中華民國第 58 屆中小學科學展覽會

作品說明書

高級中等學校組　工程學科(一)科

第三名

052302

未知環境中自主探勘、定位及導航之整合系統

學校名稱：臺北市立建國高級中學

作者：	指導老師：
高二 王維恩	許雅淳

關鍵詞：ROS、SLAM、particle filter

摘要

本研究透過ROS(機器人運作系統)實作一個能自由探勘且進行SLAM (同步定位及地圖構建)，且使用自身建構出的地圖進行導航任務之機器人系統。將探勘與定位領域結合，機器人便可以在面對完全未知的環境有一套應對措施。利用ROS機體與程式軟體分離的優點，本研究開發出的系統可以運用於任何能安裝ROS或與ROS連結之機體，供各式各樣的服務做為一系統性的路徑規劃器及運動基座。並探討此系統擅長及較不適用的環境，呈現目前機器人系統的運作能力。而日後在各區塊的技術精進，都能分別套用在此系統上，而不需要重新建構整個架構。

壹、 研究動機

近年來機器人在各領域都有長足的進步，能夠協助人們在生活及工作中的需求，以更節省資源、更有效率的方式完成任務。在各方面研究的蓬勃發展下，透過不同領域間的融合而產生的新技術更是不計其數。其中，SLAM(simultaneous localization and mapping)便是結合感測器、運動學、位置辨識等諸多領域及演算法，使機器人能在缺乏由外界提供的資訊下更好的完成定位以及繪製地圖的任務，無需仰賴以往廣泛使用的GPS或無線定位系統。

另一方面，機器人的自主探勘能力也是我們非常感興趣的。在未知或危險的區域人無法靠自己去執行任務時，機器人卻不受這些限制。探勘的過程似乎與SLAM的目標不謀而合，但目前為止運行SLAM建立地圖的過程中大多仍需要人的操控，如此將無法完全利用機械的優勢；若能從一開始建立地圖時即全權由機器人的演算法決定，就能使機器人具備完整的自主探勘及後續局部建立地圖及定位的能力。

因此，我們希望能實作出一個結合探勘與SLAM兩大領域，同時能獨立進行導航的機器人系統。

貳、 研究目的

本研究希望機器人能達成以下目標：

一、 在未知環境中以機器人本身的感測器資訊，自主決策探勘方向及順序。

二、 基於機器人本身的觀察，建立所要探勘區域之完整實況地圖

三、 無外界資源下，使用先前建好的地圖，以蒙地卡羅定位法提供機器人在地圖中的定位。

四、 在地圖建立完成後，執行基於此地圖的導航任務。

參、 研究設備及器材

一、ROS(Robot Operating System)

本系統的開發環境，主要支援 C++ 及 Python 兩種程式語言，使用版本為安裝在ubuntu 16.04的ROS Kinetic。

二、Pioneer P3-DX

本研究所使用的實體移動機器人平台。左右各有一個驅動輪，後面有一個輔助輪。

圖1、Pioneer P3-DX

三、SICK LMS100 laser

Pioneer本身沒有加裝雷射測距儀，所以外加此雷射於Pioneer上，用以執行感測周圍環境之任務。

圖2、SICK LMS100 laser

肆、 研究過程或方法

一、環境介紹

本研究使用ROS的原因是在ROS平台執行的程式(這樣的程式通常稱為一個node)之間可以很容易地互相溝通及傳輸資料(在node之間傳輸的訊息稱為topic)，且能在僅僅調整些微系統參數的情況下，以不同的機器去實現同樣的演算法。而現今網路上已有很多其他人發展出的package(程式包) ，因此我們並不需要為研究中的作品每一部分重新開發程式，只需稍加修改；再者，ROS把硬體的控制區塊和數據處理與決策的區塊分離，所以我們可以使用模擬器去替代實際的機械，去除實驗進行的突發狀況引起的危險及提高測試效率。

(一)、Gazebo(version 7.0)

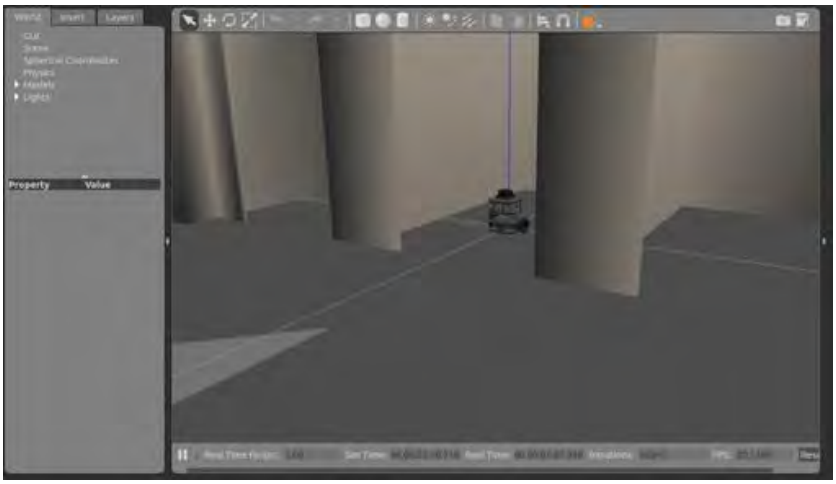


圖3、Gazebo的介面

在使用真實的機器人前，本研究先以Gazebo做為模擬器，用來在虛擬環境中實驗觀察程式有無達成我們的目標。Gazebo模擬器與其他模擬器相比有以下幾個優點：

1. 能夠跟ROS配合的很完美
2. 可以很迅速地建立出實驗的環境
3. 物理引擎優秀，能很好的模擬真實世界的狀況。
4. 靈敏度高，要更動環境時很容易。

Gazebo的機器人模型本身有配備雷射測距儀，所以開啟Gazebo後可直接進行後續的作業。

(二)、RosAria

RosAria作為將Pioneer連接上電腦的**node**，可以透過眾多**topic**將Pioneer本身的資訊或感測器的資訊傳輸給電腦；也能接收/RosAria/cmd_vel這個**topic**的資訊，並以此控制Pioneer的速度。

(三)、tf

在一個機器人系統中，各個不同感測器量測到的數據，都是相對於感測器自身的立體坐標系；然而不同的感測器位於不同的位置，而且會因為各部位的運動，隨時間改變在絕對坐標系中的位置。tf程式包的功能就是讓這些不同坐標系中的資訊能互相轉換。而系統中正在運作的tf可以用rqt_tf_tree來觀察。

