國 立 勤 益 科 技 大 學

機 械 工 程 系 碩 士 班

碩士論文

深度感測器應用於追蹤機器人之研究

A Study of a Depth Sensor Applying on a Tracking Robot

研究生：何治鋒

指導教授：邱俊智 博士

中華民國 一〇一 年 七 月

深度感測器應用於追蹤機器人之研究

A Study of a Depth Sensor Applying on a Tracking Robot

研究生：何治鋒
指導教授：邱俊智 博士

國立勤益科技大學

機械工程系碩士班

碩士論文

A Thesis Submitted to
Department of Mechanical Engineering
National Chin-Yi University of Technology
in Partial Fulfillment of the Requirements
for the Degree of Master of Science

July 2012
Taiping, Taichung, Taiwan, Republic of China

中華民國 一〇一 年 七 月

國立勤益科技大學
研究所碩士班
論文口試委員會審定書

本校 機械工程系 碩士班 何治鋒 君

所提論文 深度感測器應用於追蹤機器人之研究

合於碩士資格水準，業經本委員會評審認可。

口試委員：郭高為　　

陳正和　　

邱俊榮　　

指導教授：邱俊榮

系（所）主管：機械工程系 主任 蔡明義

中華民國 一〇一 年 七 月

誌謝

本論文能順利完成，感謝吾師邱俊智教授這二年來細心的教導，使我無論在論文、課業與日常生活上皆受益良多，更讓我在老師身上從中學習到如何處理問題，能獨立思考和統整邏輯的判斷能力來解決問題及對我將來做人處事上受用無窮，在此致上崇高的感謝。

感謝口試委員，陳正和教授與郭宗益教授特別抽空前來指導，讓我知道還有許多地方需要改進，使本論文能夠更加完整，在此特別感謝。

在這二年碩士生活其間，首先感謝聖哲、政傑、政勳、柏巖、昌育、勝凱、立凱等學長，在剛進機器人實驗室懵懵懂懂的我特別照顧，感謝明毅、元聖、培勝、蔣承、一郎、佩吟等同學，陪伴著我度過風風雨雨，不管快樂或悲傷都有你們在身邊陪伴，還有許多要感謝的同學們，雖然無法在這全說盡，但我會一直記住有你們的陪伴真好。

最後感謝我的爸、媽栽培我念完碩士，感謝哥在我有問題時細心的教導我，你們辛苦了!特別感謝我的女友及她的家人，在我有困難時幫我度過難關，且無時無刻的叮嚀我，使本論文能順利完成。

－ 治鋒 －

2012/07/31

深度感測器應用於追蹤機器人之研究

研究生：何治鋒　　指導教授：邱俊智 博士

國立勤益科技大學機械工程系碩士班

摘要

本研究主要探討微軟公司所生產的XBOX Kinect深度感測器，將它應用於追蹤機器人。使用Kinect可以取得追蹤目標的三維位置，傳統的方法中，需要二台攝影機以仿人眼的方式平行放置，再經由影像特徵辨識及複雜演算法才可以得到追蹤目標的三維位置。本研究將Kinect架設在機器人上並連接在機器人的電腦，再利用OpenNI C/C++ API撰寫Kinect的應用程式，當Kinect接收到追蹤目標的三維位置，將資料做為機器人程式追蹤目標物的依據。本研究所建立的追蹤機器人包括三個主要功能：(1)機器人與追蹤目標之相對座標系統，(2)目標追蹤系統控制機器人與追蹤目標之相對距離，(3)方向導正系統用以修正機器人與追蹤目標之相對方位。目前所建立之追蹤機器人已可達成追蹤之功能。

關鍵字 ：XBOX Kinect、深度感測器、三維位置、追蹤機器人

A Study of a Depth Sensor Applying on a Tracking Robot

Student：Chih-Feng He **Advisor：Dr. Chun-Chih Chiu**

Department of Mechanical Engineering, National Chin-Yi University of Technology

Abstract

This research studies a XBOX Kinect, a depth sensor made by Microsoft, and applies it on a tracking robot to get the three-dimensional (3D) position of the tracking target, an object tracked by the robot. In traditional ways, researchers utilize two cameras placed parallel to imitate the human's eyes and then go through the process of image feature recognition and a sequence of complex algorithm to decide the target's 3D position. In this research, a Kinect sensor is installed on top of the robot and is connected to robot's computer.Use OpenNI C/C++ API to program Kinect's application software. Once the Kinect receives the target's position, the data is used in the robot's operating software to track the target. The setup tracking robot contains three major functions: (1) a relative coordinate system between the robot and the target, (2) a target tracking system controlling the distance between the robot and its tracking target, (3) a direction modification system adjusting the orientation between the robot and its tracking target. The setup system has shown its tracking performance.

Keywords: XBOX Kinect, Depth Sensor, Three-dimensional Position, Tracking Robot

目錄

圖目錄

表目錄

第一章　緒論

高科技的時代來臨了，許多高科技產品已陸續問市，例如：3D 汽車儀表顯示板，可將駕駛人員前方數十公尺處的路況以 3D 方式呈獻於汽車儀表板上，方便駕駛觀察路況；智慧型鬧鐘，將護腕式感知器於睡前戴在手上，感知器會感知你的睡眠狀態，當計時快到了，而你的睡眠狀態又漸漸的進入深層睡眠時，鬧鐘會即時提醒，使你不容易睡過頭；智慧型台燈，內部裝有電腦，可透過無線鍵盤操作，擁有四個銀幕，可同時顯式不同資訊，如：天氣、新聞、e-mail、音樂播放器等功能。

機器人的研究已有數十年，近年來更是蓬勃發展，以往對機器人的認知是在工廠裡的機械手臂，可代替人力 24 小時運作，增加工廠的生產效率，而近年來機器人已經從工廠慢慢融入人們的日常生活中，2010 年臺北國際花卉博覽會的宣傳大使“香草寶貝”為一導覽機器人，胸前觸控式螢幕會顯示多媒體影音，加上跳舞與遊客互動，還會用各種語言跟遊客打招呼，展現了台灣科技上的實力。在電影“機械公敵”中，每個家庭都有一台智慧型機器人，是家庭中的重要成員，當你回到家時會幫你開門、倒茶、開燈等，能為你的生活帶來不少的便利，說不定過個十幾二十年現實生活中也會有此種類似情形。

1.1 機器人發展概況

早期，機器人主要是應用在工業上居多，機械手臂就是很典型的例子，到現在越來越多的工廠走入自動化作業，也都仰賴於機械手臂；機器人也可代替人類從事高度危險性的工作，如：拆炸彈、深海探勘、星球探測等，精神號(Spirit, MER-A)

和機遇號(Opportunity, MER-B)是美國國家航空暨太空總署火星探測漫遊者計劃的第一部雙胞胎火星漫遊車[1][2]，主要任務為探測火星上是否有水和生命的存在，並分析地質與岩石，評估火星環境能否支持生命的存在，如圖(1-1)、圖(1-2)所示。

圖 1-1 Spirit, MER-A

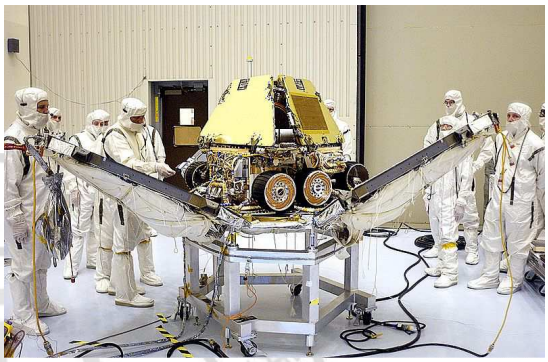


圖 1-2 Opportunity, MER-B

近年來，各國許多研究偏向於家庭用機器人、娛樂機器人、商用機器人等方面發展。麻省理工研究團隊研發出世界第一個有模擬情感的機器人 Kismet[3]，Kismet 不僅可以與人溝通，還有學習系統，更有多種仿人類表情，喜、怒、哀、樂，通通難不倒它，如圖(1-3)所示。

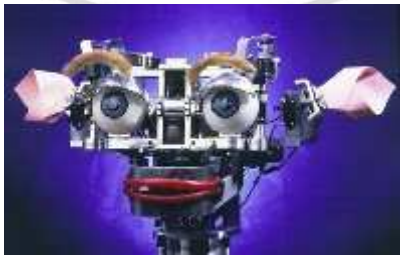


圖 1-3 情感機器人 Kismet

服務型機器人更是現代研究發展的對象，服務機器人主要從事運輸、清潔、

保安、監護等工作，國立勤益科技大學電子工程系邱詰淙等四人由國科會計畫資助，研製出點餐服務機器人[4]，此機器人擁有觸控式螢幕點餐、影像辨識、語音辨識、超音波感測器避障、室內雷射定位導航系統等多項功能，為餐飲服務生帶來不少衝擊，但由於造價非凡，還無法取代人力成本，所以離正式上市還需要加以研發改進，如圖(1-4)所示。

圖 1-4 點餐服務機器人

1.3 研究動機與目的

機器人的開發正逐漸朝向日常生活的實際應用，其中具有輔助與跟隨功能的看護型機器人正是備受關注的項目之一。以往追蹤機器人的研究大多是利用二部攝影機平行放置，利用影像重疊的原理，加上立體視覺演算法，經過較複雜的運算求出機器人與目標的相對距離，本論文則是探討只需一部 Kinect 深度感測器就能達到深度感測的效果，結合機器人及演算法達到追蹤之目的，選擇使用 Kinect 感測器是因為不必使用複雜的演算就能準確的取得深度值，並且在市面上很容易取得，價格也不算太高。Kinect 深度感測主要是由一部 RGB 彩色攝影機、紅外線

發射器及紅外線攝影機所構成，詳細規格後面章節會加以說明，控制系統是採用模糊控制演算法，機器人要追蹤的對象是人，人在走路過程中難免會忽快忽慢，左右晃動，利用模糊控制演算法使機器人能與被追隨者保持固定距離，追蹤機器人可被應用於大賣場、機場及醫院等場所為人服務。

1.4 文獻回顧

本節將回顧國內外追蹤機器人與具有跟隨能力之自走車的相關研究。

H. Kwon[5]等人利用二台獨立移動攝影機，由攝影機拍攝到的影像中估計目標距離，建立控制輸入與影像像素之間關係，並實現於移動機器人，達到追蹤人的演算法。

T. Yoshimi[6]等人利用特徵比對方式，追蹤機器人在追蹤目標前須透過目標檢測系統，如衣服紋路或顏色等檢測，判別該目標為即將追蹤之對象，方能鎖定該目標追蹤，不受外人干擾。

N. Hirai, H. Mizoguchi[7]利用視覺追蹤法，根據衣服材質及肩膀寬度，進行影像比對，達到人體追蹤效果。

C. Y. Tsai, K. T. Song[8]使用閉迴路視覺追蹤控制系統，基於影像的誤差狀態控制模型，對人臉做視覺追蹤，實現於移動機器人上。

何昭慶[9]使用一對網路攝影機，利用連續適應性均值追蹤演算法與立體視覺

演算法求出目標在空間中立體位置，此技術可應用於機器人或自走車控制。

Li, T.-H.S.[10]等人設計二台自主式移動機器人，前者裝設紅外線發射器，並設定移動軌跡，後者裝設紅外線接收器，並設計模糊目標追蹤控制，追隨前者機器人移動。

林哲瑋[11]使用 CMOS 攝影機，利用影像面積量測法，只需簡單公式就能求出感測目標之距離，並套用多車整合控制應用於自走車上，使該自走車能與前導車保持固定距離。

何育昕[12]使用聲納感測器感測自走車與目標資訊，透過模糊距離控制器估測自走車與目標相對距離，並由模糊速度控制器使自走車與目標保持固定距離，達到目標追蹤目的。

M. Chueh[13]等人設計自主機器人追蹤控制器，基於視覺量測由卡爾曼濾波器，從導引目標的作動不斷地估測機器人下一步該移動到的位置，該控制器可應用於機器人與人的互動上。

Y. Nagumo[14]等人設計自主式移動追蹤機器人，前導者必須裝配有二個 LED 光源發射器，機器人頂端裝有攝影機，會捕捉 LED 光源，當前導者移動時，機器人會跟著光源方向追蹤。

M. Kobilarov[15]等人將視覺追蹤結合雷射測距儀應用於移動機器人上，只使用視覺追蹤時，在環境較複雜的情況下，容易受外在干擾，而加上雷射測距儀後，目標如果只是暫時被遮蔽還是能追蹤。

賴一翔[16]設計全方位移動跟隨機器人，被跟隨者身上配戴紅外線光源裝置，由機器人上的攝影機透過影像處理技術判別目標方位，並由紅外線感測器感測其相對距離，就由模糊理論控制機器人跟隨。

傅培耕[17]使用雙 CCD 攝影機擷取連續影像，經影像處理比對顏色，藉由立體視覺演算法計算出目標物與自走車相對距離，實現於自走車上，使自走車有追蹤特定顏色目標能力。

1.5 論文架構

本論文總共分為五個章節，以下為各章節的介紹:

第一章 緒論

本章節介紹前言、機器人發展概況、研究動機與目的及文獻回顧。

第二章 系統架構與軟硬體介紹

本章節介紹追蹤機器人系統架構、硬體設備規格及軟體。

第三章 機器人控制策略

本章節介紹機器人移動方式、座標系統、追蹤控制系統及方向導正系統，其中追蹤控制系統包含了模糊理論、模糊控制原理及模糊控制器設計。

第四章 實驗結果與測試

本章節呈現 Matlab 軟體模擬結果、Kinect 感測器追蹤策略、追蹤控制系統測試、方向導正系統測試。

第五章 結論與未來展望

最後章節陳述本研究結論及未來展望。

第二章　系統架構與軟硬體介紹

2.1 系統架構

本系統架構包含機器人端與監控端，機器人端為本系統之核心，機器人內部即為一部電腦，以下稱之為 PC-Mini-ITX，馬達控制與感測器資訊的傳輸都是透過 USB 管理模組(USB Machine Management Module)[18]，被追蹤者的畫面與資訊會顯示於 PC-Mini-ITX 螢幕上，不過為了操作便捷，一般機器人端不配置螢幕，因此為了方便監控，監控端另有一台桌上型電腦，透過遠端遙控桌面軟體 TeamViewer6.0 來監控機器人端之畫面，系統架構圖如圖(2-1)所示。

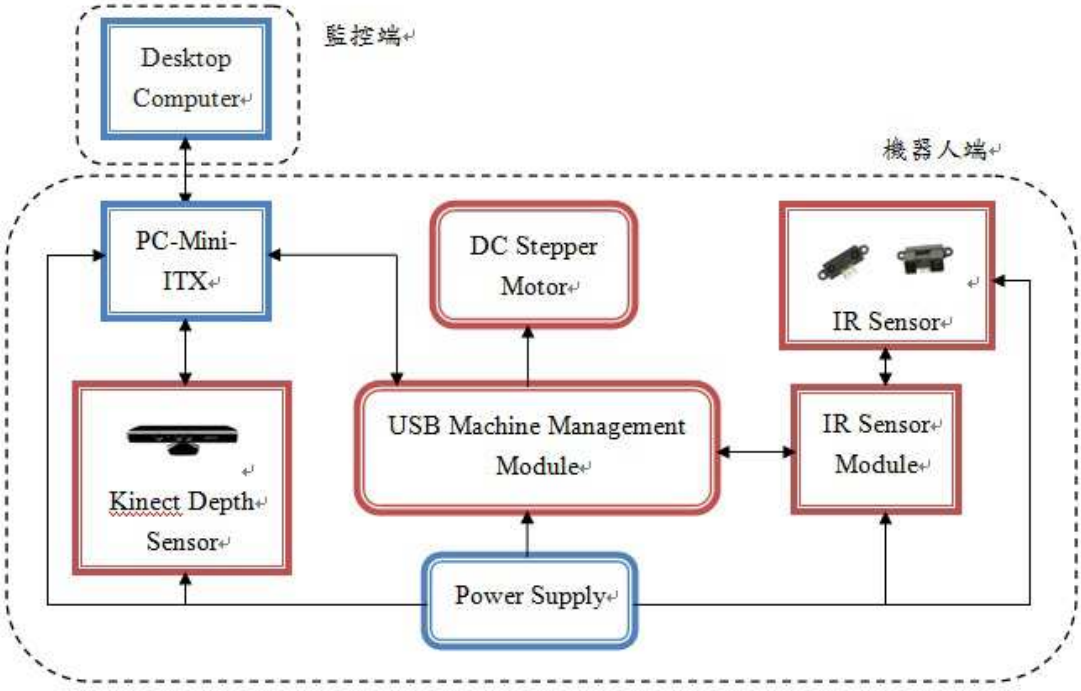


圖 2-1 系統架構

機器人端的電源皆由 Power Supply 電源系統所供應，包含 USB 管理模組、IR Sensor Module、IR Sensor、Kinect Depth Sensor、DC 步進馬達及 PC-Mini-ITX，其中較值得一提的是,Kinect感測器本身消耗的功率超過USB2.0所能提供的範圍(5V, 500mA)，因此 Kinect 感測器除了將 USB 端口接於 PC-Mini-ITX 之外，還需外接 12V 電源。Kinect 感測器資訊的傳輸選用 TCP/IP 通訊協定，TCP/IP 協定會確認資料是否送達，避免資料遺失的狀況發生。

本系統運作前的首要動作是監控端電腦與 PC-Mini-ITX 必須先透過 TeamViewer6.0 遠端遙控，監控端電腦銀幕上所顯示的畫面即為 PC-Mini-ITX 的畫面，此時即可於監控端操作 PC-Mini-ITX 裡的程式，達到無線遠端監控的功能。

2.2 整體設計

本論文所研究之追蹤機器人是將微軟公司所生產的 XBOX Kinect 深度感測器架設在 914 PC-Bot 機器人上，機器人所要追蹤的目標物為人，一般成年人的平均身高約為 150~170cm，而 914 PC-Bot 機器人高度為 53.4cm，考慮到 Kinect 感測器所能感測到的範圍，需將 Kinect 感測器架設到合適的高度。

考慮到機器人的負重，選用 33.5(H)×31 (W) cm 厚度為 0.3cm 的鋁板當層板，如圖(2-2)所示，而支撐層板的材質則選用 20(H)×20(W) cm M4 系列鋁擠型長度為 41cm，如圖(2-3)所示，由於鋁擠型方便剪裁、質地輕且抗壓性強，使機器人架高設計穩固，經設計後的追蹤機器人高度約為 90cm，如圖(2-4)所示，此高度對應 Kinect 感測器的感測範圍已達到本研究適合之高度，為了方便固定鋁擠型，拆除了原機器人的上殼，所以高度非 53.4cm+41cm。

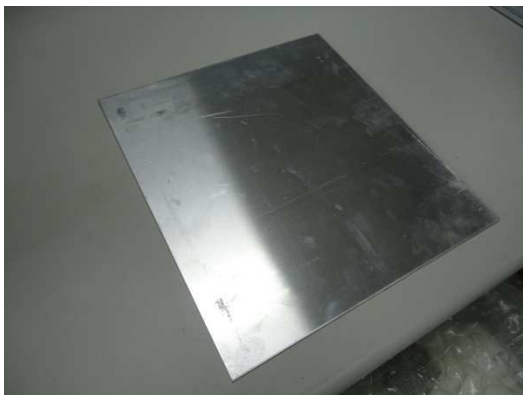


圖 2-2 鋁板實體圖

圖 2-3 鋁擠型實體圖

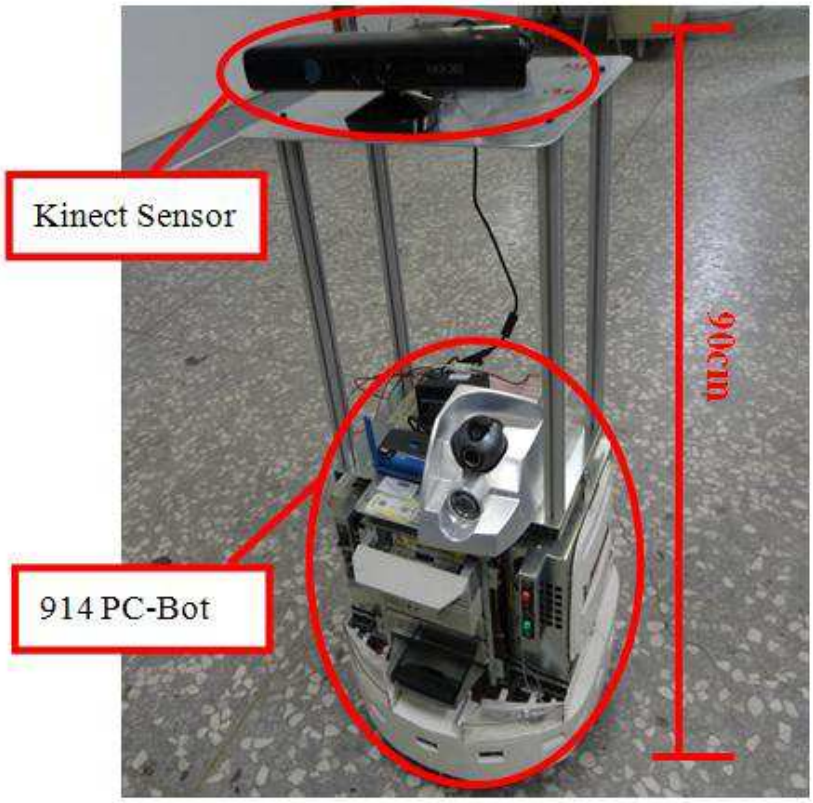


圖 2-4 追蹤機器人實體圖

2.3 914 PC-Bot 機器人

914 PC-Bot機器人是由加拿大White box Robotics公司所設計[19]，White box Robotics公司發展的重點是PC機器人，於2005年推出914 PC系列機器人，2006年在紐約的數位生活展上榮獲“最佳展式:未來技術獎”，許多大學都與White box Robotics公司有合作，並結合研究推廣教育合作。

機器人的電腦系統可為Microsoft Windows或Linux，本論文所使用的系統為Microsoft Windows，因機器人內部為電腦，所以運算速度大於一般嵌入式系統機器人，移動控制是由二個DC步進馬達驅動，而一般電腦的基本配備，主機板、硬碟、光碟機它也都具備。

