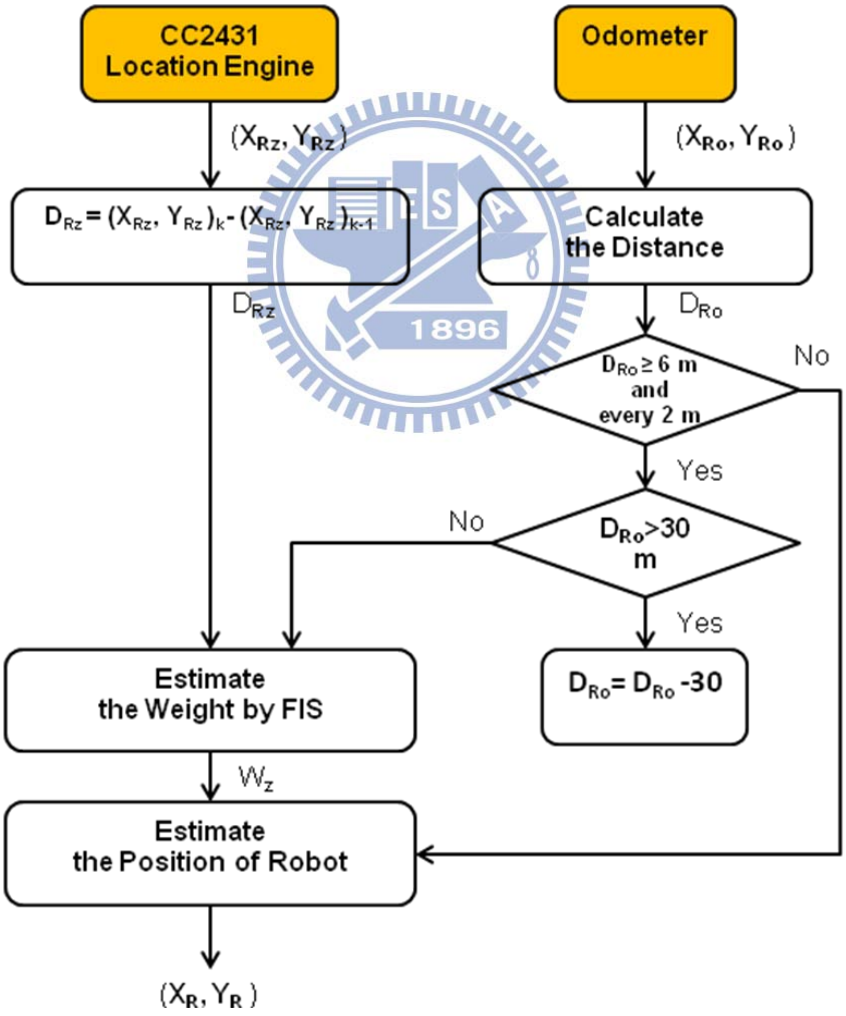

圖 3-2 機器人自我定位系統流程圖

的世界座標位置。然後每隔 2 公尺修正一次，這是為了防止 CC2431 定位引擎過多的修正而導致機器人的世界座標位置也變成隨機跳動，進而喪失了里程計定位穩定之優點。之後隨著機器人移動距離的增加，里程計的權重變小，機器人自我定位會漸漸地由 CC2431 定位引擎來主導，把里程計累積誤差的影響給降低，直到機器人移動距離超過 30 公尺後，里程計累積誤差之影響已經被修正的差不多，再將機器人的定位系統重新交給里程計主導，重新循環以上流程。

3.3 模糊邏輯資訊融合設計

對於如何分配兩種不同性值的定位系統問題，本論文使用模糊邏輯控制來解決，如圖 3-3 所示，模糊邏輯控制器包含模糊化(Fuzzifier)、推論引擎(Inference Engine)、模糊規則(Fuzzy Rule Base)以及解模糊化(Defuzzifier)四個部分。輸入為 CC2431 定位引擎的穩定度 D_{Rz} 和機器人移動距離 D_{Ro}，而輸出為 CC2431 定位引擎的權重值 W_z，只要決定好一個權重值後，另一個就能依靠比例去推算求得。

首先將兩個輸入值 D_{Rz}、D_{Ro} 進行模糊化，在輸入端的歸屬函數(Input Membership Function)設計上，將 CC2431 定位引擎的穩定度 D_{Rz} 分成五個等級，分別為定位估測差異非常小(Very Small)、定位估測差異小(Small)、定位估測差異中等(Medium)、定位估測差異大(Big)、定位估測差異非常大(Very Big) ，如圖 3-4 所示。同樣的，也將機器人移動距離 D_{Ro} 分成五的等級，分別為機器人移動距離很短(Very Short)、移動距離短(Short)、移動距離中等(Medium)、移動距離長(Long)、移動距離很長(Very Long)，如圖 3-5 所示。

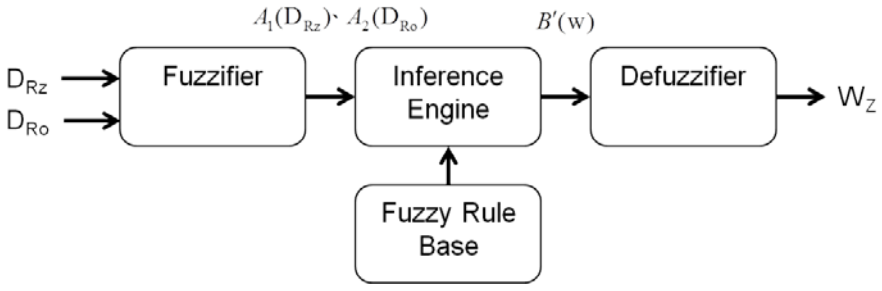

圖 3-3 模糊邏輯控制應用於機器人自我定位系統

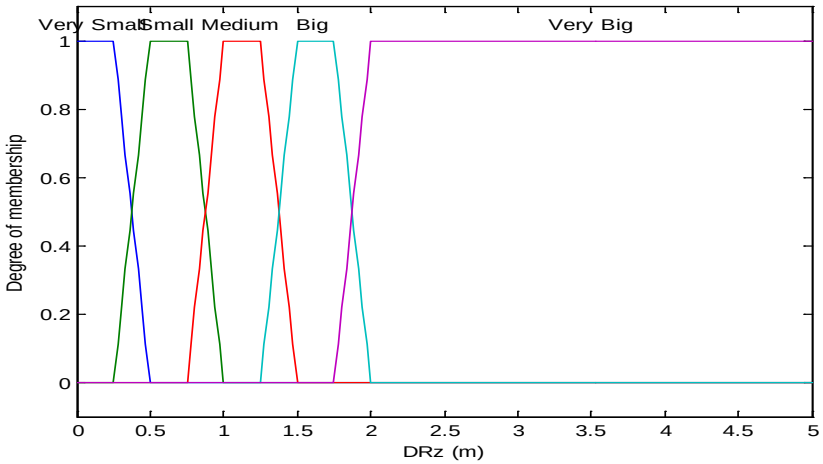

圖 3-4 CC2431 定位引擎穩定度的歸屬函數

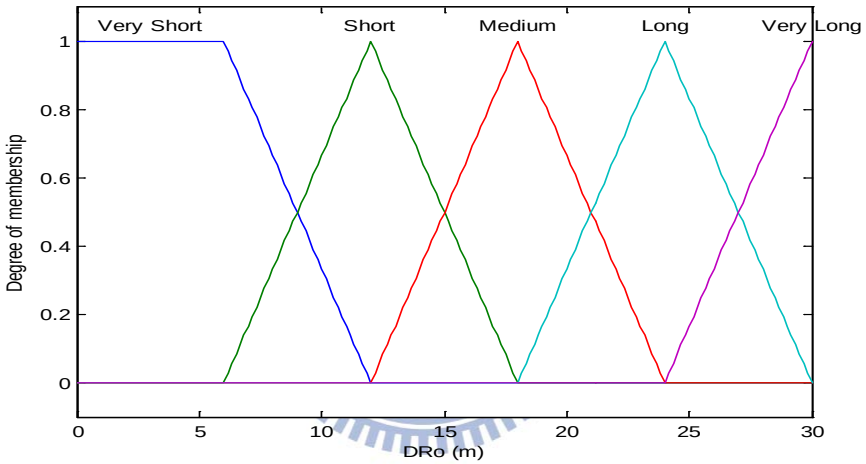

圖 3-5 機器人移動距離的歸屬函數

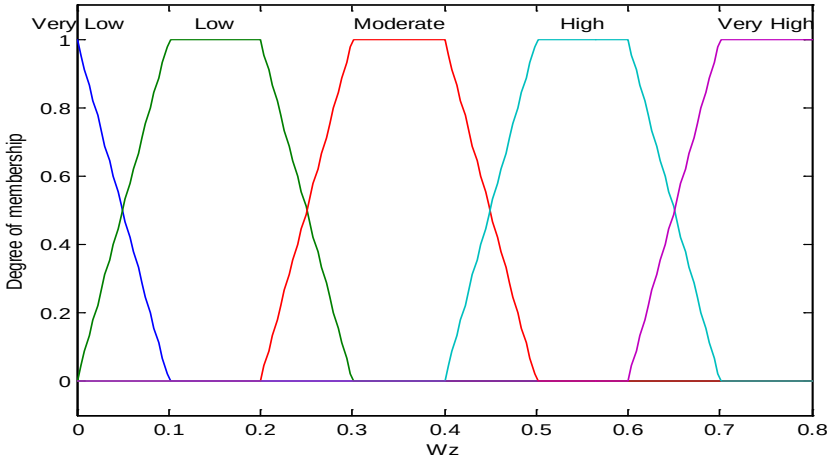

圖 3-6 CC2431 定位引擎權重值的歸屬函數

表 3-1 模糊規則表

		D_{Ro}				
		Very Short	Short	Medium	Long	Very Long
D_{Rz}	Very Small	Low	Moderate	High	Very High	Very High
	Small	Very Low	Low	Moderate	High	High
	Medium	Very Low	Very Low	Low	Moderate	Moderate
	Big	Very Low	Very Low	Very Low	Low	Low
	Very Big	Very Low	Very Low	Very Low	Very Low	Very Low

而在輸出端的歸屬函數(Output Membership Function)設計上，將 CC2431 定位引擎的權重值分成五個等級，分別為權重值很低(Very Low)、權重值低(Low)、權重值中(Moderate)、權重值高(High)、權重值很高(Very High)，如圖 3-6 所示。

透過模糊化，便能將輸入值 D_{Rz}、D_{Ro} 轉換到相應歸屬函數值 $A_1(D_{Rz})$、$A_2(D_{Ro})$，然後再將歸屬值送到推論引擎，搭配模糊規則便可得到 CC2431 定位引擎的權重輸出模糊集合 $B'(w)$。而其中模糊規則如表 3-1 所示。在機器人剛開始移動時，除非 CC2431 定位引擎的定位估測差異非常小，要不然都是給里程計多一些權重值；相反的，在機器人移動一段距離後，除非 CC2431 定位引擎的穩定度太差，要不然都是給 CC2431 定位引擎多一些權重值。而在推論引擎中，使用 Minimum Inference Engine 來計算模糊輸出值，如(3-6)所示：

$$B'(w) = \max_{l=1}^{k}[A_1^l(D_{Rz}) \wedge A_2^l(D_{Ro}) \wedge B^l(w)] \tag{3-6}$$

其中

k 為規則數

$A_1^l(D_{Rz})$ 為第 l 條模糊規則的 D_{Rz} 歸屬函數值

$A_2^l(D_{Ro})$ 為第 l 條模糊規則的 D_{Ro} 歸屬函數值

$B^l(w)$ 為第 l 條模糊規則的 CC2431 定位引擎權重之模糊集合

$B'(w)$ 為 CC2431 定位引擎的權重輸出模糊集合

最後，將推論引擎所獲得的輸出模糊集合，透過 Center of Area 來解模糊化，如(3-7)所示，其中 W_Z 是解模糊化後的明確值，為 CC2431 定位引擎的權重值。

$$W_Z = \frac{\sum B'(w) \cdot w}{\sum B'(w)} \tag{3-7}$$

透過以上模糊邏輯控制可以得到 CC2431 定位引擎的權重值，經由兩者相加必須為 1 的比例關係，可以簡單地推算出里程計的權重值，然後將各自的定位結果乘上各自的權重值，再進行相加，即為融合後的機器人的世界座標位置，如(3-8)所示：

$$\begin{bmatrix} X_R \\ Y_R \end{bmatrix} = (1 - W_z)\begin{bmatrix} X_{Ro} \\ Y_{Ro} \end{bmatrix} + W_z\begin{bmatrix} X_{Rz} \\ Y_{Rz} \end{bmatrix} \tag{3-8}$$

其中

X_R、Y_R 為融合後的機器人的世界座標位置

X_{Ro}、Y_{Ro} 為里程計定位機器人的世界座標位置

X_{Rz}、Y_{Rz} 為 CC2431 定位引擎定位機器人的世界座標位置

W_Z 為 CC2431 定位引擎的權重值

3.4 結論與討論

透過本論文所提出的融合設計，可以有效地減少機器人自身定位的誤差，對於需要找尋在不同房間或長距離的使用者，機器人不迷失自身的位置，有助於召喚任務的達成。另一方面，因為 CC2431 定位引擎無法提供絕對角度之資訊，所以無法直接對機器人的朝向角 θ 來進行修正，但是逐漸透過絕對座標來修正機器人的自身座標，有可能會間接修正到機器人朝向角 θ，使得機器人朝向角誤差不會逐漸累積變大。

第四章 使用者影像追蹤

本章節將介紹如何透過影像來偵測使用者，並且找到使用者在影像中的位置資訊，然後透過本論文所設計影像追蹤控制器，讓機器人可以追蹤使用者，並且停在使用者面前。

在影像攝影機的選擇上，對於 2D 影像攝影機來說，不使用相同背景的情況下，來偵測使用者是一件相當困難的問題，所以有些研究會透過立體攝影機(Stereo Camera)的深度影像資訊(Depth Image)來偵測使用者[19][20]。但是立體攝影機的價格昂貴，以及需要多個演算法來偵測使用者，對於需要執行多個任務的機器人而言，會相當消耗電腦資源。相較之下，Kinect 具有便宜的價格、能同時獲得彩色和深度影像資訊以及能快速準確地偵測出畫面中使用者之骨架追蹤系統(Skeletal Tracking System)，對於使用者偵測上有很大幫助，所以本論文選用 Kinect 來協助機器人偵測使用者。

4.1 Kinect 介紹與使用

Kinect 是微軟公司所開發的體感裝置，應用於遊戲主機 XBOX360，讓玩家不用手持搖控器就能進行遊戲，實現身體就是遙控器之理念。Kinect 之所以能辦到這點，全歸功於能偵測玩家動作的深度攝影機，如圖 4-1 所示，Kinect 透過中間的 RGB 攝影機來獲得彩色影像資訊，可用於辨識玩家身份。而透過左右兩邊的紅外線發射器與紅外線 CMOS 攝影機所構成的深度攝影機，可獲得深度影像資訊，經過晶片運算後，可以判斷出人體骨架各部位的位置與方向，進而達到辨識玩家的肢體動作。其詳細的規格如表 4-1 所示。

為了讓機器人能使用 Kinect 來獲得影像資訊，採用 OpenNI (Open Natural Interaction)[23]當作溝通管道。OpenNI 是一個跨平台的「自然操作」開放原始碼架構，「自然操作」包含了語音、手勢、身體動作等不需要其他特殊裝置的操作方式，讓程式開發者可以方便分析影像、聲音等相關感應器之資料，並且透過中介軟體(Middleware)的協助來使用相關技術上之應用。所以透過 OpenNI 之協助，

圖 4-1 Kinect[21]

表 4-1 Kinect 規格表[22]

sensor	Color and depth-sensing lenses Voice microphone array Tilt motor for sensor adjustment
Field of view	Horizontal field of view: 57 degrees Vertical field of view: 43 degrees Physical tilt range: ± 27 degrees Depth sensor range: 1.2m~3.5m
Data streams	320×240 16-bit depth at 30FPS 640×480 32-bit color at 30FPS 16-bit audio at 16 kHz
Skeletal Tracking System	Tracks up to 6 people, including 2 active players Tracks 20 joints per active player
Audio System	Echo cancellation system enhances voice input

可以在電腦上讀取到 Kinect 的彩色影像以及深度影像，並且經由中介軟體的幫助，也能使用骨架追蹤系統。

4.2 使用者偵測

透過 CC2431 定位引擎，機器人可以估測出使用者的所在位置，但是估測結果與真實位置一定會有所落差，造成機器人無法到達使用者面前。所以為了彌補這項缺點，在機器人靠近使用者時，透過人形偵測(Body Detection)與人臉偵測(Face Detection)兩種方法來獲取機器人與使用者之間的相對位置，使得機器人可以修正自身方向以準確到達使用者面前。

4.2.1 人形偵測

在人形偵測方面，本論文採用 OpenNI 之中介軟體所提供的骨架追蹤系統之使用者產生器(User Generator)來進行使用者的人形偵測，如圖 4-2 所示。首先由 Kinect 取得一張 320×240 的深度影像後，使用者產生器會針對使用者身體部份與背景物件進行區隔，然後透過圖像識別系統(比如人的身高有多高、人有兩隻手和兩條腿等)來判斷影像畫面中是否有使用者存在。如果影像畫面有使用者存在，則將偵測到為使用者的區域賦予標籤(label)，表示此像素值(pixel)有使用者存在，所以透過自訂顏色來取代此像素值，就能將使用者從深度影像中顯示出來，如圖 4-3 所示。而且使用者產生器可以同時偵測到 6 個使用者，因此對不同標籤賦予不同的顏色來取代該像素值，就能在深度影像中顯示出多個使用者。

此外，當持續被偵測的使用者擺出預先定義的校正用姿勢「Psi」時，如圖 4-4 所示，系統就會進入骨架追蹤系統後半部，進行人體骨架校正分析，然後追蹤人體的 20 個關節點(包含軀幹、四肢以及手指等)，透過機器學習技術(Machine Learning)所建立的龐大圖像資料庫，來組合出使用者可能呈現的姿勢。