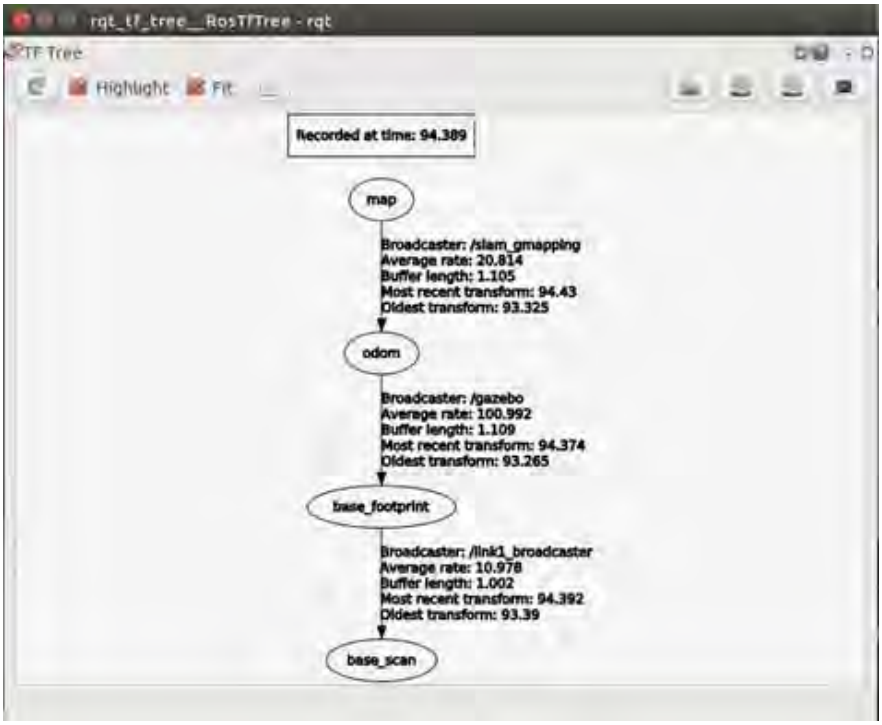


圖4、本系統的rqt_tf_tree

(四)、rviz

ROS中內建的程式，用以將各個topic的資料以圖像的方式呈現。在本研究中透過rviz就能直接觀察機器人運作的整個過程，即時地檢視實驗是否成功。

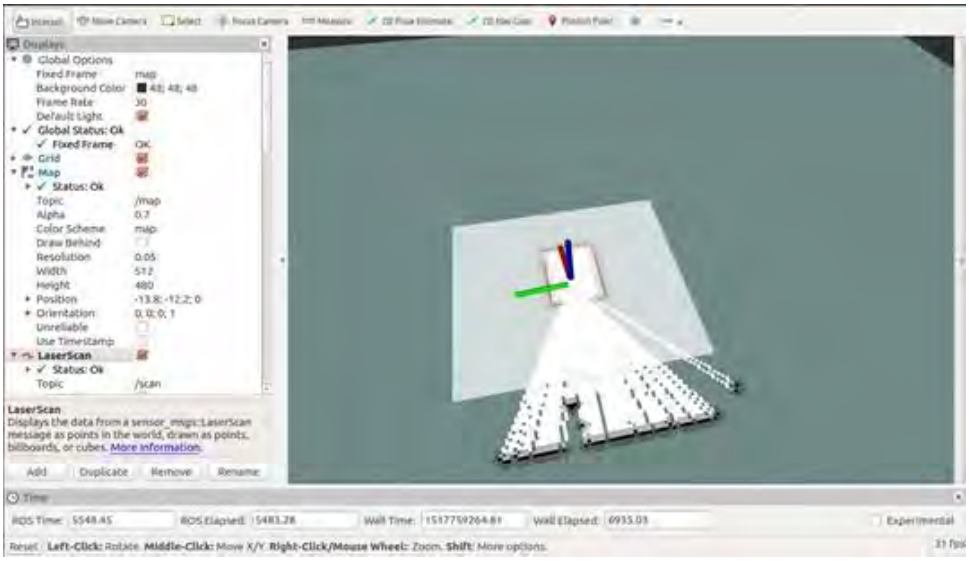


圖5、rviz的介面

(五)、將Pioneer連上ROS

1. 啟動ROS。
2. 將Pioneer以USB線連接上電腦後，執行 sudo chmod 777 /dev/ttyUSB0開啟Pioneer資訊的權限。
3. 將雷射接到電腦的網路線孔，並編輯網路連線

圖6、編輯雷射網路連線

執行 rosrun lms1xx LMS1xx_node _host:=192.168.11.100 即可產生/scan的雷射資料。

4. 執行 RosAria。

圖7、架設完成的 Pioneer P3-DX

(六)、手動控制機器人的移動

本研究是研究全自動之機器人，故需要一種手動操作方式，以用來比較自主移動及手動移動之間的差別；或是實驗過程中機器人發生問題，必須仰賴人力介入。在這裡為方便起見使用ROS內建的teleop_twist_keyboard，此node會將鍵盤資料轉換成速度資訊傳至/cmd_vel這個topic中。

圖8、teleop_twist_keyboard輸入窗口

二、理論基礎

(一)、DWA(dynamic window approach) [2]

自走機械的避障能力在任何需要運動的場合下都非常重要，沒有避障能力的機器人會在工作過程中撞到障礙物而損壞。DWA演算法透過測距感測器(/scan)及路徑感測器(/odom)，經過運算後產生控制機體速度及角速度的策略，使機器人能在避障的同時成功達到目的地。每一個時間點下的速度由幾項因素決定：

1. 速度空間的選取

首先，速度空間的範圍由機體的目前加速度及角加速度決定，這些參數是可以經由讀取機體發出的加速度 topic 得知。速度空間V_d的範圍由下式描述：

$$V_d = \{v, \omega | v \in [v_n - \dot{v}t, v_n + \dot{v}t] \text{^} \omega \in [\omega_n - \dot{\omega}t, \omega_n + \dot{\omega}t]\}$$

v_n, ω_n分別為目前的速度及角速度

$\dot{v}, \dot{\omega}$分別為目前的加速度及角加速度

t為每次指令之間的時間間隔

當然，機器人在運行過程中要避免撞到障礙物，在這情況下允許的速度範圍V_a由運動學方程可以得出：

$$V_a = \left\{v, \omega | v \leq \sqrt{2dist(v,\omega)\dot{v}_b} \text{^} \omega \leq \sqrt{2dist(v,\omega)\dot{\omega}_b}\right\}$$

$\text{dist}(v, \omega)$ 為機體在此路徑下與最近的障礙物之間的距離

$\dot{v}_b, \dot{\omega}_b$分別為機器人可產生的最大加速度及角加速度

由以上的討論，機器人這次指令下的速度必須存在於上面兩速度空間與機器人本身可允許的速度空間V_s之交集V_r：

$$V_r = V_s \cap V_d \cap V_a$$

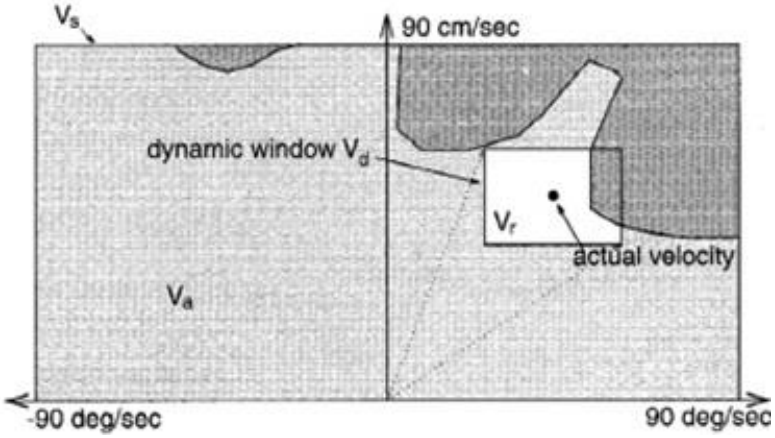


圖9、V_r示意圖

2. 目標函數$G(v,\omega)$的最佳化

在不發生碰撞的情況下，能最快抵達目標點的速度取值顯然是最好的選擇，而這可以用三個參數決定：

(1) 機器人的走向與目標點的方向匹配度 $h(v,\omega)$

(2) 與障礙物的距離 $dist(v,\omega)$

(3) 機器人在軌跡上的切線速率 $velo(v,\omega)$

將這些函數標準化使之數值皆落在$[0,1]$區間內；由此，最佳函數$G(v,\omega)$可寫為：

$$G(v,\omega)=\alpha\cdot h(v,\omega)+\beta\cdot dist(v,\omega)+\gamma\cdot(v,\omega)$$

α,β,γ為系統調整參數

而在速度空間V_r中，能使$G(v,\omega)$得到最大值的速度取值就是DWA將下的速度指令。舉例來說，若希望機器人有更好的避障機能，則可以將β參數放大；若系統環境較空曠，則將γ參數放大。

(二)、A* search algorithm

A*演算法常被用在限制條件下(如地圖中有障礙物)尋找最短路徑的問題中。

1. 計分函式

若以$G(N)$代表問題中的起點到節點N的實際距離，$H(N)$代表節點N到終點的估算距離(如直線距離，也可以是其他的簡易算法)，則A*演算法的計分函式$F(N)$為：

$$F(N)=G(N)+H(N)$$

2. 節點集合

在此演算法中節點可分為兩個集合，一為開放集(open set)，另一為關閉集(close set)。初始時關閉集僅包含起點，開放集包含鄰近起點的節點(並計算各個節點的計分函式)。路徑判斷的過程即是持續的重複以下三個步驟：

(1) 找出目前open set中計分函式F最小的節點，令此節點為S。

(2) 將S移入close set中。

(3) 對於鄰近S的每個節點K，若K

a. 存在於close set中：直接進行下個迴圈。

b. 不存在於open set或close set中：將之納入open set中並計算其計分函式。

c. 存在於open set中：

比較K原先的計分函式與根據現在路徑所計算出的計分函式，若新的$F(K)$較小，則更新$F(K)$。

抵達終點時即結束迴圈。

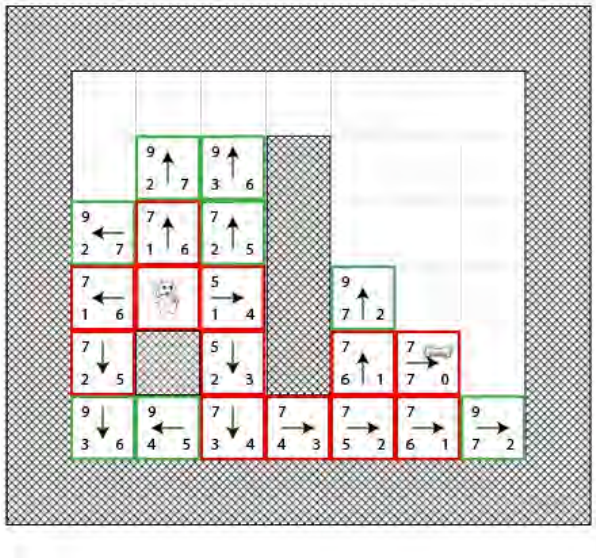


圖10、A*演算法例子

以圖10為例，每個方塊為一個節點。一個節點相鄰的節點即是除了牆壁以外相鄰的方塊。

方塊中左下角的數字為G，右下角的數字為H，左上角的數字為F(此例中估算距離的方式即是計算無障礙下的距離)。

綠色的方塊為open set的節點，紅色的方塊為close set的節點。箭頭指的是路徑的方向。而找到終點後，從終點沿著箭頭的反方向走即可求得最短路徑。

(三)、POMDP(部分可觀測Markov決策過程)