2.3.1 硬體規格

914 PC-Bot 機器人有專利設計的自我清潔滾動腳輪、八個 IR Sensor、無線網路、USB 管理模組、二個 DC 步進馬達、M2-ATX 電源供應器及一般的電腦配備[19]，914 PC-Bot 機器人規格如表(2-1)所示。

表 2-1 914 PC-Bot 機器人規格

規格明細	內容
高度、重量	53.4cm、25kg
負重	5kg
爬坡斜度	8°
專利	自我清潔滾動球體腳輪
驅動	二個 DC 步進馬達

管理模組	USB 管理模組-馬達控制與 I/O 模板介面
感測組件	八個 IR 感測器
頭部組件	一個網路攝影機
網路	202.11g 無線 USB 轉接器
電源	二個M2-ATX電源供應器、 SONEIL 12V 智慧型電池充電器(3A)、 2 x 12V 9Ah (45W)鉛酸蓄電池
電腦配備	IGoLoic i3899 Mini-ITX主機板、 Intel Core 2 DUO 2GHz處理器、 1 Gbyte of DDR2 RAM、 80 GB 2.5”SATA硬碟、 Slim-styleDVD-ROM/CD-RW Combo Drive、 5.25”computer Bay 揚聲器

2.3.2 驅動馬達

本論文之機器人是由二個DC步進馬達(Stepping Motor)驅動二輪來移動，步進馬達與一般馬達比較具有以下特點[20]：

- 可以利用數位信號直接控制開迴路(open loop)，無需編碼器(Encoder)使系統簡單化。
- 可以使旋轉速度隨脈波信號的頻率成比例變化，使速度控制範圍增廣。
- 容易啟動、停止、正逆轉、變速、反應性好。
- 馬達的旋轉角度與輸入脈波數完全成比例。
- 角度誤差小、誤差不累積。
- 由於無電刷(Brush)，馬達本身的零件數少，信賴性高。

步進馬達的運作是由驅動電路切換馬達的勵磁(Exciting)，控制電路控制馬達的加減速、步進數及位置，利用檢出電路將馬達的角度與速度回傳至控制電路，其基本構成圖如圖(2-5)所示。

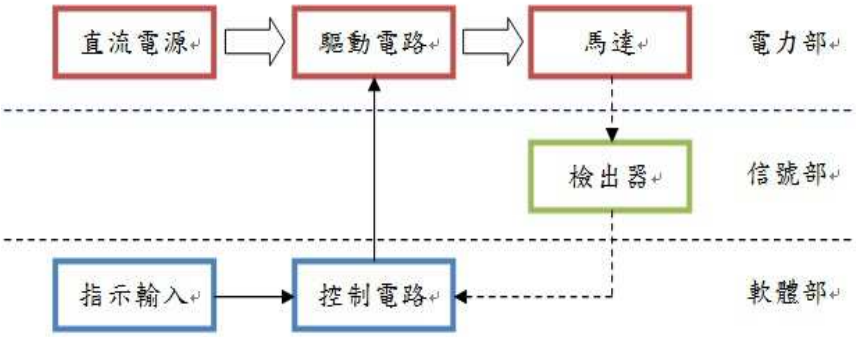


圖 2-5 步進馬達基本構成圖

2.3.3 電源系統

914 PC-Bot 機器人的總電源是由二個 12V 9Ah 鉛酸電池所提供，如圖(2-6)所示，透過二個 M2-ATX Power Supply 轉換電路後提供各元件使用，分別提供 PC-Mini-ITX 功率為 60W、電壓/電流 5V/1A 與 12V/2A，IR Sensor 電壓與電流為 5V/0.4A，Kinect Depth Sensor 電壓/電流為 12V/0.3A，USB 管理模組電壓與電流為 12V/3A。

圖 2-6 鉛酸電池

M2-ATX：

M2-ATX 是智慧型車用電源供應器[21]，最高輸出功率可達 160W，輸入電壓之範圍最高為 24V，最低為 6V，輸入電流限制為 15A，輸出電壓可為 5V、3.3V、12V、-12V，M2-ATX 的邏輯圖如圖(2-7)所示，實體圖如圖(2-8)所示，規格表如表(2-2)所示。

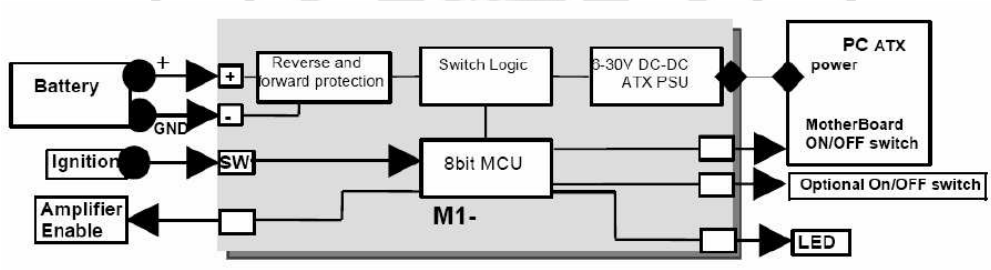


圖 2-7 M2-ATX 邏輯圖

圖 2-8 M2-ATX 實體圖

表 2-2 M2-ATX 規格表

規格明細	內容
最低作業電力	6V
最高作業電力	24V
輸入限制	15A
最高輸出功率	160W
作業溫度	-40℃ to 85℃
保存溫度	-55℃ to 125℃
平均故障率(MTBF)	192,000 hrs@50℃ , 96,000hrs@65℃
PCB 尺寸	160×45mm

2.3.4 感測組件

914 PC-Bot 機器人擁有八個紅外線感測器，都是 Sharp 公司所生產的，其中五個型號為 GP2Y0A21YK，如圖(2-9)所示，有效量測距離約 10~80cm，輸入電壓為 5V，輸出電壓為 0~3V，量測距離與輸出電壓關係[22]如圖(2-10)所示，位置配置示意圖[23]如圖(2-11)所示。

圖 2-9 GP2Y0A21YK 紅外線感測器

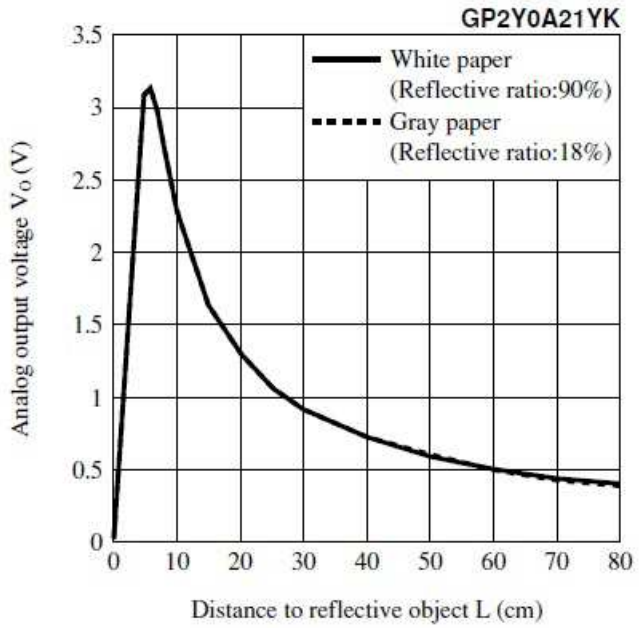


圖 2-10 GP2Y0A21YK 量測距離與輸出電壓關係圖

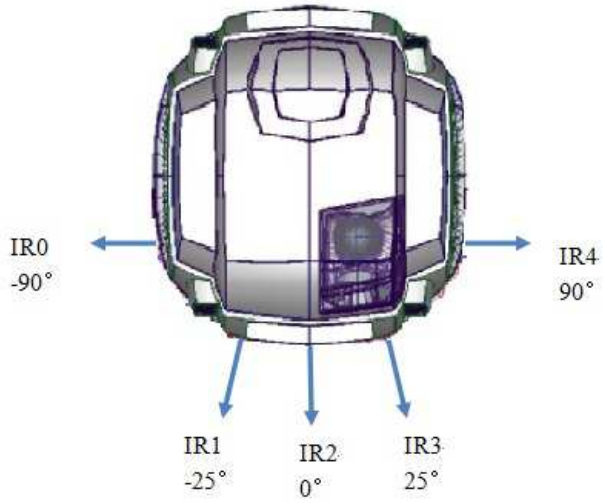


圖 2-11 GP2Y0A21YK 位置配置示意圖

另三個型號為 GP2Y0A02YK，如圖(2-12)所示，主要作為量測前方高低落差是否過大，如樓梯或台階等，所以位置配置是將感測器朝前下方，有效量

測距離約 20~150cm，輸入電壓為 5V，輸出電壓為 0~3V，量測距離與輸出電壓關係[24]如圖(2-13)所示，位置配置示意圖[23]如圖(2-14)所示，紅外線感測器位置配置實體圖如圖(2-15)所示。

圖 2-12 GP2Y0A02YK 紅外線感測器

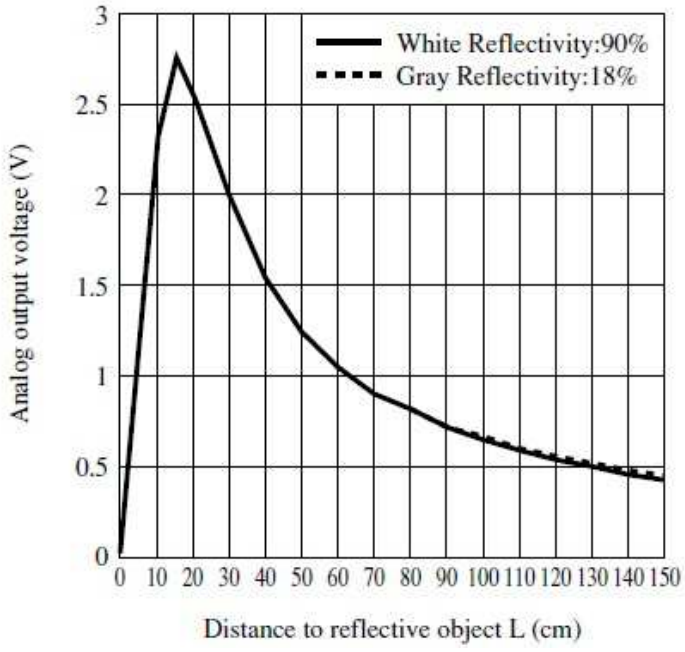


圖 2-13 GP2Y0A02YK 量測距離與輸出電壓關係圖

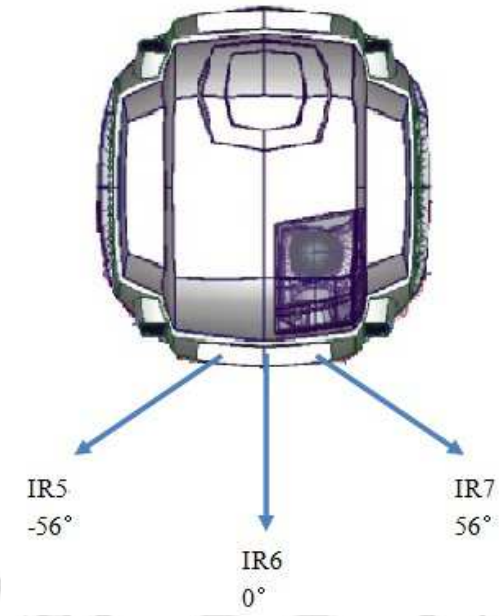


圖 2-14 GP2Y0A02YK 位置配置示意圖

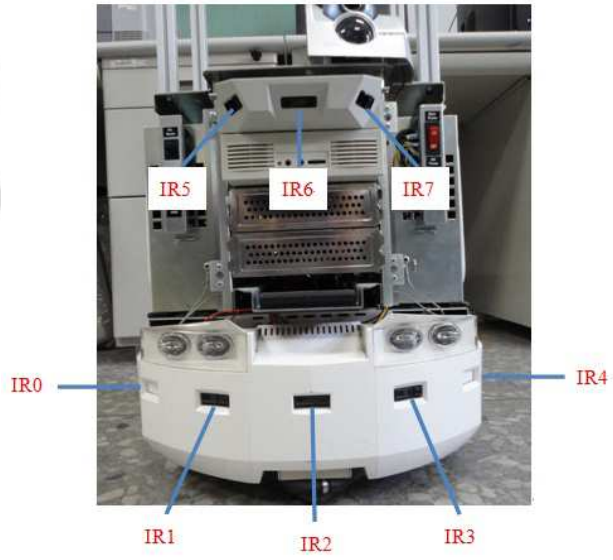


圖 2-15 紅外線感測器位置配置實體圖

二種紅外線感測器的內部結構皆相同[24]，如圖(2-16)所示。輸入電源為 5V，由 LED 端發射紅外線至待測物、經由待測物反射至接收端，在透過內部

的 Distance Measuring IC，計算出電壓，輸出的訊號為類比訊號 0~3V。

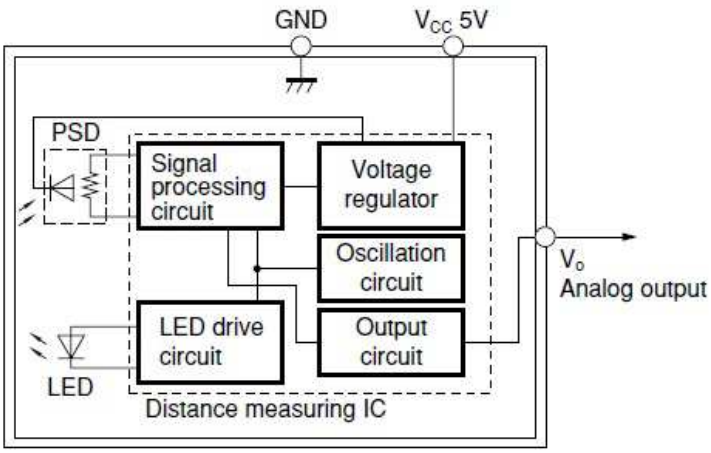


圖 2-16 紅外線感測器內部結構圖

2.3.5 開發軟體

本論文用來開發機器人的軟體為 Microsoft Visual Studio C#，C#(發音為 C Sharp)是微軟公司為.NET Framework 量身訂做的程式語言[25]，.NET Framework 為微軟.NET 跨平台功能必要的轉換程式，C#在 2000 年 6 月由微軟公司所推出，繼承了 C/C++與 Java 的特點。.NET 是微軟根據通用語言基礎架構(Common Language Infrastructure，CLI)規格所設計的軟體，CLI 規格雖是微軟規定，但任何廠商都可加入實作，.NET 即是微軟版的 CLI 實作軟體。

C#程式語言兼具翻譯與直譯的功能，中介語言檔使 C#程式有跨平台(Platform Independence)的功能，在任何裝有 C#翻譯程式的作業系統上翻譯好的程式可以在任何裝有.NET Framework 的作業系統上執行，其 C#跨平台運作流程圖如圖(2-17)所示。

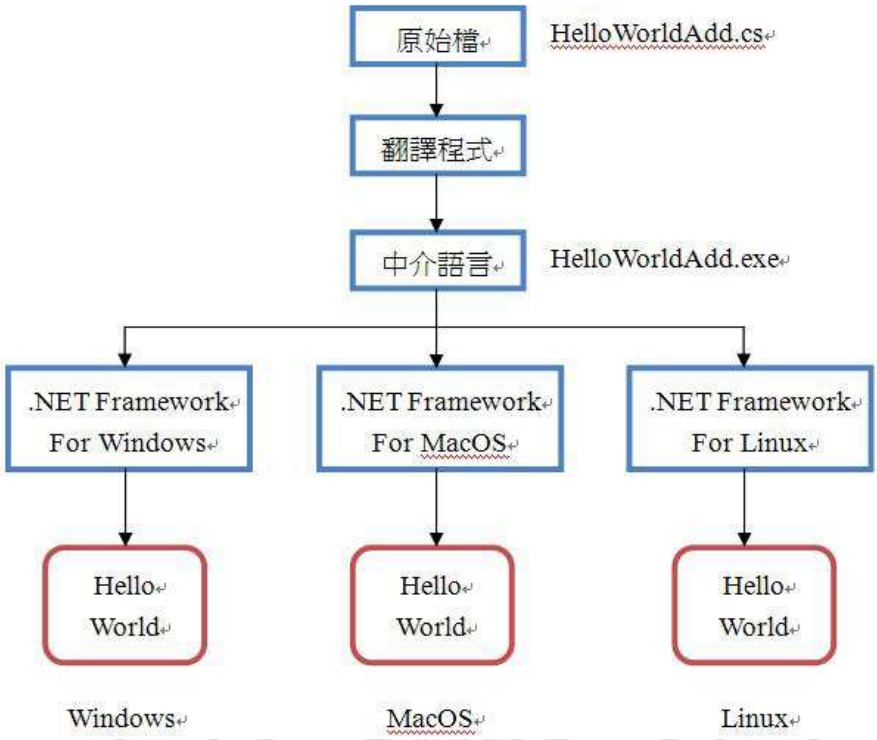


圖 2-17 C#跨平台運作流程圖

2.4 Microsoft Kinect 深度感測器

Kinect 深度感測器為 Microsoft 公司於 2010 年底推出的最新一代 3D 深度感測器，當時正流行任天堂的 Wii 遙控器，玩家只要拿著 Wii 揮動，內部加速度計會計算出其動作並顯示於遊戲中角色，而 Kinect 為一感測器，將其放置於適當距離，玩家手上不需配戴任何裝置或遙控，玩家做動作時，Kinect 感測器會將玩家的彩色影像與 3D 深度影像經過內部晶片運算後，顯示於遊戲中角色，此時玩家可真正體會到自然體感遊戲。

2.4.1 硬體規格

Kinect 感測器的鏡頭有三個，如圖(2-18)所示，其中間為彩色攝影機，擷取彩色影像，左邊為紅外線發射器，右邊為紅外線攝影機，擷取紅外線反射的 3D 深度影像，Kinect 擁有內建陣列式麥克風，可經由多組麥克風收音，經由比對後消除雜音，使聲音傳遞更清楚，因此 Kinect 感測器可擷取三種資訊，分別為彩色影像、3D 深度影像及聲音訊號[26]，如圖(2-19)所示，Kinect 感測器規格如表(2-3)所示。

圖 2-18 Kinect 感測器實體圖

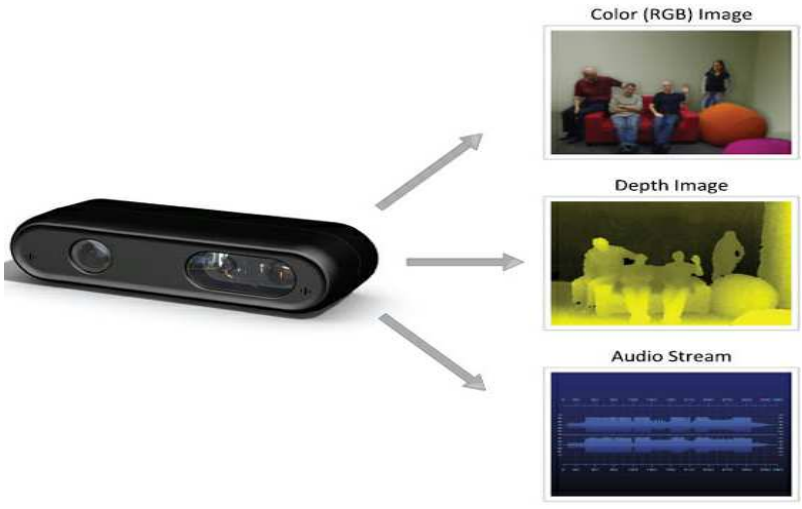


圖 2-19 Kinect 擷取的三種資訊

表 2-3 Kinect 規格表

規格明細	內容
水平視野角度	57°
垂直視野角度	43°
實體傾斜範圍	±28°
操作範圍	0.8m – 3.5m
深度感測器	640×480 16-bit 30fps
彩色攝影機	640×480 32-bit 30fps
聲音	16-bit 16 kHz
尺寸(WxHxD)	28cm x 3.5cm x 6cm

2.4.2 技術合作

Kinect 感測器是由 Microsoft 公司與 PrimeSense 公司合作推出的 3D 深度感測器，感測技術的開發是 PrimeSense 公司的Light Coding™感測技術，其是以面型近紅外線雷射光(Near-IR Light, CLASS1)對整個場景進行編碼，再以影像感測器(Standard CMOS Image Sensor)接收已編碼過之場景的紅外線影像，並將接收到之紅外線影像訊號，經由內部 SoC 晶片(PrimeSense PS1080 SoC)進行演算轉變為深度值[26]，最後再將處理過的資訊傳回至控制主機，如圖(2-20)所示。

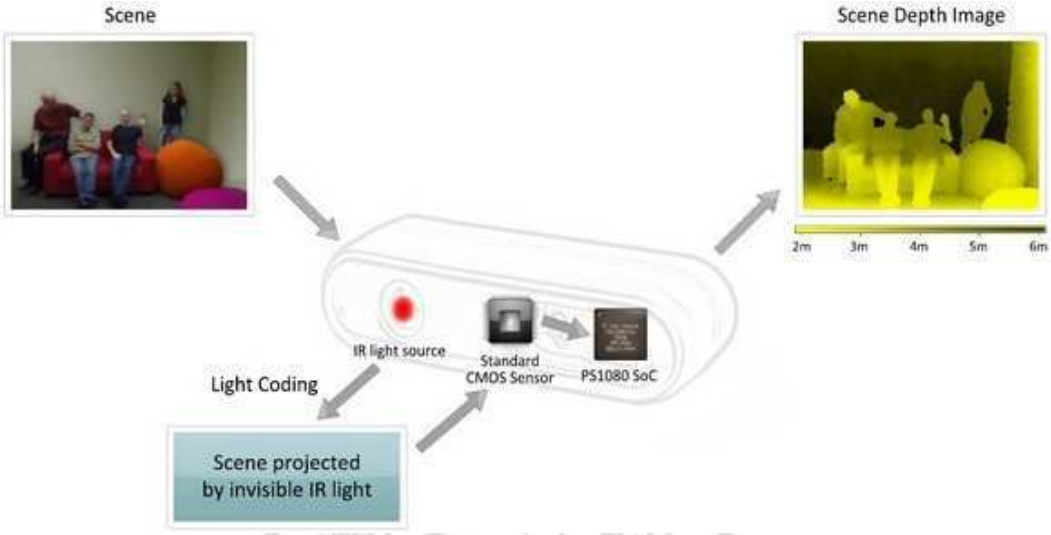


圖 2-20 Kinect 內部流程圖

2.4.3 開發軟體

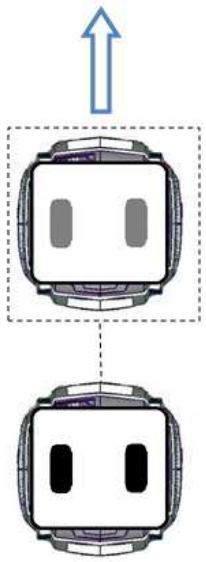
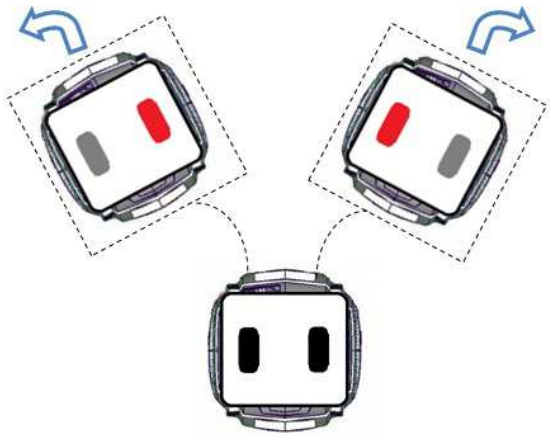
本論文用來開發 Kinect 感測器的軟體為 Microsoft Visual Studio C++，其中使用了 OpenNI(Open Natural Interaction)組織[27]所釋出的 API，OpenNI 是為了推廣自然互動領域所成立的組織，提供免費的 OpenNI Framework 開發框架給應用程式開發者使用，OpenNI Framework 提供許多高階功能方便開發者撰寫，如手勢辨識等，同時也提供了原始影像資料的存取，如深度影像資訊、彩色影像資訊等，該開發框架可客製許多特殊應用，像是 NITE (Natural Interaction Technology for End-user)就是 PrimeSense 所提供的[28]，用來將原始資料經過演算產生人體骨架資訊，NITE 提供了人體骨架 API 介面讓開發者使用，人體各關節的位置資訊都能透過此 API 來獲得，如肩膀、膝蓋、人體的重心等，在此應用到的是抓取人體重心的位置資訊作為感測目標位置的依據。