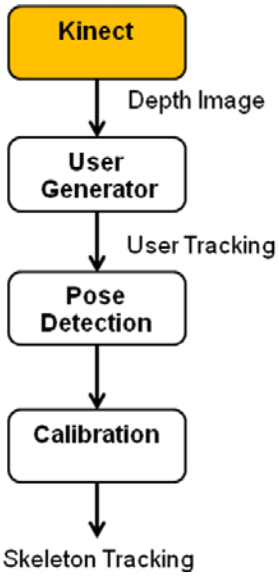

圖 4-2 骨架追蹤系統流程圖

圖 4-3 人形偵測之結果

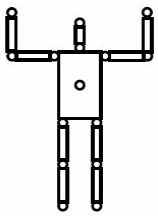

圖 4-4 校正用姿勢「Psi」[24]

對於使用者產生器可能偵測出多個人形的結果，本論文採用最近偵測的人形當作是使用者。為了獲得使用者與機器人之間的相對位置，本論文採用使用者人形的質心座標來當作位置資訊。當使用者產生器偵測到深度畫面中的使用者後，對使用者所佔有的影像區域進行質心運算，如(4-1)所示，如此就能計算出使用者在深度畫面中的質心座標，結果如圖 4-5 所示。

$$\begin{bmatrix} X_{U\_Depth} \\ Y_{U\_Depth} \end{bmatrix} = \begin{bmatrix} \dfrac{\sum_{i=1}^{k} x_i}{k} \\ \dfrac{\sum_{i=1}^{k} y_i}{k} \end{bmatrix} \tag{4-1}$$

其中

X_{U_Depth}、Y_{U_Depth} 為使用者在深度畫面中的質心座標

x_i、y_i 為使用者在深度畫面中所佔有的像素位置

k 為使用者在深度畫面中所佔有的像素數量

圖 4-5 使用者在深度畫面中的質心座標

4.2.2 人臉偵測

在人臉偵測方面，本論文希望能準確找到使用者人臉，如果直接使用膚色偵測來找尋畫面中人臉區域，容易造成錯誤的人臉偵測發生，所以透過 Kinect 取得一張 320×240 的彩色影像後，一開始先採用 OpenCV (Open Source Computer Vision)[25]所提供的 Haar-Like Features 來找出畫面中可能的人臉區域。

Haar-Like Features 是由 Viola[26]以及 Lienhart[27]所提出的人臉偵測方法，因為人的眼睛周圍、鼻子下方或是嘴唇下方等都是容易形成陰影的區域，造成明顯的亮度差異，所以可以將此亮度差異當作特徵來使用。

透過 AdaBoost Learning Algorithm 從上千張的人臉樣本中，挑選出具有代表人臉特徵的矩形圖案，其矩形圖案可以為不同型式，但都是由黑白兩色所構成，這些矩形圖案就稱為 Haar-Like Features，如圖 4-6 所示，有邊緣特徵、線特徵以及中心包圍特徵等類型。Haar-Like Features 是利用黑白區塊間的亮度差異來進行人臉偵測，Haar-Like Features 會移動到影像畫面中每一個區域來計算此區域的亮度差異是否超過閥值(此閥值為訓練階段所決定)，如果超過閥值，則為人臉特徵，否則為不是人臉特徵。

為了在畫面中快速準確地找出人臉區域，透過不停改變 Haar-Like Features 的大小和位置，讓 Haar-Like Features 能在畫面中找出大量的人臉特徵，並且透過 Integral Image 運算的協助，來減少大量運算時間，達到及時偵測效果。如圖

4-7 所示，Integral Image 運算方法是將像素點(x, y)位置的左上角所有灰色方塊範圍內的像素值進行加總，求得所有像素點所包含的像素值，如此一來，在計算黑白區塊間的亮度差異時，只要引用 Integral Image 的結果，就能輕鬆推求出畫面中任何矩陣所包含的像素值(比如要求得 R_{1234}，可以透過 R_D-R_C-R_B+R_A 來獲得)，而不用重新計算。

最後使用 Cascaded Classifier 方法來判斷感興趣的影像區域是否為人臉區域。每一個代表人臉特徵的 Haar-Like Features 都是一種 Classifier，當感興趣的影像區域成功地通過一連串的 Classifier 後，就會被判定為人臉區域，反之，如果在其中一個 Classifier 遭到否決，就不為人臉區域，如圖 4-8 所示。

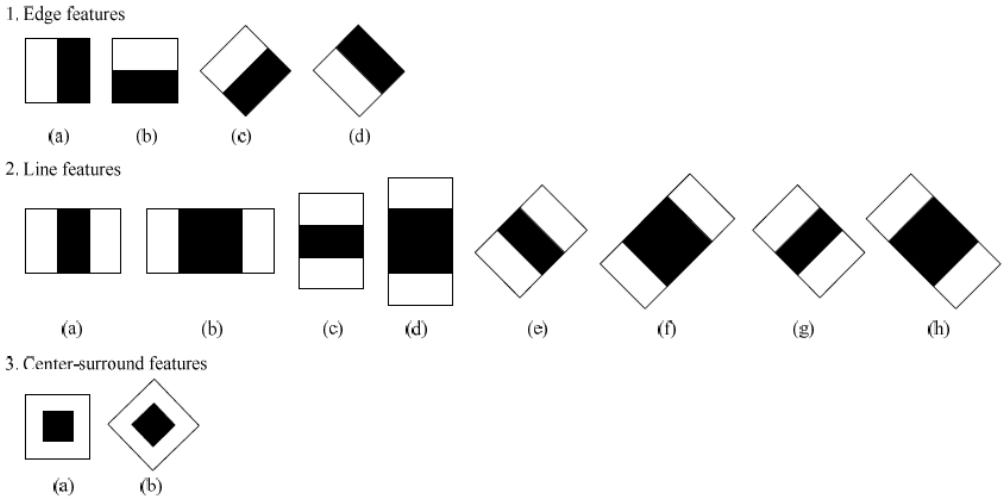

圖 4-6 Haar-Like Features[26]

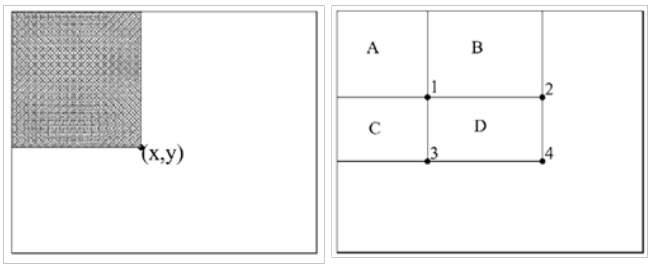

圖 4-7 Integral Image 運算[27]

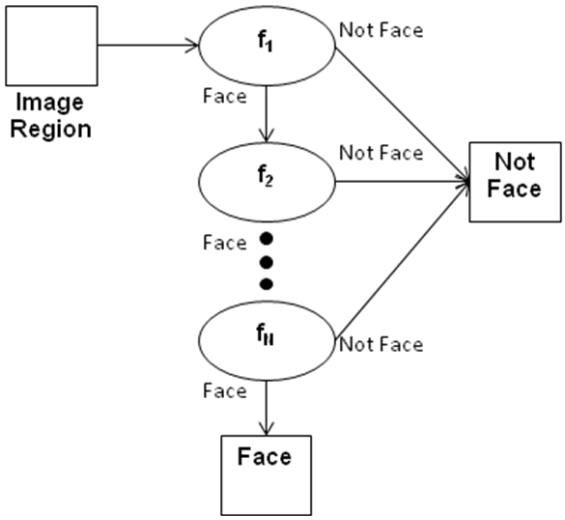

圖 4-8 Cascaded Classifier 應用於人臉偵測

圖 4-9 Haar-Like Features 人臉偵測之結果

由於 Haar-Like Features 人臉偵測可能會找出畫面中為人臉與多個非人臉的影像區域，如圖 4-9 所示。因此，在 Haar-Like Features 人臉偵測所找出的這些區域中，本論文透過人臉膚色密度來協助找出何者為真正的人臉區域。

在進行人臉膚色計算之前，由於在 RGB 空間的影像會因為光源的亮度強弱，使得同一個物體的顏色在影像畫面中呈現出不同的顏色，這樣可能會造成錯誤的人臉膚色計算，所以要先將 RGB 空間轉換到 Normalized Color Coordinates(NCC)[28]空間以減少 R 與 G 對光源變化的靈敏度(在實際應用上，B

對光線的靈敏度較低，因此忽略)，其轉換的公式如(4-2)、(4-3)所示：

$$r = \frac{R}{R+G+B} \tag{4-2}$$

$$g = \frac{G}{R+G+B} \tag{4-3}$$

其中

R、G、B 分別為 RGB 空間的紅、綠、藍像素值

r、g 分別為 NCC 空間的紅、綠像素值

轉換到 NCC 空間後，可以從 r 與 g 的關係中找出人臉膚色範圍[29]，如圖 4-10 所示，可以看出人臉膚色的分布範圍相當集中，所以定義兩個二次方程式來找出人體膚色的分布範圍公式，分別為人臉膚色的上界 g_{up} 和人臉膚色的下界 g_{down}，其公式如(4-4)、(4-5)所示：

$$g_{up} = -1.376r^2 + 1.0743r + 0.1452 \tag{4-4}$$

$$g_{down} = -0.776r^2 + 0.5601r + 0.1766 \tag{4-5}$$

而白色光(r=0.33，g=0.33)也會包含在人臉膚色的分布範圍內，因此我們必須將白色光過濾掉，其白色光如(4-6)所示：

$$\mathrm{W} = (r - 0.33)^2 + (g - 0.33)^2 \tag{4-6}$$

整合以上公式，人臉膚色的範圍如(4-7)所示：

$$Skin = \begin{cases} 1\ ,\ (g < g_{up})\ \mathrm{and}\ (g > g_{down})\ \mathrm{and}\ \mathrm{w} > 0 \\ 0\ ,\ otherwise \end{cases} \tag{4-7}$$

其中

$Skin = 1$ 為膚色，$Skin = 0$ 非膚色

透過人臉膚色公式就可以計算出每一個 Haar-Like Features 人臉偵測所找出的影像區域之人臉膚色範圍大小，並且將人臉膚色範圍大小除以影像區域大小即可得

出人臉膚色密度，如(4-8)所示：

$$\text{Desity}_{\text{Skin}} = \frac{\sum Skin}{width \times height} \tag{4-8}$$

其中

$\text{Desity}_{\text{Skin}}$ 為人臉膚色密度

Skin 為人臉膚色範圍大小

width 、*height* 分別為影像區域的寬和長

最後，計算出每一個 Haar-Like Features 人臉偵測所找出的影像區域之人臉膚色密度後，將人臉膚色密度最大的影像區域定義為真正的人臉區域，其結果如圖 4-11 所示，而完整的人臉偵測之流程圖如圖 4-12 所示。

為了獲得使用者與機器人之間的相對位置，本論文採用使用者人臉的中心座標與人臉寬度來當作位置資訊，如(4-9)所示：

$$\begin{bmatrix} X_{U\_RGB} \\ Y_{U\_RGB} \\ D_{U\_RGB} \end{bmatrix} = \begin{bmatrix} x + \frac{width}{2} \\ y + \frac{height}{2} \\ width \end{bmatrix} \tag{4-9}$$

其中

X_{U_RGB}、Y_{U_RGB} 為彩色畫面中的人臉中心座標

x、*y* 為彩色畫面中的人臉區域左上角座標

width、*height* 為彩色畫面中的人臉區域之寬和長

D_{U_RGB} 為彩色畫面中的人臉寬度

透過人臉的中心座標可以讓機器人修正自身方向來轉向使用者，而透過人臉的寬度數值可以估測機器人與使用者的遠近關係，決定是否要前進到使用者面前或是緊急後退以免撞上使用者，讓機器人能與使用者保持安全上的距離。

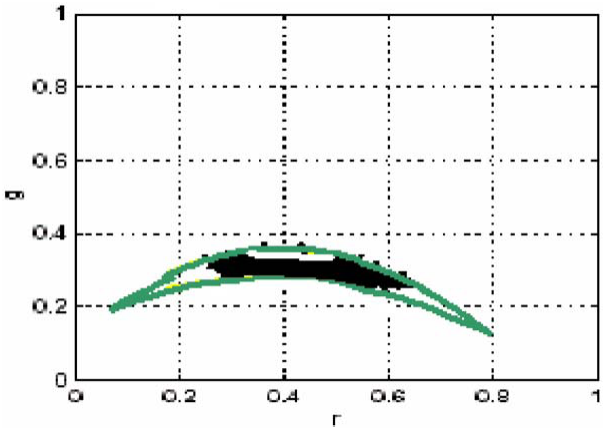

圖 4-10 人臉膚色在 *rg* 座標中的分佈[29]

圖 4-11 人臉偵測之結果

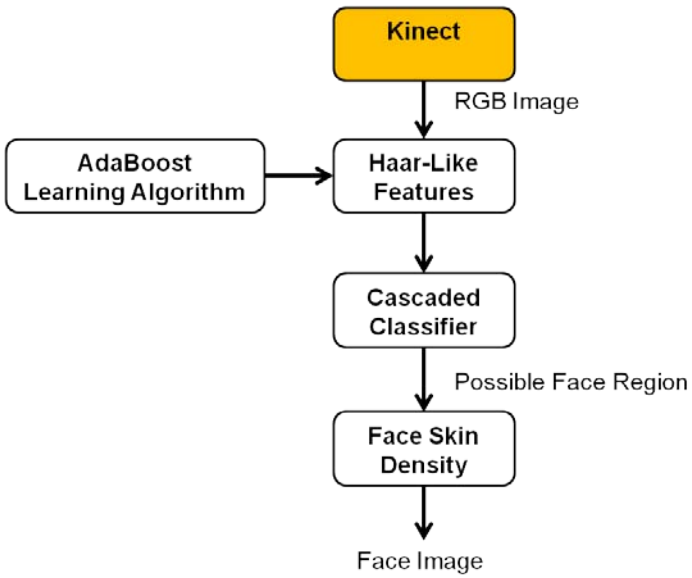

圖 4-12 人臉偵測流程圖

4.3 影像追蹤控制器設計

在影像追蹤控制上要找出明確的運動模型不是一件簡單的事，而模糊 邏輯控制對於這種不確定的資料具有良好處理能力，使得不需透過精確的運動模型亦能進行控制，所以本論文採用模糊邏輯控制來設計影像追蹤控制器，讓機器人可以透過影像資訊來追蹤使用者，其架構圖如圖 4-13 所示。

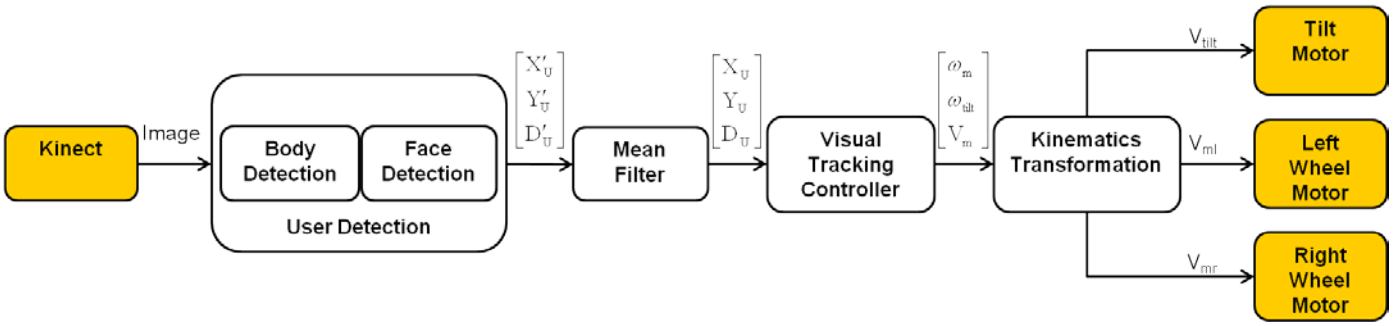

圖 4-13 影像追蹤控制器架構圖

影像追蹤控制器的輸入訊號為使用者偵測(User Detection)之輸出[X_U, Y_U, D_U]，其中 X_U、Y_U 是使用者在影像畫面中的位置座標，讓機器人可以知道使用者與機器人的相對方向，D_U 是使用者在影像畫面中的人臉寬度，讓機器人可以估測使用者與機器人的距離。現今之使用者偵測為每 0.1 秒輸出一次偵測結果，而使用者偵測之輸出為人形偵測與人臉偵測所獲得機器人與使用者之間的相對位置，所以在此要先對不同的影像資料進行整合(Image Date Fusion)，分別為以下三種情況：

情況1. 機器人只偵測到使用者人形：

$$\begin{bmatrix} X'_U \\ Y'_U \\ D'_U \end{bmatrix} = \begin{bmatrix} X_{U\_Depth} \\ Y_{U\_Depth} \\ 45 \end{bmatrix} \tag{4-12}$$

當機器人接近使用者時，會先偵測到使用者人形，但是使用者人形並沒有人臉寬度，所以先設定人臉寬度值為 45，此值在本論文所設計的影像追蹤控制器中，是讓機器人能持續保持前進，直到偵測到使用者人臉後，轉為其他種情況。

情況2.　機器人只偵測到使用者人臉：

$$\begin{bmatrix} X'_U \\ Y'_U \\ D'_U \end{bmatrix} = \begin{bmatrix} X_{U\_RGB} \\ Y_{U\_RGB} \\ D_{U\_RGB} \end{bmatrix} \tag{4-12}$$

情況3.　機器人偵測到使用者人形與人臉：

$$\begin{bmatrix} X'_U \\ Y'_U \\ D'_U \end{bmatrix} = W_{Depth} \begin{bmatrix} X_{U\_Depth} \\ Y_{U\_Depth} \\ 45 \end{bmatrix} + W_{RGB} \begin{bmatrix} X_{U\_RGB} \\ Y_{U\_RGB} \\ D_{U\_RGB} \end{bmatrix} \tag{4-12}$$

$$W_{Depth} = [0.5 \quad 0.5 \quad 0]$$

$$W_{RGB} = [0.5 \quad 0.5 \quad 1]$$

透過權重的分配來整合兩種不同的影像資訊，本論文採用平分方式來決定 X'_U、Y'_U，由於受限於深度偵測的距離，在機器人距離使用者很近的情況下會偵測不到人形，所以 D'_U 完全採用人臉寬度來決定機器人的前進或後退。