MDP(Markov決策過程)是面對部份隨機，部份可由決策者控制的狀態下，找出最佳化評分函數的模型。而POMDP與MDP不同的地方則在於POMDP問題中，無法直接的測量系統的狀態，測量與狀態本身有一定的不確定性存在，與機器人所面對的各種問題相同。

MDP與其他最佳化模型的不同點在於，MDP問題可以視為一個遞迴過程，即現在的狀態只與前一步的狀態有關，而不需考慮之前任何的變化，減少計算量的需求。

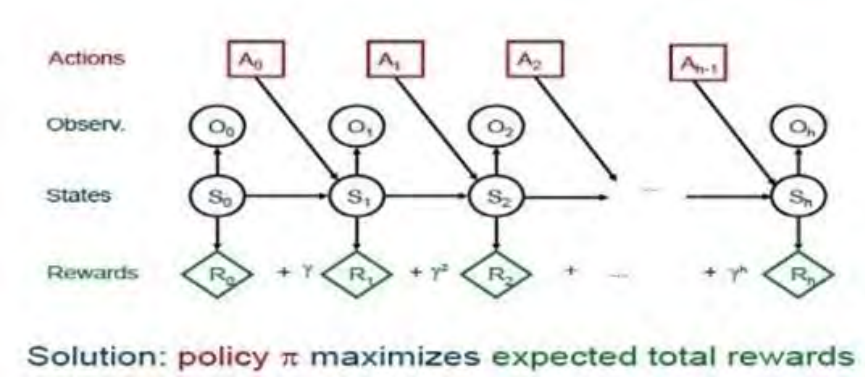


圖11、POMDP決策過程

(四)、Bayes filter

在機器人領域中誤差的處理非常重要，原因是感測器在測量環境數據時，一定會產生誤差，而且以現今的測量技術這樣的誤差就足以導致錯誤的決策。考慮量測值誤差分布的情況下，我們必須用「機率」的概念來描述我們所要量測的狀態。而時刻 t 時狀態 x 的機率分布$\mathrm{Be}(x_t)$定義為：

$$\mathrm{Be}(x_t) = p(x_t|z_{0:t}, u_{0:t})$$

其中$z_{0:t}$為從初始到時刻 t 所有的測量，$u_{0:t}$為從初始到時刻 t 對系統做的所有操作。Bayes filter則透過遞迴的方式尋找出目前狀態的機率分布，從而求出所要的狀態。我們假設此狀態符合POMDP模型，則Bayes filter可經由兩個步驟完成狀態的迭代：

1. Motion model

機器人透過路徑感測器得知u_t，以$\mathrm{p}(x_t|x_{t-1}, u_{t-1})$表示運動學推得的狀態轉移，而對於所有的$x_{t-1}$都會有相應的轉移狀態機率分布，所以全部的加權總合就是運動

學模型給出的第一次狀態迭代：

$$Be_0(x_t) = \int p(x_t|x_{t-1}, u_{t-1}) \cdot Be(x_{t-1})\, dx_{t-1}$$

2. Observation model

Bayes filter的第二步驟則由機器人的環境感測器(雷射、聲納)修正motion model給出的狀態機率分布。以$\mathrm{p}(z_t|x_t)$表示感測器的測量分布，修正後的狀態機率分布為：

$$\mathrm{Be}(x_t) = \alpha \cdot p(z_t|x_t) \cdot Be_0(x_t)$$

其中α為狀態機率分布的歸一化常數。

應用上可根據需求使用不同的方式實現Bayes filter，particle filter便是其中一種方法。

(五)、particle filter

particle filter以隨機取樣(sampling)得出離散的機率分布(particles)，更新狀態時以三個步驟進行。

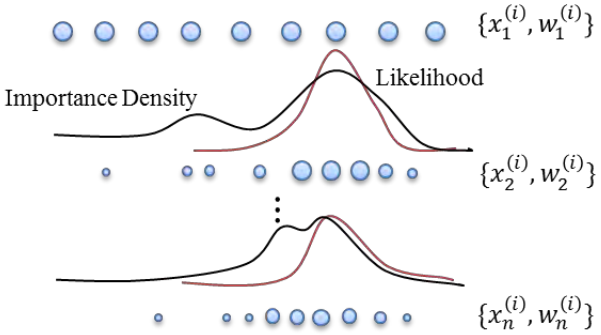


圖12、particle filter

1. sample

particle filter以隨機取樣(sample, particle)及每個樣本的權重去描述機率密度，所以particle filter的第一步即是在機率空間中取樣(如：以感測器量測)，若取的樣本數越多，則越能正確的描述機率密度函數。

2. weight

對系統做一次操作後，先以理論模型去預測系統的變化，若我們假設操作後的機率分布函數為g，實際取樣的分布函數為f，則權重w定為f/g。

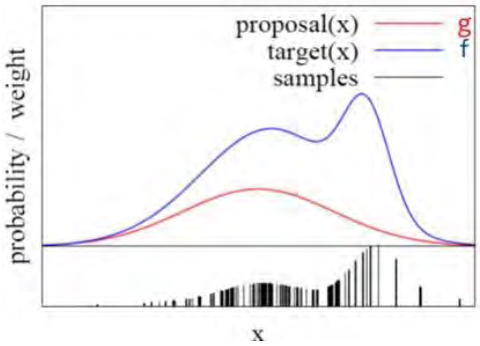


圖13、sampling, weighting示意圖

3. resample

接著去除權重較小的狀態(即可能性較小的狀態)，留下可能性較大的狀態集合；然後將所有的權重皆規一化，再重複做第一步驟。經過數次的循環後，樣本數即會收斂到接近真實的狀態。

三、系統架構

以上介紹了本研究的理論基礎，而實驗的目的即是設法將這些理論結合，使系統不單單具備一項能力。在開發系統之前，必須先明瞭每一個小系統在整體中的功能及定位。整體而言，系統可以由下圖說明：

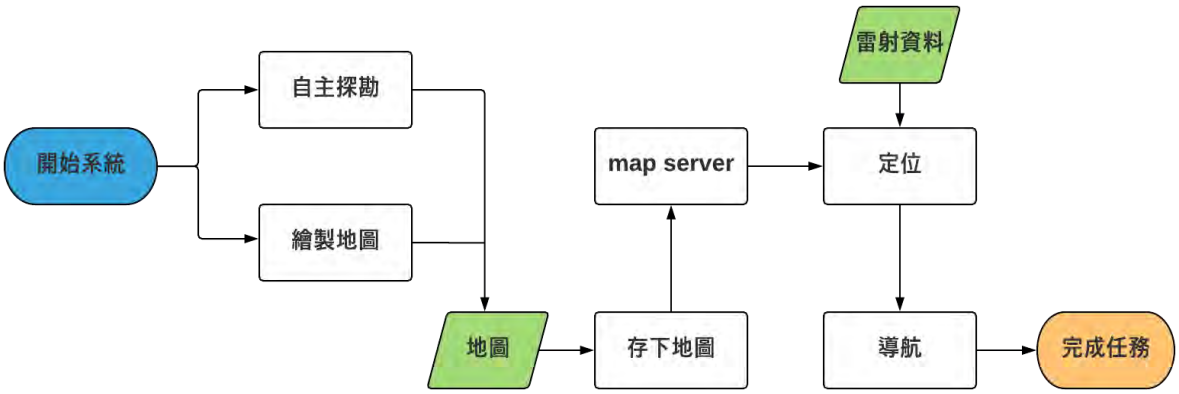


圖14、系統架構圖

(一)、自主探勘

選定好要進行作業的空間範圍後，接著就是自主探勘的過程。自主探勘包含兩個主題；第一是避障的問題，第二是決定移動的策略：

1. 避障(使用DWA)

2. 移動策略

規劃策略的部分本研究使用[3]中所提出的ase_exploration，此算法將探勘過程視為一個POMDP問題，而所求出的路徑要滿足在其他可能決策中，得到的地圖信息是最大的。其中，找尋路徑的方式則是利用particle filter：以控制速度與路徑當作進行迭代的狀態，經過數次迭代後，以權重最大的路徑及速度作為探索的指令。最後將控制速度的指令傳送至 move_base 程式包的node中，機器人就會依指令執行探勘任務。