第三章　機器人控制策略

3.1 移動方式

本論文所使用之機器人屬於二輪驅動輪型機器人,雖然機器人擁有三個輪子，但其前輪為無動力的惰輪，做為前懸吊系統的支撐點，並有自我清潔滾動球體腳輪，作用是第三個支撐點來支撐機器人，並且可使移動軌跡較為順暢；而左、右二輪為整台機器人移動的動力來源，可獨立控制左右輪速度，作業上不會互相影響，利用左右輪的速度差可讓機器人直線前進、左轉彎、右轉彎等移動方式，如表(3-1)所示。

表 3-1 直線前進與轉彎比較表

左右輪速度相同，即直線前進	右輪速度大於左輪，即左轉彎 左輪速度大於右輪，即右轉彎
	

機器人移動控制可分為以下五種：

- 直線前進：當機器人右輪正轉，左輪反轉，且左右輪速度相同時，即直線前進；反之，當機器人右輪反轉，左輪正轉，且左右輪速度相同時，即直線後退。
- 原地左轉：當機器人左右輪正轉，且左右輪速度相同時，即原地左轉。
- 原地右轉：當機器人左右輪反轉，且左右輪速度相同時，即原地右轉。
- 左轉彎：當機器人右輪正轉，左輪反轉，且右輪速度大於左輪速度時，即左轉彎。
- 右轉彎：當機器人右輪正轉，左輪反轉，且左輪速度大於右輪速度時，即右轉彎。

3.2 座標系統

在設計機器人控制系統之前必須先設定好座標位置，本論文使用 Kinect 深度感測器取得被追蹤者的三維位置，藉由 OpenNI 組織所釋放出的 API 撰寫程式，在此用到的是追蹤人體骨架的 API，並且抓取被追蹤者的重心位置資料回傳給機器人端，座標原點設定為 Kinect 感測器擺放位置，即 Kinect 感測器本身，而回傳的座標位置(x,y,z)即為機器人與被追蹤者的相對位置，如圖(3-1)所示，在此座標系統並非一般所見的右手法則，而是左手法則，原因為 OpenNI 組織釋放出的 API 本身設計的關係，在此沿用了原來的設計。

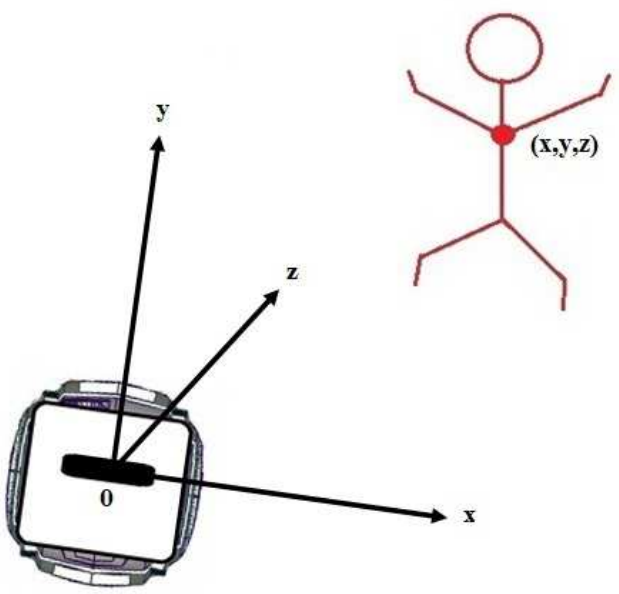


圖 3-1 座標位置示意圖

3.3 追蹤控制系統

機器人能保持固定距離的追蹤人是很困難的，因為人在走路時無法保持固定速度前進，走路的速度難免會忽快忽慢，為了使機器人能與被追蹤者保持固定距離，本論文設計了模糊控制器來實現目標追蹤的控制系統，以下將介紹模糊理論與模糊控制原理，進一步說明本模糊控制器設計步驟。

3.3.1 模糊理論

Fuzzy 在字典的解釋為:模糊不清的、朦朧的意思，有人會認為模糊不是好事，讓人感覺不夠嚴謹或者太隨便，但是在人們的日常生活中，模糊卻隨時隨地伴隨著我們，如有人問:「現在幾點鐘?」，通常會回答:「九點半左右。」;又問:「你身高多少?」，通常會回答:「大約 170 公分」，這些回答都是會被人接受，幾乎不

會有人去追究「現在是九點二十八分二十五秒嗎?」或「你的身高是 169.13 公分嗎?」這些精確的答案，那就表示我們可以接受這些模糊概略的回答。

模糊理論是 1965 年由美國加州柏克萊大學 L.A.Zadeh 教授於「資訊與控制」(Information and Control)學術雜誌上發表「模糊集合」(Fuzzy Sets)所提出[29]，其概念為任何的事物不該只有 0 和 1 的分別，而應該是有程度的分別，最早被應用在蒸氣渦輪引擎上[30]，其設計概念是將人類對燃燒煤炭的經驗，轉換為模糊的判斷規則，並以歸屬函數將其歸屬程度量化，最後就可推論出控制量的大小。

◆ 模糊集合

模糊理論是以模糊集合(Fuzzy Sets)為基礎，而模糊集合是由傳統明確集合(Crisp Sets)的推廣[31-33]，傳統的明確集合是屬於二元邏輯，不是 0 就是 1 的特徵函數(Characteristic Function)表示法，即某元素對某集合的關係只有”屬於”與”不屬於”兩種，而模糊集合是屬於多元邏輯，不是只有 0 與 1 的表示，而是 0 與 1 之間歸屬函數(Membership Function)的表示法，歸屬度(Membership Grade)愈接近 1 表示某元素歸屬於某集合的程度愈高或愈大，傳統明確集合與模糊集合的比較如表(3-2)所示。

表 3-2 明確集合與模糊集合比較

Crisp Sets	Fuzzy Sets
使用 0 或 1 的特徵函數	使用 0 到 1 的歸屬函數
強調非此即彼的關係	接受亦此亦彼的關係
只接受精確不模糊的資訊	可接受模糊不精確的資訊
硬性的二分法	軟性的分類法

以下舉簡單的例子說明傳統明確集合與模糊集合的差別，以明確集合定義「中年」年齡，即 30 歲以下不屬於中年年齡，50 歲以上也不屬於中年年齡，只有 30 歲到 50 歲才屬於中年年齡，「中年」定義的明確集合如圖(3-2)所示。

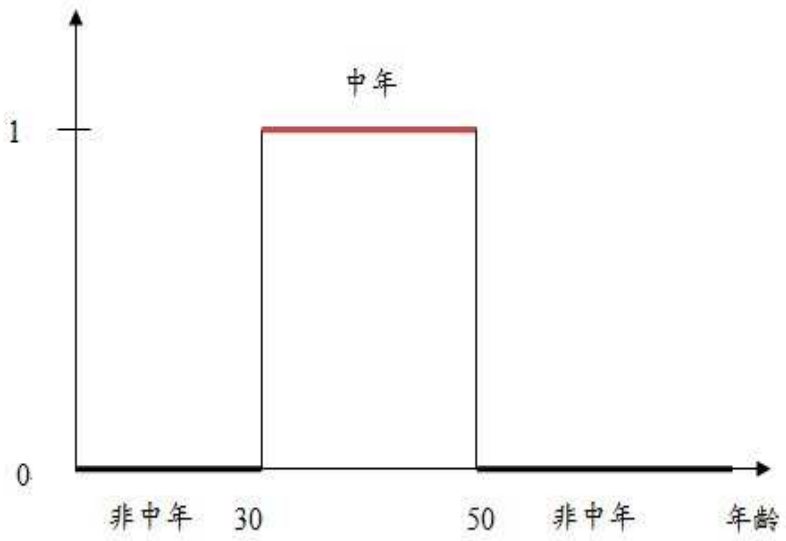


圖 3-2 「中年」定義的明確集合

以模糊集合定義「中年」年齡，其是以歸屬度來判別中年年齡的程度，當年齡為 30 歲時，歸屬中年的程度為 0.5，當年齡為 40 歲時，歸屬中年的程度為 1，當年齡為 55 歲時，歸屬中年的程度為 0.3，如此一來能有較彈性的判別，而非硬性的判別，「中年」定義的模糊集合如圖(3-3)所示。

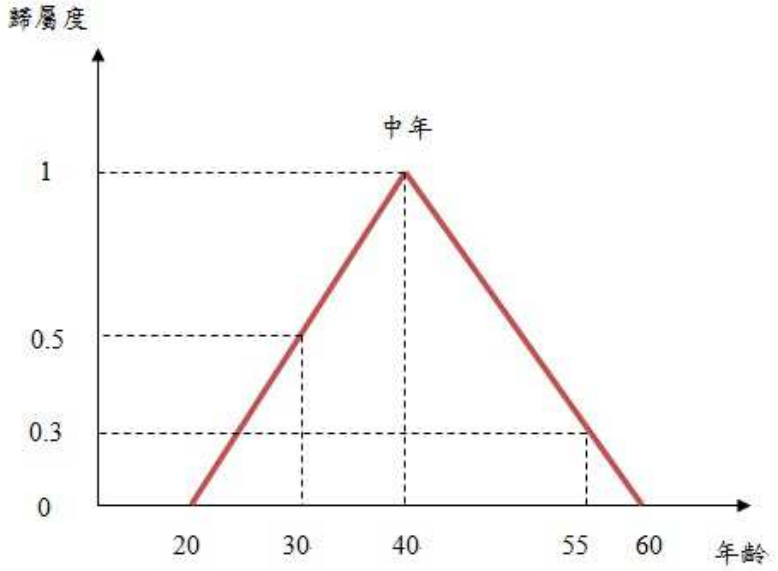


圖 3-3 「中年」定義的模糊集合

◆ 歸屬函數

歸屬函數是模糊理論的基本概念，用來描述模糊集合的性質，透過歸屬函數我們可以對模糊集合進行量化，也才能利用精確的數學方法分析與處理模糊性的資訊，而表示某元素歸屬某模糊集合程度的數值介於 0 到 1 之間，歸屬函數的選擇並無通用的定理，即不具有客觀性，通常是靠著人類對某事物所累積的經驗法則來建立歸屬函數。

歸屬函數可分為數值(Numerical)及函數(Functional)兩種定義方式，數值化定義方式又稱離散化歸屬函數，而函數化定義方式又稱連續化歸屬函數，以下將對兩種定義方式加以說明。

● 離散化歸屬函數(Discrete Membership Function)

直接給定有限模糊集合內每個元素的歸屬度，並以向量形式表達，此向量的大小與論域(Universe of Discourse)離散化的程度有關，如圖(3-4)所示，其特點是簡單明瞭，容易建立模糊關係矩陣，且節省記憶空間及函數換算時間，但是論域離

散化的離散間距往往有以下影響:(1)離散間距太大則論域空間的分割會過於粗略，但是向量維度小，處理起來比較容易；(2)離散間距小則系統較為完整及精確，但向量維度大，處理較耗時間及較需記憶體空間。

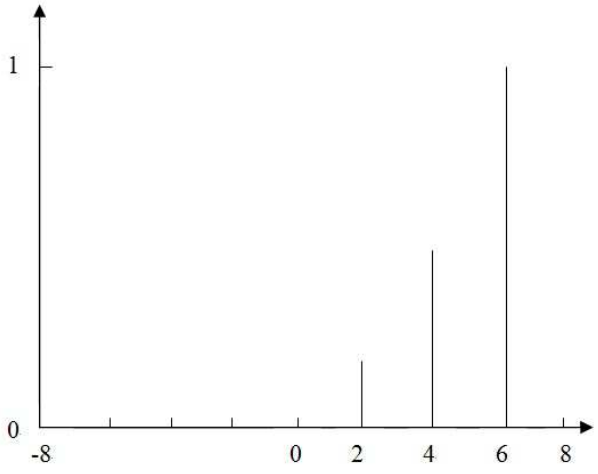


圖 3-4 離散化歸屬函數

- 連續化歸屬函數(Continuous Membership Function)

通常以連續化歸屬函數表示無限元素特性，一般連續化歸屬函數有五種不同形式的函數來描述模糊集合，其為 S 函數、Z 函數、Π函數、三角形與梯形，實際應用在模糊控制時，用三角形與梯形建立歸屬函數就能有滿意的結果，此五種函數表示如式(3.1-3.5)，其圖形如圖(3-5)所示。

(1) S 函數(S Function)為單調遞增型歸屬函數

$$S(x;\alpha,\beta,\gamma)=\begin{cases}0, & x\le\alpha\\ 2\left(\frac{x-\alpha}{\gamma-\alpha}\right)^2, & \alpha\le x\le\beta\\ 1-2\left(\frac{x-\gamma}{\gamma-\alpha}\right)^2, & \beta<x\le\gamma\\ 1, & x\ge\gamma\end{cases} \tag{3.1}$$

(2) Z 函數(Z Function)為單調遞減型歸屬函數

$$Z(x;\alpha,\beta,\gamma)=\begin{cases}1, & x\le\alpha\\ 1-2\left(\frac{x-\alpha}{\gamma-\alpha}\right)^2, & \alpha\le x\le\beta\\ 2\left(\frac{x-\gamma}{\gamma-\alpha}\right)^2, & \beta<x\le\gamma\\ 0, & x\ge\gamma\end{cases} \tag{3.2}$$

(3) Π函數(Pi Function)為 S 函數及 Z 函數的合成，同時具有遞增與遞減性值

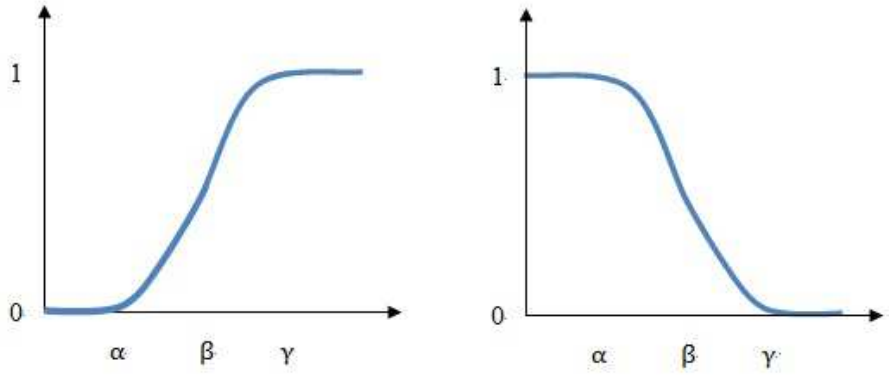
$$\Pi(x;\beta,\gamma)=\begin{cases}S\left(x;\gamma-\beta,\gamma-\frac{\beta}{2},\gamma\right), & x\le\gamma\\ 1-S\left(x;\gamma,\gamma+\frac{\beta}{2},\gamma+\beta\right), & x\ge\gamma\end{cases} \tag{3.3}$$

(4) 三角形函數(Triangular Shape Function)

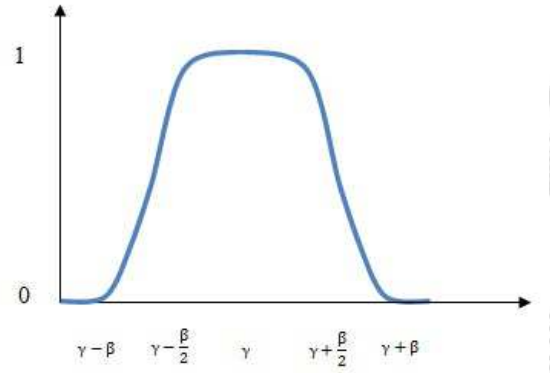
$$\mu(x)=\begin{cases}0, & x<a_1\\ \frac{x-a_1}{a-a_1}, & a_1\le x<a\\ 1, & x=a\\ \frac{b_1-x}{b_1-a}, & a<x\le b_1\\ 0, & x>b_1\end{cases} \tag{3.4}$$

(5) 梯形函數(Trapezoid Shape Function)

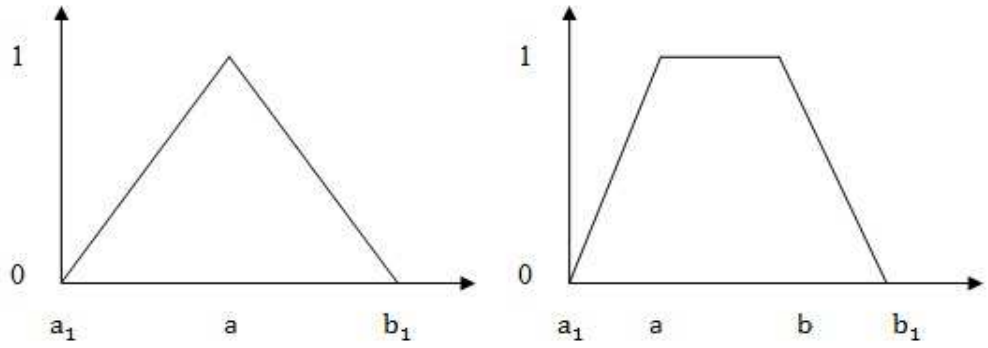
$$\mu(x)=\begin{cases}0, & x<a_1\\ \frac{x-a_1}{a-a_1}, & a_1\le x<a\\ 1, & a\le x\le b\\ \frac{b_1-x}{b_1-b}, & b<x\le b_1\\ 0, & x>b_1\end{cases} \tag{3.5}$$



(a) S 函數　　(b) Z 函數



(c)Π 函數



(d) 三角形函數　　(e) 梯形函數

圖 3-5 連續化歸屬函數

- 單值型歸屬函數(Singleton Membership Function)

單值型歸屬函數為特例，當 x 等於 y 時，歸屬函數 A(x)為 1 其餘為 0，通常用於模糊規則的後件部，如圖(3-6)所示。

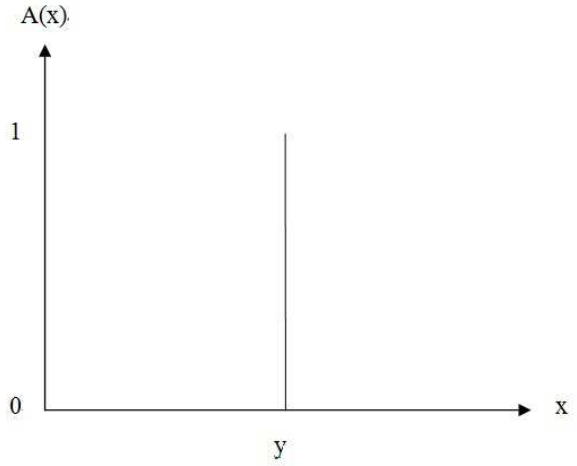


圖 3-6 單值型歸屬函數

3.3.2 模糊控制原理

傳統控制器在設計時，受控體必須先有明確的數學模式，再根據此數學模式來設計控制器，但對於非線性、不確定性及時變性的系統時，僅能達到近似的數學描述，效果就沒預期的好。模糊控制器是由模糊理論所發展出來，擺脫了傳統控制器須透過明確數學模式來建立的局限，典型的範例為倒單擺的直立，Yamakawa[34]成功使用了七條模糊規則讓倒單擺直立起來。

模糊控制器的基本理念為人類的經驗法則，以語言變數(Linguistic Variable) IF-THEN 條件式描述系統的輸入與輸出關係，模擬人類對此受控體的控制經驗或操作行為，IF 部分又稱前件部，THEN 部分又稱後件部，經由模糊推論工廠(Fuzzy Inference Engine)模仿人類推論方式，將條件式語句轉為控制策略，模糊控制系統