為了避免影像或是機器人運動時所造成的雜訊干擾，會先將輸入訊號透過平均濾波器(Mean Filter)來進行濾波，如(4-13)所示，對連續時間內的 *n* 筆輸入訊號進行平均：

$$\begin{bmatrix} X_U \\ Y_U \\ D_U \end{bmatrix} = \frac{\sum_{i=1}^{n} \begin{bmatrix} X'_U \\ Y'_U \\ D'_U \end{bmatrix}_i}{n} \tag{4-13}$$

之後，將濾波後的輸入訊號進行模糊化，在輸入端的歸屬函數設計上，將 X_U 分成五個等級，分別為使用者在畫面很左邊(Very Left)、使用者在畫面左邊(Left)、使用者在畫面中間(Middle)、使用者在畫面右邊(Right)、使用者在畫面很右邊(Very Right)，如圖 4-14 所示。同樣的，將 Y_U 分成五個等級，分別為使用者在畫面很上面(Very Up)、使用者在畫面上面(Up)、使用者在畫面中間(Middle)、使用者在畫面下面(Down)、使用者在畫面很下面(Very Down) ，如圖 4-15 所示。

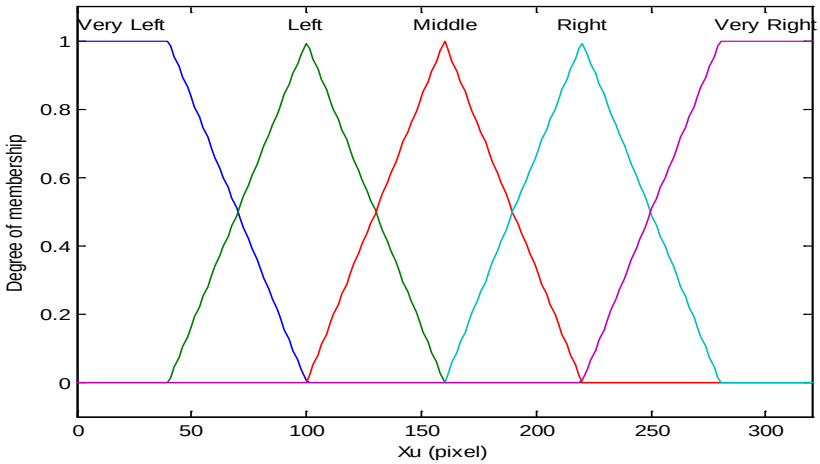

圖 4-14 X_U 的歸屬函數

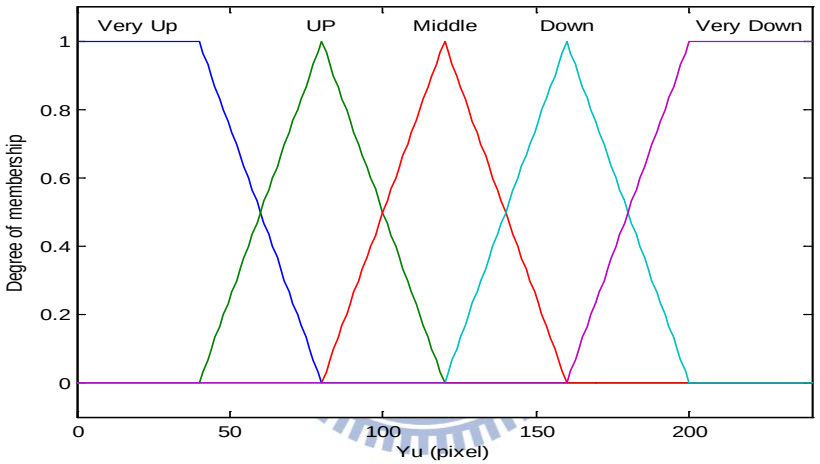

圖 4-15 Y_U 的歸屬函數

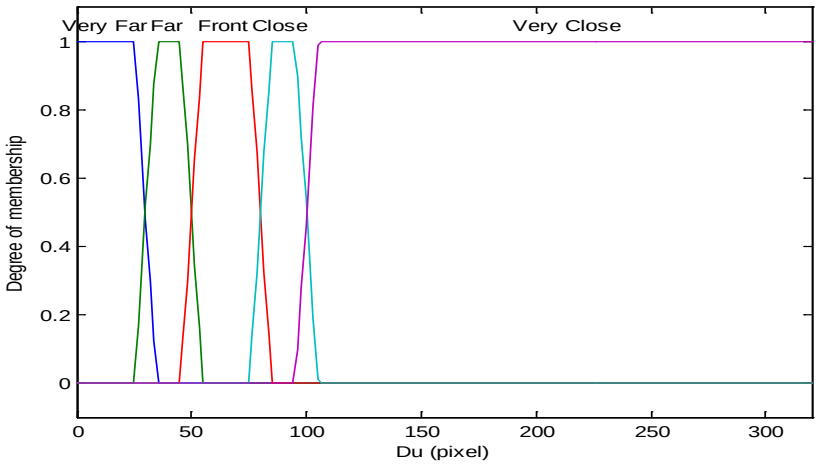

圖 4-16 D_U 的歸屬函數

D_U 的設計是透過實驗來找出使用者距離機器人的關係，本論文希望機器人能與使用者保持約 60 公分之安全距離，而此時人臉寬度為 55 左右，所以依此為標準，將 D_U 分成五個等級，分別為使用者距離機器人很遠(Very Far)、使用者距離機器人遠(Far)、使用者在機器人面前 (Front)、使用者距離機器人近(Close)、使用者距離機器人很近(Very Close) ，如圖 4-16 所示。

影像追蹤控制器的輸出為機器人速度[ω_m、ω_{tilt}、V_m]，其中 ω_m 是機器人本體的角速度，讓機器人可以修正自身的方向來轉向使用者，ω_{tilt} 是機器人頭部上下俯仰之速度，讓機器人可以修正頭部方向來注視著使用者，V_m 機器人本體的線速度，讓機器人可以前進或後退。在輸出端的歸屬函數(Output Membership Function)設計上，將 ω_m 分成五個等級，分別為快速向左轉(Turn Left Fast)、向左轉(Turn Left)、停止(Stop)、向右轉(Turn Right)、快速向右轉(Turn Right Fast)，如圖 4-17 所示，當使用者在畫面的邊緣時，機器人能以設定的最大轉速±0.785 rad/sec 來快速轉向使用者。同樣的，將 ω_{tilt} 分成五個等級，分別為快速向上轉(Turn Up Fast)、向上轉(Turn Up)、停止(Stop)、向下轉(Turn Down)、快速向下轉(Turn Down Fast)，如圖 4-18 所示，設定最大轉速為±0.175 rad/sec 來快速轉向使用者。最後，將 V_m 也分成五個等級，分別為快速後退(Move Backward Fast)、後退(Move Backward)、停止(Stop)、前進(Move Forward)、快速前進(Move Forward Fast)，如圖 4-19 所示，根據不同的人臉寬度，來決定機器人的線速度，最大線速度設定為±30 cm/sec。

透過模糊化，便能將濾波後的輸入訊號[X_U、Y_U、D_U]轉換到相應歸屬函數值 $A_1(X_U)$、$A_2(Y_U)$、$A_3(D_U)$，然後再將歸屬值送到推論引擎，搭配模糊規則便可計算出所相應的機器人速度。而其中模糊規則設計如下：

If the X_U is on the very left side of the screen, then the robot turns right fast.

If the X_U is on the left side of the screen, then the robot turns right.

If the X_U is on the middle side of the screen, then the robot stops.

If the X_U is on the right side of the screen, then the robot turns left.

If the X_U is on the very right side of the screen, then the robot turns left fast.

If the Y_U is on the very up side of the screen, then the robot's head turns up fast.

If the Y_U is on the up side of the screen, then the robot's head turns up.

If the Y_U is on the middle side of the screen, then the robot's head stops.

If the Y_U is on the down side of the screen, then the robot's head turns down.

If the Y_U is on the very down side of the screen, then the robot's head turns down fast.

If the D_U is very close from the robot, then the robot moves backward fast.

If the D_U is close from the robot, then the robot moves backward.

If the D_U is in the front of the robot, then the robot stops.

If the D_U is far from the robot, then the robot moves forward.

If the D_U is very far from the robot, then the robot moves forward fast.

在解糢糊化方面，一樣使用 Minimum Inference Engine 來計算模糊輸出值。然後透過 Center of Area 來解模糊化，計算出相應的機器人速度輸出[ω_m、ω_{tilt}、V_m]。最後將機器人速度轉換到機器人左右兩輪與頭部馬達之速度(Kinematics Transformation)，如(4-14)~(4-16)所示，如此機器人就能根據使用者的影像資訊來追蹤使用者，而完整的使用者影像追蹤流程圖如下圖 4-20 所示。

$$V_{ml} = V_m + \frac{E \cdot \omega_m}{2} \tag{4-14}$$

$$V_{mr} = V_m - \frac{E \cdot \omega_m}{2} \tag{4-15}$$

$$V_{tilt} = \omega_{tilt} \tag{4-16}$$

其中

V_{ml}、V_{mr} 為機器人左輪、右輪之速度，單位 cm/sec

V_{tilt} 為機器人頭部上下俯仰之速度，單位 cm/sec

ω_m、V_m 為影像追蹤控制器所輸出的機器人本體角速度、線速度，單位 cm/sec

ω_{tilt} 為影像追蹤控制器所輸出的機器人頭部上下俯仰之速度，單位 cm/sec

E 為機器人兩輪之間的距離，單位 cm

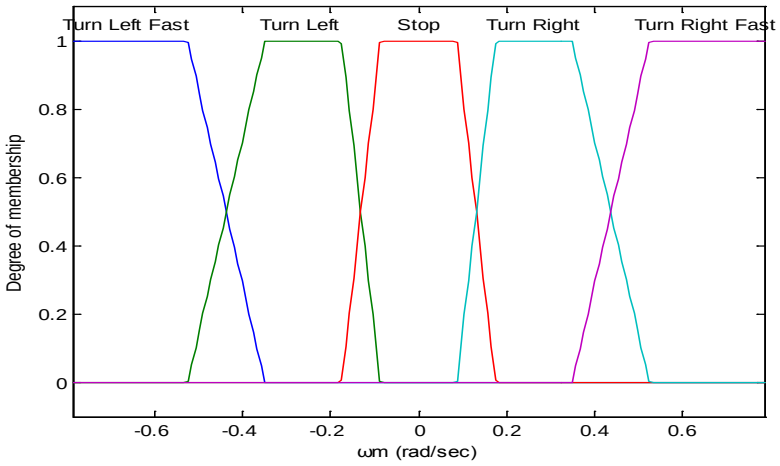

圖 4-17 ω_m 的歸屬函數

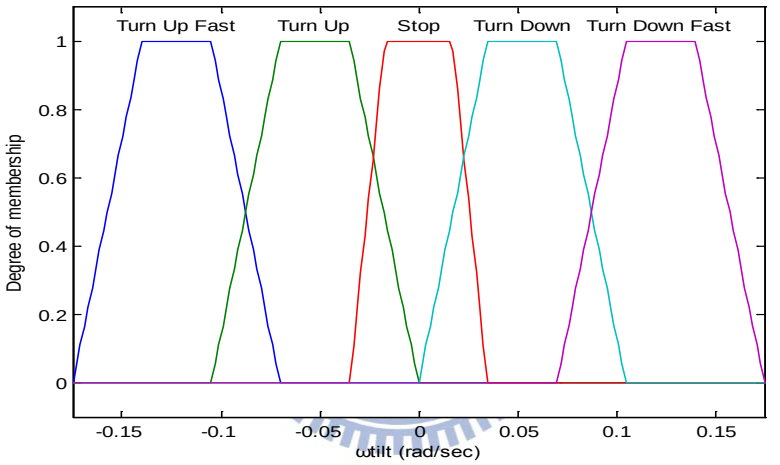

圖 4-18 ω_{tilt} 的歸屬函數

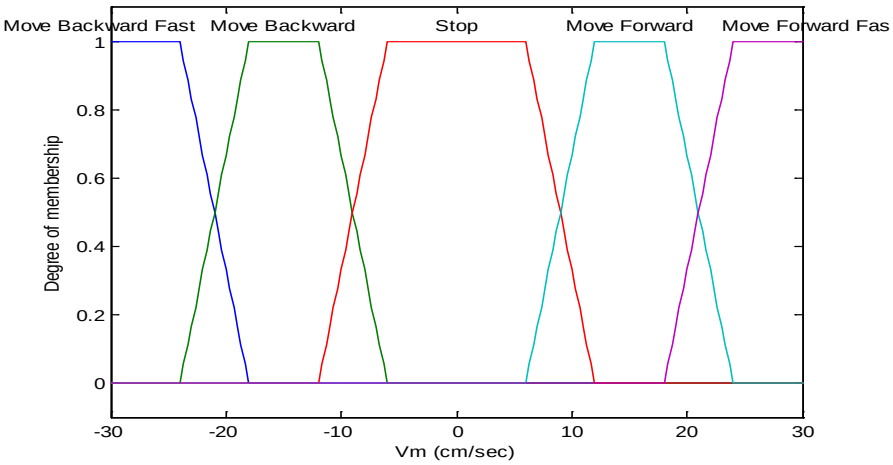

圖 4-19 V_m 的歸屬函數

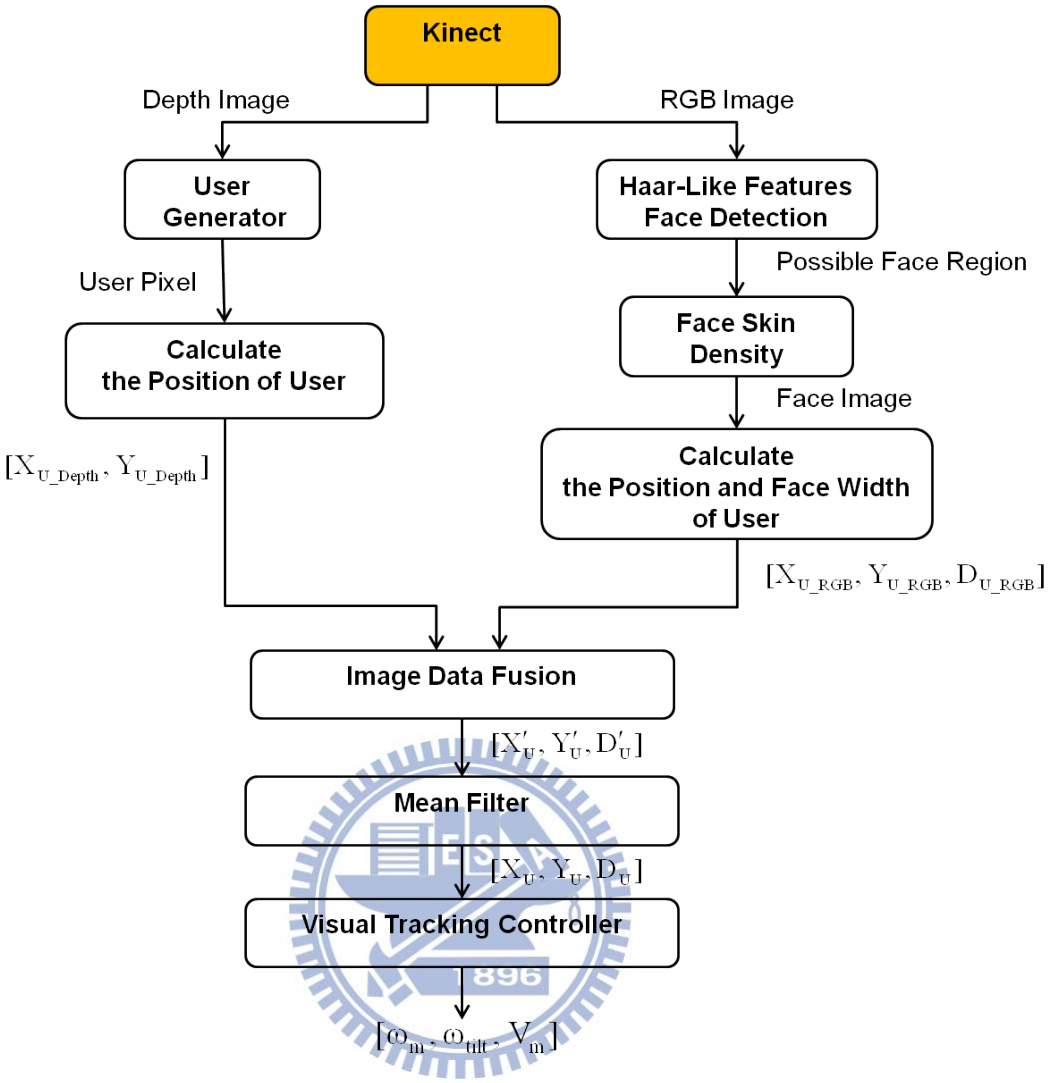

圖 4-20 使用者影像追蹤流程圖

4.4 結論與討論

本論文經由 Kinect 所提供的深度影像，使用了使用者產生器來偵測使用者人形，並且計算使用者在畫面中的人型質心座標。然後經由 Kinect 所提供的彩色影像，使用了 Haar-Like Features 人臉偵測來找出畫面中可能為人臉的影像區域，接著使用人臉膚色密度來找出真正為人臉的影像區域，並且計算使用者人臉的中心座標與人臉寬度。將這些影像資訊進行整合後，透過本論文所設計的影像追蹤控制器，讓機器人可以修正自身方向，朝向使用者前進，並且在適當的距離下，停在使用者面前，完成召喚任務。

第五章 機器人召喚之導航系統

本章節將會介紹機器人的導航控制系統，以及如何整合自我定位系統以及影像追蹤控制器，使得機器人能順利導航到使用者面前。

5.1 系統架構

完整的機器人召喚系統架構圖如圖 5-1 所示，當使用者召喚機器人時，機器人會透過 CC2431 定位引擎來進行使用者定位，估測出使用者的位置[X_{Uz}, Y_{Uz}]，接著使用自主導航系統前往使用者位置。透過雷射的感測資訊，可以讓機器人閃避障礙物而不會撞上，而在機器人移動的途中，自我定位系統也會適時地來修正機器人的自身座標，讓機器人不會迷航。然後在機器人接近使用者時，透過 Kinect 所提供的影像來進行使用者偵測，計算出使用者在畫面的位置資訊[X_U, Y_U, D_U]，接著使用影像追蹤控制器來追蹤使用者，讓機器人可以修正自身方位來朝向使用者前進，適時地停在使用者面前，完成召喚任務。