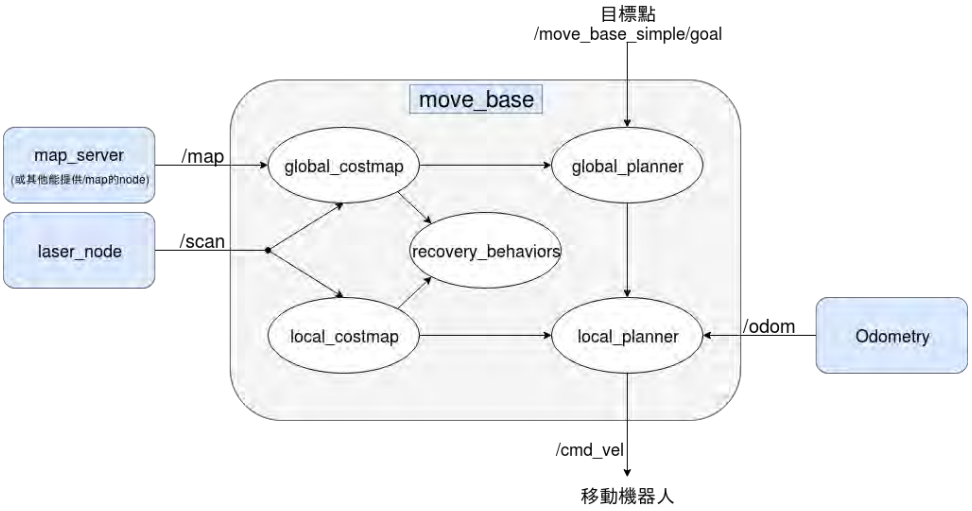


圖15、move_base中的資訊流動

首先，將一個既有的/map的topic套用進全域的costmap。costmap是ROS以數值來表示地圖中障礙物的map，一般來說障礙物所在的位置costmap會以254代表，而空曠的區域則為0。透過雷射資訊可以隨時建立出目前的costmap，判斷地圖中哪些區域容易碰壁，哪些區域是空曠的。而路徑規劃器(planner)直接依據costmap規劃路徑，global planner以達到終點為目標建立路徑，local planner則配合路徑感測器得知機器人實際的軌跡建立安全的路徑。最終得到的速度指令藉由/cmd_vel這個topic直接控制機器人。

當機器人判斷自己「困住」的時候(如距離障礙物太近)，move_base的另外一個topic，recovery_behavior，即會以圖17的步驟設法解決：

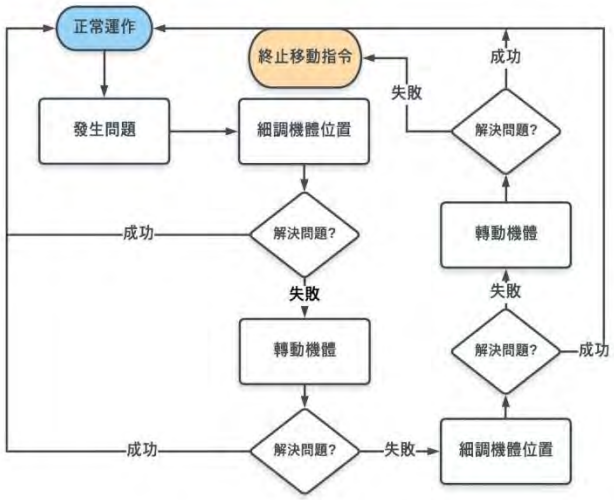


圖16、recovery_behavior示意圖

(二)、繪製地圖

透過機器人的雷射掃描環境，可以偵測環境中的障礙物與空曠的地方。ROS的官方網站上有Gmapping的程式包能夠拿來做SLAM，所以為方便起見我們在本研究中使用Gmapping來做SLAM，在邊移動邊掃描的情況下建立地圖。

圖17、Gmapping(rviz)與真實地圖

如圖17，Gmapping在運作時，會將空曠的區域在rviz上以白色上色，黑色則代表障礙物或牆壁，灰色則代表尚未探索的區域。

Gmapping與前面的移動策略同樣是利用particle filter來進行雷射資訊及機器人運動模型的匹配，而為了增進運動模型的精確度，其中在單純的運動學之外還經過了

EKF(extended Kalman filter)的處理。除此之外，Gmapping在得知機體位置的同時，尚具備建立地圖的任務，在這樣的情況下一般的particle filter能力有限，必須以改良後的Rao-Blackwellized particle filter來處理 [4]。此改良成功的將建立地圖及定位兩狀態分開處理，且減少particle filter取樣與再取樣的次數及數量，使機器人能夠在雷射測距器掃描後即得到準確的地圖，幾乎不需後製時間。

(三)、定位

地圖建立好之後，機器人便可以結合自身藉由測距器得到的周圍環境資訊與整體地圖的資訊，去比對並找出自己在地圖中的位置，達成定位的目的。這裡採用的機制為AMCL(adaptive monte-carlo localization)，此定位法即是在利用particle filter推算機體位置(狀態)的同時，視定位狀況變更取樣(particles)的數量以減少計算量，提升運作效率。

(四)、導航

若以上步驟皆成功達成，機器人基本上對於環境就有全面性的了解。任何機器人在地圖上有能力抵達的座標點都能使用前面提及的move_base抵達，當我們對電腦輸入目標座標後，機器人就會規畫路徑抵達，完成任務。

在導航任務中，global planner使用A* search algorithm，local planner則利用DWA。

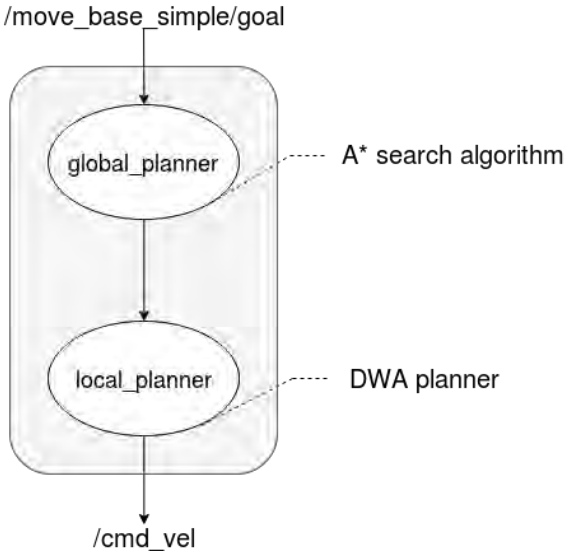


圖18、路徑規劃器所使用的演算法

由於在規畫路徑時擁有的資訊量遠多於自主探勘時的情況，所以執行導航任務的時間遠比探勘所花的時間還要少。

四、實驗流程

(一)、launch file

在ROS中可以用launch file的方式同時啟動多個node，將同時會用到的node放在同個launch file中可以節省許多功夫；launch file中也能將系統參數一併寫入，所以修改參數時不必重新編輯及編繹程式。

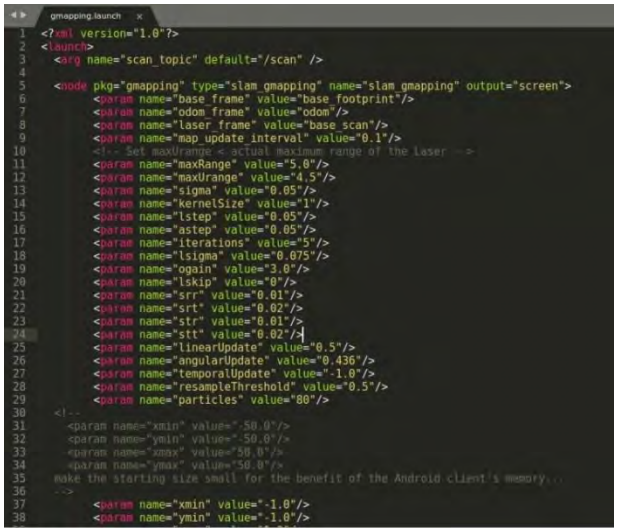


圖19、本研究使用的gmapping launch file

本研究中的實驗分成兩種，其一為在Gazebo模擬器上運作的實驗；其二為使用 Pioneer P3-DX之實驗。實驗則分為兩部分進行，第一部分為自主SLAM與普通的SLAM之間的比較，第二部分為定位及導航的評估。