基本架構如圖(3-7)所示。

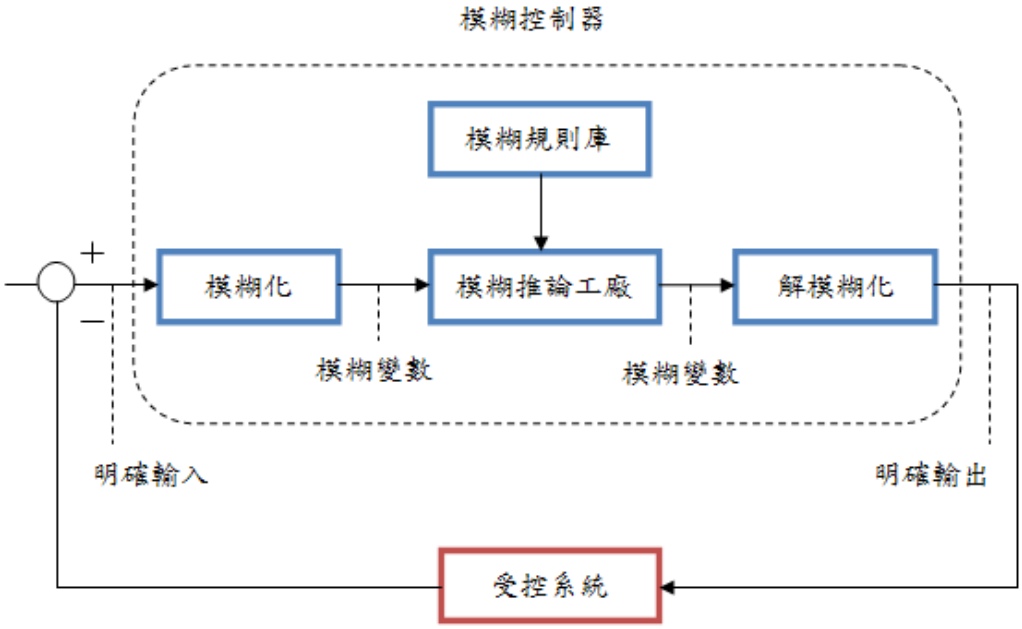


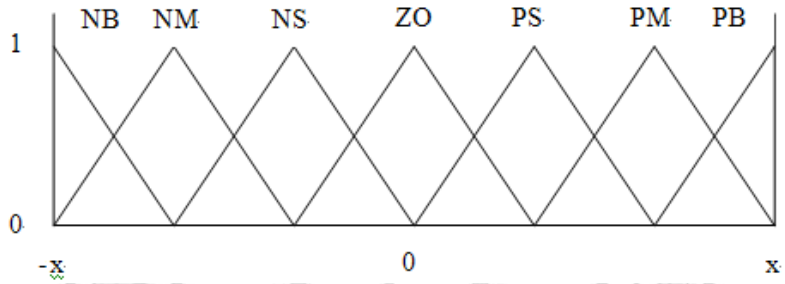
圖 3-7 模糊控制系統基本架構

設計模糊控制器必須先定義好輸入及輸出變數並模糊化,再定義模糊規則庫，經由模糊推論工廠，最後解模糊化，其步驟如下：

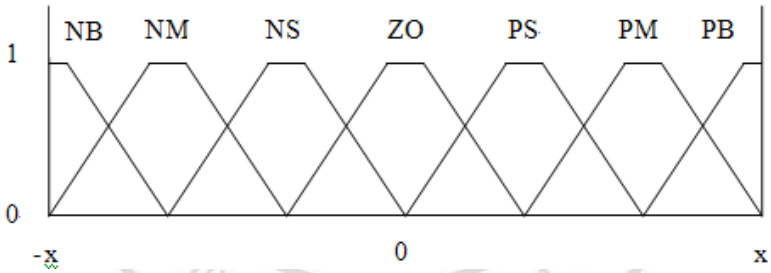
步驟一. 模糊化策略

對受控系統而言感測器所量測到的資訊為明確的數值，而模糊控制器是以條件式規則為控制策略，為了讓量測到的資訊與模糊控制器結合，模糊化是首要的步驟，模糊化是根據輸入變數及輸出變數的變動範圍決定語言變數的論域，通常以誤差量(e)和誤差變化量(ė)作為輸入變數，設計者經由專業知識和經驗，將語言變數論域分成數個模糊集合,論域中模糊集合的標示可定義為:NB(Negative Big)、

NM (Negative Medium)、NS (Negative Small)、 ZO(Zero)、PS(Positive Small)、PM (Positive Medium)、PB (Positive Big)，此標示可自行定義，而這些模糊集合採用的歸屬函數有 S 函數、Z 函數、Π函數、三角形函數、梯形函數五種，通常以三角形歸屬函數及梯形歸屬函數較常使用，如圖(3-8)所示。



(a) 三角形歸屬函數



(b) 梯形歸屬函數

圖 3-8 歸屬函數圖形

步驟二. 設計模糊規則庫

模糊規則庫代表整個模糊控制系統的思考法則，由 IF-THEN 的條件式法則組合而成，以描述系統的輸入及輸出的關係，以雙輸入單輸出系統為例，其模糊法則表示如下：

R_k：If e is A_i and ė is B_j Then y is C_{ij}

i=1,…,m　j=1,…,n　k=(i-1)*n+j

其中R_k代表第 k 條法則，A_i、B_j及C_{ij}代表語言變數的歸屬函數，假設 e 模糊集合的語言變數有 m 個，ė模糊集合的語言變數有 n 個，則模糊法則有 m*n 條法則，e及ė代表輸入變數，y 代表輸出變數。

若將輸入變數分別放置陣列之垂直與水平方向，則可得模糊規則庫如表(3-3)所示，模糊規則庫建立的方法有二種：

- 經由控制系統學習修正控制
- 直接轉換操作員的操作技巧與經驗

表 3-3 標準模糊規則庫

		ė						
		NB	**NM**	**NS**	**ZO**	**PS**	**PM**	**PB**
e	**NB**	NB	NB	NB	NB	NM	NS	ZO
	NM	NB	NB	NM	NS	NS	ZO	PS
	NS	NB	NM	NS	NS	ZO	PS	PM
	ZO	NM	NM	NS	ZO	PS	PM	PM
	PS	NM	NS	ZO	PS	PS	PM	PB
	PM	NS	ZO	PS	PM	PM	PB	PB
	PB	ZO	PS	PM	PB	PB	PB	PB

步驟三. 擬定模糊推論工廠

設輸入變數為e及ė，輸出變數為 y，常見之模糊推論工廠為乘積推論及最小推

論，若有 R_1及R_2二條規則，如下：

R_1：If e is A_1 and ė is B_1 Then y is C_{11}

R_2：If e is A_2 and ė is B_2 Then y is C_{22}

(1) 乘積推論工廠(Product Inference Engine)

α_1為$A_1(e)$與$B_1(\dot{e})$的乘積，α_2為$A_2(e)$與$B_2(\dot{e})$的乘積，即$\alpha_1 = A_1(e) \cdot B_1(\dot{e})$，$\alpha_2 = A_2(e) \cdot B_2(\dot{e})$，如圖(3-9)所示。

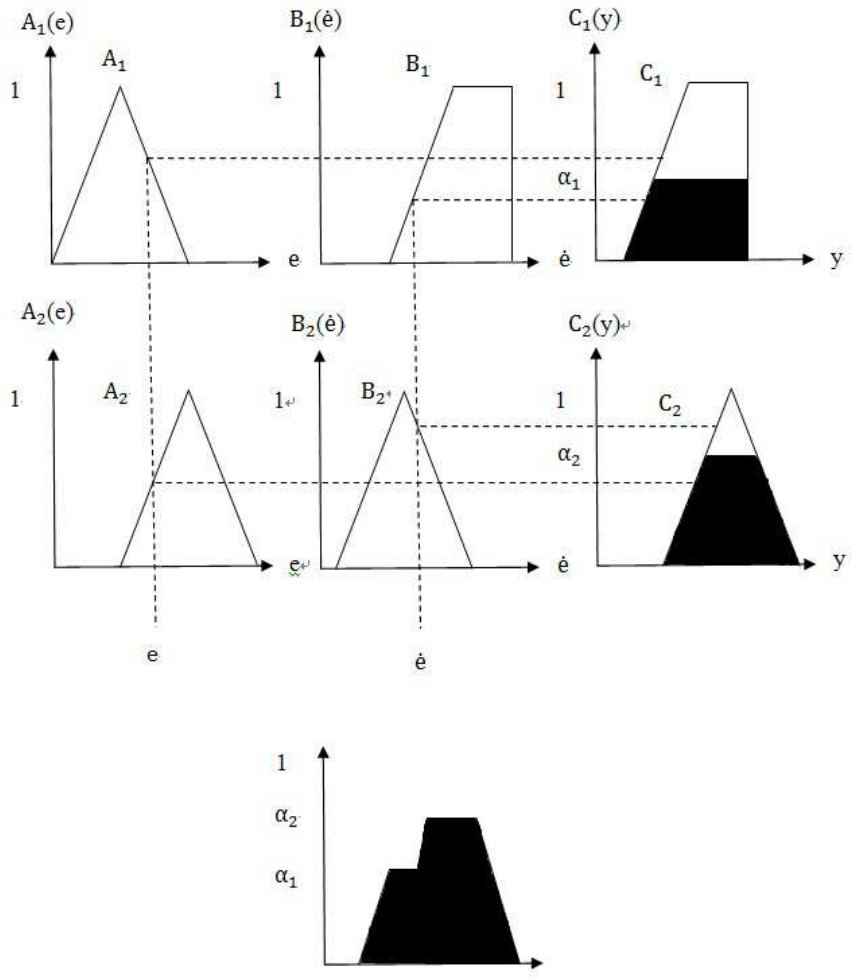


圖 3-9 乘積推論法

其中$A_1(e)$為輸入變數 e 在歸屬函數A_1的歸屬度，$B_1(\dot{e})$為輸入變數$\dot{e}$在歸屬函數B_1的歸屬度， C_1為輸出變數歸屬函數

(2) 最小值推論工場(Minimum Inference Engine)

α_1為$A_1(e)$與$B_1(\dot{e})$的最小值，α_2為$A_2(e)$與$B_2(\dot{e})$的最小值，$\alpha_1 = \min[A_1(e), B_1(\dot{e})]$，$\alpha_2 = \min[A_2(e), B_2(\dot{e})]$，如圖(3-10)所示。

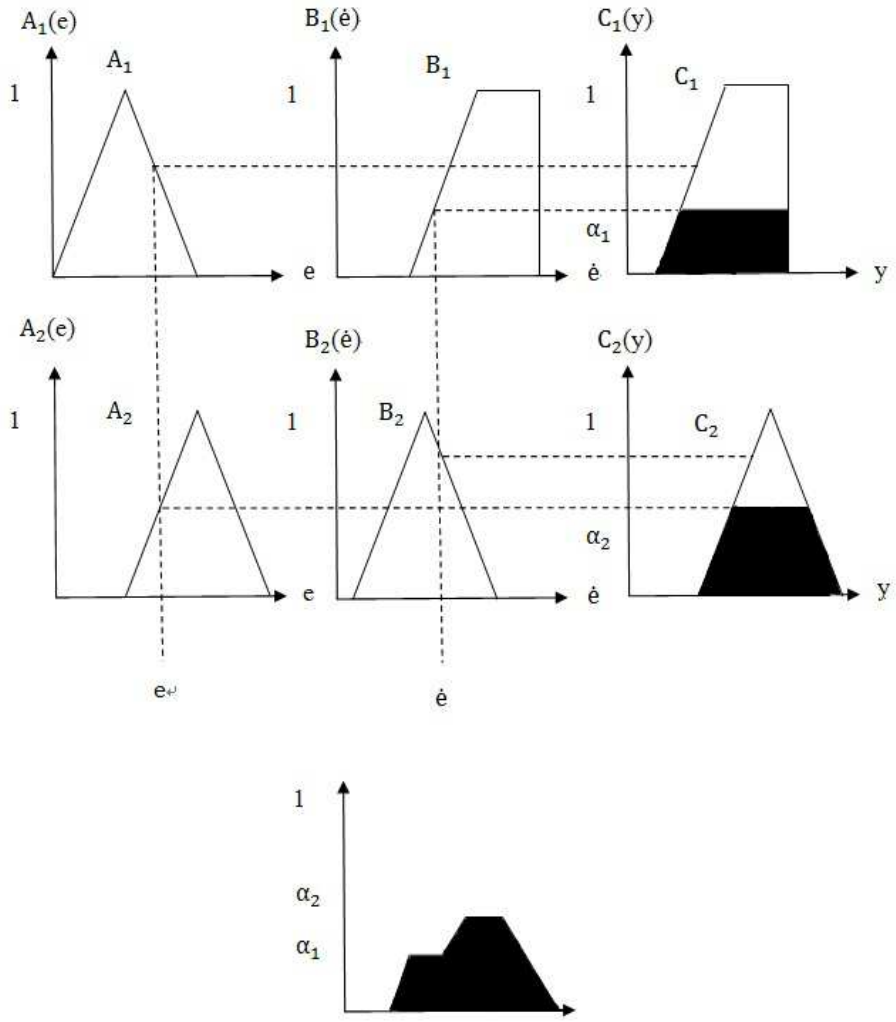


圖 3-10 最小值推論法

步驟四. 選擇解模糊化方法

解模糊化剛好跟模糊化相反，是將模糊推論工廠推論出的歸屬函數量化為明確的輸出變數，以便實際應用於系統之操作，設輸出變數歸屬函數 C 的論域範圍為 a 到 b，輸出變數為 y，較常見解模糊化方法為以下幾種：

(1) 重心解模糊化法(Center of Gravity Defuzzification, CGD)

$$y_{CGD} = \frac{\sum_{i=1}^{n} y_i c_i(y_i)}{\sum_{i=1}^{n} c_i(y_i)} \qquad (3.6)$$

(2) 面積和之中心解模糊化法(Center of Sum Defuzzification, CSD)

$$y_{CSD} = \frac{\sum_{i=1}^{n} \int_a^b y c_i(y) dy}{\sum_{i=1}^{n} \int_a^b c_i(y) dy} \qquad (3.7)$$

(3) 中心平均值解模糊化法(Center Average Defuzzification, CAD)又稱高度解模糊化法(Height Defuzzification)，$h(C_i)$表示每個C_i之高度，p_i表示C_i未經矮化前中心點

$$y_{CAD} = \frac{\sum_{i=1}^{n} p_i h(c_i)}{\sum_{i=1}^{n} h(c_i)} \qquad (3.8)$$

解模糊化的結果要滿足以下三大準則：

- 合理性：在人類的直覺上感覺輸出變數是合理的
- 計算簡單：是為了在控制使用上的方便
- 連續性：模糊集合形狀有些許變化，輸出變數變化不會太大

3.3.3 模糊控制器設計

- 定義輸入與輸出變數

本模糊控制器設計為兩輸入一輸出系統，輸入變數分別為距離誤差(e_d)與距離誤差變化量(Δe_d)分別如式(3.9)及式(3.10)所示。

$$e_d(n) = d(n) - d_f(n) \tag{3.9}$$

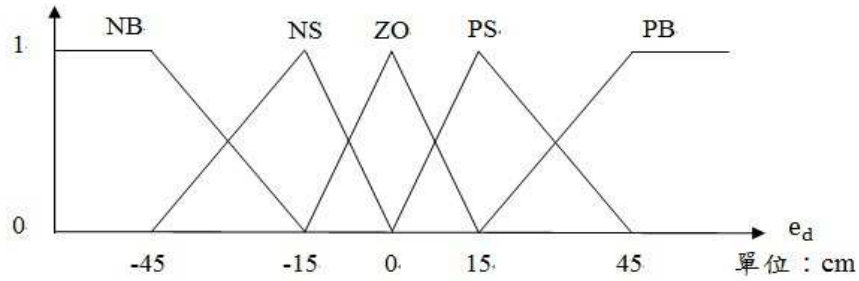
$$\Delta e_d = e_d(n) - e_d(n-1) \tag{3.10}$$

其中 d 為機器人與被追蹤者相對距離，d_f為機器人與被追蹤者欲保持之固定距離，n 為當下感測器回傳第 n 筆資料，n-1 為感測器回傳第 n-1 筆資料。

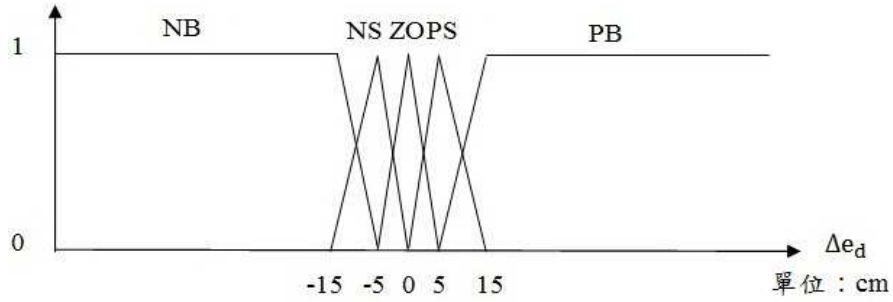
控制器的輸出變數為馬達轉向與轉速，即機器人移動速度(v)，其範圍為-10 到 10 之間，單位為 cm/s。

- 選擇歸屬函數

輸入變數與輸出變數定義後，必須將輸入與輸出變數模糊化，本論文輸入變數e_d及Δe_d選用三角形歸屬函數，將其論域分成五種不同程度的語言變數，分別是負大(NB)、負小(NS)、零(ZO)，正小(PS)及正大(PB)，如圖(3-11)所示。



(a) e_d歸屬函數



(b) Δe_d歸屬函數

圖 3-11 輸入變數歸屬函數

輸出變數選用單值歸屬函數，將其論域分成五種不同程度的語言變數，分別是分別是負大(NB)、負小(NS)、零(ZO)，正小(PS)及正大(PB)，如圖(3-12)所示。

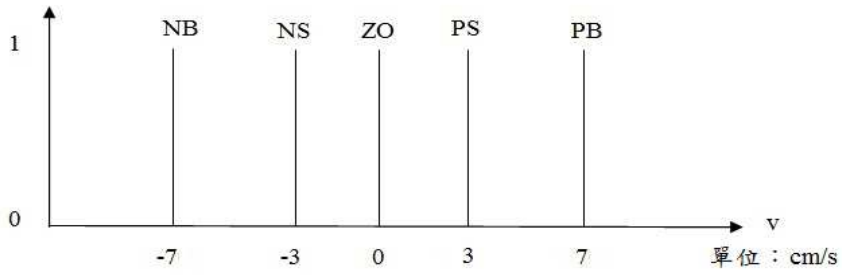


圖 3-12 輸出變數歸屬函數

- 建立模糊規則庫

規則庫的建立是將模糊化後的歸屬函數以 IF-THEN 的條件式法則表示，以描述輸入與輸出變數的對應關係，本論文追蹤控制系統之模糊規則庫如表(3-4)所示。

表 3-4 追蹤控制系統模糊規則庫

		e_d				
		NB	**NS**	**ZO**	**PS**	**PB**
Δe_d	**NB**	NB	NB	NB	NS	ZO
	NS	NB	NS	NS	ZO	PS
	ZO	NB	NS	ZO	PS	PB
	PS	NS	ZO	PS	PS	PB
	PB	ZO	PS	PB	PB	PB

本追蹤控制系統兩輸入變數皆為五種程度語言變數，其模糊規則共有$5^2 = 25$條規則，在此舉其中一條規則做說明。

If e_d is PB and Δe_d is PB Then v is PB

假設距離誤差(e_d)為正大(PB)，且距離誤差變化量(Δe_d)為正大(PB)，則機器人移動速度(v)為正大(PB)。表示當機器人與被追蹤者相對距離很遠，且相對距離越來越遠的情況下，負責控制機器人移動的馬達輸出為正轉且速度快，使機器人速度變快，距離方能保持在欲保持的相對距離。其 25 條模糊規則如下：

R_1: If e_d is NB and Δe_d is NB Then v is NB

R_2: If e_d is NB and Δe_d is NS Then v is NB

R_3: If e_d is NB and Δe_d is ZO Then v is NB

R_4: If e_d is NB and Δe_d is PS Then v is NS

R_5: If e_d is NB and Δe_d is PB Then v is ZO

R_6: If e_d is NS and Δe_d is NB Then v is NB

R_7: If e_d is NS and Δe_d is NS Then v is NS

R_8: If e_d is NS and Δe_d is ZO Then v is NS

R_9: If e_d is NS and Δe_d is PS Then v is ZO

R_{10}: If e_d is NS and Δe_d is PB Then v is PS

R_{11}: If e_d is ZO and Δe_d is NB Then v is NB

R_{12}: If e_d is ZO and Δe_d is NS Then v is NS

R_{13}: If e_d is ZO and Δe_d is ZO Then v is ZO

R_{14}: If e_d is ZO and Δe_d is PS Then v is PS

R_{15}: If e_d is ZO and Δe_d is PB Then v is PB

R_{16}: If e_d is PS and Δe_d is NB Then v is NS

R_{17}: If e_d is PS and Δe_d is NS Then v is ZO

R_{18}: If e_d is PS and Δe_d is ZO Then v is PS

R_{19}: If e_d is PS and Δe_d is PS Then v is PS

R_{20}: If e_d is PS and Δe_d is PB Then v is PB

R_{21}: If e_d is PB and Δe_d is NB Then v is ZO

R_{22}: If e_d is PB and Δe_d is NS Then v is PS

R_{23}: If e_d is PB and Δe_d is ZO Then v is PB

R_{24}: If e_d is PB and Δe_d is PS Then v is PB

R_{25}: If e_d is PB and Δe_d is PB Then v is PB

- 模糊推論與解模糊化

模糊規則庫建立完成後，須經過模糊推論工廠進行模糊推論，本論文使用最小值推論工廠進行模糊推論，再利用重心解模糊化法(CGD)將模糊推論出的歸屬函數進行解模糊化，求得輸出值v_{CGD}，如式(3.11)所示。

$$V_{CGD} = \frac{\sum_{i=1}^{25} v_i B_i(e_d, \Delta e_d)}{\sum_{i=1}^{25} B_i(e_d, \Delta e_d)} \quad (3.11)$$

其中v_i為輸出變數的單值；$B_i(e_d, \Delta e_d)$為觸發歸屬函數之歸屬程度。

3.4 方向導正系統

為了使機器人能正確的朝向被追蹤者方向前進，本研究設計了方向導正系統，當被追蹤者行進方向改變時，機器人能跟著做方向導正的動作，使機器人能保持在被追蹤者的後方，由於一般人在行進中身體難免會有小幅度的左右晃動，若機器人也跟著做修正會導致不必要的晃動，在此設計方向導正門檻值條件，作為機器人是否要修正方向的依據。

假設x_1為原始座標，x_2為方向偏移後座標，d_1為機器人與被追蹤者原相對距離，d_2為偏移後機器人與被追蹤者相對距離，θ為機器人將轉動的角度，由三角關係可得出式(3.12)與式(3.13)，其三角關係圖如圖(3-13)所示。

$$d_2 = \sqrt{{d_1}^2 + (x_2 - x_1)^2} \tag{3.12}$$

$$\theta = cos^{-1}\frac{d_1}{d_2} \tag{3.13}$$

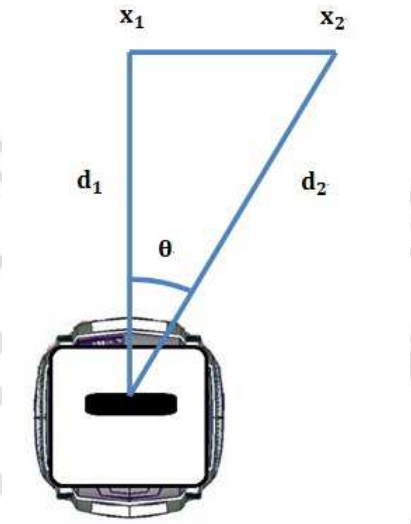


圖 3-13 方向偏移三角關係圖

3.5 程式控制流程

本研究機器人控制由二個主程式所構成，一個是控制 Kinect 感測器感測被追蹤者三維位置，並透過 TCP/IP 通訊協定，將三維資訊回傳給第二個主程式，此主程式接收到資訊後經過計算決定機器人動向，機器人控制程式流程圖如圖(3-14)所示。