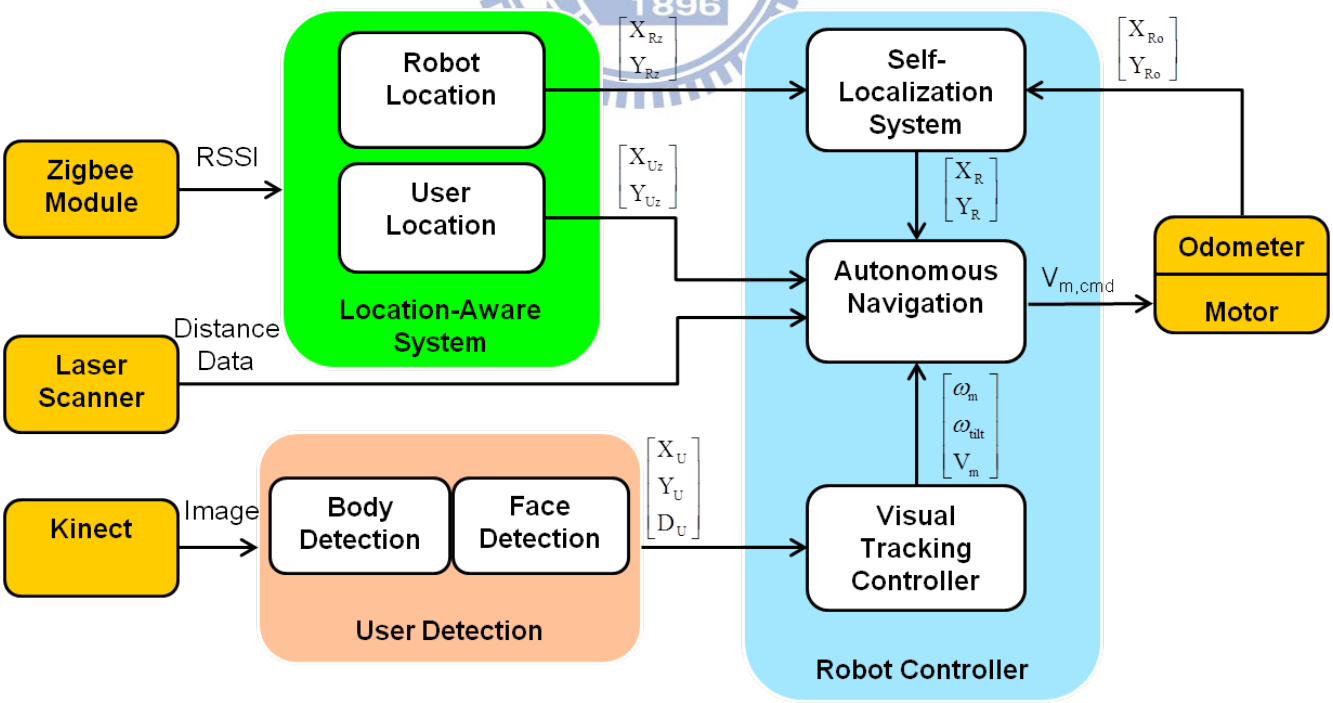

圖 5-1 系統架構圖

5.2 自主導航系統

在自主導航系統(Autonomous Navigation)方面，如圖 5-2 所示，本論文採用[15]所提出一個行為融合的方法設計機器人之導航控制系統當作為基礎，進一步整合自我定位系統與影像追蹤控制器，讓機器人能在長時間、長距離進行導航工作下，仍然不會迷失自己的位置，並且順利找到召喚者。

在導航控制系統方面，透過模糊邏輯控制器來設計三種導航行為，分別為閃避障礙物行為(Obstacle Avoidance)、沿牆行走行為(Wall Following)以及目標追蹤行為(Goal Seeking)，藉由雷射掃描儀(Laser Scanner)來收集周遭環境資訊，作為個別行為模糊控制器的輸入，來決定在當時環境下，三種行為的輸出表現，接着透過 Kohonen 模糊分類網路(FKCN)與建立好的規則表，計算出三種行為的融合權重，接著將三種行為的輸出表現進行融合，最後產生一個改變機器人前進的轉速差，讓機器人擁有隨著環境變化而自行調整能力。

5.2.1 融合設計

在目標追蹤行為上，透過自我定位系統之輸出(X_R, Y_R)與 CC2431 定位引擎定位使用者之位置(X_{Uz}, Y_{Uz})，來計算兩者之間的夾角 θ，當作目標追蹤行為之輸入，如此一來，就能有效降低機器人自身定位誤差之影響，使得機器人可以順利朝向使用者前進。

在召喚任務中，首先機器人會透過導航控制系統來前往使用者的估測位置，當機器人接近使用者時，機器人會透過第四章所提出的影像技術來追蹤使用者，使得機器人可以準確地朝向使用者前進。但是在機器人移動途中，錯誤的影像偵測有可能會導致機器人盲目追蹤，進而導致召喚任務失敗。所以本論文設計一個融合權重 W_F 來決定影像追蹤控制器的參與度。當機器人距離使用者很遠時，就算影像畫面中疑似有使用者存在，導致影像追蹤控制器產生速度輸出，機器人也應該繼續保持導航任務，而不受影響。當機器人距離使用者越近時，影像偵測到真正的使用者之可能性就會增加，所以影像追蹤控制器的參與度也應該增加，使

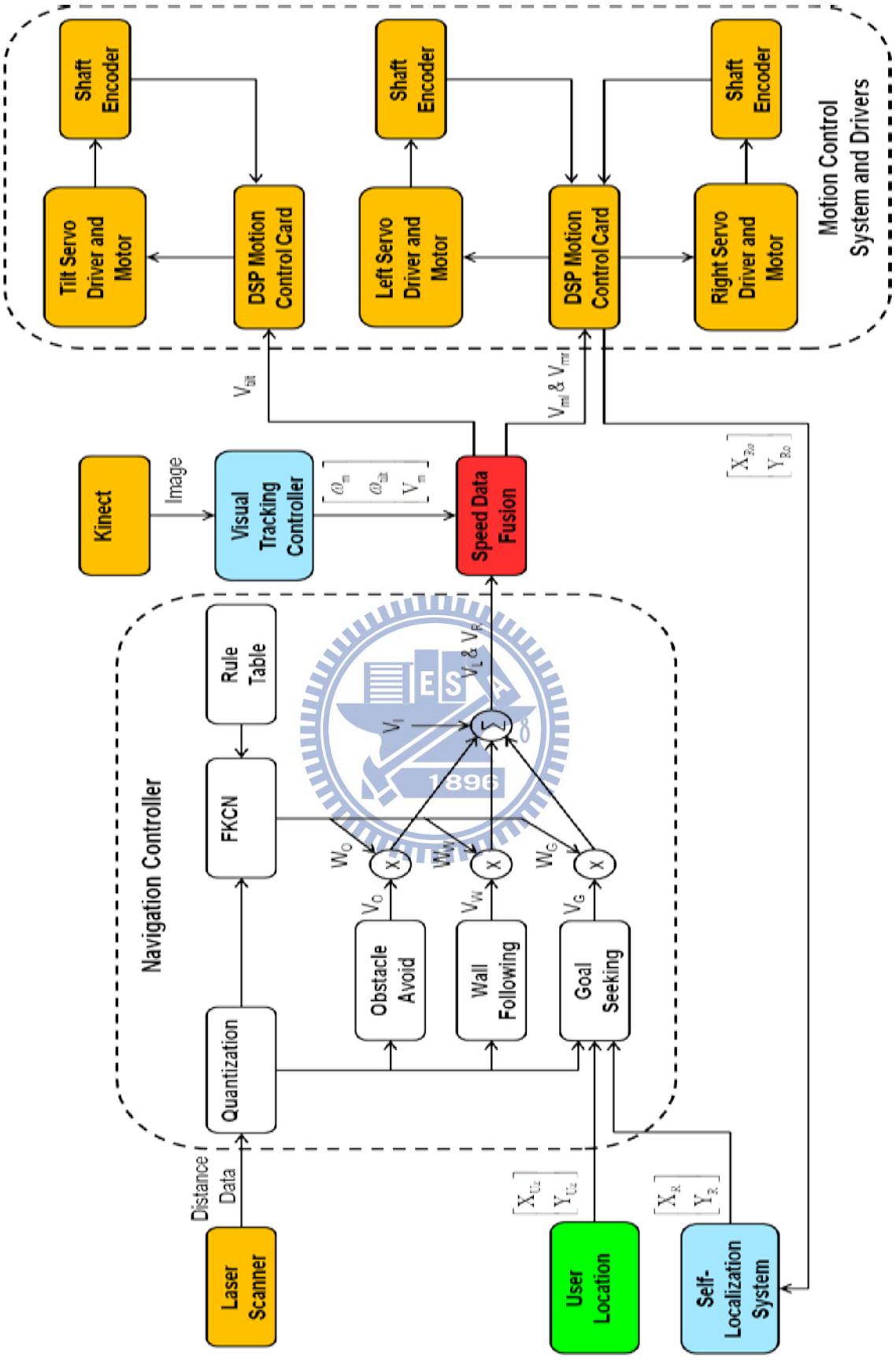

圖 5-2 自主導航系統架構圖

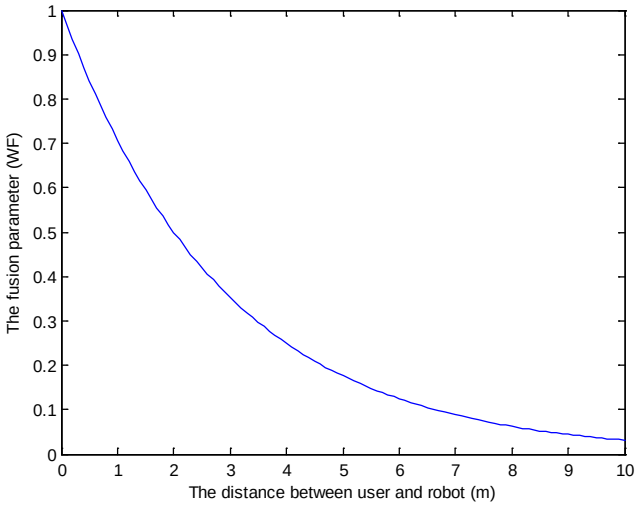

圖 5-3 W_F 和機器人與使用者之間相對距離之關係

得機器人能逐漸地修正自身方向，朝向使用者前進，並且在快接近使用者時，讓機器人完全採用影像追縱控制器之速度輸出，避免導航控制系統閃避使用者之情況發生。

一個根據機器人與使用者之間距離來決定影像追蹤控制器之參與度的融合權重，如圖 5-3 與(5-1)所示，為一個指數倒數。當機器人與使用者之間的相對距離越近時，則 W_F 的值會越大，反之，當機器人與使用者之間的相對距離越遠，則 W_F 的值會越小。

$$W_F = \frac{1}{e^{0.347D_{R\_U}}} \tag{5-1}$$

其中

W_F 是決定影像追蹤控制器之參與度的權重值

D_{R_U} 是機器人與使用者之間的相對距離，單位 m

0.347 表示為機器人距離使用者 2 公尺時，影像追蹤控制器的權重值為 0.5

本論文基於 Kinect 深度感測距離最大至 3.5 公尺，以及避免導航過程中，錯誤的影像偵測發生與機器人電腦資源浪費，所以設定機器人在距離使用者 4 公尺

以內時，才開啟影像偵測。當機器人開始影像偵測後，就會計算出融合權重 W_F 來決定導航控制系統與影像追蹤控制器之間的參與度，如(5-2)~(5-6)所示：

$$V_L = V_l + W_G V_G + W_O V_O + W_W V_W \tag{5-2}$$

$$V_R = V_l - W_G V_G - W_O V_O - W_W V_W \tag{5-3}$$

$$V_{ml} = (1 - W_F) V_L + W_F (V_m + \frac{E \cdot \omega_m}{2}) \tag{5-4}$$

$$V_{mr} = (1 - W_F) V_R + W_F (V_m - \frac{E \cdot \omega_m}{2}) \tag{5-5}$$

$$V_{tilt} = W_F \omega_{tilt} \tag{5-6}$$

其中

V_L、V_R 為導航控制系統所輸出的機器人左輪、右輪之速度，單位 cm/sec

V_l 為機器人之線速度，單位 cm/sec

V_G、V_O、V_W 為目標追蹤行為、閃避障礙物行為、沿牆行走行為之速度，單位 cm/sec

W_G、W_O、W_W 為目標追蹤行為、閃避障礙物行為、沿牆行走行為之權重

W_F 是決定影像追蹤控制器之參與度的權重值

V_{ml}、V_{mr} 為機器人左輪、右輪之速度，單位 cm/sec

V_{tilt} 為機器人頭部上下俯仰之速度，單位 cm/sec

ω_m、V_m 為影像追蹤控制器所輸出的機器人本體角速度、線速度，單位 cm/sec

ω_{tilt} 為影像追蹤控制器所輸出的機器人頭部上下俯仰之速度，單位 cm/sec

E 為機器人兩輪之間的距離，單位 cm

為了避免閃避障礙物行為來影響機器追蹤使用者，當機器人距離使用者 1.5 公尺以內以及有偵測到使用者人臉的情況下，機器人應該完全採用影像追蹤控制器之速度輸出來追蹤使用者，如(4-14)~(4-16)所示。

5.3 結論與討論

本論文將導航控制系統當作基礎，進一步的融合了自我定位系統與影像追蹤控制器，讓機器人在長時間、長距離的移動下，可以有效地降低自身的定位誤差，並且透過導航控制系統與影像追蹤控制器的融合，讓機器人可以逐漸地修正自身的方位，朝向使用者前進。

第六章 實驗結果

本章節介紹如何透過實驗來驗證第三章所提出的自我定位系統，可以有效地減少機器人自身定位的誤差，以及第四章的影像追蹤控制器，能找出影像畫面中的使用者，並且影像追蹤使用者，最後整合各章節所提出的機器人召喚系統，可以順利找到遠距離的使用者，並且移動到使用者面前。

6.1 機器人硬體架構

本論文以實驗室自行研發的看護機器人 RoLA 當作實驗平台，機器人本體如圖 6-1 所示，採用雙獨立輪式之運動機構，搭配平台後方的輔助輪，使得機器人能夠平順穩定的在室內環境中移動。機器人之頭部為 Pan-tilt 機構，可以控制頭部上下左右四個方向之轉動，並且搭載 Kinect 感測器，用以實現影像追蹤使用者之功能。機器人前方裝有 SICK 公司所生產的雷射掃描儀(LMS291-S05)，偵測機器人前方水平 0°~180°是否有障礙物存在，讓機器人可以進行閃避。在機器人內部放置 Zigbee 模組，負責接收或傳送資料到 Zigbee 無線感測網路之協調介面以及定位機器人自身位置之盲點。機器人主控電腦為工業電腦(NOVA-9452-S20)，搭載雙核心處理器與完善的周邊介面，進行機器人的資料處理、程式運算以及控制各項硬體運作。在馬達控制方面，採用實驗室自行研發的 DSP 馬達控制卡以及驅動電路，能同時控制兩組馬達，讓馬達執行電腦端所下達的速度命令，並且透過馬達軸編碼器(Encode)所回傳的馬達轉動資料，讓電腦端可以計算出機器人移動的情形。在電源供應方面，在機器人的底層放置了可重覆充電使用的 12V 碳酸電池，透過 DC12V 轉 DC5V 的電源供應器，可以提供給不同電量需要之硬體所使用。最後，機器人前方嵌有觸控式螢幕，可以讓使用者與機器人進行互動。

機器人的控制系統架構如圖 6-2 所示，以工業電腦為中心，負責 連結所有周邊硬體，下方為馬達控制系統，分別為 DSP 馬達控制卡、驅動電路以及機器人的左右輪馬達與頭部馬達，上方為 Zigbee 協調介面，負責接收或傳送資料到 Zigbee