(二)、實驗第一部分

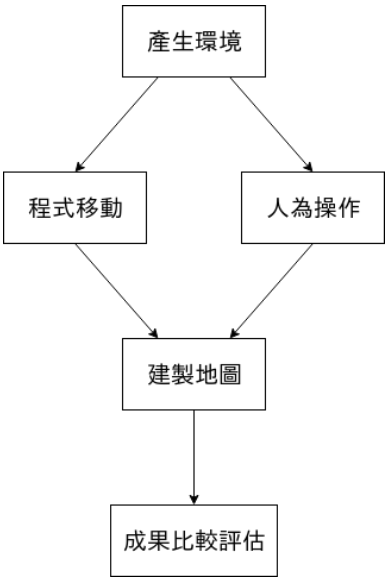


圖20、實驗第一部分流程

1. 產生環境

   用不同的環境觀察在哪些情況下系統運作良好，哪些容易發生錯誤。

2. 運作過程

   (1) 依照前面提及的系統架構圖運作，觀察參數的變化對實驗的影響，不同的環境中機器人的反應。

   (2) 探勘與SLAM完成後，以ROS的map_server程式包中之map_saver將地圖存下，供後續的定位及導航時使用。

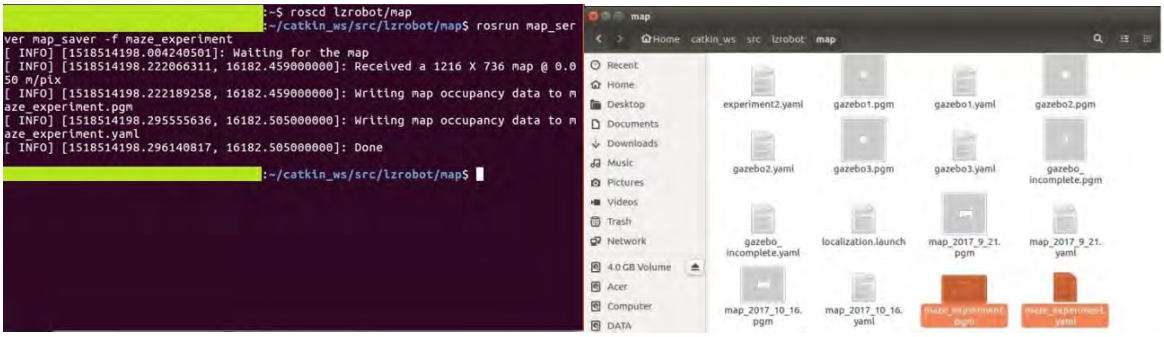


圖21、使用map_saver儲存SLAM建好的地圖

3. 結果評估

   (1) 計算自主探勘SLAM在不同環境中的發生問題的頻率。

   (2) 比較自動與人為操作產生之地圖的完成率。

   (3) 以實測結果分析系統參數對系統執行任務的影響。

(三)、實驗第二部分

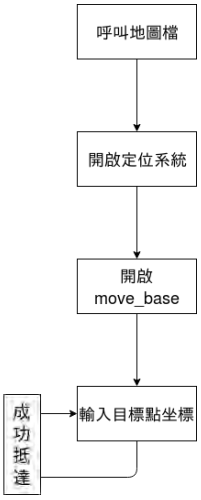


圖22、實驗第二部分流程

1. 利用map_server將先前建好的地圖與ROS連結。
2. 開啟定位系統

   因為amcl必須仰賴運動模型的迭代才能定出正確的位置，所以剛開啟不能馬上得知機器人位置。
3. 完成導航任務

   若定位成功，則開啟move_base後，任意輸入地圖上的點，機器人都可以抵達。
4. 結果評估

   實驗第二部份成功的關鍵就是能夠定位成功，依實驗結果，探究以下兩點：

   (1) 不同環境下之定位成功率。

   (2) 成功定位後，導航任務之效率及準確率。

伍、 研究結果

一、在Gazebo上運行

本研究使用willow garage地圖進行實驗，因為此地圖面積很大，為方便起見，即以地圖不同區域當做不同環境，將地圖中要探勘的區域用障礙物圍起來。

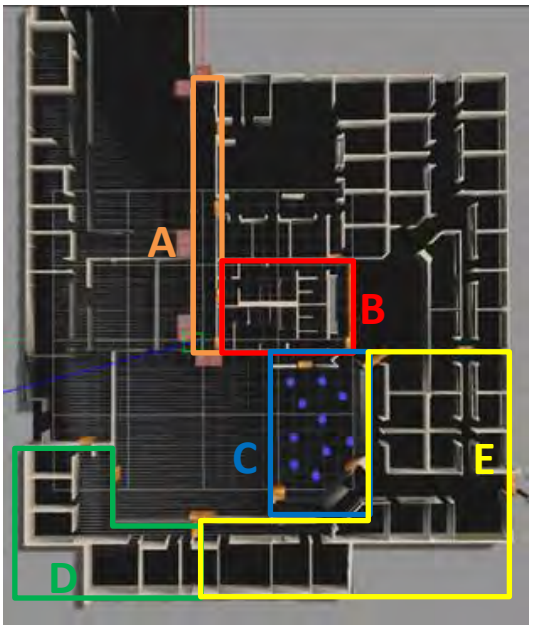


圖23、willow garage實驗區域(以不同色框分別)

Gazebo中機器人的模型以差動輪進行轉彎，附帶雷射測距儀。設定最大速率為0.50m/s，最大角速率1.0rad/s。

(一)、自主探勘問題發生情況

1. A區(長廊)

在長廊的環境中，本系統可以安全的完成任務，問題發生率為0。

2. B區(多房間區域)

在運行過程中平均遇到約三次問題，通常是要離開房間時，與牆壁距離太近造成系統中斷，仰賴 move base 中的 recovery behavior 進行恢復。

3. C區(空曠環境中添加障礙物)

多次實驗問題發生率低，問題原因主要是設定的運行速度太快，雷射數據更新後的運算還未來的及實際的運動及產生擦撞，以較低的速度運行就不會發生問題。

4. D區(轉角處)

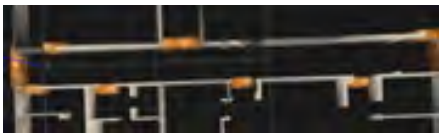
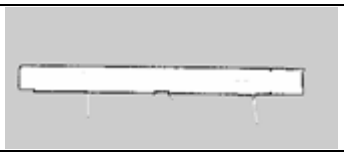
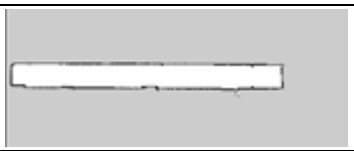
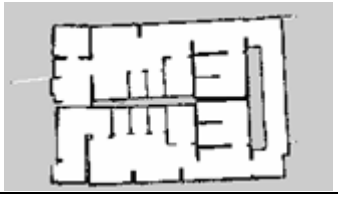
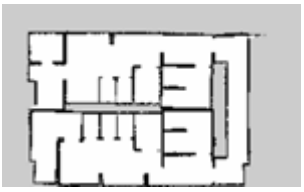
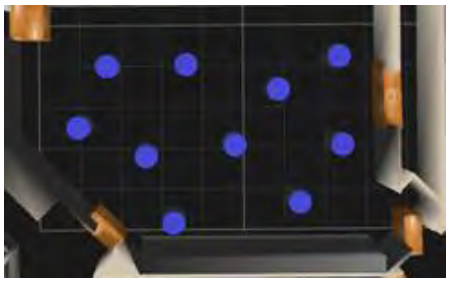
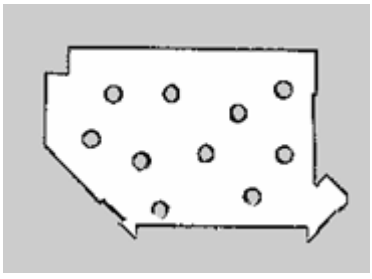
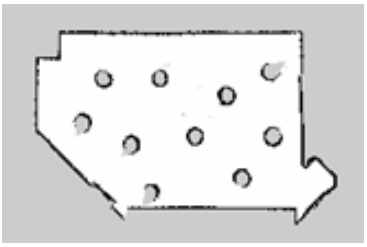
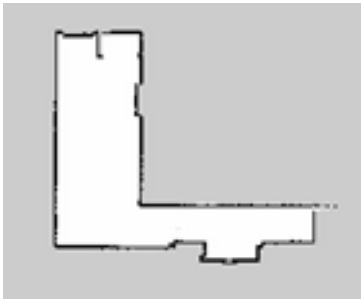
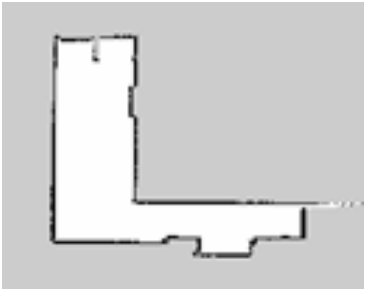
相對於機體大小較大的彎道機器人都能順利通過。

5. E區(多種環境區域)

有時候探勘完畢後並沒有將房間內部完全建置於地圖上。

(二)、人為操控與自主決策對比(同出發點)

表1、各區域自主SLAM建製地圖

	真實地圖	人為操控機體	機器人自主決策
A區			
花費時間		1分58秒	3分58秒
白色pixel數(完成率)		9515	9666(100%)
B區			
		7分21秒	13分9秒
		16794	16453(98.0%)
C區			
		3分43秒	6分11秒
		16743	16744(100%)
D區			
		1分3秒	2分45秒
		8210	8216(100%)

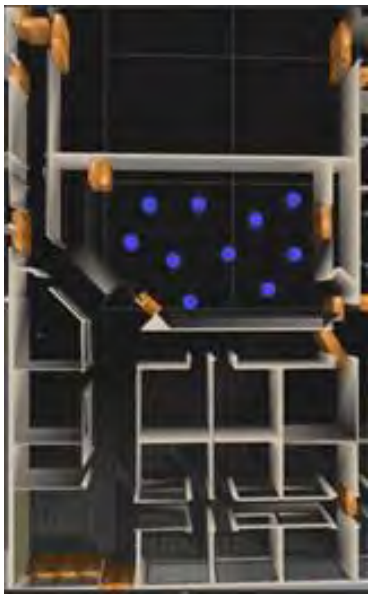
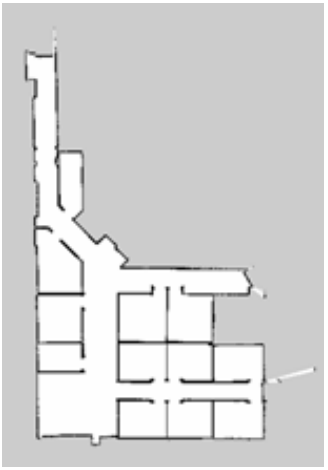
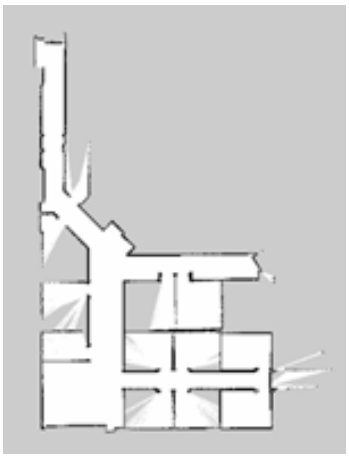
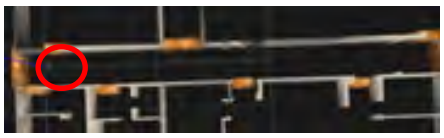
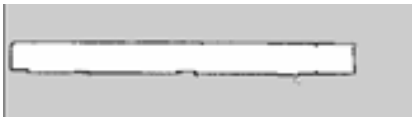
E區			
		11分56秒	19分22秒
		45991	39882(86.7%)