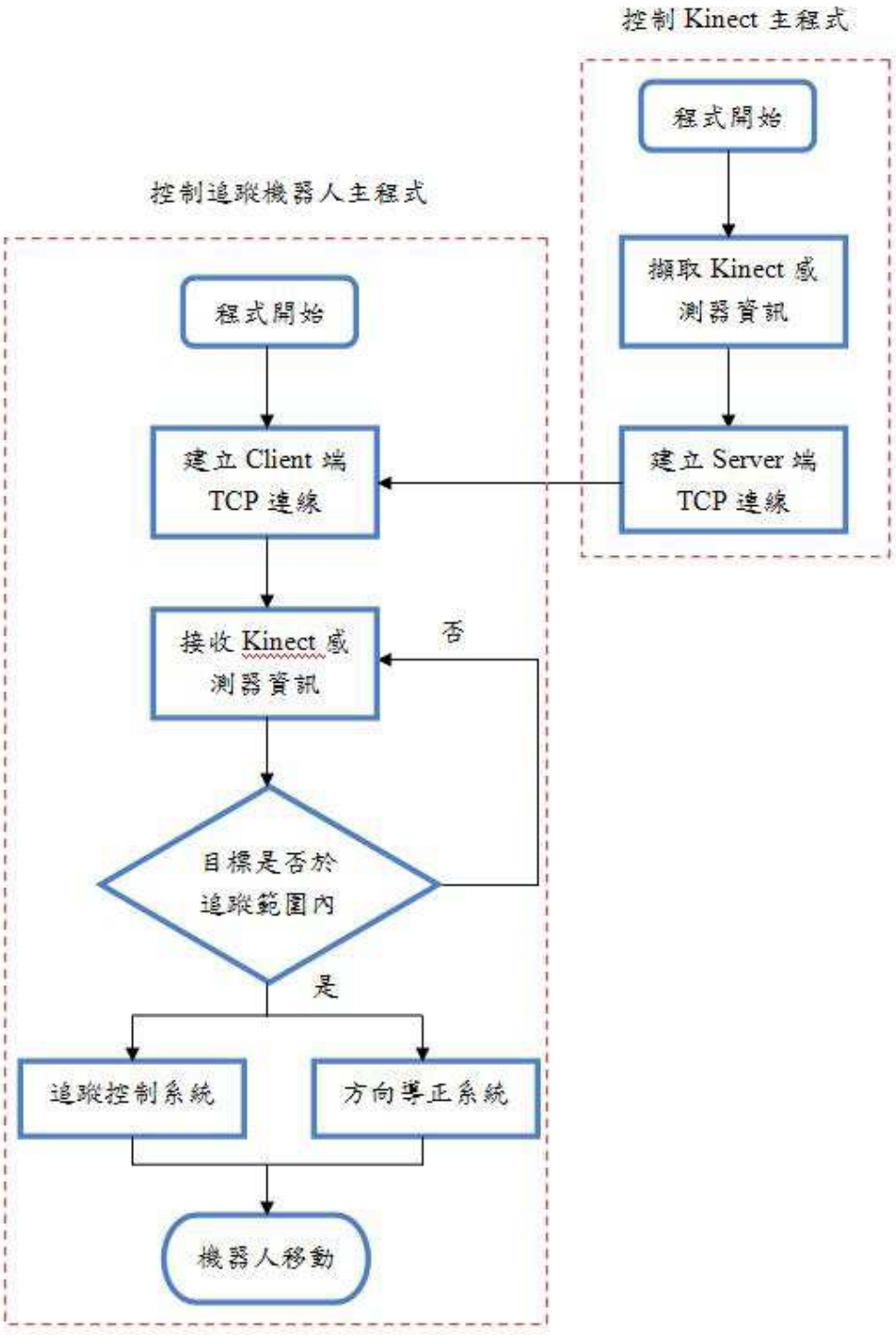


圖 3-14 機器人控制程式流程圖

第四章　實驗結果與測試

4.1 Matlab 軟體模擬

本論文使用 Matlab 軟體的 Fuzzy Logic Toolbox 進行模糊控制器設計與模擬，其輸入變數為 e 及 de，歸屬函數選用三角形歸屬函數，輸入皆為五種程度語言變數，其模糊規則共有$5^2 = 25$條規則，輸出變數為 v，歸屬函數選用單值型歸屬函數，模糊推論選用最小值模糊推論法，解模糊化法選擇重心法，計算出的結果如圖(4-1)所示，有顏色區域表示該輸入有觸發到的幾條規則與其歸屬程度，以此圖為例，e=0 觸發到第 11、12、13、14 及 15 條規則，歸屬程度為 1，de=0 觸發到第 3、8、13、18 及 23 條規則，歸屬程度為 1，當二個輸入皆有觸發到的規則才會進行模糊推論動作，即第 13 條，再由重心法解模糊化求出輸出 v=0。

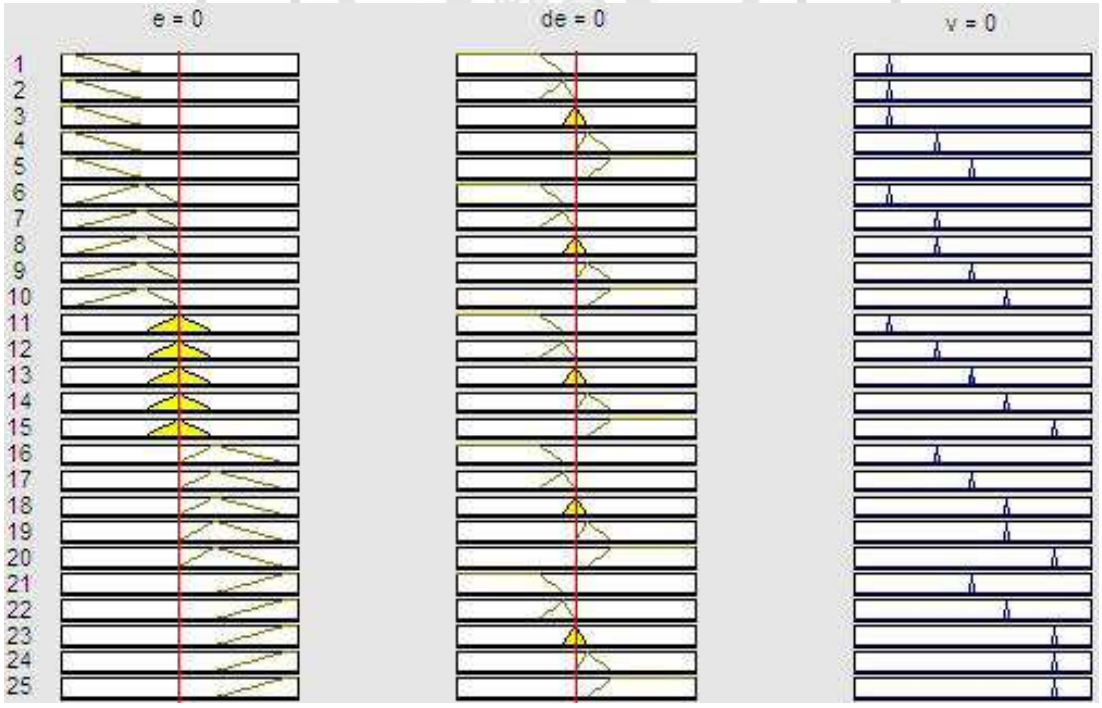


圖 4-1 模擬結果

當輸入改變，例：e =7.5，de=2.5，則輸出隨即改變為 v=1.5，如圖(4-2)所示，其輸入與輸出關係圖如圖(4-3)所示。

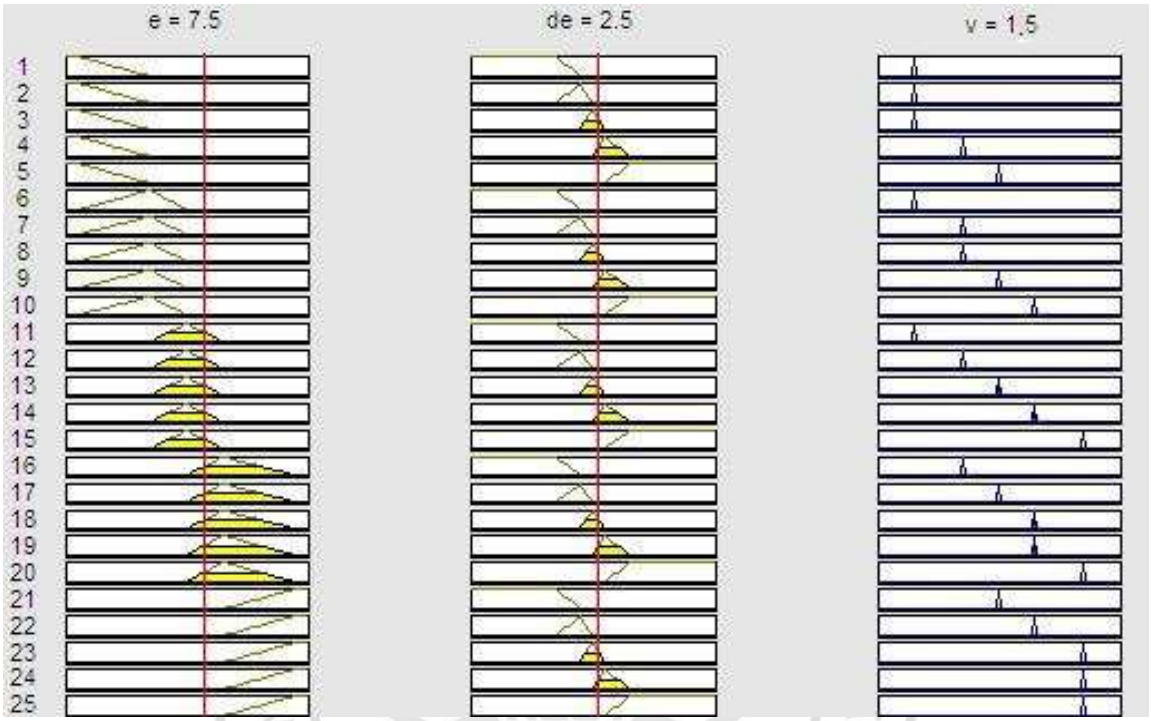


圖 4-2 輸入改變後結果

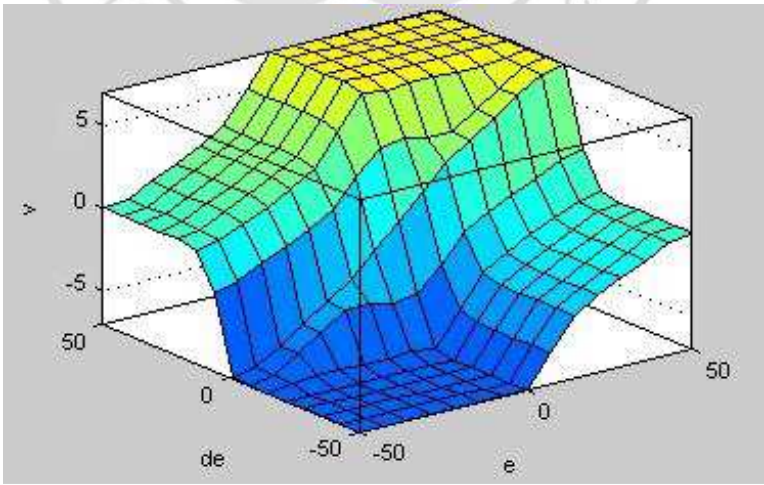


圖 4-3 輸入與輸出關係圖

將多組輸入(e)(de)代入模糊控制器模擬，得出以下結果，如表(4-1)所示，由此表可看出輸入與輸出關係符合預其結果。

表 4-1 多組輸出輸入結果

第 N 次	e	de	V	第 N 次	e	de	v
1	45	15	7	27	22.5	-15	-2.25
2	45	7.5	7	28	15	15	7
3	45	5	7	29	15	7.5	4
4	45	2.5	7	30	15	5	3
5	45	0	7	31	15	2.5	3
6	45	-2.5	5	32	15	0	3
7	45	-5	3	33	15	-2.5	1.5
8	45	-7.5	2.25	34	15	-5	0
9	45	-15	0	35	15	-7.5	-0.75
10	30	15	7	36	15	-15	-3
11	30	7.5	5	37	7.5	15	7
12	30	5	5	38	7.5	7.5	4.33
13	30	2.5	5	39	7.5	5	3
14	30	0	5	40	7.5	2.5	1.5
15	30	-2.5	3.33	41	7.5	0	1.5
16	30	-5	1.5	42	7.5	-2.5	0
17	30	-7.5	0.6	43	7.5	-5	-1.5
18	30	-15	-1.5	44	7.5	-7.5	-2.6
19	22.5	15	7	45	7.5	-15	-5.
20	22.5	7.5	4	46	0	15	7
21	22.5	5	4	47	0	7.5	4
22	22.5	2.5	4.33	48	0	5	3
23	22.5	0	4	49	0	2.5	1.5
24	22.5	-2.5	2.6	50	0	0	0
25	22.5	-5	0.75	51	0	-2.5	-1.5
26	22.5	-7.5	0	52	0	-5	-3

53	0	-7.5	-4	77	-22.5	0	-4
54	0	-15	-7	78	-22.5	-2.5	-4.33
55	-7.5	15	5	79	-22.5	-5	-4
56	-7.5	7.5	2.6	80	-22.5	-7.5	-4
57	-7.5	5	1.5	81	-22.5	-15	-7
58	-7.5	2.5	0	82	-30	15	1.5
59	-7.5	0	-1.5	83	-30	7.5	-0.6
60	-7.5	-2.5	-1.5	84	-30	5	-1.5
61	-7.5	-5	-3	85	-30	2.5	-3.33
62	-7.5	-7.5	-4.33	86	-30	0	-5
63	-7.5	-15	-7	87	-30	-2.5	-5
64	-15	15	3	88	-30	-5	-5
65	-15	7.5	0.75	89	-30	-7.5	-5
66	-15	5	0	90	-30	-15	-7
67	-15	2.5	-1.5	91	-45	15	0
68	-15	0	-3	92	-45	7.5	-2.25
69	-15	-2.5	-3	93	-45	5	-3
70	-15	-5	-3	94	-45	2.5	-5
71	-15	-7.5	-4	95	-45	0	-7
72	-15	-15	-7	96	-45	-2.5	-7
73	-22.5	15	2.25	97	-45	-5	-7
74	-22.5	7.5	0	98	-45	-7.5	-7
75	-22.5	5	-0.75	99	-45	-15	-7
76	-22.5	2.5	-2.6	100			

4.2 Kinect 感測器追蹤策略

為了讓 Kinect 感測器辨別特定目標，而不是有人進入感測範圍就追蹤，所以本研究設定了啟動感測器追蹤的手勢，當未擺出啟動手勢時，感測器並不會追蹤此人，畫面上顯示此人的顏色為單一色，如圖(4-4)所示，且感測器回傳的資訊都為零，如圖(4-5)所示。

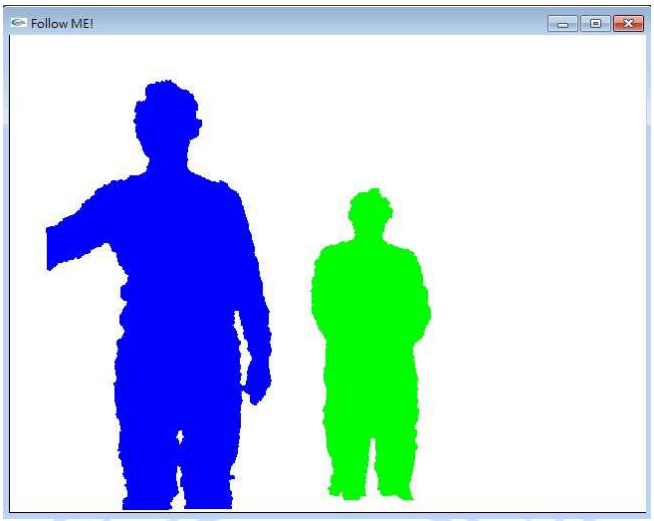


圖 4-4 未啟動追蹤前畫面

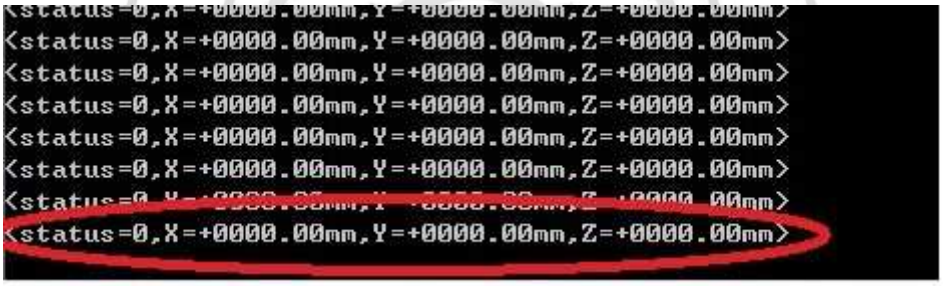


圖 4-5 未啟動追蹤前回傳資訊

當此人擺出啟動感測器追蹤的手勢時，該人在畫面上顯示即變為彩色，如圖(4-6)所示，表示該人為感測器目前所追蹤的特定目標，且感測器回傳資訊將會是該人目前與感測器相對應的三維位置，如圖(4-7)所示，此實驗由二個研究人員進行測試，清楚的看出感測器目前所追蹤的目標為右手邊這位研究人員，且當感測器正在追蹤某人時，其他人也擺出此手勢是不會有反應的，除非先前感測器追蹤的人再擺一次手勢解除追蹤。

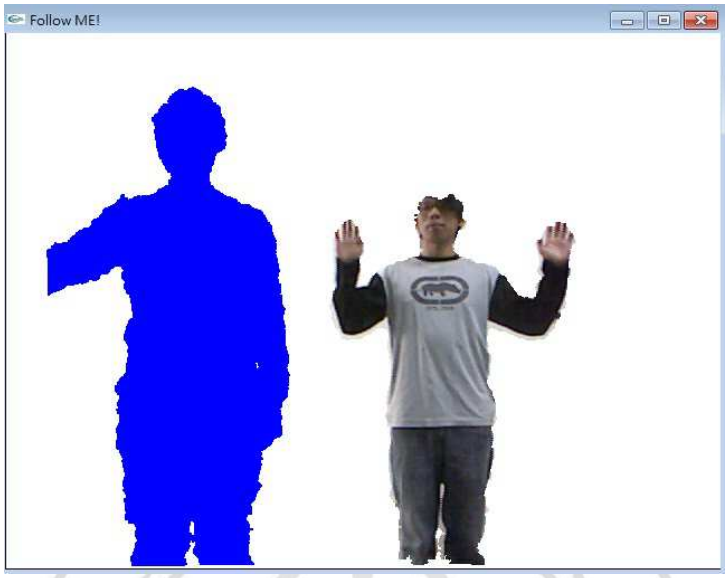


圖 4-6 啟動追蹤後畫面

圖 4-7 啟動追蹤後回傳資訊

4.3 追蹤控制系統測試

當感測器成功的啟動追蹤目標時，會將此人三維位置資訊回傳給機器人端主程式，機器人端主程式接收到資訊後並且經過計算後，判別 z 軸座標是否於安全距離內，即啟動左右輪馬達追蹤此人，追蹤控制系統程式流程圖如圖(4-8)所示，

追蹤控制系統測試如圖(4-9)所示。

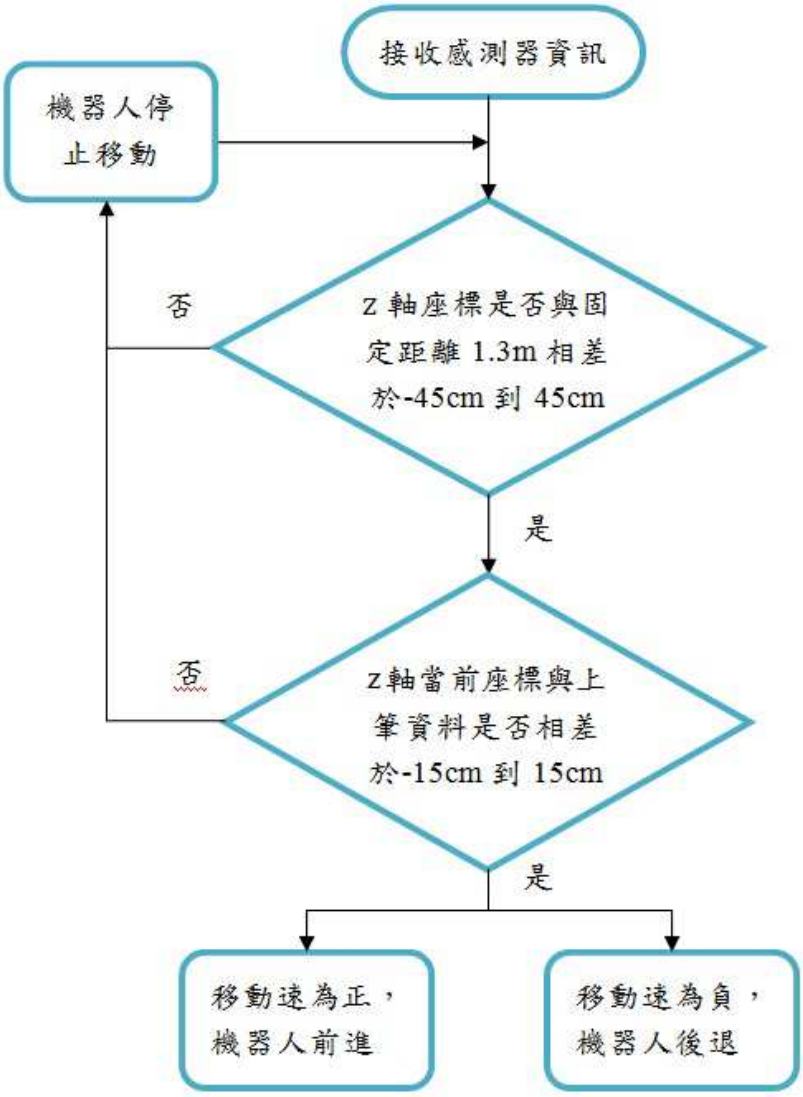


圖 4-8 追蹤控制系統程式流程圖

圖 4-9 追蹤控制系統測試

4.4 方向導正系統測試

本研究設計了方向導正系統,確保機器人前進方向與被追蹤者前進方向相同，當被追蹤者往左偏移時，機器人隨即左轉，當被追蹤者往右偏移時，機器人隨即右轉，方向導正系統程式流程圖如圖(4-10)所示，方向導正系統測試如圖(4-11)所示，