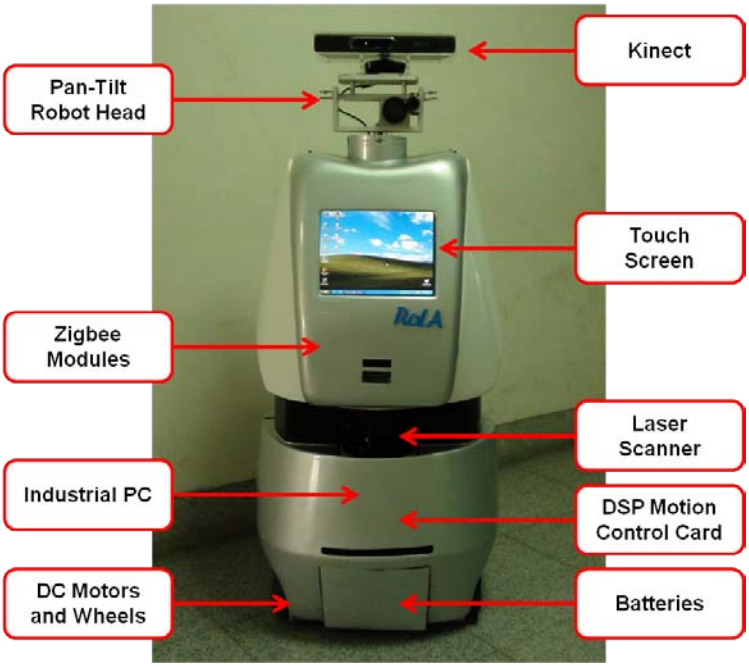

圖 6-1 看護機器人 RoLA

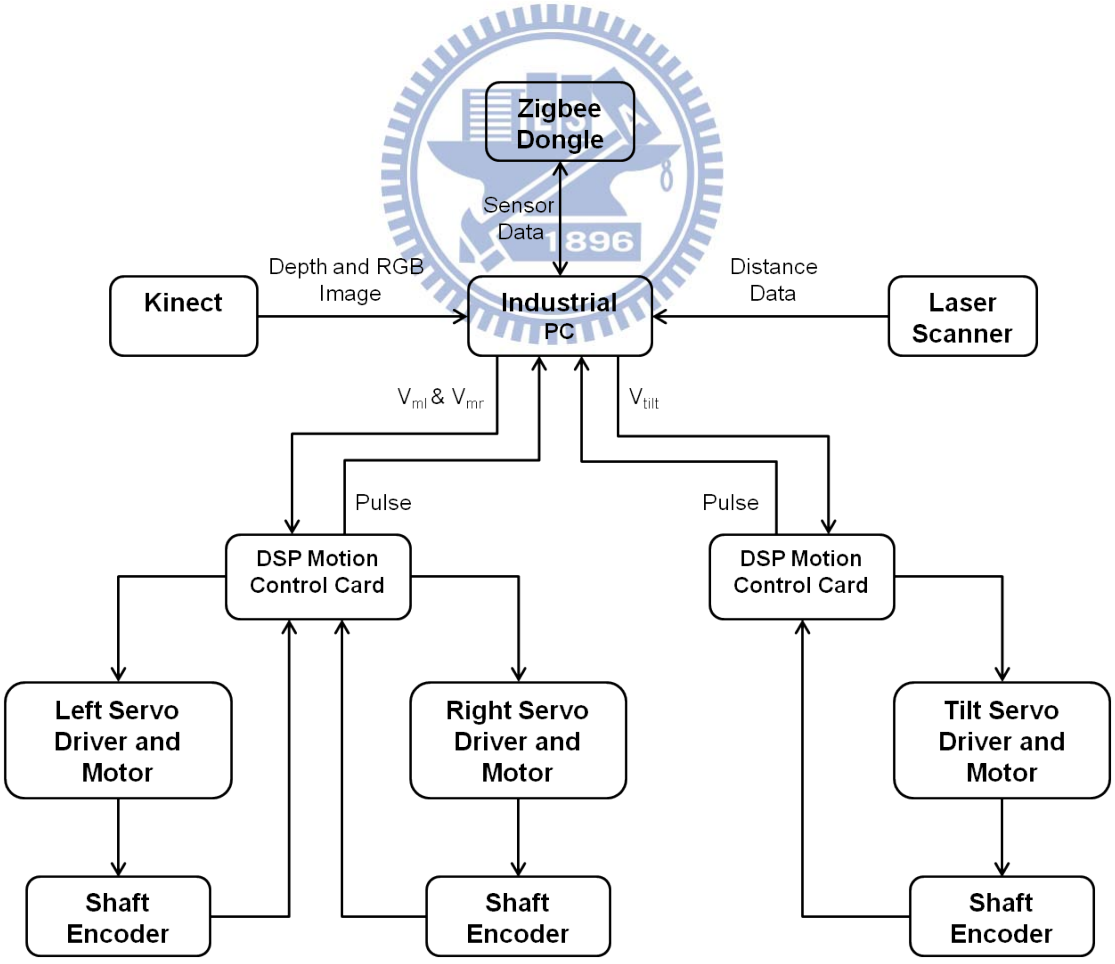

圖 6-2 機器人系統控制架構圖

無線感測網路，左方為 Kinect 感測器，可以獲取深度與彩色影像，右方為雷射掃描儀，偵測前方是否障礙物存在。

6.2 機器人自我定位實驗

本實驗主要目的為驗證自我定位系統在機器人在長距離的移動下之性能，所以設計以下兩個實驗，分別為機器人直線移動 60 公尺與機器人轉彎移動 60 公尺。

6.2.1 機器人直線移動 60 公尺

實驗環境為實驗室外的走廊，如圖 6-3 所示，環境中佈置了 12 個參考點，每個參考點均放置於高度相同的小圓凳上，彼此相隔 6 公尺交叉擺放，來提供機器人身上的盲點所需要的定位資訊。機器人位於起始的世界座標(1, 0)，所設定的初始姿態資訊為(0, 0, 90°)，機器人以線速度 22 cm/sec 往正前方(Y 方向)移動到終點的世界座標(1, 60)，其移動距離總共為 60 公尺，過程中，機器人透過融合自我定位系統之導航控制來修正自身定位，並且同時記錄里程計與自我定位系統兩種定位資料。

實驗結果如圖 6-4 至圖 6-7 所示，綠色虛線為里程計之定位紀錄，藍色圓圈為 CC2431 定位引擎之定位紀錄，紅色實線為自我定位系統之定位紀錄，由圖可以發現，當機器人移動越久，里程計估測 X 軸方向位置的誤差會明顯地越來越大，而使用自我定位系統可以逐漸地將定位誤差修正回來，保持在一個可接受的誤差範圍內。實驗數據如表 6-1 所示，當機器人移動 6 公尺時，自我定位系統開始修正定位誤差，將機器人移動距離 6 公尺與 CC2431 定位引擎之定位穩定度(50, 75)進行資料融合，得到 CC2431 定位引擎的權重值(0, 0)，因為機器人移動距離不長以及定位穩定度不是很高，所以此刻自我定位系統不進行任何修正，然後機器人每移動 2 公尺後，自我定位系統都會進行定位修正，直到機器人移動 30 公尺時，自我定位系統會進行最後一次定位修正，將機器人移動距離 30 公尺與 CC2431 定位引擎之定位穩定度(0, 255)進行資料融合，得到 CC2431 定位引擎的權重值(0.75, 0)，讓里程計定位機器人的世界座標(-76, 3119)與 CC2431 定位引擎

定位機器人的世界座標(150, 3025)進行權重分配，計算修正後的機器人世界座標(93, 3119)，然後重新循環以上流程。最後，當機器人移動完 60 公尺時，本論文記錄下機器人所停下的世界座標位置，然後與里程計以及自我定位系統所估測的機器人世界座標相比，來比較定位估測誤差大小，結果如表 6-2 所示，經由三次相同實驗可發現，使用自我定位系統的平均誤差為 56 公分，比起使用里程計定位的平均誤差為 1284 公分，可以有效減少機器人自身定位的誤差。

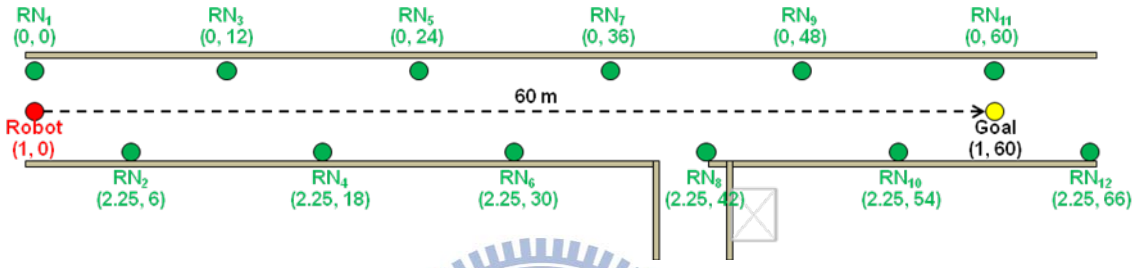

圖 6-3 機器人直線移動 60 公尺之實驗環境

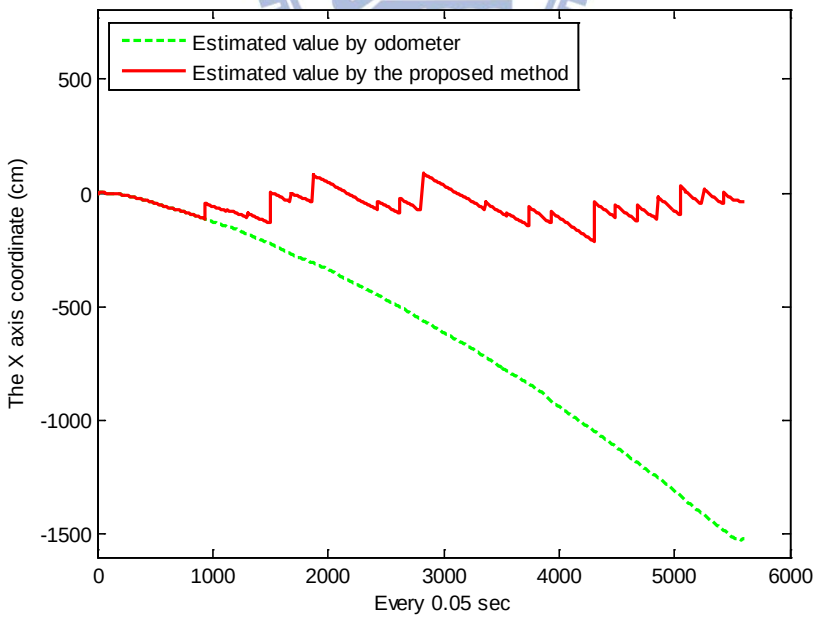

圖 6-4 機器人直線移動 60 公尺之 X 軸估測值

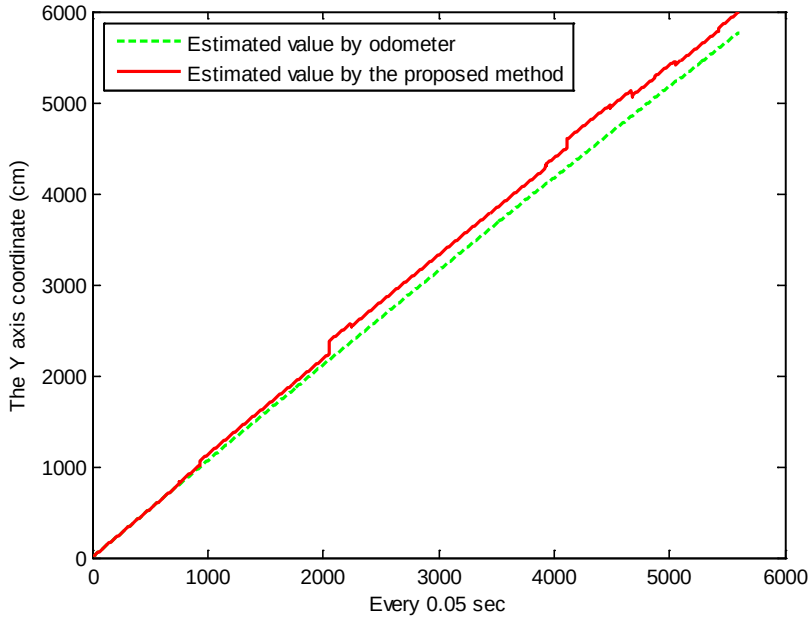

圖 6-5 機器人直線移動 60 公尺之 Y 軸估測值

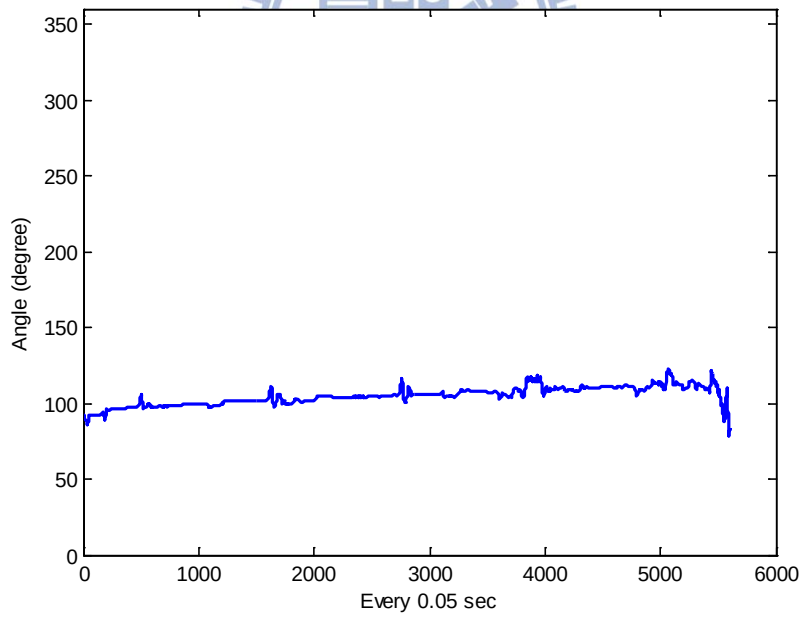

圖 6-6 機器人直線移動 60 公尺之朝向角估測值

表 6-1 機器人直線移動 60 公尺之定位實驗數據

D_{Ro}	$(X_{Ro}, Y_{Ro})_{old}$	(X_{Rz}, Y_{Rz})	D_{Rz}	W_Z	$(X_{Ro}, Y_{Ro})_{new}$
6 m	(-54, 597)	(50, 750)	(50, 75)	(0, 0)	(-54, 597)
8 m	(-82, 795)	(150, 1075)	(175, 75)	(0, 0.075)	(-82, 817)
10 m	(-113, 1013)	(125, 1175)	(25, 0)	(0.283, 0.283)	(-46, 1060)
12 m	(-78, 1257)	(-50, 1700)	(75, 500)	(0.15, 0)	(-74, 1257)
14 m	(-109, 1454)	(-25, 1800)	(50, 500)	(0.216, 0)	(-91, 1454)
16 m	(-131, 1650)	(150, 1650)	(0, 0)	(0.483, 0.483)	(4, 1650)
18 m	(-40, 1846)	(75, 1850)	(75, 25)	(0.35, 0.55)	(0, 1848)
20 m	(-39, 2041)	(150, 2000)	(0, 0)	(0.616, 0.616)	(77, 2036)
22 m	(34, 2237)	(25, 2525)	(50, 50)	(0.483, 0.483)	(29, 2377)
24 m	(-19, 2570)	(-25, 2525)	(25, 25)	(0.75, 0.75)	(-23, 2536)
26 m	(-72, 2731)	(-25, 2750)	(0, 225)	(0.75, 0)	(-36, 2731)
28 m	(-86, 2924)	(0, 3075)	(25, 225)	(0.75, 0)	(-21, 2925)
30 m	(-76, 3119)	(150, 3025)	(0, 255)	(0.75, 0)	(93, 3119)
36 m	(-73, 3695)	(150, 3575)	(0, 100)	(0.15, 0)	(-40, 3695)
38 m	(-100, 3885)	(75, 4200)	(75, 350)	(0.075, 0)	(-88, 3885)
40 m	(-142, 4079)	(150, 4675)	(0, 500)	(0.283, 0)	(-59, 4079)
42 m	(-132, 4265)	(25, 4400)	(0, 25)	(0.35, 0.35)	(-77, 4312)
44 m	(-146, 4499)	(150, 4725)	(150, 25)	(0, 0.416)	(-146, 4593)
46 m	(-210, 4783)	(150, 5175)	(0, 500)	(0.483, 0)	(-36, 4783)
48 m	(-105, 4970)	(50, 4900)	(50, 75)	(0.35, 0.35)	(-51, 4946)
50 m	(-121, 5132)	(50, 5025)	(50, 25)	(0.416, 0.616)	(-49, 5066)
52 m	(-117, 5254)	(25, 5300)	(0, 100)	(0.683, 0.283)	(-20, 5267)
54 m	(-94, 5452)	(75, 5400)	(25, 0)	(0.75, 0.75)	(32, 5413)
56 m	(-47, 5595)	(50, 5600)	(25, 0)	(0.75, 0.75)	(25, 5599)
58 m	(-48, 5784)	(25, 5825)	(0, 25)	(0.75, 0.75)	(6, 5815)

表 6-2 機器人直線移動 60 公尺之誤差值

Times	Localization Error by Odometer (cm)			Localization Error by Proposed Method (cm)		
	X-axis	Y-axis	Position	X-axis	Y-axis	Position
1	1506	258	1528	21	35	41
2	1171	74	1173	33	49	59
3	1147	93	1151	51	44	67
Average	1275	142	1284	35	43	56

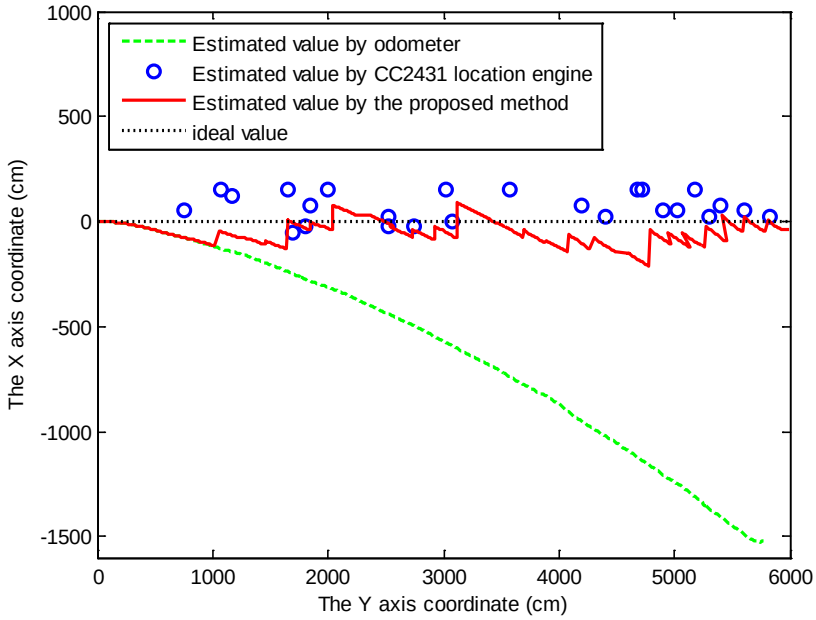

圖 6-7 機器人直線移動 60 公尺之移動軌跡

6.2.2 機器人轉彎移動 60 公尺

實驗環境為實驗室外的走廊，如圖 6-8 所示，環境中佈置了 13 個參考點，每個參考點均放置於高度相同的小圓凳上，彼此相隔 6 公尺交叉擺放，來提供機器人身上的盲點所需要的定位資訊。機器人位於起始的世界座標(1, 4.5)，所設定的初始姿態資訊為(0, 0, 90°)，機器人以線速度 22 cm/sec 往正前方(Y 方向)移動到轉折點的世界座標(1, 34.5)，然後向右轉 90° 後，往正右方(X 方向)移動到終點的世界座標(31, 34.5)，其移動距離總共為 60 公尺，過程中，機器人透過融合自我定位系統之導航控制來修正自身定位，並且同時記錄里程計與自我定位系統兩種定位資料。