表1中以建置出的地圖之白色格點數作為探勘完成度的判據，圖中有些多餘的白色區塊，成因為障礙物並未遮好後方區域，但這樣的區塊面積很小所以可以忽略其對完成度的影響。

(三)、探勘起點的選擇與自主探勘效率

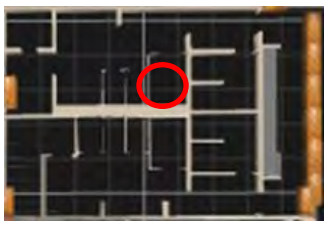
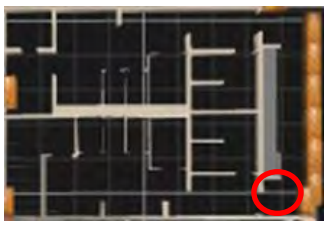
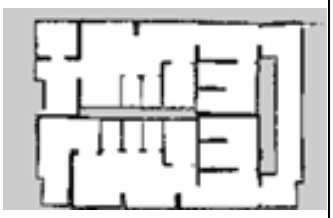
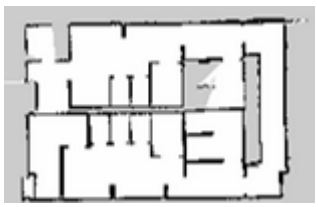
1. A區

表2、A區起點不同之第一部分實驗結果

起點		
花費時間	3分58秒	9分51秒
問題發生次數	0	0
繪製的地圖		
完成度	9666(100%)	9746(100%)

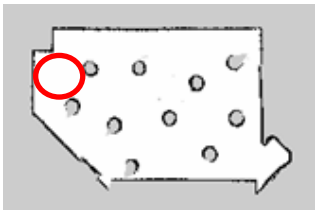
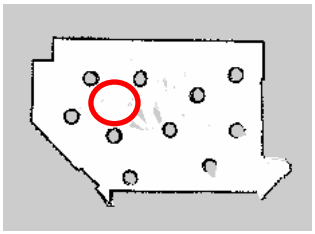
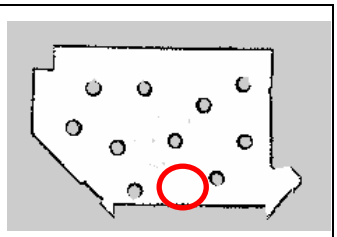
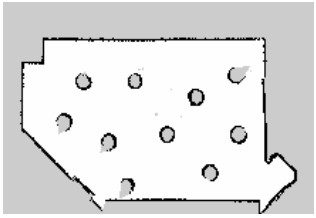
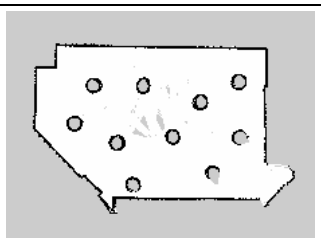
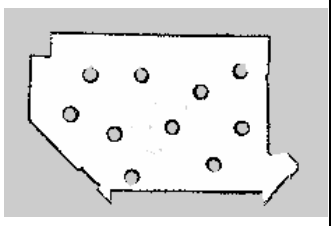
2. B區

表3、B區起點不同之第一部分實驗結果

起點			
花費時間	13分9秒	17分1秒	9分48秒
問題發生次數	2	4	1
繪製的地圖			
完成度	16453(98.0%)	15978(95.1%)	9938(59.2%)

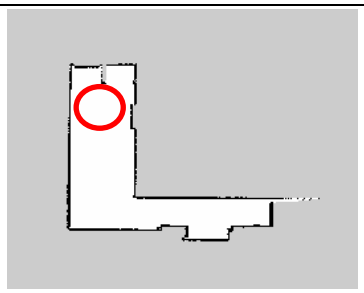
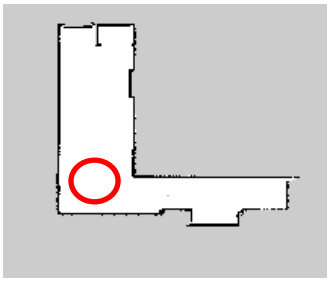
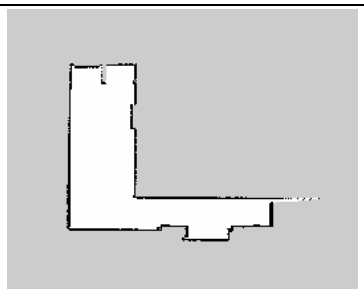
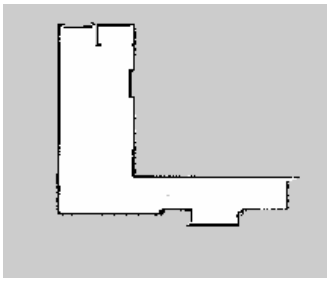
3. C區

表4、C區起點不同之第一部分實驗結果

起點			
花費時間	6分11秒	6分2秒	5分41秒
問題發生次數	0	1	0
繪製的地圖			
完成度	16744(100%)	16564(98.9%)	16617(99.2%)

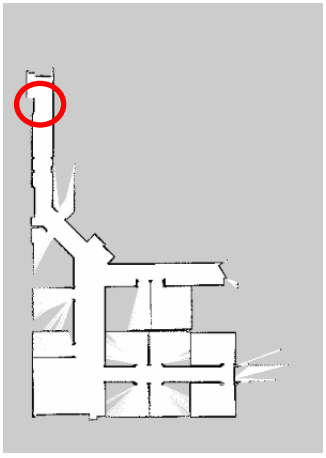
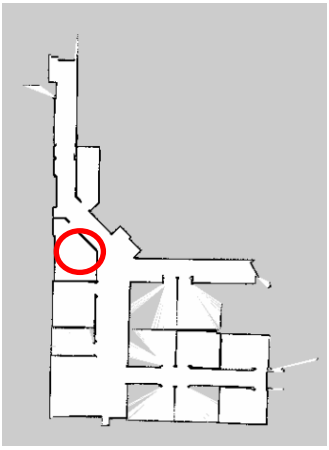
4. D區

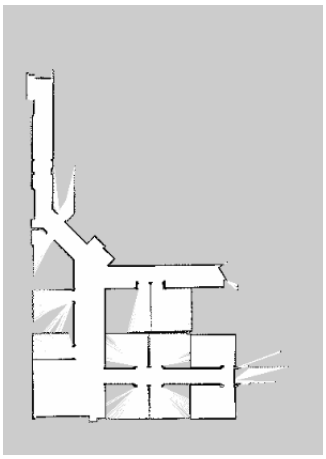
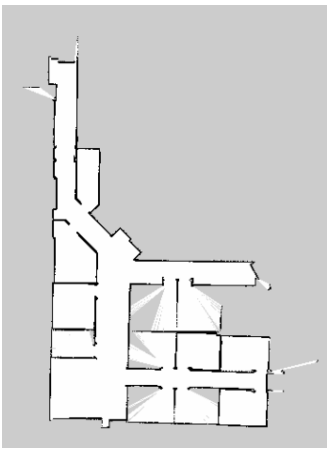
表5、D區起點不同之第一部分實驗結果

起點		
花費時間	2分45秒	3分28秒
問題發生次數	0	0
繪製的地圖		
完成度	8216(100%)	8235(100%)

5. E區

表6、E區起點不同之第一部分實驗結果

起點		
花費時間	19分22秒	18分10秒
問題發生次數	1	0

繪製的地圖		
完成度	39882(86.7%)	43289(94.1%)

二、在實體環境中運作

圖24、自主SLAM在Pioneer P3-DX上的運作情形

如圖24中的電腦螢幕所見，機器人經過的地方可以立刻畫出地圖，並在運行中可以同時避障。

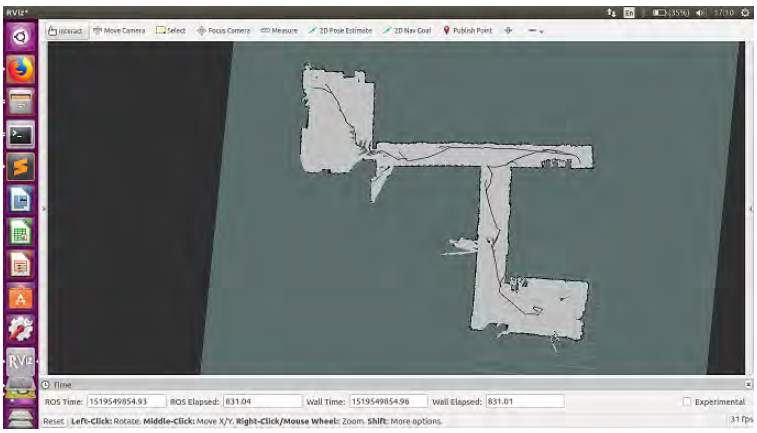


圖25、自主SLAM在實際環境中繪製的地圖與行經路徑

三、定位情形

(一)、空間相似度高區域(A區，B區)