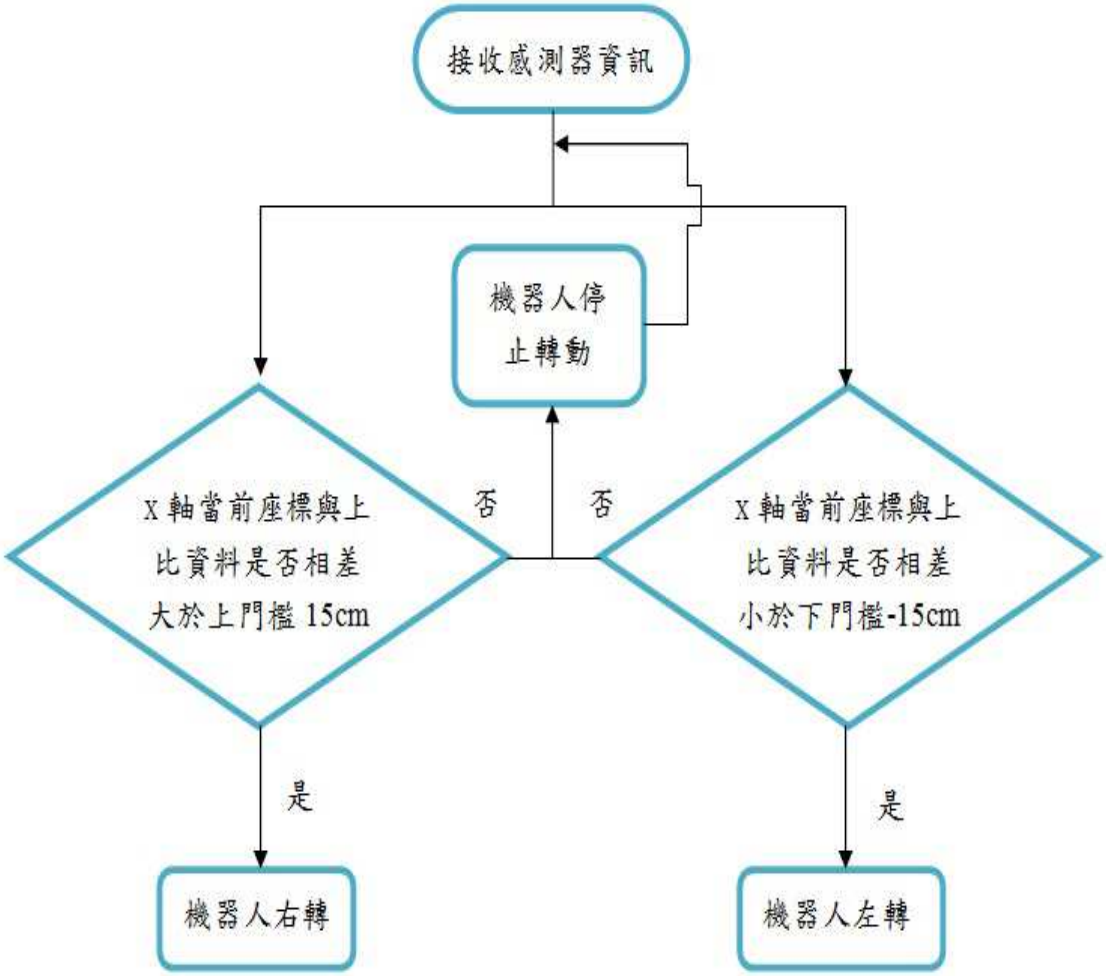


圖 4-10 方向導正系統程式流程圖

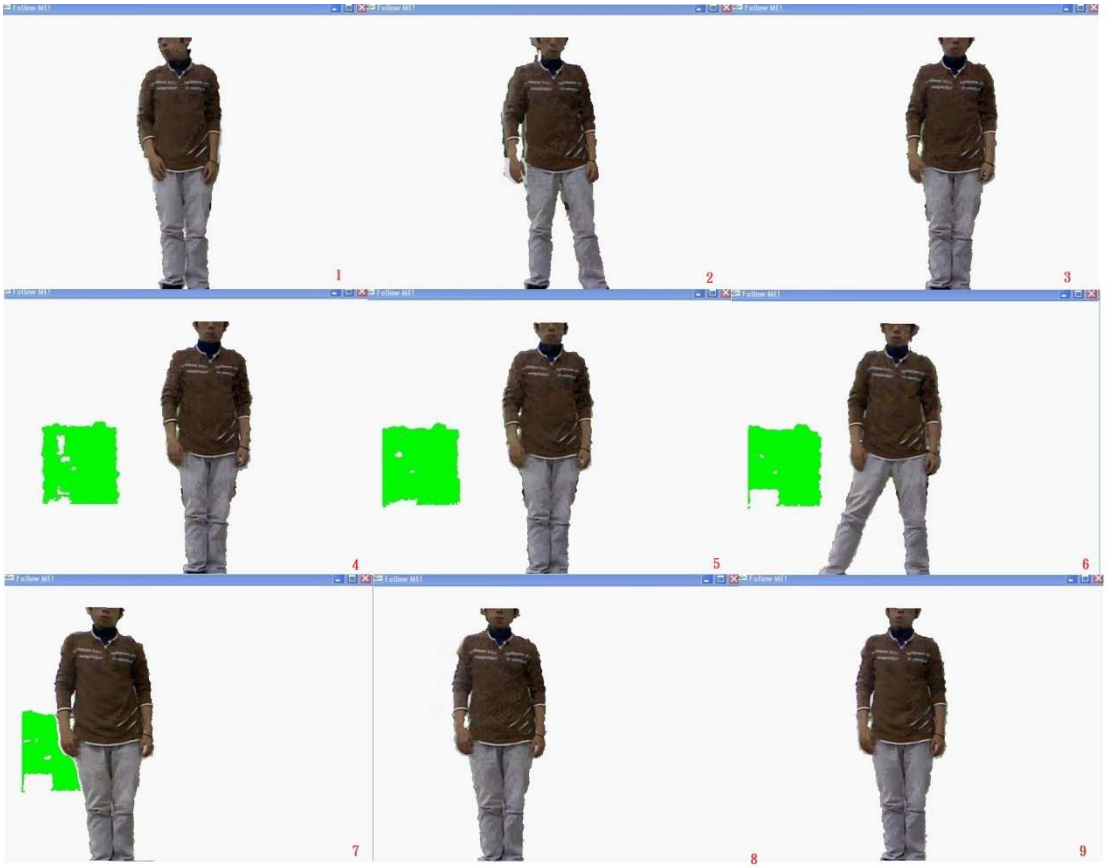


圖 4-11 方向導正系統測試

由於一般人在行進中身體難免會有小幅度的左右晃動，若機器人也跟著做修正會導致移動上不必要的晃動，而人走一步的距離約 15cm，在此設計為方向導正門檻值條件，作為機器人是否要修正方向的依據，當$x_1 - x_2$的值低於下門檻時，機器人即左轉，反之，當高於上門檻時，機器人即右轉，其$x_1 - x_2$資訊如圖(4-10)所示。

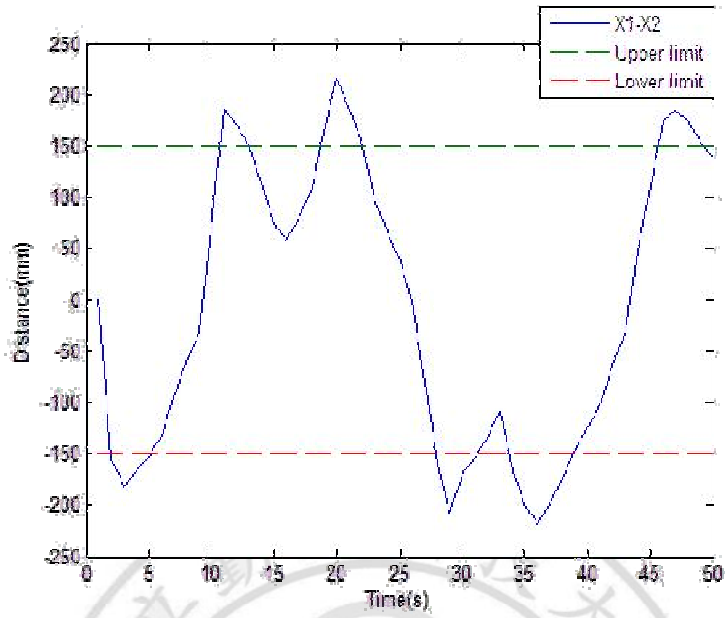


圖 4-12　$x_1 - x_2$ 資訊

第五章　結論與未來展望

5.1 結論

本論文透過較新的技術，使追蹤機器人的研究領域上有更進一步的發展，並透過實驗測試歸納出以下幾點結論：

1. 僅使用一部 Kinect 感測器實現了目標三維位置的追蹤，打破以往研究需使用到二台攝影機的限制。
2. 設計一啟動感測器追蹤手勢，使感測器能正確的辨識此目標是否為追蹤目標，讓錯誤追蹤的機率大大減少。
3. 透過 TCP/IP 通訊協定成功的將三維資訊回傳給機器人端電腦，使 Kinect 感測器與 914 PC-Bot 機器人結合，並且透過追蹤控制系統與方向導正系統，使機器人能夠保持固定距離與方向追蹤，達到追蹤機器人之目的。

本研究已達到初步追蹤機器人的功能，隨著科技的進步，服務型機器人愈來愈普及，相信不久的將來類似的技術將會被廣泛應用，如果此技術能夠應用在機場、大賣場、美術館等，範圍較大且需要提供服務的場合，想必能提高我們的生活品質。

5.2 未來展望

本研究雖然已達到追蹤機器人的目的，若要實際應用於生活上仍有些地方需要克服，故本論文針對未來可改進的方向提出以下幾點建議：

1. 控制策略方面，本論文中並未考慮到機器人避障問題，未來可加入機器人避障系統，避免追蹤過程中障礙物的干擾，其他控制策略部分，未來

可再加入更多的控制法則，提升機器人整體的靈活度。

2. 追蹤目標方面，目前機器人追蹤目標並無給定特定對象，只要 Kinect 感測器當時並無追蹤目標，有人在感測器感測範圍內擺出啟動追蹤手勢，機器人將會對那個人進行追蹤，即每個能擺出此手勢的人都可使機器人對他進行追蹤，如未來需要給定特定對象追蹤，可考慮加上 RFID 系統進行目標的辨認，可使追蹤目標更為明確。

3. 移動速度方面，由於原廠機器人的速度給定在每秒-10 到 10 公分之間，導致移動速度上稍嫌不足，未來可考慮更換馬達或其他方法，使機器人速度增加，達到更有效的追蹤效果。

參考文獻

1. The Planetary Society, http://www.planetary.org
2. Mars Exploration Rover project, NASA/JPL document NSS ISDC, 2001
3. C. Breazeal, “Sociable Machines: Expressive Social Exchange Between Humans and Robots”, Sc.D. dissertation, Department of Electrical Engineering and Computer Science, MIT, 2000
4. 邱詰淙、陳種成、鐘毅宏、沈育龍，點餐服務機器人，智慧型機器人產品創意競賽，2008
5. H. Kwon, Y. Yoon, J.B. Park and A.C. Kak, “Person Tracking with a Mobile Robot using Two Uncalibrated Independently Moving Cameras”, IEEE International Conference on Robotics andAutomation, Barcelona , pp. 2877 - 2883, 2005
6. T. Yoshimi, M. Nishiyama, T. Sonoura, H. Nakamoto, S. Tokura, H. Sato, F.Ozaki, N. Matsuhira and H. Mizoguchi, “Development of a Person Following Robot with Vision Based Target Detection”, IEEE/RSJ International Conference on Intelligent Robots and Systems, Beijing, pp. 5286 - 5291, 2006.
7. N. Hirai, H. Mizoguchi, “Visual Tracking of Human Back and Shoulder for Person Following Robot”, The Proceedings of IEEE International Conference on Advanced Intelligent Mechatronics, pp. 527-532, 2003
8. C. Y. Tsai and K. T. Song, “Face Tracking Interaction Control of a Nonholonomic Mobile Robot”, IEEE/RSJ International Conference on Intelligent Robots and Systems, Beijing, pp. 3319 - 3324, 2006
9. 何昭慶，影像伺服控制與三維追蹤之研究，國立台灣科技大學電機工程系，

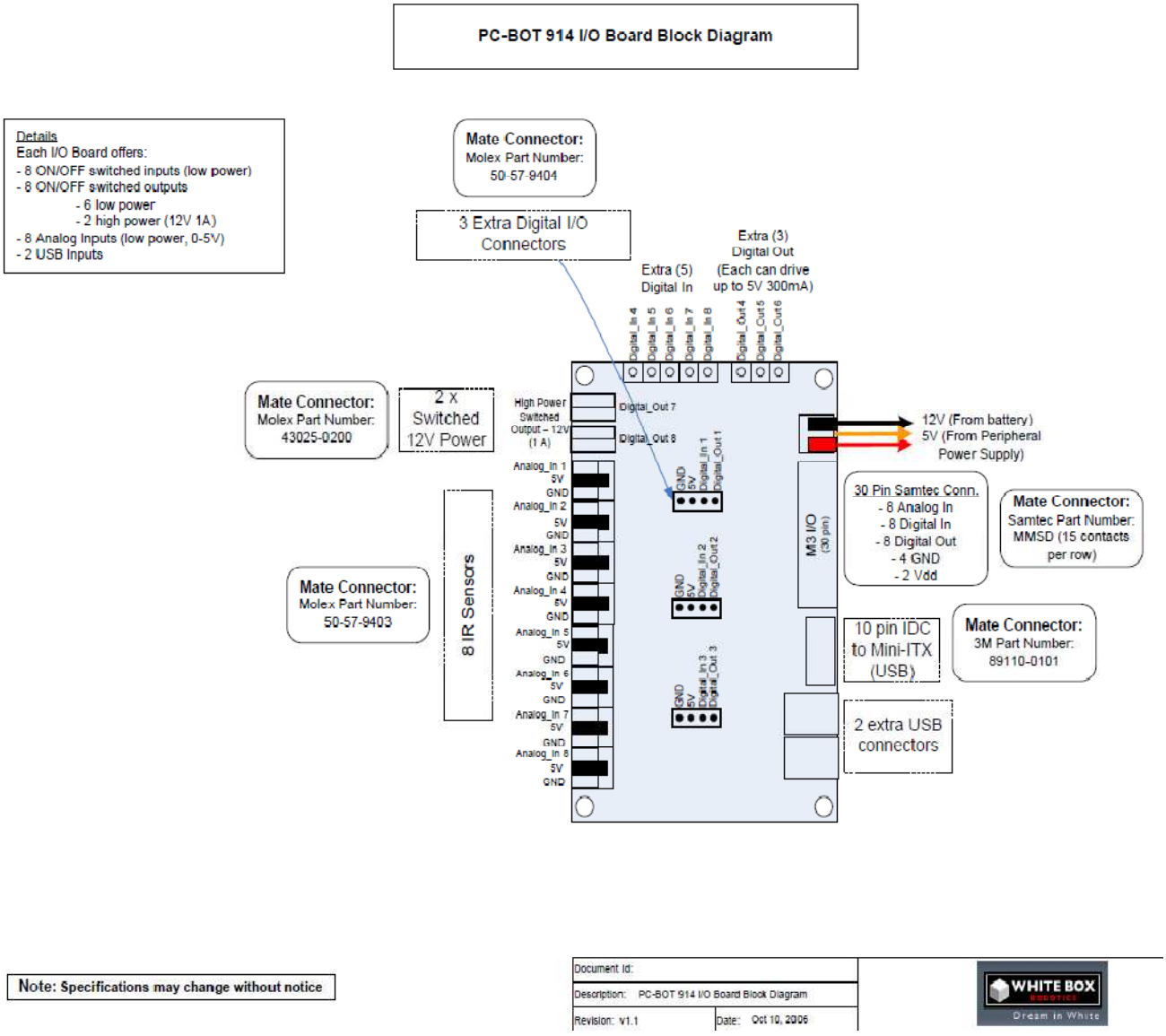
博士論文，2008

10. Li T.-H.S, Shih-Jie Chang and Wei Tong, “Fuzzy Target Tracking Control of Autonomous Mobile Robots by Using Infrared Sensors”, IEEE Transactions on Fuzzy System, Vol.12, No.4, 2004
11. 林哲瑋，基於即時影像之模糊追隨控制，國立台北科技大學自動化科技研究所，碩士論文，2007
12. 何育昕，自動導航車之模糊追隨控制，國立台北科技大學電機工程系所，碩士論文，2005
13. M. Chueh, Au Yeung, Y .L.W, Lei K.-P.C and Joshi S.S, “Following Controller for Autonomous Mobile Robots Using Behavioral Cues”, IEEE Transactions on Industrial Electronics, Vol.55, No.8, 2008
14. Y. Nagumo, A. Ohya, “Human Following Behavior of an Autonomous Mobile Robot Using Light-Emitting Device”, IEEE International Workshop on Robot and Human Interactive Communication, pp.225-230, 2001
15. M. Kobilarov, G. Sukhatme, J. Hyams and P. Batavia, “People Tracking and Following with Mobile Robot Using an Omnidirectional Camera and a Laser”, IEEE International Conference on Robotics and Automation, pp. 557-562, 2006
16. 賴一翔，具搜尋與避障之自動跟隨機器人，國立中央大學電機工程學系，碩士論文，2009
17. 傅培耕，即時物體追蹤之立體視覺導引自走車，中原大學機械工程學系，碩士論文，2004
18. “914_PC-BOT_Extreme-Version”, White Box Robotics Inc., 2007
19. White Box Robotics Official Website, http://www.whiteboxrobotics.com

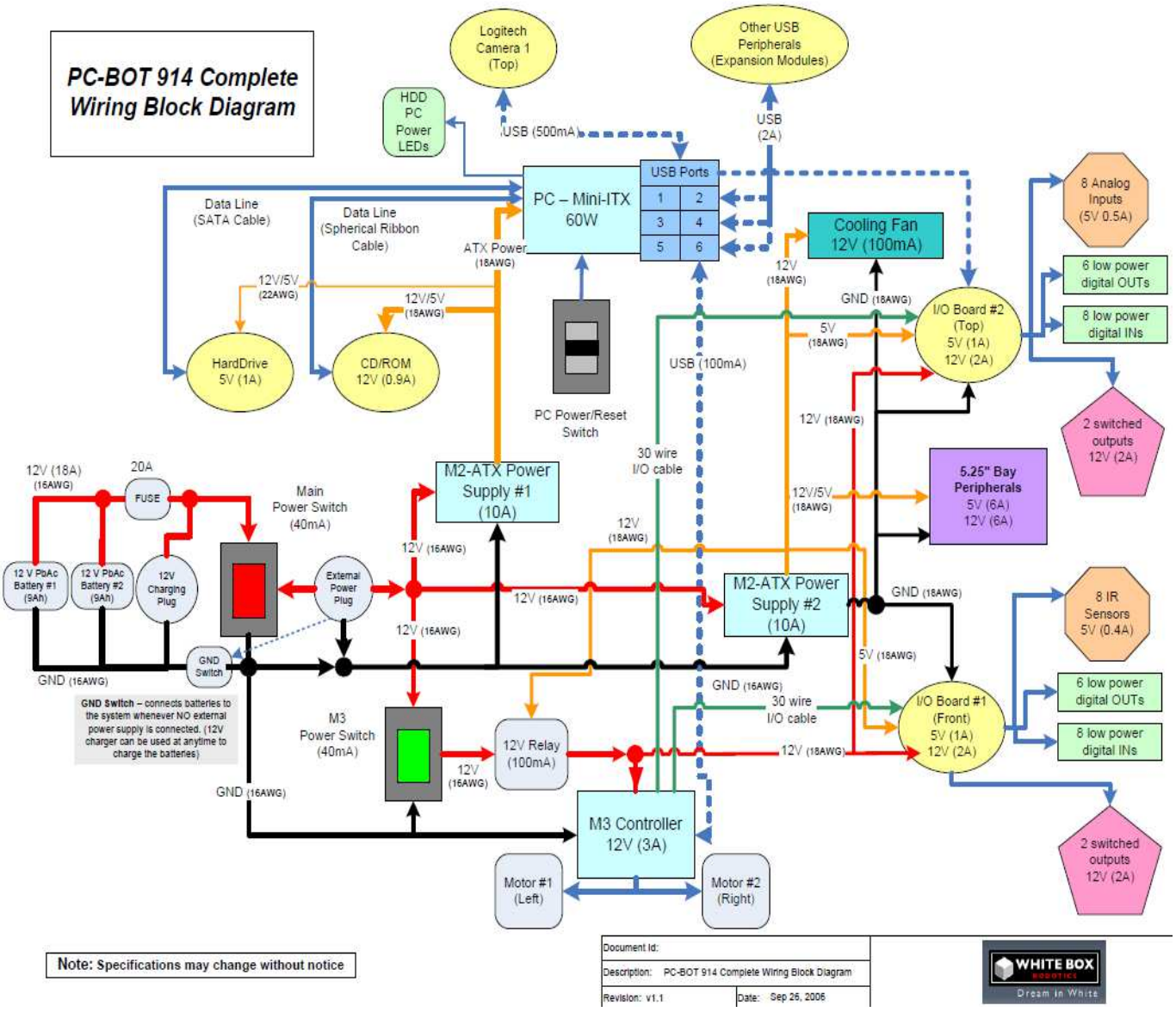
20. 陳連春，步進馬達原理與活用要訣，建興出版社，2000
21. “M2-ATX Installation Guide”, Mini-Box Inc., 2009
22. “Sharp-GP2Y0A21YK”, Sharp Corporation Inc., 2006
23. “PC-Bot_Tech_Spec-Infra-redSensorsv1.2”, White Box Robotics Inc., 2007
24. “Sharp-GP2Y0A02YK”, Sharp Corporation Inc., 2006
25. 陳文冰，跟我學 Visual C# 20008，碁峰資訊股份有限公司，2008
26. “PrimeSense Reference Design 1.08”, PrimeSense Inc., 2010
27. OpenNI Organization Official Website, http://www.openni.org/
28. “NITE Controls User Guide”, PrimeSense Inc., 2010
29. L. A. Zadeh, “Fuzzy Sets”, Information and Control, Vol.8, pp.338–353, 1965
30. E.H.Mamdani,“Application of Fuzzy Algorithms for Control of Simple Dynamic Plant”, IEEE Proceedings of the Institution of Electrical Engineers, Vol.121, No.12, 1974
31. 王文俊，認識 Fuzzy-第三版，全華圖書股份有限公司，2007
32. 楊英魁，Fuzzy 控制，全華科技圖書股份有限公司，1992
33. 李允中、王了璠、蘇木春，全華科技圖書股份有限公司，2003
34. T.Yamakawa,“Stabilization of an Inverted Pendulum by a High-Speed Logic Controller Hardware System”, Fuzzy Sets and Systems, Vol.32, NO.2, pp.161-180, 1989