實驗結果如圖 6-9 至圖 6-12 所示，綠色虛線為里程計之定位紀錄，藍色圓圈為 CC2431 定位引擎之定位紀錄，紅色實線為自我定位系統之定位紀錄，由圖可以發現，機器人在前 30 公尺中，里程計估測 X 軸方向位置的誤差會隨著機器人移動距離增加而明顯地越來越大，然後在後 30 公尺中，里程計估測 Y 軸方向位置的誤差會隨著機器人移動距離增加而明顯地越來越大，但是透過自我定位系

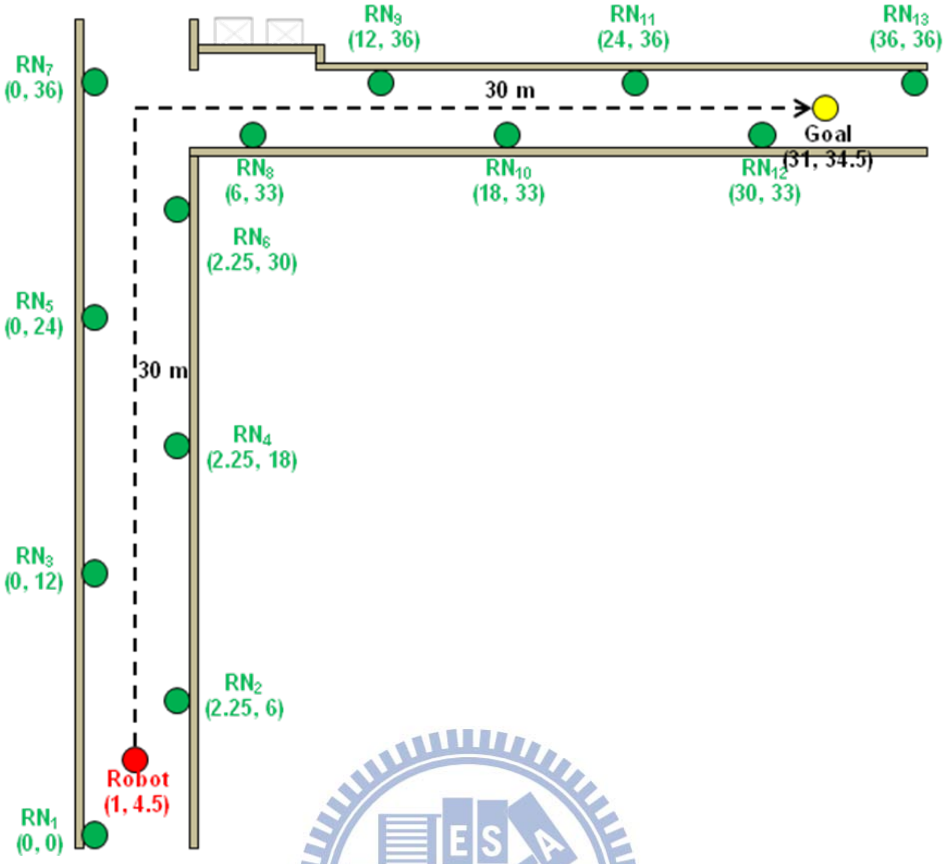

圖 6-8 機器人轉彎移動 60 公尺之實驗環境

統可以逐漸地將定位誤差修正回來，保持在一個可接受的誤差範圍內。實驗數據如表 6-3 所示，當機器人移動 6 公尺時，自我定位系統開始修正定位誤差，將機器人移動距離 6 公尺與 CC2431 定位引擎之定位穩定度(50, 75)進行資料融合，得到 CC2431 定位引擎的權重值(0, 0)，因為機器人移動距離不長以及定位穩定度不是很高，所以此刻自我定位系統不進行任何修正，然後機器人每移動 2 公尺後，自我定位系統都會進行定位修正，直到機器人移動 30 公尺時，自我定位系統會進行最後一次定位修正，將機器人移動距離 30 公尺與 CC2431 定位引擎的之位穩定度(100, 25)進行資料融合，得到 CC2431 定位引擎的權重值(0.35, 0.75)，讓里程計定位機器人的世界座標(11, 2846)與 CC2431 定位引擎定位機器人的世界座標(150, 2850)進行權重分配，計算修正後的機器人世界座標(59, 2849)，然後重新循環以上流程。最後，當機器人移動完 60 公尺時，本論文記錄下機器人所停

下的世界座標位置，然後與里程計以及自我定位系統所估測的機器人世界座標相比，來比較定位估測誤差大小，結果如表 6-4 所示，經由三次相同實驗可發現，使用自我定位系統的平均誤差為 68 公分，比起使用里程計定位的平均誤差為 949 公分，可以有效減少機器人自身定位的誤差。

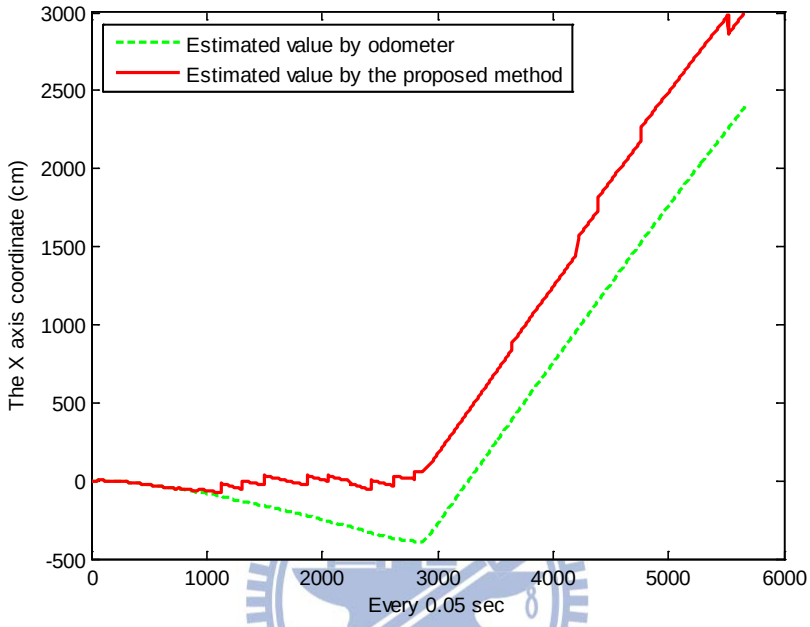

圖 6-9 機器人轉彎移動 60 公尺之 X 軸估測值

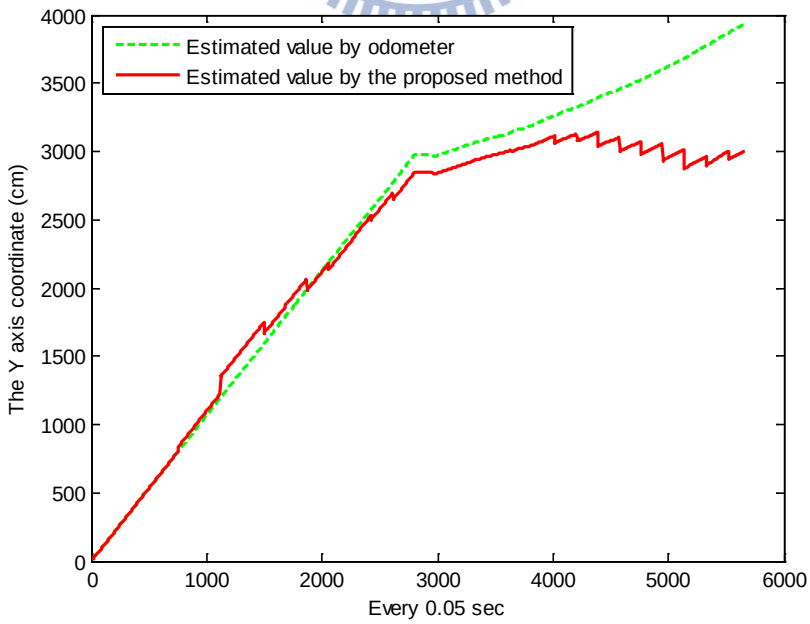

圖 6-10 機器人轉彎移動 60 公尺之 Y 軸估測值

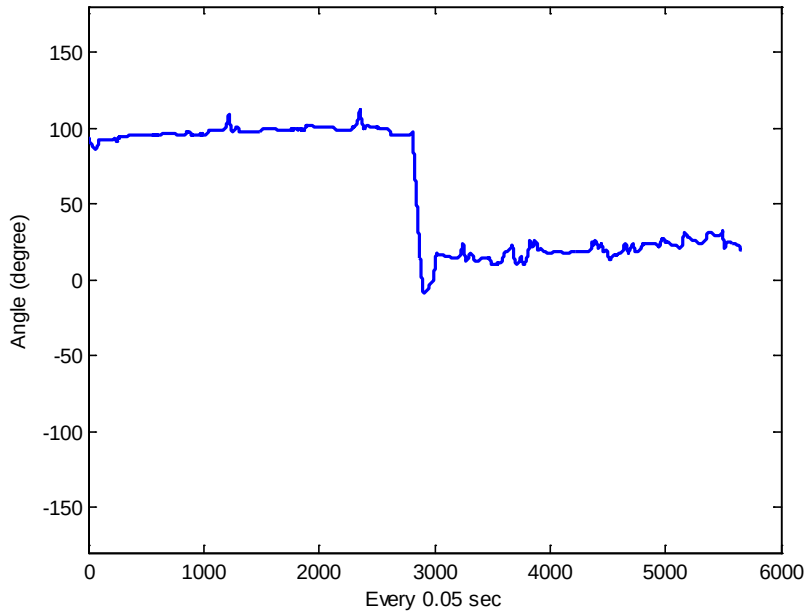

圖 6-11 機器人轉彎移動 60 公尺之朝向角估測值

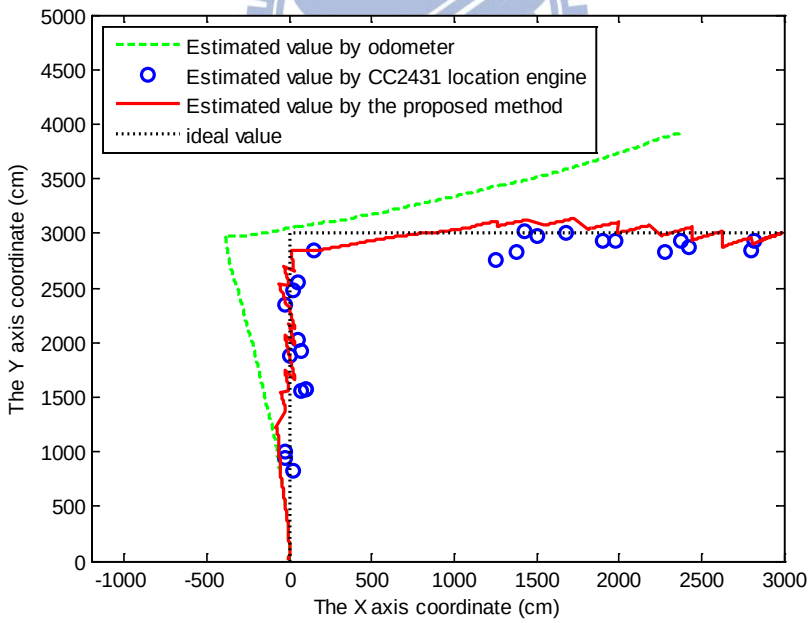

圖 6-12 機器人轉彎移動 60 公尺之移動軌

表 6-3 機器人轉彎移動 60 公尺之定位實驗數據

D_{Ro}	$(X_{Ro}, Y_{Ro})_{old}$	(X_{Rz}, Y_{Rz})	D_{Rz}	W_Z	$(X_{Ro}, Y_{Ro})_{new}$
6 m	(-32, 599)	(25, 825)	(50, 75)	(0, 0)	(-32, 599)
8 m	(-53, 798)	(-25, 950)	(25, 0)	(0.217, 0.217)	(-47, 831)
10 m	(-67, 1030)	(-25, 1000)	(0, 100)	(0.283, 0)	(-55, 1030)
12 m	(-79, 1228)	(100, 1575)	(25, 25)	(0.35, 0.35)	(-16, 1350)
14 m	(-52, 1547)	(75, 1550)	(25, 25)	(0.416, 0.416)	(0, 1548)
16 m	(-26, 1747)	(100, 1575)	(0, 0)	(0.483, 0.483)	(34, 1664)
18 m	(3, 1861)	(0, 1875)	(50, 0)	(0.35, 0.55)	(2, 1868)
20 m	(-28, 2066)	(75, 1925)	(25, 0)	(0.616, 0.616)	(35, 1979)
22 m	(-2, 2176)	(50, 2025)	(25, 100)	(0.683, 0.283)	(33, 2337)
24 m	(0, 2331)	(-25, 2350)	(25, 125)	(0.75, 0.35)	(-18, 2337)
26 m	(-59, 2533)	(25, 2475)	(25, 50)	(0.75, 0.55)	(3, 2501)
28 m	(-30, 2698)	(50, 2550)	(25, 100)	(0.75, 0.35)	(29, 2646)
30 m	(11, 2846)	(150, 2850)	(100, 25)	(0.35, 0.75)	(59, 2849)
36 m	(641, 2966)	(1250, 2750)	(125, 150)	(0, 0)	(641, 2966)
38 m	(836, 3011)	(1375, 2825)	(50,75)	(0.075, 0.075)	(877, 2997)
40 m	(1070, 3049)	(1425, 3025)	(200, 25)	(0, 0.283)	(1070, 3042)
42 m	(1260, 3112)	(1500, 2975)	(150, 25)	(0, 0.35)	(1260, 3064)
44 m	(1449, 3125)	(1675, 3000)	(0, 25)	(0.416, 0.416)	(1543, 3073)
46 m	(1732, 3140)	(1900, 2925)	(0, 0)	(0.483, 0.483)	(1813, 3036)
48 m	(2003, 3100)	(1975, 2925)	(50, 0)	(0.35, 0.55)	(1993, 3003)
50 m	(2182, 3071)	(2375, 2925)	(50, 25)	(0.416, 0.616)	(2262, 2981)
52 m	(2447, 3058)	(2425, 2875)	(100, 25)	(0.283, 0.683)	(2441, 2933)
54 m	(2625, 3013)	(2275, 2825)	(200, 0)	(0, 0.75)	(2625, 2872)
56 m	(2807, 2961)	(2800, 2850)	(75, 50)	(0.55, 0.55)	(2803, 2900)
58 m	(2984, 2997)	(2825, 2925)	(0, 0)	(0.75, 0.75)	(2864, 2943)

表 6-4 機器人轉彎移動 60 公尺之誤差值

Times	Localization Error by Odometer (cm)			Localization Error by Proposed Method (cm)		
	X-axis	Y-axis	Position	X-axis	Y-axis	Position
1	587	985	1147	20	60	63
2	349	832	902	55	23	60
3	276	749	798	78	22	81
Average	404	855	949	51	35	68

6.3 使用者影像追蹤實驗

本實驗主要目的在於驗證影像偵測與影像追蹤控制器之性能，實驗環境為實驗室外的走廊，機器人初始姿態資訊為(0, 0, 90°)，透過 Kinect 感測器所提供的深度與彩色影像來偵測使用者的人形與人臉，並且計算與整合其使用者影像資訊，然後使用影像追蹤控制器來追蹤使用者。過程如圖 6-13 至圖 6-15 所示，使用者首先繞著機器人左邊移動，此時機器人偵測到影像畫面中的使用者偏向左邊後，機器人旋轉追上使用者，如圖 6-13 與圖 6-14 中 A 至 B，以及圖 6-15(a)至圖 6-15(f)所示，在使用者繞了半圈後，開始往機器人前進，此時機器人偵測到影像畫面中的使用者人臉寬度變大後，機器人後退與使用者保持適當距離，如圖 6-13 與圖 6-14 中 B 至 C，以及圖 6-15(f)至圖 6-15(h)所示，最後使用者再繞著機器人左邊移動一段距離後，往後移動，此時機器人偵測到影像畫面中的使用者偏向走邊以及使用者人臉寬度變小後，機器人趕緊修正自身方位，朝向使用者前進，並且停在使用者面前，如圖 6-13 與圖 6-14 中 C 至 D，以及圖 6-15(h)至圖 6-15(l)所示。實驗結果顯示機器人能偵測到影像畫面中的使用者，並且透過影像追縱控制器，讓機器人修正自身方向來追隨使用者。