依照AMCL(adaptive monte-carlo localization)進行的定位容易在這種區域失敗，原因為在進行雷射資料比對時會有很多相似的結果，導致系統無法判斷何為正確的位置。如圖26，定位出的位置落在一條稍長的帶狀區域：

圖26、A區的定位情況

(二)、空間相似度低區域(C區，D區，E區)

amcl很快就能將位置收斂到正確的區域。

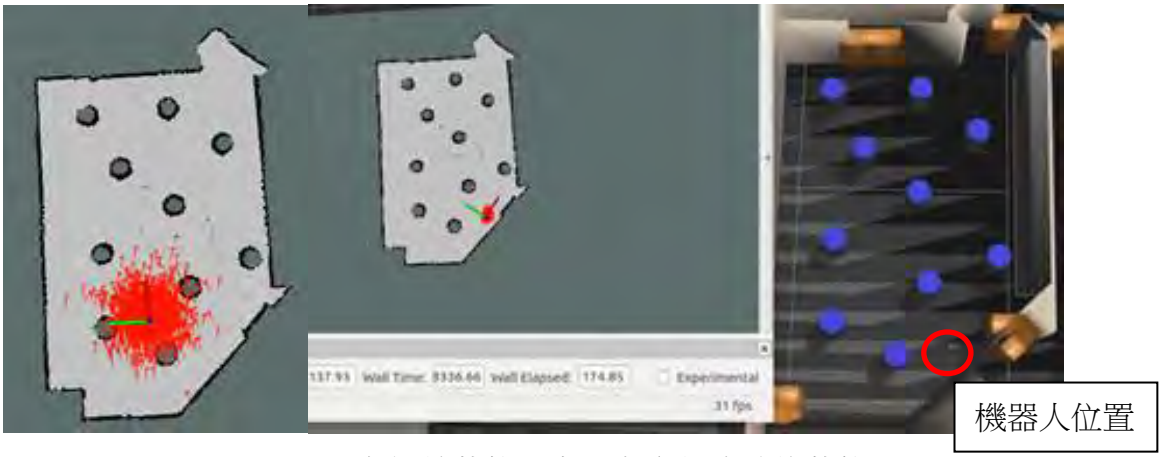


圖27、左圖為初始狀態，右圖為定位成功的狀態

四、導航效果

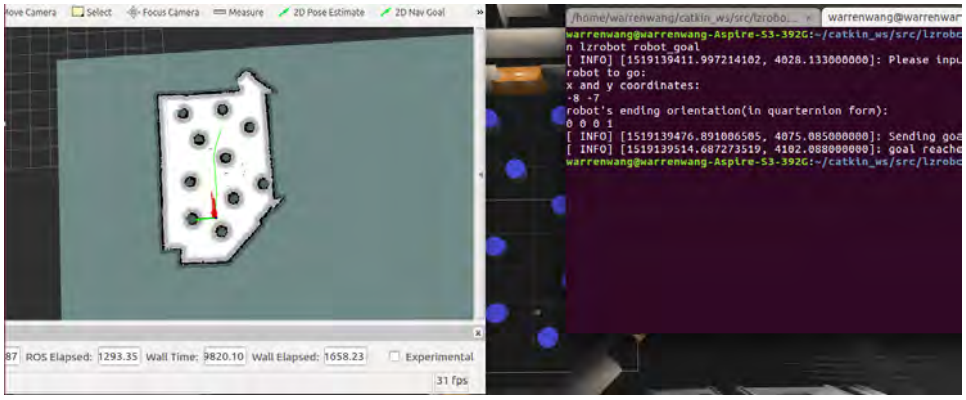


圖28、輸入要抵達的點座標、成功抵達目標點

如圖28，本系統之導航功能可以成功且很快的規畫路徑及移動到目的地。

陸、討論

一、不同區域下自主SLAM的運作結果

由實驗結果可以觀察到本系統對於太狹窄(相對機體而言)的空間較容易發生失誤，其他環境都有不錯的結果。

二、路徑規劃器參數

在本研究中關注幾個對自主SLAM成功率與效率影響較顯著的參數：

(一)、costmap參數：

1. inflation radius：costmap中把障礙物的開始納入考量的最遠距離。

2. cost scaling factor：costmap中數值以指數下降的倍率，越大則costmap中判斷為「危險」的區域越少。

(二)、path planner參數：

path_distance_bias、goal_distance_bias、occdist_scale分別對應到DWA目標函數中的三個調整參數。

1. path_distance_bias對應機體在軌跡上的切線速率。

2. goal_distance_bias對應機體的走向與目標點的方向匹配度

3. oddist_scale對應路徑上與最近障礙物的距離。

以預設的參數值無法通過狹窄的門口，因costmap的危險區域會包含門口。

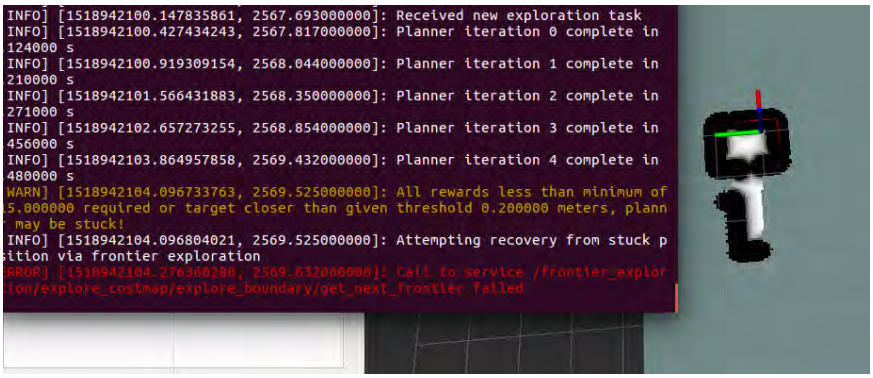


圖29、inflation radius太大引起的costmap錯誤

表7、預設的參數

inflation radius(公尺)	0.50
cost scaling factor	10.0
path_distance_bias	32.0
goal_distance_bias	10.0
occdist_scale	0.01

表8、調整後參數

inflation radius(公尺)	0.20
cost scaling factor	15.0
path_distance_bias	20.0
goal_distance_bias	10.0
occdist_scale	0.05

主要改變為將不能通行的區域變小，使其能穿過窄門口。但這樣調整的同時也會使避障的能力降低，故調高occidst_scale。

三、定位能力

為了降低系統的運算量，本系統會將可能的位置散布於設定的點附近，所以運用此定位法時，盡量將機器人位置設定於原點附近會容易成功。當然實際使用時，我們通常都會知道起點的位置，也就能將amcl設定於起點位置；而若是要對全地圖做particle filter，計算量會非常龐大且耗時，不符合定位任務的需求。

四、導航

只要定位成功後，下目標後即刻就能得出路徑，

柒、 結論

一、研究總結

本研究在繪製 2D 地圖的情況下做得很好，除了太狹窄的區域外，對於不同的環境修改一下演算法的參數就能達到很高的成功率，不容易在實驗的過程中碰壁或逗留在一個區域很久。而定位方面，環境較為複雜的區域(如雜亂的室內環境)有很好的效果，單調或空間相似度較高的環境較容易定位失敗；若定位成功後，機體能很好的完成導航任務。

二、未來展望

針對本研究尚可發展及精進的部分，我們將往幾個層面拓展：

(一)、對不同環境用特化的運動模式進行自主SLAM以增進效率及降低失誤率。

(二)、在不影響地圖的精確度下以最短的探索時間完成SLAM。

(三)、利用 Lidar 或攝影機取代雷射以將此研究從二維空間拓展至三維空間。

(四)、在對amcl不利的區域運用其他方法進行定位任務。

(五)、在時變的環境中(如有人的場所)區分運動及靜止的物體，適用於更多樣的工作環境。

捌、參考資料及其他

[1] Documentation-ROS wiki(wiki.ros.org)

[2] Dieter Fox, Wolfram Burgard, and Sebastian Thrun. *The dynamic window approach to collision avoidance*, IEEE Robotics & Automation Magazine, pages 23-33, 1997

[3] Mikko Lauri, Risto Ritala. *Planning for robotic exploration based on forward simulation,* Robotics and Autonomous Systems, Vol 83, (2016), pp. 15-31, DOI: 10.1016/j.robot.2016.06.008

[4] Giorgio Grisetti, Cyrill Stachniss, Wolfram Burgard. *Improved Techniques for Grid Mapping with Rao-Blackwellized Particle Filters*, IEEE Transactions in Robotics, 2006