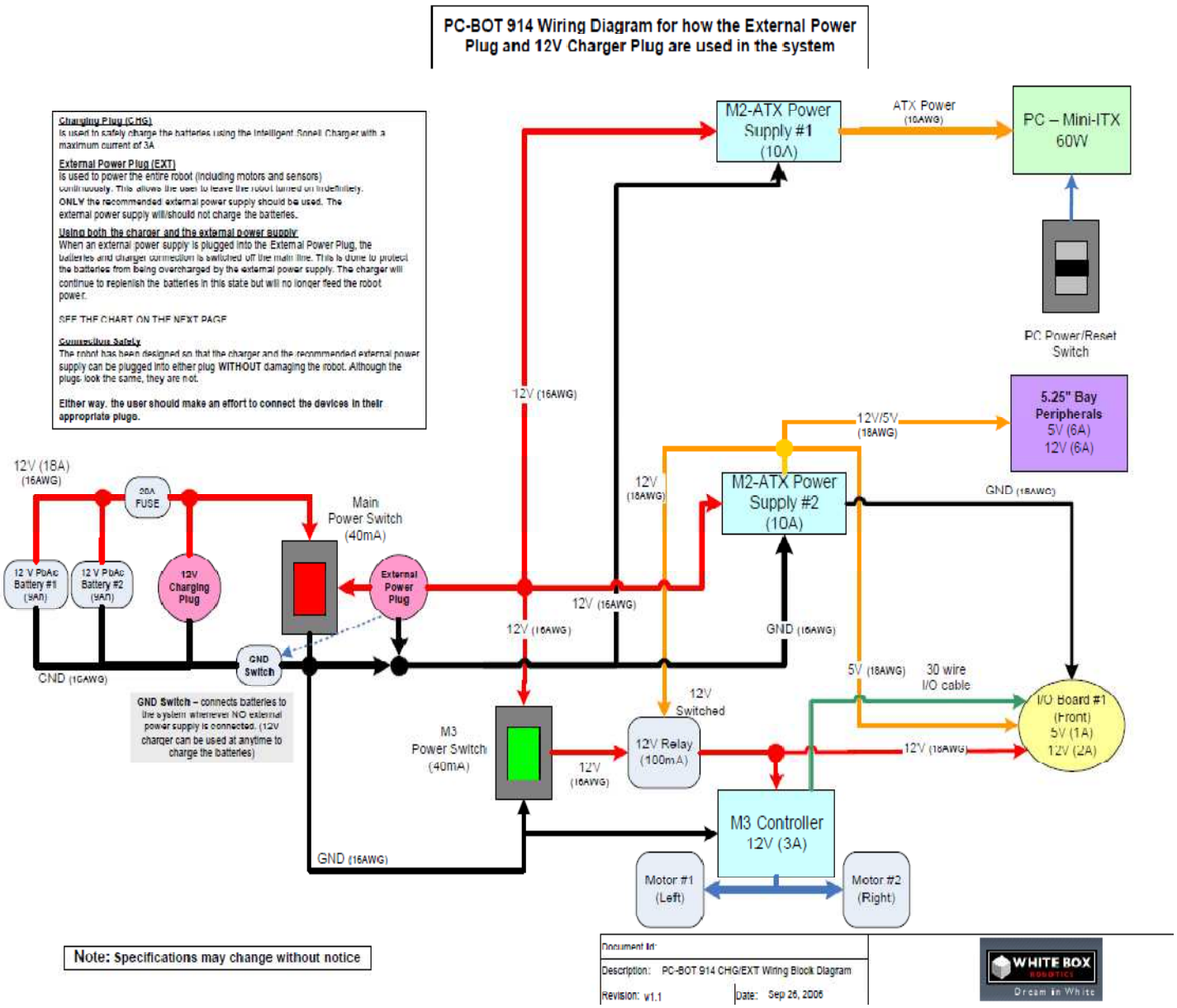
附錄

附錄一：PC-BOT 914 I/O Board Block Diagram



附錄二：PC-BOT 914 Complete Wiring Block Diagram




PC-BOT 914 Wiring Diagram for how the External Power Plug and 12V Charger Plug are used in the system
Charging Plug (CHG)
is used to safely charge the batteries using the Intelligent Sonell Charger with a maximum current of 3A
External Power Plug (EXT)
is used to power the entire robot (including motors and sensors) continuously. This allows the user to leave the robot turned on indefinitely. ONLY the recommended external power supply should be used. The external power supply will/should not charge the batteries.
Using both the charger and the external power supply
When an external power supply is plugged into the External Power Plug, the batteries and charger connection is switched off the main line. This is done to protect the batteries from being overcharged by the external power supply. The charger will continue to replenish the batteries in this state but will no longer feed the robot power.
SEE THE CHART ON THE NEXT PAGE
Connection Safety
The robot has been designed so that the charger and the recommended external power supply can be plugged into either plug WITHOUT damaging the robot. Although the plugs look the same, they are not.
Either way, the user should make an effort to connect the devices in their appropriate pluge.
M2-ATX Power Supply #1 (10A)
ATX Power (16AWG)
PC – Mini-ITX 60W
PC Power/Reset Switch
12V (16AWG)
12V/5V (18AWG)
5.25" Bay Peripherals 5V (6A) 12V (6A)
12V (18A) (16AWG)
20A FUSE
Main Power Switch (40mA)
12V (18AWG)
M2-ATX Power Supply #2 (10A)
GND (18AWG)
12 V PbAc Battery #1 (9Ah)
12 V PbAc Battery #2 (9Ah)
12V Charging Plug
External Power Plug
12V (16AWG)
GND (16AWG)
GND Switch
GND (16AWG)
GND Switch – connects batteries to the system whenever NO external power supply is connected. (12V charger can be used at anytime to charge the batteries)
5V (18AWG)
30 wire I/O cable
I/O Board #1 (Front) 5V (1A) 12V (2A)
12V Switched
M3 Power Switch (40mA)
12V Relay (100mA)
12V (16AWG)
12V (16AWG)
M3 Controller 12V (3A)
GND (16AWG)
Motor #1 (Left)
Motor #2 (Right)
Note: Specifications may change without notice
Document Id:
Description: PC-BOT 914 CHG/EXT Wiring Block Diagram
Revision: v1.1
Date: Sep 26, 2006
WHITE BOX ROBOTICS
Dream in White

PC-BOT 914 Wiring Diagram for how the External Power Plug and 12V Charger Plug are used in the system

Using the BATTERY CHARGER

	Connected to Plug:	
Main Switch	**CHG Plug**	**EXT Plug**
ON	Slows down battery discharge and powers the robot	Slows down battery discharge and powers the robot
OFF	Charges battery	No effect

Using the RECOMMENDED EXTERNAL POWER SUPPLY

	Connected to Plug:	
Main Switch	**CHG Plug**	**EXT Plug**
ON	No effect	Powers the robot indefinitely
OFF	No effect	Powers the robot indefinitely

PC-BOT Run Time: (note: the times listed are estimates)

Autonomous movement, Charger NOT connected	2 hours
No Autonomous movement, Charger NOT connected	4-5 hours
No Autonomous movement, Charger IS connected	8-9 hours
No Autonomous movement, Ext Power supply connected	Continuous

Battery Charge Time
2 hours to recharge batteries from auto-shutoff point

Note: Specifications may change without notice

Document Id:	
Description: PC-BOT 914 CHG/EXT Connection Chart	
Revision: v1.1	Date: Sep 26, 2006

WHITE BOX ROBOTICS
Dream in White

PC-BOT 914 Power Budget

Note: Batteries currently used in the PC-Bot are 12V 18Ah capacity. Maximum current drawn should not exceed 18A (216W). (This will be fuse limited).
The values listed below are max current drawn for the given item, the total of which goes well beyond the maximum allowed current.
Some combination of the following, adding up to under 216W, can be used in the system.

Total System Power			
Item	**Quantity**	**Maximum Current/Voltage (From the Batteries)**	**Total Wattage**
M2-ATX Power Supply (For PC)	1	12V~7.67A (92W)	92W
M2-ATX Power Supply (For Peripherals)	1	12V~9.88A (118.6W)	118W
M3 Controller	1	12V~3A (36W)	36W
I/O Board - Switched 12V Outputs (Headlights, etc.)	2	12V~2A (24W)	48W
Main Power Lighted Switch	1	12V~0.04A (0.6W)	0.6W
M3 Power Lighted Switch	1	12V~0.04A (0.6W)	0.6W
		Total Possible:	**295.2W**
		Total Allowed:	**216W**

Note: More detailed view of M2-ATX Power Supplies shown below:

M2-ATX Power Supply (For PC)		
Item	**Quantity**	**Maximum Current/Voltage (Drawn by the item)**
Mini-ITX	1	60W
HDD (40Gig)	1	5V~1A (5W)
CD/DVD ROM	1	12V~0.9A (12W)
USB Web Camera	2	5V~1A (5W)
Other USB Peripherals	4	5V~2A (10W)
	Total Possible:	**92W**

M2-ATX Power Supply (For Peripherals)		
Item	**Quantity**	**Maximum Current/Voltage (Drawn by the item)**
M3 Cooling Fan	1	12V~0.12A (1W)
Top I/O Board - **Sensors/Analog/Digital IN-OUT ONLY**	1	12V~0.6A (7.1W)
5.25" Bay Peripherals	6	5V~6A, 12V~6A (102W)
Front I/O Board - **Sensors/Analog/Digital IN-OUT ONLY**	1	12V~0.6A (7.1W)
Relay for M3 and I/O 12V outputs	1	12V~0.117A (1.4W)
	Total Possible:	**118.6W**

Note: More detailed view of the I/O board shown below:

I/O Board - Sensors/Analog/Digital IN-OUT ONLY		
Item	**Quantity**	**Maximum Current/Voltage (Drawn by the item)**
IR (Analog) Sensors	8	5V~0.4A (2W)
5V low power switched outputs	3	5V~0.9A (4.5W)
Digital outputs	3	5V~0.12A (0.6W)
	Total Possible:	**7.1W**

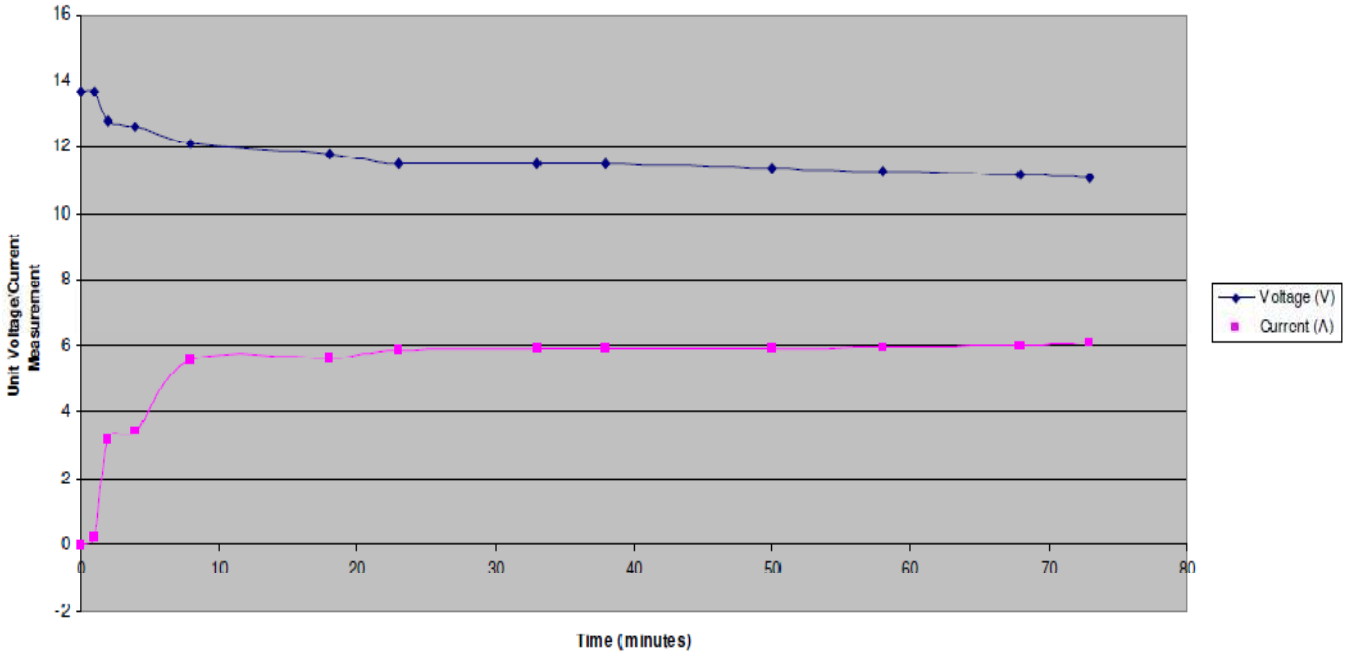
Note: Specifications may change without notice

Document Id:	
Description: PC-BOT 914 Power Budget	
Revision: v1.1	Date: Sep 26, 2006

WHITE BOX ROBOTICS
Dream in White

PC-BOT 914 Maximum (Worst Case Scenario) Current/ Voltage Usage over time

Current/Voltage Discharge on PC-BOT



Details	Time	Voltage	Current
Robot Power Turned OFF	0	13.7	0
Main Power Switch ON	1	13.7	0.22
PC Booting up Windows started	2 4	12.8 12.6	3.2 3.4
Brian GUI started, headlights turned ON, motors moving with full torque (100% duty cycle), USB web camera and wireless device running.	8	12.1	5.6
Continuous Movement	18	11.8	5.63
Continuous Movement	23	11.5	5.85
Continuous Movement	33	11.5	5.9
Continuous Movement	38	11.5	5.9
Continuous Movement	50	11.35	5.9
Continuous Movement	58	11.3	5.97
Continuous Movement	68	11.2	6
Auto-Shutoff	73	11.1	6.1

This chart indicates worst case scenario current discharge of the PC-Bot. The motors were run at full torque (100% duty cycle), all sensors, USB peripherals (camera, wireless network and mouse) and headlights were turned ON. The movement was continous at low speed which leads to maximum current drain.

Note: Specifications may change without notice

Document Id:	
Description: PC-BOT 914 Maximum Current/Voltage Usage	
Revision: v1.1	Date: Sep 26, 2006

WHITE BOX ROBOTICS Dream in White

附錄三：M2-ATX Installation Guide

Before you start...

Please take a moment and read this manual before you install the M2-ATX in your
vehicle. Often times, rushing into installing the unit can result in serious damage to your M2-ATX board, computer and probably your car's electrical system.

The M2-ATX board has several wires that need to be installed in various places.
When installing, **always double check the polarity** of your wires with a voltmeter.

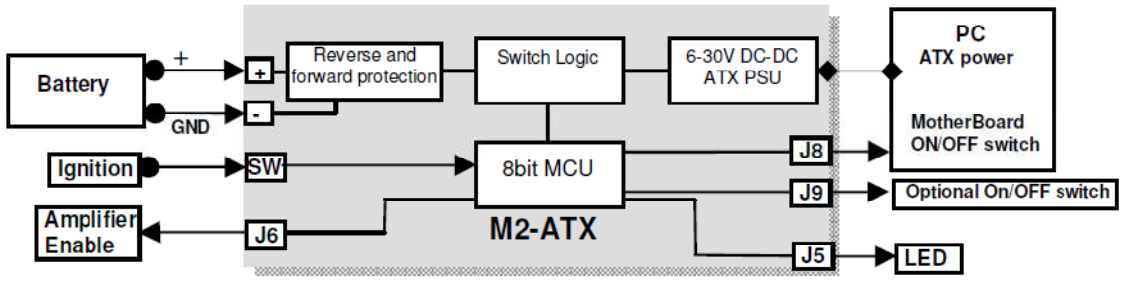
Avoid using the cigarette plug as a power source, often times the contacts are not capable of delivering high current to your PC.

1.0 Introduction

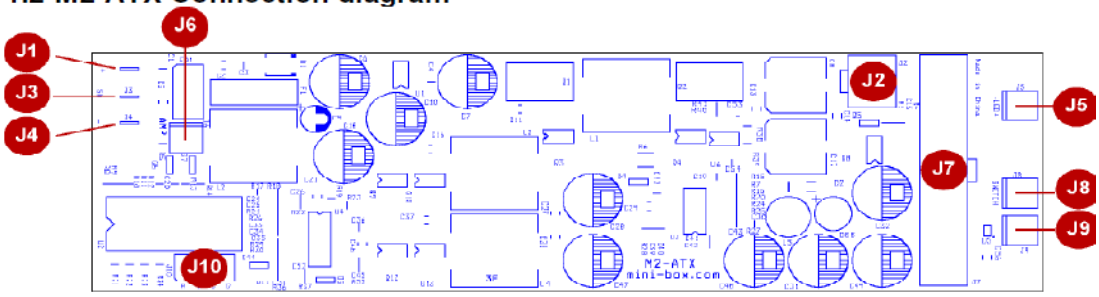
Thank you for purchasing the M2-ATX power sequencer / vehicle ATX power supply.

The M2-ATX was designed to work with a wide variety of main boards such as the VIA mini-ITX motherboards as well as Pentium-M, Celeron or certain full power Core Duo systems.

1.1 M2-ATX Logic Diagram



1.2 M2-ATX Connection diagram



M2-ATX, top view

Power Input Connectors

J1 Battery + (un-switched battery, positive)
J3 Ignition (To start connect to Battery +). Do not use in the standard PSU mode (mode P0)
J4 Battery - (negative)

Controls and Settings

J6 Controls amplifier via remote ON/OFF. Left pin is RMT, Right pin is GND
J8 To motherboard ON/OFF switch
J10 User jumper settings (A,B,C,D)
J9 To external ON/OFF switch (optional, J8 is in parallel with J9)

Power Output Connectors

J2 Optional P4-12V power
J7 ATX power connector (to motherboard)
J5 To LED (optional)

Jumper attached=ON						
A	B	C	D	P	Off-delay (All rails ON)	Hard-off (5VSB)
OFF	OFF	OFF	OFF	P0	Standard PSU mode	
ON	OFF	OFF	OFF	P1	5s + 1min AutoLatch*	1 min
OFF	ON	OFF	OFF	P2	5s+ 1min AutoLatch*	2 hour
ON	ON	OFF	OFF	P3	5s+ 1min AutoLatch*	NEVER
OFF	OFF	ON	OFF	P4	30s + 1min AutoLatch*	2 hour
ON	OFF	ON	OFF	P5	30s + 1min AutoLatch*	NEVER
OFF	ON	ON	OFF	P6	30min	NEVER
ON	ON	ON	OFF	P7	3 hour	NEVER
OFF	OFF	OFF	ON	P8	10 min	1 hour
ON	OFF	OFF	ON	P9	15 min	2 hours
OFF	ON	OFF	ON	P10	1 hour	75min

IMPORTANT: Always use the "Hibernate" feature, never use "Standby" as it can severely discharge your battery over extended periods of time.

NEVER use "Hard-off = NEVER" settings unless you understand the risks of battery depletion. "Hard-off=NEVER always keeps 5VSB rail ON!

***AutoLatch** is active during the fist 60s of PC operation (and only during the first 60 seconds). For example, If Ignition is turned ON and then OFF right away, M2-ATX will latch Ignition in ON position for the next 60 seconds,

allowing your operating system to fully come up. This will prevent disk drive corruption or systems that remain hung in the ON position. After the first 60 seconds of system operation, the AutoLatch feature will be removed and system will shut down at as governed by the "Off-delay" setting.

P0: In this mode, the M2-ATX behaves like a regular ATX power supply. If J6 is connected to the motherboard, M2-ATX will also send a gratuitous "ON pulse" to the motherboard right after power is first applied. **If IGNITION is connected to the battery, the unit will shut down if battery is < 11.2V**. This is to protect the battery from over-discharging. If Ignition is not hooked up, M2-ATX will operate from as low as 6V. For more information, please consult the "UPS mode" documentation on the product page.

P1 (recommended): Sends ON pulse to motherboard when ignition is ON for more than 5 seconds, sends OFF pulse to motherboard **5 seconds** after ignition is turned off. Waits another **60 seconds** and then shuts down 5VSB to conserve battery. In this mode, the M2-ATX consumes less than 0.5mA. **This is our recommended setting.**

1.2 Power challenges in a Vehicle PC

The 5V Standby Problem: One of most difficult tasks of operating a PC in a vehicle is power consumption while the computer is OFF. Even when your computer is OFF or in Suspend, it will still consume about 50-150mA on the 5VSB rail. *No matter how big your battery is, you will eventually drain it if proper actions are not taken.*

The M2-ATX is addressing these issues by cutting off the 5VSB rail after a pre-defined amount of time (see jumper chart, HARDOFF). When 5VSB is always active (HARDOFF=Never), M2-ATX constantly monitors the battery levels. When battery level drops below 11V for more than one minute, M2-ATX will shut down and re-activate only when the input voltage is > 12V.

Engine Cranks, under-voltage and over-voltage situations. Another difficult task is maintaining stable 3.3V, 5V, 12V and -12V power to your PC. While car batteries are rated at 12V, they actually provide voltages in between 7-16V (engine cranks) or as high as 80 volts (load dump). Most of the times, your battery will stay at 13.5V (while car is running) but extra precautions need to take place in order to prevent such situations. M2-ATX can operate as low as 6V and as high as 28V while providing strict regulation on all rails along with input voltage clamping and reverse protection.

Loud amplifier pops when PC starts. If your PC is connected to your car amplifier, you will hear a loud pop when the computer is first started. The M2-ATX has an 'anti-thump' control that will keep your amp OFF while the PC starts. Simply connect J6 to your amplifier remote control pins to activate the 'anti-thump' feature.

2.0 Mode of operation

1) Ignition=OFF. Nothing happens. M2-ATX is waiting for ignition signals.

2) Ignition=ON. M2-ATX waits for few seconds then turns on the 5VSB rail. After another second M2-ATX sends an "ON" signal to the motherboard via the 2 wires connected to the motherboard's ON/OFF pins. The motherboard will turn ON and your system should start

booting. The Ignition state will be latched for 60 more seconds so that the motherboard will have a change to come up in a clean manner.

3) Ignition=ON. Your computer will remain ON.

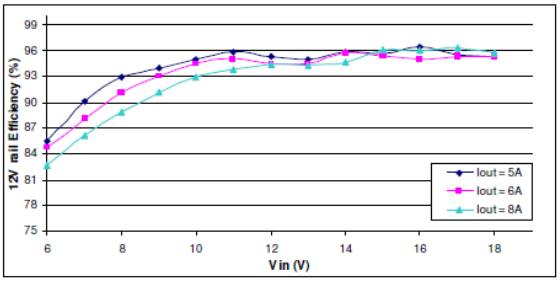
4) Ignition=OFF. M2-ATX waits for "OFFDELAY" in seconds (see jumper chart) and then it turns the motherboard OFF by sending a signal to the motherboard's ON/OFF switch. Your computer should turn off gracefully (shutdown procedure). After shutdown, 5VSB will still be provided for another "HARDOFF" seconds. In the event where the shutdown process is longer than "HARDOFF" (Operating System gets frozen, etc), power will be shut down hard, turning off all power rails. During the HARDOFF procedure, the battery levels will be constantly monitored to prevent deep discharge situations.

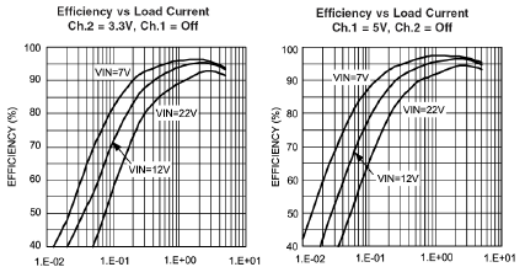
5) M2-ATX will go to step 1, if ignition is tuned ON again.