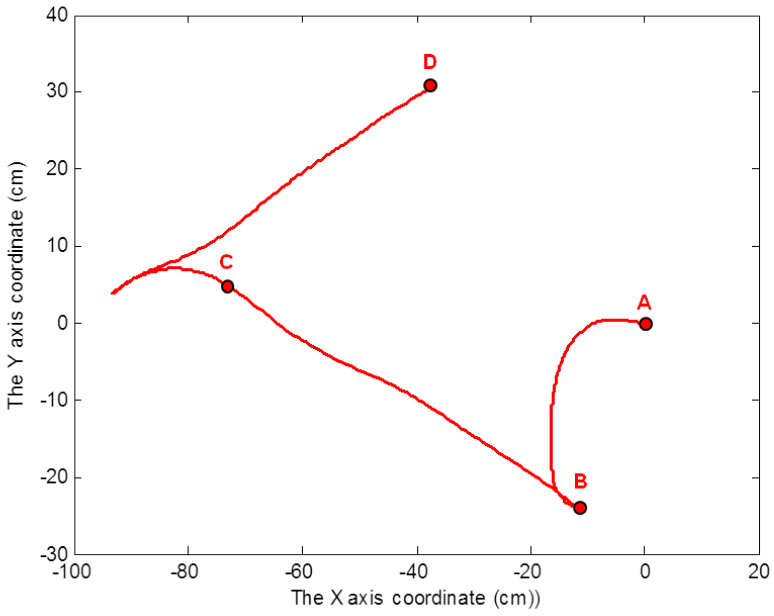

圖 6-13 機器人影像追蹤使用者之移動軌跡

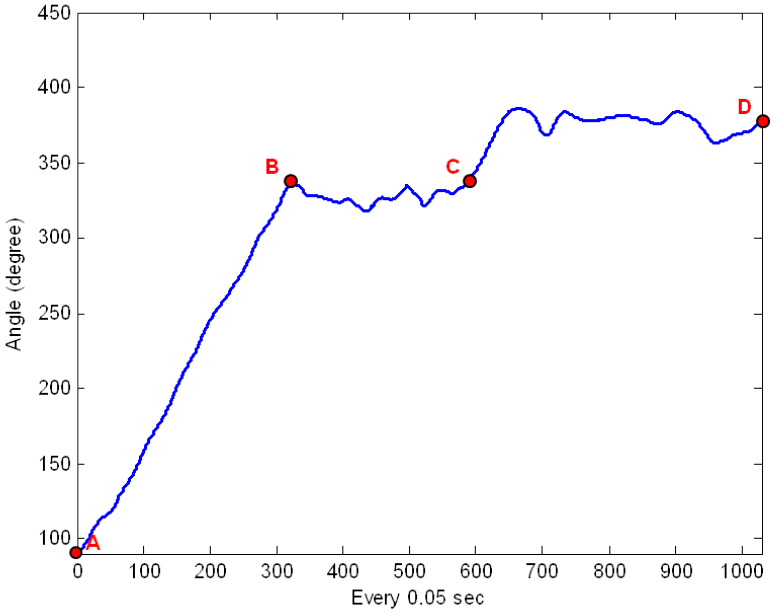

圖 6-14 機器人影像追蹤使用者之朝向角估測值

6.4 短距離召喚機器人實驗

本實驗主要目的在於驗證機器人召喚系統可以讓機器人在具有障礙物之環境下，順利找到短距離的使用者，並且移動到使用者面前。短距離實驗環境為實驗室與實驗室外的走廊，如圖 6-16 所示，每個房間均佈置了 4 個參考點，彼此交叉擺放並且放置於高度相同的小圓凳上，來提供使用者與機器人身上的盲點所需要的定位資訊。使用者位於世界座標(1, 7)，機器人位於世界座標(4.5, 3)，所設定的初始姿態資訊為(0, 0, 90°)以及線速度 13 cm/sec。一開始使用者按下 Zigbee 模組上的按鍵來發射召喚訊號，如圖 6-18(a)至圖 6-18(b)所示，當機器人收到召喚訊號後，立即透過 CC2431 定位引擎來估測出使用者位置，其結果為(1.25, 6.42)，接著將使用者位置當作自主導航系統之輸入，讓機器人開始往使用者方向前進。使用者在機器人左下方，機器人應該往左下方移動，但是一開始機器人左邊有障礙物存在，導致閃避障礙物行為輸出較大，使得機器人閃避左邊障礙物而朝向門口移動，然後穿越門，並且閃避走廊上的障礙物，往使用者方向前進，如圖 6-18(c)至圖 6-18(i)，當機器人距離使用者 4 公尺時，影像追蹤使用者之功能

圖 6-15 機器人影像追蹤使用者之實驗過程

開始運作，計算使用者在影像畫面之位置資訊，接著透過影像追蹤控制器，讓機器人能修正自身方向，移動到使用者面前，如圖 6-18(j)至圖 6-18(l)所示，而完整的機器人移動軌跡如圖 6-17 所示，機器人總共移動距離約 10 公尺。最後，當機器人停在使用者面前後，本論文記錄使用者與機器人之間的距離以及機器人面對使用者正面的角度差，結果如表 6-5 所示，經由三次相同實驗之結果，機器人與使用者之間的平均距離為 69 公分，機器人面對使用者正面的平均角度差為 12°，代表機器人能順利找到使用者，並且停在使用者面前。

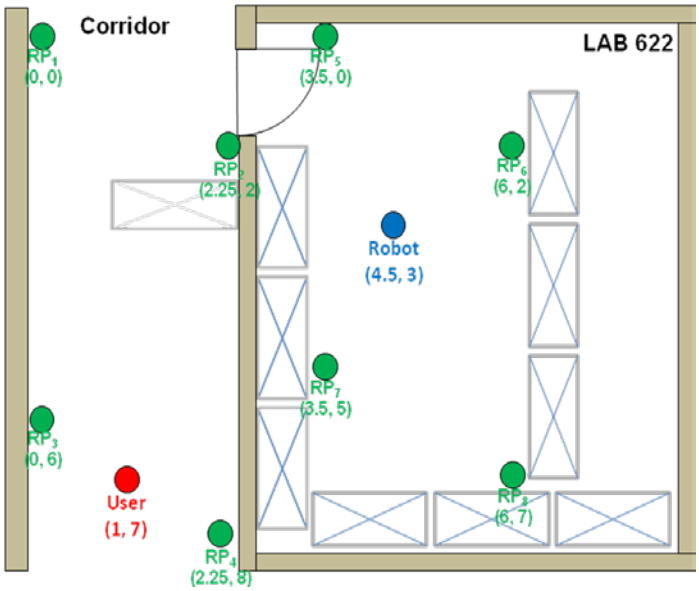

圖 6-16 短距離召喚機器人之實驗環境

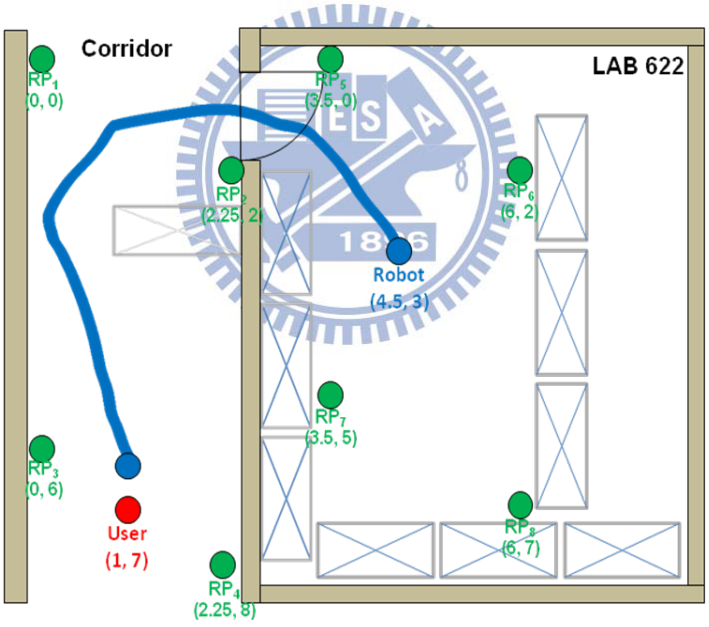

圖 6-17 短距離召喚機器人之移動軌跡

6.5 長距離召喚機器人實驗

本實驗主要目的在於驗證機器人召喚系統可以讓機器人在具有障礙物之環境下，順利找到長距離的使用者，並且移動到使用者面前。長距離實驗環境為實驗室到電梯口的走廊，如圖 6-19 所示，實驗室佈置了 4 個參考點，在走廊上佈置了 8 個參考點，彼此交叉擺放並且放置於高度相同的小圓凳上，來提供使用者

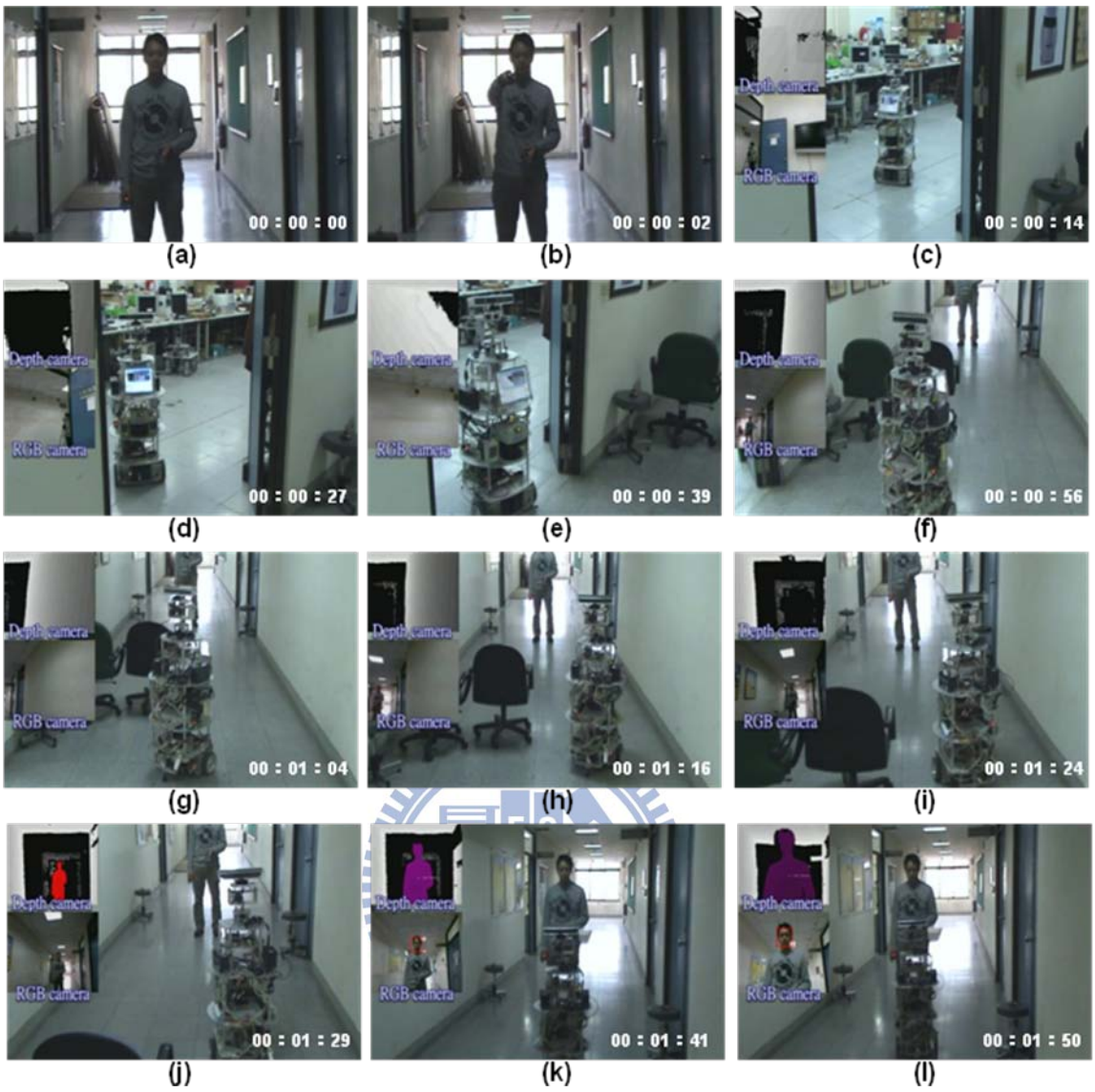

圖 6-18 短距離召喚機器人之實驗過程

表 6-5 短距離召喚機器人之實驗結果

Times	The distance between the robot and user	The angle between the robot and user
1	70 cm	10°
2	65 cm	12°
3	72 cm	15°
Average	69 cm	12°

與機器人身上的盲點所需要的定位資訊。使用者位於世界座標(6.5, 35)，機器人位於世界座標(4.5, 5)，所設定的初始姿態資訊為(0, 0, 90°)以及線速度 22 cm/sec。一開始使用者按下 Zigbee 模組上的按鍵來發射召喚訊號，如圖 6-21(a)至圖 6-21(b)所示，當機器人收到召喚訊號後，立即透過 CC2431 定位引擎來估測出使用者位置，其結果為(5.25, 35.42)，接著將使用者位置當作自主導航系統之輸入，讓機器人開始往使用者方向前進。透過導航行為，機器人順利穿越門，閃避環境中的障礙物，往使用者方向前進，如圖 6-21(c)至圖 6-21(o)，機器人移動過程中，自我定位系統會持續地修正機器人自身座標，以改善里程計所累積的定位誤差，自我定位系統之實驗數據如表 6-6 所示，當機器人移動 6 公尺時，自我定位系統開始修正定位誤差，將機器人移動距離 6 公尺與 CC2431 定位引擎之定位穩定度(150, 24)進行資料融合，得到 CC2431 定位引擎的權重值(0, 0.15)，讓里程計定位機器人的世界座標(-346, 490)與 CC2431 定位引擎定位機器人的世界座標(-200, 975)進行權重分配，計算修正後的機器人世界座標(-346, 562)，然後機器人每移動 2 公尺後，自我定位系統都會進行定位修正，直到機器人移動 30 公尺時，自我定位系統進行最後一次定位修正，將機器人移動距離 30 公尺與 CC2431 定位引擎的之位穩定度(125, 125)進行資料融合，得到 CC2431 定位引擎的權重值(0.35, 0.35)，讓里程計定位機器人的世界座標(-244, 2929)與 CC2431 定位引擎定位機器人的世界座標(-255, 2950)進行權重分配，計算修正後的機器人世界座標(-233, 2936)，然後重新循環以上流程。當機器人距離使用者 4 公尺時，影像追蹤使用者之功能開始運作，計算使用者在影像畫面之位置資訊，接著透過影像追蹤控制器，讓機器人能修正自身方向，移動到使用者面前，如圖 6-21(p)至圖 6-21(r)所示，而完整的機器人移動軌跡如圖 6-20 所示，機器人總共移動距離約 35 公尺。最後，當機器人停在使用者面前後，本論文記錄使用者與機器人之間的距離以及機器人面對使用者正面的角度差，結果如表 6-7 所示，經由三次相同實驗之結果，機器人與使用者之間的平均距離為 68 公分，機器人面對使用者正面的平均角度差為 30°，代表機器人能順利找到使用者，並且停在使用者面前。