[5] Ray Wenderlich, *Introduction to A* Pathfinding,* (https://www.raywenderlich.com/4946/introduction-to-a-pathfinding)

【評語】052302

1. 此作品為 PioneerP3-DX 和 ROS 的應用實作計畫，設計周全，並包含因果探究及論述。

2. 研究透過探勘與定位領域結合，開發機器人可使用自身建構出的地圖進行導航任務。整合機電、雷射掃描、及機械學習技術，學理及實驗過程說明清楚，具有不錯的實驗成果產出。

3. 在建構工程系統的任務之時，若能同步建立好更聚焦的使用情境，以實際發揮應用，更可建立本專題的獨特性和創新性。

作品海報

摘要

本研究透過ROS(機器人運作系統)實作一個能自由探勘且進行SLAM(同步定位及地圖構建)，且使用自身建構出的地圖進行導航任務之機器人系統，供各式各樣的服務做為一系統性的路徑規劃器及運動基座。並探討此系統擅長及不擅長的環境，呈現目前機器人系統的運作能力。

圖1、正在執行本研究系統的Pioneer P3-DX

研究動機

SLAM與探勘兩大領域的研究讓機器人能以有系統的方式探索未知或危險的環境，然而原先SLAM的過程常需要有人在旁操作，將此過程自動化可以消除對人未知的危險。再賦予系統導航任務的能力，呈現自主SLAM之成效。

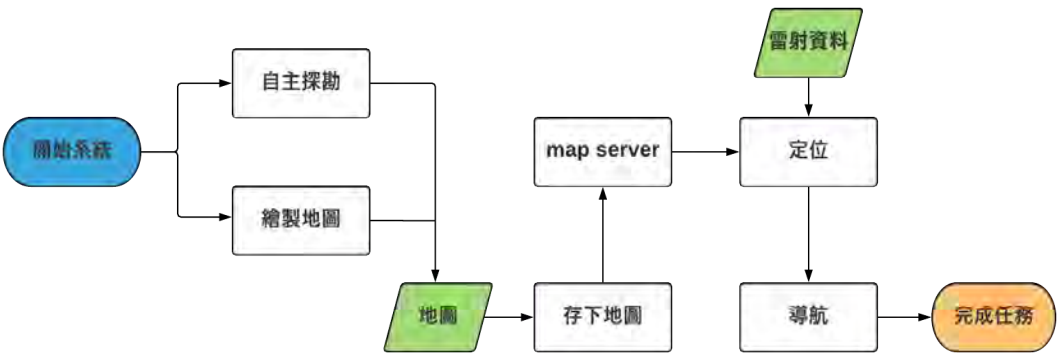
研究目的

本研究希望能開發出一結合探索、繪製地圖、定位及導航功能的機器人系統，能夠自主完成複合任務。

研究過程

系統架構

本研究使用安裝在ubuntu 17.04的ROS kinetic做為開發環境。



理論基礎

探勘	• Dynamic Window Approach • Particle Filter • Partially Observable Markov Decision Process
繪製地圖	• Gmapping • (Rao-Blackwellized Particle Filter)
定位	• Adaptive Monte-Carlo Localization • (Particle Filter)
導航	• A* Search Algorithm • Dynamic Window Approach

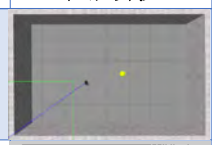
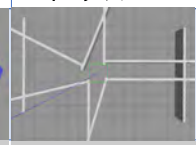
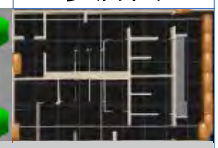
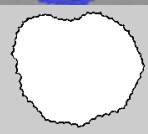
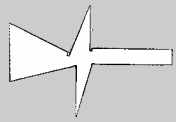
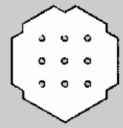
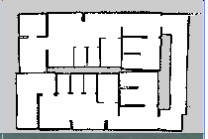
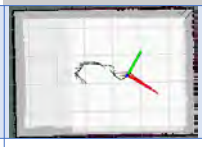
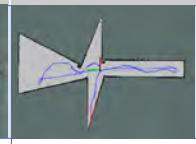
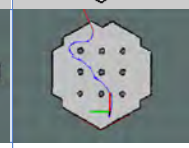
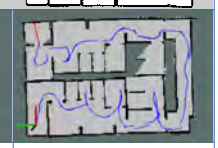
實驗步驟

架設環境及機器人 → 開啟探索(存下地圖) → 地圖建好後定位(雷射匹配地圖) → 快速導航

研究結果

虛擬環境(Gazebo)

本研究以Gazebo模擬器(version 7.0)進行實驗，透過模擬器能夠很容易根據運行的狀況調整參數，以及比較人為操作SLAM與自主SLAM之效果。

區域類型	長方形	曲線邊界	十字路口	障礙物	多房間
實際地圖					
人為操作					
自主決策及路徑					
時間	1’29”	2’02”	7’09”	2’10”	12’24”
pixel數	11525	15514	10362	7884	16265
完成率	100%	99.3%	98.8%	100%	96.9%
效率比	0.51	0.98	0.47	0.51	0.58

實體實驗(Pioneer P3-DX)

圖2(上下)、居家環境

圖3、居家環境探勘路徑及地圖

圖4、辦公室環境

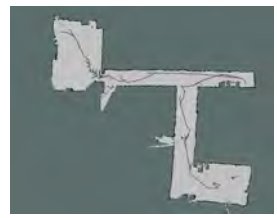


圖5、工作環境探勘路徑及地圖

定位及導航

使用Adaptive Monte-Carlo定位時，必須先讓機器人在環境中移動。使其先收斂到正確的位置，定位成功後進行導航。

區域類型	長方形	曲線邊界	十字路口	障礙物	多房間
定位偏差	<50cm	<35cm	<20cm	<15cm	<25cm
收斂時間	15s	11s	9s	6s	11s

導航系統對於先前探勘繪製的地圖上沒有的障礙物也能做出避障，更安全的完成任務。

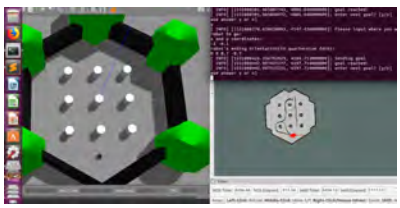


圖6、導航至指定點

圖7、導航期間避障

討論

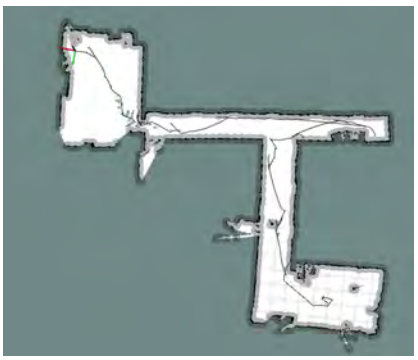


圖12、討論自主SLAM的優缺點

1. 在預設的參數下，環境中寬度大於機器人兩倍寬以上的區域，機器人能夠很快速地通過並成功繪出準確的地圖。

2. 系統在面對較狹窄的環境或較突然的轉角，決策時間通常會較久，原因是在接近目標點同時須更注意避障的問題。

3. 由於本實驗所利用的測距感測器為雷射感測器，故在環境中有玻璃、窗戶等透光性物體會使繪出的障礙物邊界失真。

4. 為了降低系統的運算量，本系統在運行Adaptive Monte-Carlo localization時將樣本散布於設定的點附近，加速位置收斂。

5. 若機器人所在環境很不規則，則AMCL的成功率越高。

6. 雷射為二維的感測器，所以面對高於或低於雷射且突出的障礙物時因無法偵測到而有碰撞的可能性。

7. 導航系統能夠很快速的規畫路徑並抵達目的地。

結論

本研究成功地開發出了探勘、繪地圖及導航的複合功能機器人。可以在完全自主的情況下有效探索未知環境且很少發生失誤。而定位方面，環境較為複雜的區域(如雜亂的室內環境)有很好的效果，單調或空間相似度較高的環境較容易定位失敗；若定位成功後，機體能很好的完成導航任務。

未來展望

1. 對不同環境用特化的運動模式進行自主SLAM。

2. 利用Lidar或攝影機取代雷射以將此研究從二維空間拓展至三維空間。

3. 對AMCL不利的區域用如影像的方式定位。

4. 使用攝影機添加辨識物體及人的功能於系統。

5. 將此研究實際應用於生活，如：室內服務機器人、搜救機器人。

參考資料

[1] wiki.ros.org
[2] Dieter Fox, Wolfram Burgard, and Sebastian Thrun. *The dynamic window approach to collision avoidance*, IEEE Robotics & Automation Magazine, pages 23-33, 1997
[3] Mikko Lauri, Risto Ritala. *Planning for robotic exploration based on forward simulation,* Robotics and Autonomous Systems, Vol 83, (2016), pp. 15-31, DOI: 10.1016/j.robot.2016.06.008
[4] Giorgio Grisetti, Cyrill Stachniss, Wolfram Burgard. *Improved Techniques for Grid Mapping with Rao-Blackwellized Particle Filters*, IEEE Transactions in Robotics, 2006
[5] https://www.raywenderlich.com/4946/introduction-to-a-pathfinding