3.0 M2-ATX Characteristics

Minimum Input Operating. voltage	6V
Maximum input Operating voltage	24V (clamping will occur at 25-27V)
Deep-Discharge shutdown threshold	11.2V
Input current limit (fuse protected)	15A (15A mini-blade fuse)
Max Output Power	160 Watts
Operating temperature	-40 to +85* degrees Celsius
Storage temperature	-55 to +125 degrees Celsius
MTBF	192,000 hrs @ 50C, 96,000 hrs @65C
Efficiency (Input 9-16V)	>94%, all rails combined, 50% load.
PCB size	160x45mm
Input connectors	Faston 0.25" terminal
Output Connector	ATX Power 20 pin (Molex P/N 39-01-2200

*Units starts failing at ~115 Celsius. Operating at temperatures above 85C / 185F will drastically reduce the MTBF. When operating at high temperatures or fanless operation, must reduce PSU load by 25%.





Maximum Power Characteristics

Output Rail	Current (Max)	Current Peak (<60 seconds)	Regulation
5V	8A	10A	1.5%
3.3V	8A	10A	1.5%
5VSB	1.5A	2A	1.5%
-12V	0.15A	0.2A	5-%
12V	8A* (see below)	10A	2%

Total power = 169.9 Watts
When operating at 24V or extreme temperatures, de-rate by 30-50%, ventilation will be required.

12V Rail Output Current

Input (V)	12V rail current	Input (V)	12V rail current
6V	4A	11V	8A
7V	5A	12V	8A
8V	6A	14V	8.5A
9V	7A	14-18V	9A
10V	8A	20-26V	7A

For low input voltage (6-10V) ventilation might be required for peak load

附錄四：Kinect Sensor Source Code(C/C++)

```
/*----------------------------------------
        - OpenNI Kinect Project -

    Program name : Follow ME
    ----------------------------------------*/



#include <stdio.h>
#include <stdlib.h>
#include <iostream>
#include <math.h>
#include <vector>
#include <string>

#include <glut.h>
#include <XnCppWrapper.h>

#include <windows.h>
#include <winsock2.h>
#include <ws2tcpip.h>



#pragma comment (lib, "Ws2_32.lib")



#define RUN    1
#define STOP 0
#define DEFAULT_BUFLEN          512
#define DEFAULT_IP_ADDRESS "127.0.0.1"
#define DEFAULT_PORT            "19882"
```

```
using namespace std;



// Global Variables //-------------------------------------------------------------
// Universal //-----------------------------------------------------
short int status         = STOP;
XnUserID    operatorID = 100;


// Kinect //-----------------------------------------------------------
xn::Context                 mContext;                 // Create Context
xn::DepthGenerator      mDepthGenerator;      // Create Depth     Generator
xn::ImageGenerator      mImageGenerator;       // Create Image     Generator
xn::UserGenerator       mUserGenerator;         // Create User       Generator


// Server //-----------------------------------------------------------
int    iResult;
int    iSendResult;
char sendbuf[DEFAULT_BUFLEN];    // Send Buffer
char recvbuf[DEFAULT_BUFLEN];    // Recv Buffer
int    recvbuflen = DEFAULT_BUFLEN;
WSADATA wsaData;
SOCKET ListenSocket = INVALID_SOCKET;
SOCKET ClientSocket = INVALID_SOCKET;
struct addrinfo *result = NULL;
struct addrinfo hints;



// Kinect Callback Function //-----------------------------------------------
// callback function of user generator: new user
void XN_CALLBACK_TYPE
NewUser( xn::UserGenerator &generator,
            XnUserID                 user,
            void                     *pCookie )
```

```
{
	if ( status != RUN ) {
		mUserGenerator.GetPoseDetectionCap().StartPoseDetection("Psi", user);
	}

	cout << "New user identified: " << user << endl;
}

// callback function of user generator: lost user
void XN_CALLBACK_TYPE
LostUser( xn::UserGenerator &generator,
		XnUserID		user,
		void			*pCookie )
{
	mUserGenerator.GetPoseDetectionCap().StopPoseDetection( user );
	if ( operatorID == user ) {
		status = STOP;
		operatorID = 100;
	}

	cout << "User " << user << " lost" << endl;
}

// callback function of pose detection: pose start
void XN_CALLBACK_TYPE
PoseDetected( xn::PoseDetectionCapability &poseDetection,
			const XnChar		*strPose,
			XnUserID		user,
			void			*pCookie)
{
	if ( status == RUN )
	{
		status = STOP;
		operatorID = 100;
```

```
        }
        else if ( status == STOP )
        {
                status = RUN;
                operatorID = user;
        }

        cout << "Pose " << strPose << " detected for user " <<    user << endl;
}



// OpenGL Callback Function //----------------------------------------------
void kinectUpdate()
{
        XnPoint3D userPosition;
        XnFloat x, y, z;

        // Update date
        mContext.WaitAndUpdateAll();

        switch (status)
        {
                case RUN:
                        mUserGenerator.GetCoM( operatorID, userPosition);

                        x = userPosition.X;
                        y = userPosition.Y;
                        z = userPosition.Z;
                        break;

                case STOP:
                        x = 0;
                        y = 0;
                        z = 0;
```

```
                    break;
        }

        // Prepare data which will be sent out.
        sprintf(sendbuf, "<status=%i,X=%+08.2fmm,Y=%+08.2fmm,Z=%+08.2fmm>", status, x, y, z);

        // Send Data
        iSendResult = send( ClientSocket, sendbuf, (int)strlen(sendbuf), 0 );
        if (iSendResult == SOCKET_ERROR) {
                printf("send failed with error: %d\n", WSAGetLastError());
                closesocket(ClientSocket);
                WSACleanup();
        }

        // Output Information
        printf("-----------------------------------------------\n");
        printf("<status=%i,X=%+08.2fmm,Y=%+08.2fmm,Z=%+08.2fmm>\n", status, x, y, z);
        printf("Bytes sent: %d\n", iSendResult);
}

void drawUsers()
{
        xn::SceneMetaData mUserMap;
        XnLabel iLabel;
        mUserGenerator.GetUserPixels(0, mUserMap);                                      // Get User Map
        const XnRGB24Pixel *rgb24ImageMap = mImageGenerator.GetRGB24ImageMap();    // Get RGB Image Map

        glBegin(GL_POINTS);
        for( int y = 0; y < 480; ++y )
        {
                for( int x = 0; x < 640; ++x )
                {
                        iLabel = mUserMap(x, y);
```

```
                // Setup Vertex Color
                if ( operatorID == iLabel ) {
                        glColor3f( ( (rgb24ImageMap[640*y+x]).nRed      )/255.0,
                                        ( (rgb24ImageMap[640*y+x]).nGreen )/255.0,
                                        ( (rgb24ImageMap[640*y+x]).nBlue    )/255.0 );
                }
                else if ( iLabel == 1 )          {
                        glColor3f(1.0, 0.0, 0.0);
                }
                         else if ( iLabel == 2 ) {
                        glColor3f(0.0, 1.0, 0.0);
                }
                         else if ( iLabel == 3 ) {
                        glColor3f(0.0, 0.0, 1.0);
                }
                         else if ( iLabel == 4 ) {
                        glColor3f(1.0, 1.0, 0.0);
                }
                         else if ( iLabel == 5 ) {
                        glColor3f(0.0, 1.0, 1.0);
                }
                         else if ( iLabel == 6 ) {
                        glColor3f(1.0, 0.0, 1.0);
                }
                         else {
                        glColor3f(1.0, 1.0, 1.0);
                }

                glVertex2f(-(x-320)*0.0015625*100, -(y-240)*0.0020833*100);
            }
        }
        glEnd();
}
```

```
void display()
{
	glClear(GL_COLOR_BUFFER_BIT);
	kinectUpdate();
	drawUsers();
	glutSwapBuffers();
	Sleep(33);
}

void idle()
{
	glutPostRedisplay();
}

void initOpenGL()
{
	glClearColor(0.0, 0.0, 0.0, 1.0);
	glColor3f (1.0, 1.0, 1.0);
}

void reshape(int w, int h)	// 改變視窗大小時會自動傳入"寬","高"
{
	glViewport(0, 0, w, h);	// 縮放顯示比例到視窗大小
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	glOrtho (-50.0, 50.0, -50.0, 50.0, -1.0, 1.0);
}


// Main Program //-------------------------------------------------------
int main( int argc, char **argv )
{
	// Server //-----------------------------------------------------
	// Initialize Winsock
```

```
iResult = WSAStartup(MAKEWORD(2,2), &wsaData);
if (iResult != 0) {
        printf("WSAStartup failed with error: %d\n", iResult);
        return 1;
}

ZeroMemory(&hints, sizeof(hints));
hints.ai_family      = AF_INET;
hints.ai_socktype = SOCK_STREAM;
hints.ai_protocol = IPPROTO_TCP;
hints.ai_flags       = AI_PASSIVE;

// Resolve the server address and port
iResult = getaddrinfo(DEFAULT_IP_ADDRESS, DEFAULT_PORT, &hints, &result);
if ( iResult != 0 ) {
        printf("getaddrinfo failed with error: %d\n", iResult);
        WSACleanup();
        return 1;
}

// Create a SOCKET for connecting to server
ListenSocket = socket(result->ai_family, result->ai_socktype, result->ai_protocol);
if (ListenSocket == INVALID_SOCKET) {
        printf("socket failed with error: %ld\n", WSAGetLastError());
        freeaddrinfo(result);
        WSACleanup();
        return 1;
}

// Setup the TCP listening socket
iResult = bind( ListenSocket, result->ai_addr, (int)result->ai_addrlen);
if (iResult == SOCKET_ERROR) {
        printf("bind failed with error: %d\n", WSAGetLastError());
        freeaddrinfo(result);
```

```
        closesocket(ListenSocket);
        WSACleanup();
        return 1;
}

freeaddrinfo(result);

iResult = listen(ListenSocket, SOMAXCONN);
if (iResult == SOCKET_ERROR) {
        printf("listen failed with error: %d\n", WSAGetLastError());
        closesocket(ListenSocket);
        WSACleanup();
        return 1;
}

// Accept a client socket
ClientSocket = accept(ListenSocket, NULL, NULL);
if (ClientSocket == INVALID_SOCKET) {
        printf("accept failed with error: %d\n", WSAGetLastError());
        closesocket(ListenSocket);
        WSACleanup();
        return 1;
}

// No longer need server socket
closesocket(ListenSocket);



   // Kinect //-------------------------------------------------------
// Initial Context and Setup Generators
mContext.Init();
mDepthGenerator.Create( mContext );
mImageGenerator.Create( mContext );
```

```
// Map Output Mode
XnMapOutputMode mapMode;
mapMode.nXRes = 640;
mapMode.nYRes = 480;
mapMode.nFPS    =    30;
mDepthGenerator.SetMapOutputMode( mapMode );
mImageGenerator.SetMapOutputMode( mapMode );

// Correct View Port
mDepthGenerator.GetAlternativeViewPointCap().SetViewPoint( mImageGenerator );
mUserGenerator.Create( mContext );

   // Register Callback Functions //----------------------
   // Create Callback Handles
XnCallbackHandle hUserCB, hPoseCB;

// User Generator Callback Function
mUserGenerator.RegisterUserCallbacks( NewUser, LostUser, NULL, hUserCB );

// Pose Detection capability Callback Function
mUserGenerator.GetPoseDetectionCap().RegisterToPoseCallbacks( PoseDetected, NULL, NULL, hPoseCB );

// Start Generate Data
mContext.StartGeneratingAll();


// OpenGL //-------------------------------------------------------
glutInit(&argc, argv);
glutInitWindowPosition(200, 200);                    // 視窗位置
glutInitWindowSize(640, 480);                         // 視窗大小
glutInitDisplayMode (GLUT_DOUBLE | GLUT_RGB);   // 使用 Double Buffer
glutCreateWindow("Follow ME!");                       // 建立視窗
initOpenGL();
```

```
	glutDisplayFunc(display);
	glutReshapeFunc(reshape);   //視窗大小改變時做的動作
	glutIdleFunc(idle);             //閒置一段時間(沒事件發生)時呼叫一次

	// 等待事件迴圈
	glutMainLoop();



	// Stop and Shutdown //-------------------------------------------
	// Server //-----------------------------------------------------
	// shutdown the connection since we're done
	iResult = shutdown(ClientSocket, SD_SEND);
	if (iResult == SOCKET_ERROR) {
		printf("shutdown failed with error: %d\n", WSAGetLastError());
		closesocket(ClientSocket);
		WSACleanup();
		return 1;
	}
	// clean-up
	closesocket(ClientSocket);
	WSACleanup();

	// Kinect //-----------------------------------------------------
	// Stop and clean-up
	mContext.StopGeneratingAll();
	mContext.Release();

	return 0;
}
```

附錄五：Tracking Robot Source Code(C#)

```
using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using System.Net.Sockets;

using System.Threading;

using System.Text.RegularExpressions;

using System.IO;



namespace Follow_You

{

    public partial class Form1 : Form

    {

        //宣告網路資料流變數

        NetworkStream myNetworkStream;

        //宣告 Tcp 用戶端物件
```

```
TcpClient myTcpClient;



//機器人控制參數

private bool m3 = true;

private bool LR = true;

private int Power = 20;

private double z_f = 1300;

private double z_e = 0;

private double z_e_o = 0;

private double x_o = 0;

private double x_n = 0;



public Form1()

{

    InitializeComponent();



    m31.Open();

    m31.Connect();



    //以字串變數設定目錄及名稱

    string path = @"C:\Documents and Settings\Owner\Desktop\Follow You - 改 - 追蹤\data";
```

```
        try
        {
            if (Directory.Exists(path))//檢查目錄是否存在
            {
                MessageBox.Show("目錄已經存在。");
            }
            //產生目錄
            DirectoryInfo di = Directory.CreateDirectory(path);
            MessageBox.Show("目錄建立成功。");
        }
        catch (Exception ex)
        {
            MessageBox.Show("The directory process failed:{0}", ex.ToString());
        }
        finally { }

    }

    private void sensors1_Load(object sender, EventArgs e)
    {

        if (backgroundWorker1.IsBusy != true)
```

```
        {
            backgroundWorker1.RunWorkerAsync();
        }
    }



    //寫入資料
    void WriteData()
    {
        String strTest = "";
        //將字串轉 byte 陣列，使用 ASCII 編碼
        Byte[] myBytes = Encoding.ASCII.GetBytes(strTest);

        Console.WriteLine("建立網路資料流 !!");
        //建立網路資料流
        myNetworkStream = myTcpClient.GetStream();

        Console.WriteLine("將字串寫入資料流  !!");
        //將字串寫入資料流
        myNetworkStream.Write(myBytes, 0, myBytes.Length);

    }
```

```
        //讀取資料
        private void ReadData()
        {
            Console.WriteLine("從網路資料流讀取資料 !!");
            //從網路資料流讀取資料
            while (true)
            {
                int bufferSize = myTcpClient.ReceiveBufferSize;
                byte[] myBufferBytes = new byte[49];
                myNetworkStream.Read(myBufferBytes, 0, 49);
                string kinectdata = System.Text.Encoding.ASCII.GetString(myBufferBytes);
                //取得資料並且解碼文字
                Console.WriteLine(kinectdata);


                z_e_o = z_e;
                x_n = x_data;
                double x_de = x_n - x_o;


                //Match The Date
                string pattern =
@"status=(\d),X=([+-]\d\d\d\d.\d\d)mm,Y=([+-]\d\d\d\d.\d\d)mm,Z=([+-]\d\d\d\d.\d\d)mm";
                MatchCollection matches = Regex.Matches(kinectdata, pattern);
```

```
//match後之動作
foreach (Match match in matches)
{
    double s_data = System.Convert.ToDouble(match.Groups[1].ToString());
    double x_data = System.Convert.ToDouble(match.Groups[2].ToString());
    double y_data = System.Convert.ToDouble(match.Groups[3].ToString());
    double z_data = System.Convert.ToDouble(match.Groups[4].ToString());
    z_e = z_data - z_f;
    double z_de = z_e - z_e_o;

    if (z_e > 450 && z_de > 0)
    {
        m31.Drive(10, 70, 70, Power);
    }
    if (z_e > 450 &&   -25< z_de && z_de <= 0 )
    {
        m31.Drive(10, 50, 50, Power);
    }

    if (z_e > 450 && -50 < z_de && z_de <= -25)
```

```
{
    m31.Drive(10, 30, 30, Power);
}

if (z_e > 450 && -75 < z_de && z_de <= -50)
{
    m31.Drive(10, 22, 22, Power);
}

if (z_e > 450 && -150 < z_de && z_de <= -75)
{
    m31.Drive(10, 0, 0, Power);
}

if (75 < z_e && z_e <= 150 && 150 < z_de )
{
    m31.Drive(10, 70, 70, Power);
}

if (75 < z_e && z_e <= 150 && 75 < z_de && z_de <= 150)
{
    m31.Drive(10, 43, 43, Power);
```

```
}

if (75 < z_e && z_e <= 150 && 50 < z_de && z_de <= 75)
{
    m31.Drive(10, 30, 30, Power);
}

if (75 < z_e && z_e <= 150 && 0 < z_de && z_de <= 50)
{
    m31.Drive(10, 15, 15, Power);
}

if (75 < z_e && z_e <= 150 && -25 < z_de && z_de <= 0)
{
    m31.Drive(10, 0, 0, Power);
}

if (75 < z_e && z_e <= 150 && -50 < z_de && z_de <= -25)
{
    m31.Drive(10, -15, -15, Power);
}
```

```
if (75 < z_e && z_e <= 150 && -75 < z_de && z_de <= -50)
{
    m31.Drive(10, -26, -26, Power);
}

if (75 < z_e && z_e <= 150 && -150 < z_de && z_de <= -75)
{
    m31.Drive(10, -50, -50, Power);
}

if (-75 < z_e && z_e <= 75 && 150 < z_de )
{
    m31.Drive(10, 70, 70, Power);
}

if (-75 < z_e && z_e <= 75 && 75 < z_de && z_de <= 150)
{
    m31.Drive(10, 40, 40, Power);
}

if (-75 < z_e && z_e <= 75 && 50 < z_de && z_de <= 75)
{
```

```
        m31.Drive(10, 30, 30, Power);
    }

    if (-75 < z_e && z_e <= 75 && 25 < z_de && z_de <= 50)
    {
        m31.Drive(10, 15, 15, Power);
    }

    if (-75 < z_e && z_e <= 75 && 0 < z_de && z_de <= 25)
    {
        m31.Drive(10, 0, 0, Power);
    }

    if (-75 < z_e && z_e <= 75 && -25 < z_de && z_de <= 0)
    {
        m31.Drive(10, -15, -15, Power);
    }

    if (-75 < z_e && z_e <= 75 && -50 < z_de && z_de <= -25)
    {
        m31.Drive(10, -30, -30, Power);
```

```
                    }


                    if (-75 < z_e && z_e <= 75 && -75 < z_de && z_de <= -50)
                    {
                        m31.Drive(10, -40, -40, Power);
                    }


                    if (-75 < z_e && z_e <= 75 && -150 < z_de && z_de <= -75)
                    {
                        m31.Drive(10, -70, -70, Power);
                    }


                    if ((z_data > 500 && z_data < 1000 && m31.VelocityLeft != 0) || (x_data > 100 &&
m31.VelocityLeft != 0) || (x_data < -200 && m31.VelocityLeft != 0))
                    {
                        m31.VelocityLeft = 0;
                        m31.VelocityRight = 0;
                        m3 = false;
                        LR = false;
                        m31.Close();
                    }
```

```
if (m3 == false)
{
    m3 = true;
    LR = true;
    m31.Open();
    m31.Connect();
    m31.VelocityLeft = 0;
}

if (x_de < -150 && LR == true)
{
    m31.Rotate(10, 10, Power, PC_BOT.M3.rotate.Anticlockwise);
    //m31.VelocityLeft = Velocity - 25;
    //m31.Drive(10, Velocity - 50, Velocity, Power);
    //LR = true;
}

if (x_de > 150 && LR == true)
{
    m31.Rotate(10, 10, Power, PC_BOT.M3.rotate.Clockwise);
    //m31.VelocityRight = Velocity - 25;
    //m31.Drive(10, Velocity, Velocity - 50, Power);
```

```
                    //LR = true;
                }

                if (x_de <= 150 && x_de >= -150 && m31.VelocityLeft != 0)
                {
                    LR = false;
                    m31.Close();
                }

                if (LR == false)
                {
                    LR = true;
                    m31.Open();
                    m31.Connect();
                    m31.VelocityLeft = 0;
                }

                //下列範例會開啟，或建立檔案 (如果尚不存在)，然後再附加資訊至檔案結尾。
                FileStream fs = new FileStream("C:\\Documents and Settings\\Owner\\Desktop\\Follow You - 改
- 追蹤\\data\\data.txt", FileMode.Append, FileAccess.Write, FileShare.Write);
                fs.Close();
                //將資料錄寫到文字檔
```

```
                    StreamWriter sw = new StreamWriter("C:\\Documents and Settings\\Owner\\Desktop\\Follow
You - 改 - 追蹤\\data\\data.txt", true, Encoding.Unicode);
                    sw.Write("status = "+s_data + "\r\n");
                    sw.Write("x = "+x_de + "\r\n");
                    sw.Write("y = "+ y_data + "\r\n");
                    sw.Write("z = "+z_data + "\r\n");
                    sw.Write("z = "+z_e + "\r\n");
                    sw.Write("z = "+z_de + "\r\n");
                    sw.Write("Power = "+m31.Power + "\r\n");
                    sw.Write("VelocityLef ="+m31.VelocityLeft + "\r\n");
                    sw.Write("VelocityRight ="+m31.VelocityRight + "\r\n");
                    sw.Write(DateTime.Now + "\r\n");
                    sw.Close();
                }
            }
        }

        private void backgroundWorker1_DoWork(object sender, DoWorkEventArgs e)
        {
            client();
```

```
}

private void client()
{
    Form1 myNetworkClient = new Form1();

    //取得主機名稱
    string hostName = "127.0.0.1";

    //取得連線 IP 位址
    int connectPort = 19882;
    //建立 TcpClient 物件
    myNetworkClient.myTcpClient = new TcpClient();
    try
    {
        //測試連線至遠端主機
        myNetworkClient.myTcpClient.Connect(hostName, connectPort);
        Console.WriteLine("Connection Successful !!\n");
    }
    catch
    {
        Console.WriteLine
```

```
                    ("主機 {0} 通訊埠 {1} 無法連接  !!", hostName, connectPort);

                return;

            }

            myNetworkClient.WriteData();

            myNetworkClient.ReadData();

            Console.ReadKey();

        }


    }

}
```