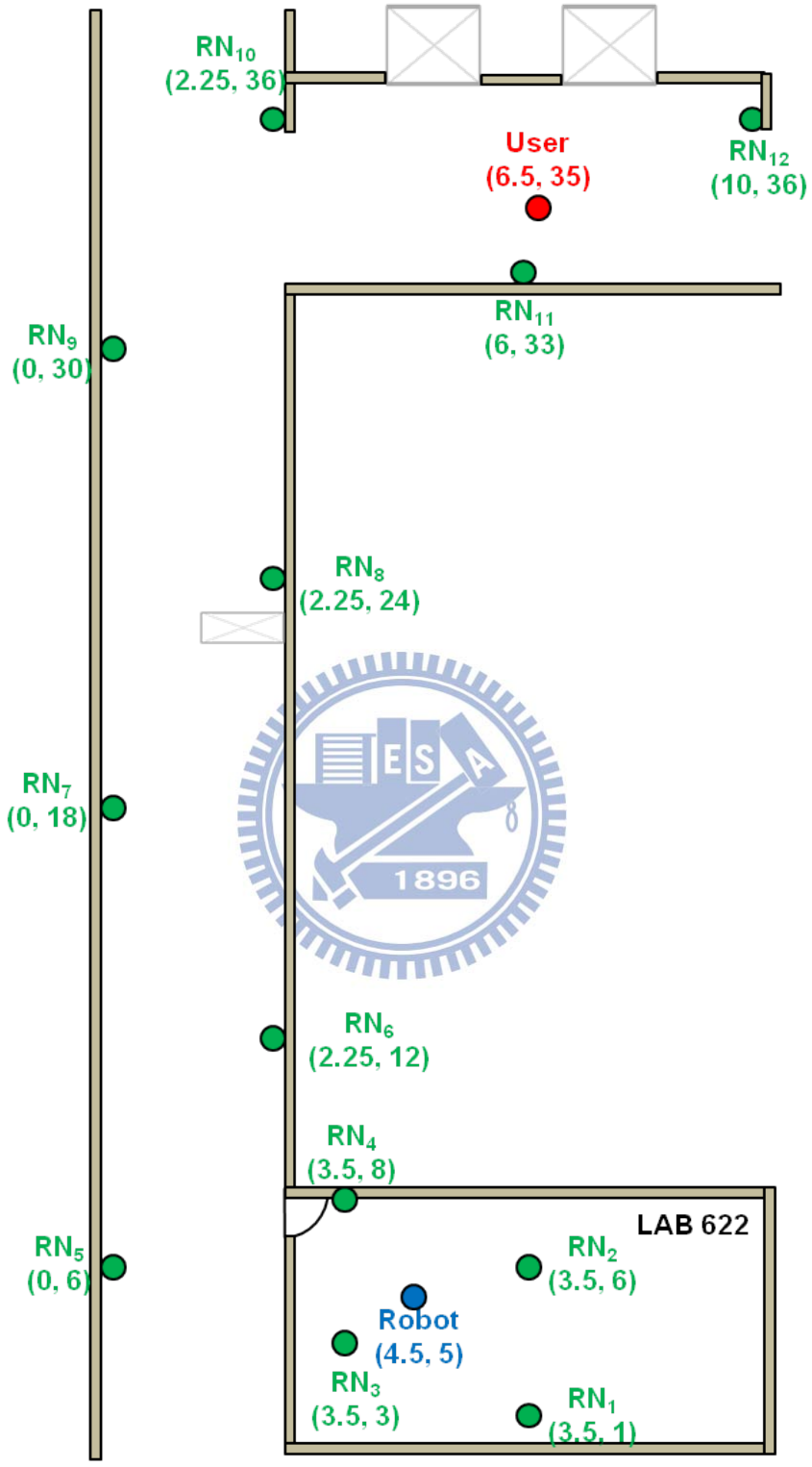

圖 6-19 長距離召喚機器人之實驗環境

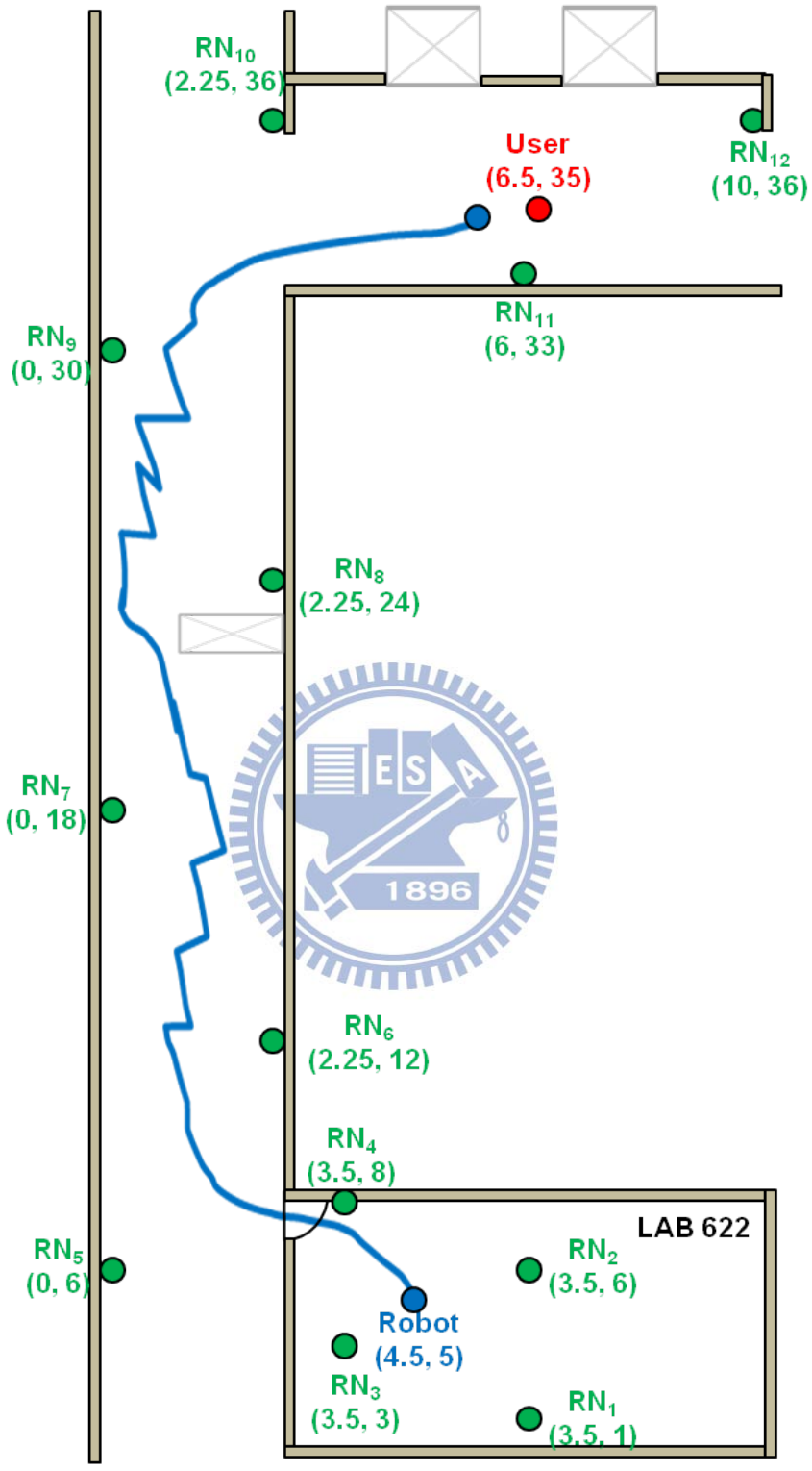

圖 6-20 長距離召喚機器人之移動軌跡

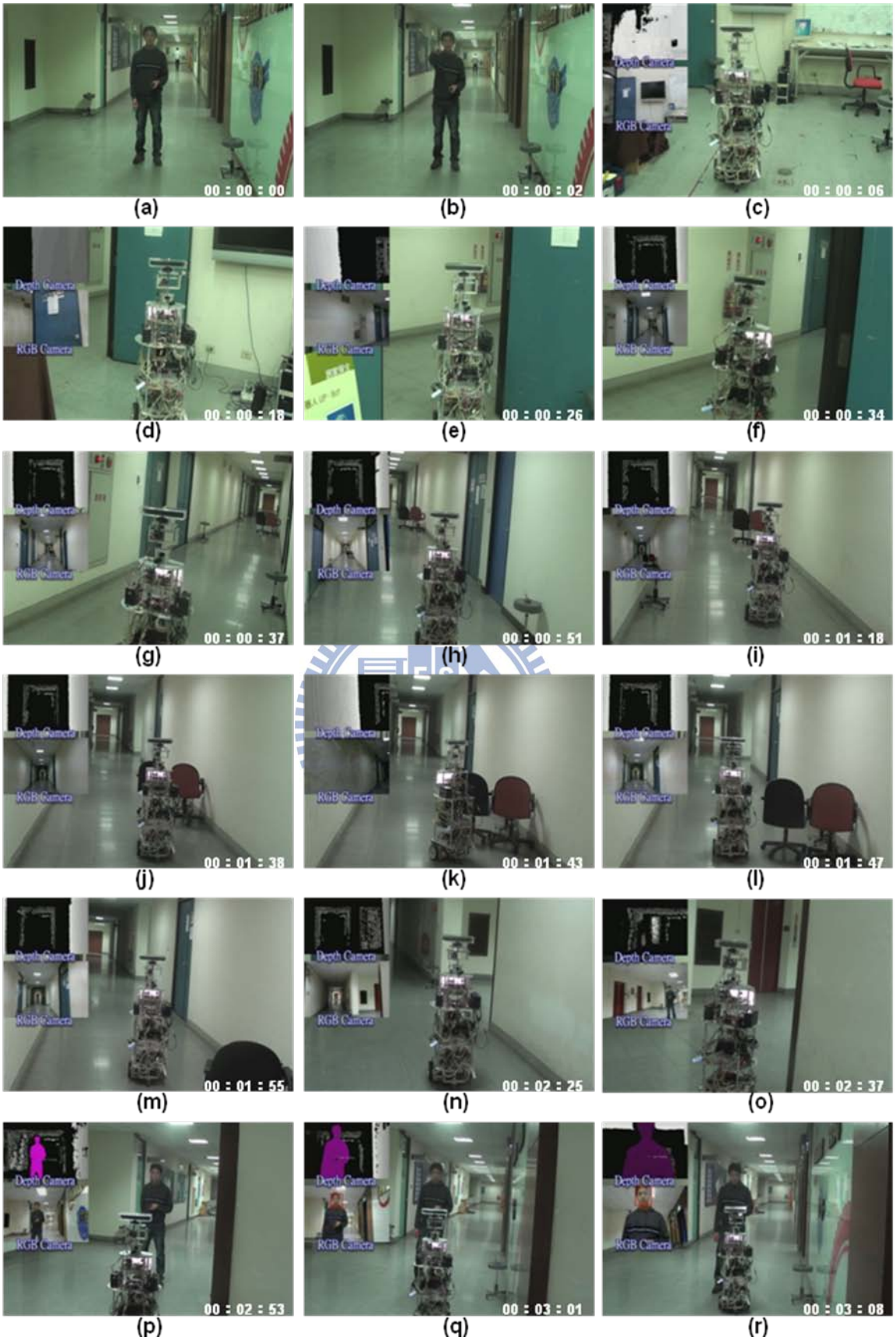

圖 6-21 長距離召喚機器人之實驗過程

表 6-6 長距離召喚機器人之定位實驗數據

D_{Ro}	$(X_{Ro}, Y_{Ro})_{old}$	(X_{Rz}, Y_{Rz})	D_{Rz}	W_Z	$(X_{Ro}, Y_{Ro})_{new}$
6 m	(-346, 490)	(-200, 975)	(150, 25)	(0, 0.15)	(-346, 562)
8 m	(-380, 777)	(-200, 1175)	(0, 150)	(0.216, 0)	(-341, 777)
10 m	(-363, 988)	(-200, 1325)	(0, 75)	(0.283, 0.12)	(-317, 1029)
12 m	(-339, 1235)	(-200, 1475)	(0, 50)	(0.35, 0.15)	(-290, 1271)
14 m	(-319, 1474)	(-375, 1500)	(0, 25)	(0.416, 0.416)	(-342, 1485)
16 m	(-367, 1686)	(-350, 1525)	(50, 0)	(0.283, 0.483)	(-362, 1608)
18 m	(-387, 1810)	(-200, 1775)	(150, 225)	(0, 0)	(-387, 1810)
20 m	(-444, 1999)	(-400, 1800)	(200, 100)	(0, 0.216)	(-444, 1956)
22 m	(-448, 2162)	(-375, 2150)	(0, 0)	(0.683, 0.683)	(-398, 2153)
24 m	(-422, 2354)	(-200, 1900)	(175, 175)	(0.15, 0.15)	(-389, 2286)
26 m	(-423, 2483)	(-200, 2425)	(125, 500)	(0.35, 0)	(-345, 2483)
28 m	(-380, 2680)	(-300, 2775)	(75, 0)	(0.55, 0.77)	(-345, 2751)
30 m	(-244, 2929)	(-255, 2950)	(125, 125)	(0.35, 0.35)	(-233, 2936)

表 6-7 長距離召喚機器人之實驗結果

Times	The distance between the robot and user	The angle between the robot and user
1	70 cm	35°
2	68 cm	23°
3	69 cm	32°
Average	68 cm	30°

第七章 結論與未來工作

7.1 結論

本論文主要目的在於設計一個機器人召喚系統，讓機器人偵測到使用者召喚訊號後，機器人可以不受環境距離、障礙物等影響，順利找到使用者，並且移動到使用者面前。

首先，本論文採用 CC2431 定位引擎來進行室內定位，透過交叉式的參考點佈置，估測出使用者位置，讓機器人透過自主導航系統來前往使用者，實驗結果顯示，在 8 個參考點的定位環境下，CC2431 定位引擎的平均定位誤差為 1.3 公尺左右，此距離已經足夠讓 Kinect 感測器擷取到使用者，並且透過影像來追蹤使用者。

本論文提出的自我定位系統，整合里程計與 CC2431 定位引擎之定位資訊，可以改善機器人因為移動過久、運作時間過長所產生的定位誤差，實驗結果顯示，在 13 個參考點的定位環境下，機器人移動 60 公尺後的平均定位誤差為 68 公分，表示自我定位系統可以有效地減少機器人自身定位的誤差，對於需要尋找長距離的使用者之情況下，機器人不會迷失自身座標，有助於召喚任務之達成。

透過 Kinect 感測器所提供的深度影像，使用使用者產生器來偵測使用者人形，並且計算使用者人形的質心座標，同時，透過 Kinect 感測器所提供的彩色影像，使用 Haar-Like Features 人臉偵測與人臉膚色密度來偵測使用者人臉，並且計算使用者人臉的中心座標與人臉寬度，將這些影像資料進行整合後，透過本論文設計的影像追蹤控制器，讓機器人可以修正自身方向來朝向使用者前進，並且在適當的距離下，停在使用者面前。

本論文將導航控制系統當作基礎，整合自我定位系統與影像追蹤控制器，讓機器可以閃避環境中的障礙物，並且有效地降低機器人自身定位的誤差，還能影像偵測使用者，讓機器人可以修正自身方向來朝向使用者前進，在適當的距離下，停在使用者面前，實驗結果顯示，在 12 個參考點的定位環境下，機器人可

以找到距離 35 公尺的使用者，並且機器人停在使用者面前的平均距離為 68 公分，而機器人面對使用者正面的平均角度差為 30°，表示機器人能順利找到使用者，並且停在使用者面前，完成召喚任務。

7.2 未來工作

機器人召喚系統還有一些地方值得進一步研究：

在影像方面，機器人雖然可以偵測出多個使用者，但是卻無法知道哪位使用者才是真正召喚機器人的使用者，所以可以進一步整合影像辨識功能，讓機器人在多人環境下，可以找出真正召喚機器人的使用者。此外，加入使用者側面或是使用者的背後等影像偵測，可以幫助機器人找到不同站姿的使用者。

在自我定位系統方面，機器人雖然透過 CC2431 定位引擎所提供的絕對座標來修正自身座標，但是卻沒有絕對角度來修正機器人的朝向角 θ，所以可以使用影像校正方式，讓機器人每次任務結束後，回到休息站來進行影像校正機器人的朝向角 θ。此外，對於融合里程計與 CC2431 定位引擎之定位資訊是透過模糊邏輯系統來決定融合數值，這是一種經驗法則，不是最佳比例的融合數值，所以使用 Extended Kalman Filter 來找出最佳比例的融合數值，可以使得自我定位系統達到最佳化。

使用同時定位與地圖建立(Simultaneous Localization And Mapping, SLAM)技術於機器人自我定位，其定位誤差會比使用無線感測網路定位機器人來得小許多，不僅解決機器人自我定位問題，透過建立環境地圖也幾乎解決機器人導航問題，所以使用 SLAM 技術，可以讓機器人自我定位更準確，順利導航到使用者身邊，有助於召喚任務之達成。此外，機器人有了環境地圖後，加入一個實用的路徑規劃系統，可以幫助機器人在複雜環境下，快速找到使用者。

參考文獻

[1] Kai-Tai Song, Chi-Yi Tsai, Fu-Sheng Huang, Jung-Wei Hong, Chen-Yang Lin, Chun-Wei Chen and Zhi-Sheng Lin, “Development of the robot of living aid: RoLA,” in *Proc. of IEEE International Conference on Automation and Logistics (IEEE ICAL)*, pp. 443-448, Qingdao, China, Sep. 2008.

[2] Yoo Oh, Jae Yoon, Ji Park, Mina Kim and Hong Kim, “A name recognition based call-and-come service for home robots,” in *IEEE Transactions Consumer Electronics*, vol.54, no.2, pp.247-253, May 2008.

[3] B. C. Park, K. D. Ban, K. C. Kwak, and H. S. Yoon, “Sound source localization based on audio-visual information for intelligent service robots,” *Int. Symposium on Advanced Intelligent Systems*, pp.364-367, Sep. 2007.

[4] Suk Lee, Kyoung Nam Ha and Kyung Chang Lee, "A pyroelectric infrared sensor-based indoor location-aware system for the smart home," in *IEEE Transactions Consumer Electronics*, vol.52, no.4, pp.1311-1317, Nov. 2006.

[5] Guangming Song, Jun Zhang, Xiaofeng Ye, Yanpeng Niu and Aiguo Song, "Wireless sensor and actuator network system for calling home robots," in *Proc. of IEEE International Conference Information and Automation (IEEE ICIA)*, pp.1553-1558, Harbin, China, June 2010.

[6] Guolin Sun, Jie Chen, Wei Guo and K. J. Ray Liu, "Signal processing techniques in network-aided positioning: a survey of state-of-the-art positioning designs," in *IEEE Signal Processing Magazine*, vol.22, no.4, pp.12-23, July 2005.

[7] F. Gustafsson and F. Gunnarsson, "Mobile positioning using wireless networks: possibilities and fundamental limitations based on available wireless network measurements," in *IEEE Signal Processing Magazine*, vol.22, no.4, pp.41-53, July 2005.

[8] Dae Geun Hwang, Jae Ho Hwang, Sung Jeen Jang and Jae Moung Kim, "A fast ToA position estimation technique based on MHP pulse," in *Proc. of International Symposium Communications and Information Technology(ISCIT)*, pp.1472-1476, Korea, Sep. 2009.

[9] F. Gustafsson and F. Gunnarsson, "Positioning using time-difference of arrival measurements," in *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol.6, pp. VI- 553-6, April 2003.

[10] Rong Peng and M.L. Sichitiu, "Angle of arrival localization for wireless sensor networks," in *2006 3rd Annual IEEE Communications Society on Sensor and Ad Hoc Communications and Networks*, Reston, pp.374-382, VA, USA, Sep. 2006.

[11] Jeffrey Hightower and Gaetano Borriello, "A Survey and Taxonomy of Location Sensing Systems for Ubiquitous Computing", Technical Report UW CSE 01-08-03, University of Washington, Department of Computer Science and Engineering, Washington, 2001.

[12] P. Bahl and V. N. Padmanabhan, "RADAR: an in-building RF-based user location and tracking system ," in *Proc. of IEEE INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies*, pp.775-784, Tel Aviv, 2000.

[13] K. Kaemarungsi and P. Krishnamurthy, "Properties of indoor received signal strength for WLAN location fingerprinting," in *Proc. of The First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services*, Boston, Massachusetts, pp.14-23, USA, Aug. 2004.

[14] B. Li, J. SALTER, A. G. DEMPSTER and C. RIZOS, "Indoor positioning techniques based on Wireless LAN," in *Wireless Broadband & Ultra Wideband Communications*, pp.13-16, March 2006.

[15] 林鎮源, "移動式機器人之行為融合控制器設計," 國立交通大學電機與

控制工程研究所碩士論文，2005.

[16] K. Aamodt, "CC2431 Location Engine," Texas Instruments, July 2006.

[17] "A True System-on-Chip solution for 2.4 GHz IEEE 802.15.4 / ZigBee datasheet," Texas Instruments, 2009.

[18] Li Wenzhong, “ZigBee 2006 wireless networks and wireless location Combat,” Beijing University of Aeronautics and Astronautics Press, 2008.

[19] Liyuan Li, Hoe J. K. E., Shuicheng Yan, and Xinguo Yu, "ML-fusion based multi-model human detection and tracking for robust human-robot interfaces," in *Applications of Computer Vision*, pp.1-8, Dec. 2009.

[20] Keita Itoh, Takashi Kikuchi, Hiroshi Takemura and Hiroshi Mizoguchi, "Development of a person following mobile robot in complicated background by using distance and color information," in *The 32nd Annual Conference of the IEEE Industrial Electronics Society*, pp.3839-3844, Paris, Nov. 2006.

[21] 台灣 Kinect 官網
http://www.xbox.com/zh-TW/Kinect

[22] Kinect technical details
http://www.play.com/Games/Xbox360/4-/10296372/Project-Natal/Product.html

[23] Open NI
http://openni.org/

[24] 透過 OpenNI / NITE 分析人體骨架
http://kheresy.wordpress.com/2011/01/28/detecte_skeleton_via_openni_part1/

[25] OpenCV
http://sourceforge.net/projects/opencvlibrary/

[26] P. Viola and M. Jones, “Rapid object detection using a boosted cascade of simple features,” *in Proc. of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. I-511-I-518, 2001.

[27] Rainer Lienhart and Jochen Maydt. “An Extended Set of Haar-like Features for Rapid Object Detection,” *in Proc. of International Conference on Image Processing*, pp. 900-903, 2002.

[28] M. Soriano, B. Martinkauppi, S. Huovinen and M. Laaksonen, “Skin detection in video under changing illumination conditions,” *in Proc. of 15th International Conference on Pattern Recognition*, pp. 839-842, Istanbul, Turkey, 2000.

[29] C. C. Chiang, W. K. Tai, M. T. Yang, Y. T. Huang and C. J. Huang, “A Novel Method for Detecting Lips, Eyes and Faces in Real Time,” *Real-Time Imaging*, vol. 9, no. 4, pp. 277-287, 2003.