

國立勤益科技大學

電機工程系碩士班

碩士論文

基於深度強化學習之室內移動機器人路徑規劃與導航系統研究

Research on Path Planning and Navigation System of Indoor Mobile Robot Based on Deep Reinforcement Learning

**研究生：林栩永**

**指導教授：陳碧雲博士**

**白能勝博士**

**中華民國一一二年七月**

**(學位考試委員會口試審定書)**

【說明】

1.封面後應附上**口試審定書**(掃描檔)，正本需繳交圖書館。

2.務必確認簽名處、日期皆已填寫。

基於深度強化學習之室內移動機器人路徑規劃與導航系統研究

研究生：林栩永 指導教授：陳碧雲博士  
白能勝博士

國立勤益科技大學電機工程系碩士班

摘要

本論文提出了一種用於室內移動機器人的自主導航系統架構，結合端到端自動駕駛(End-to-End Autonomous Driving, E2E)和傳統導航演算法的優點。這個架構旨在克服傳統導航算法過度依賴高精度定位以及E2E在無法檢測目標物時難以做出優良決策的挑戰。利用深度強化學習在模擬環境中訓練了神經網路，並引入了行為複製(Behavior Cloning)的方法來穩定訓練過程。透過這種方法，訓練好的神經網路能夠僅依靠2D-LiDAR和攝影機拍攝的影像進行動作決策，不再依賴高精度定位系統，從而克服了傳統導航算法過度依賴高精度定位的挑戰。在現實環境中，使用YOLO-v7 tiny模型進行室內環境中目標物的檢測。當距離目標物較遠時，使用A\*和DWA演算法進行路徑規劃，以確保系統能夠安全且有效地導航。這些演算法能夠找到全域最優路徑並進行局部避障，從而實現了室內環境中的自主導航。

關鍵詞：深度強化學習、行為複製、YOLO-v7 tiny、A\*演算法、DWA演算法

Research on Path Planning and Navigation System of Indoor Mobile Robot Based on Deep Reinforcement Learning

|  |  |  |
| --- | --- | --- |
| Student: Hsu-Yung Lin | Advisor: | Pi-Yun Chen  Neng-Sheng Pai |
| Department of Electrical Engineering  National Chin-Yi University of Technology | | |

Abstract

This paper presents a self-navigation system architecture for indoor mobile robots that integrates the advantages of end-to-end autonomous driving (E2E) and traditional navigation algorithms. The proposed framework aims to address the challenges faced by traditional navigation algorithms, which overly rely on high-precision localization, and E2E approaches, which struggle to make optimal decisions when unable to detect target objects. By employing deep reinforcement learning, the neural network is trained in a simulated environment, with the incorporation of behavior cloning to enhance training stability. Consequently, the trained neural network is capable of making action decisions solely based on 2D-LiDAR and camera images, eliminating the dependency on high-precision localization systems and overcoming the excessive reliance observed in traditional navigation algorithms. In real-world scenarios, the YOLO-v7 tiny model is employed for target object detection in indoor environments. When the target object is situated at a considerable distance, path planning is performed using A\* and DWA algorithms to ensure safe and efficient navigation. These algorithms facilitate the discovery of globally optimal paths and enable local obstacle avoidance, thereby enabling autonomous navigation in indoor environments.

Keywords: Deep Reinforcement Learning, Behavior Cloning, YOLO-v7 tiny, A\* Algorithm, DWA Algorithm.

誌謝

請在此寫入論文誌謝，以一頁為原則。

目次

[摘要 i](#_Toc135654444)

[Abstract ii](#_Toc135654445)

[誌謝 iv](#_Toc135654446)

[目次 v](#_Toc135654447)

[表目次 vii](#_Toc135654448)

[圖目次 viii](#_Toc135654449)

[第一章 緒論 1](#_Toc135654450)

[1.1 前言 1](#_Toc135654451)

[1.2 文獻探討 2](#_Toc135654452)

[1.3 論文目標 4](#_Toc135654453)

[1.4 論文架構 4](#_Toc135654454)

[第二章 背景理論 6](#_Toc135654455)

[2.1 深度學習 6](#_Toc135654456)

[2.2 強化學習 8](#_Toc135654457)

[2.3 深度強化學習 12](#_Toc135654458)

[第三章 系統架構與軟硬體介紹 18](#_Toc135654459)

[3.1 系統架構 18](#_Toc135654460)

[3.2 硬體介紹 19](#_Toc135654461)

[3.2.1 邊緣運算平台 19](#_Toc135654462)

[3.2.2 影像擷取設備 21](#_Toc135654463)

[3.2.3 2D LiDAR 22](#_Toc135654464)

[3.2.4 電腦端 23](#_Toc135654465)

[3.3 開發環境及各種軟體介紹 23](#_Toc135654466)

[3.3.1 深度學習框架 24](#_Toc135654467)

[3.3.2 機器人作業系統(Robot Operating System, ROS) 24](#_Toc135654468)

[第四章 基於強化學習的導航避障 26](#_Toc135654469)

[4.1 模擬環境搭建 26](#_Toc135654470)

[4.2 動作空間與狀態空間設計 28](#_Toc135654471)

[4.3 獎勵設計 29](#_Toc135654472)

[4.4 網路架構 32](#_Toc135654473)

[4.5 環境流程設計 35](#_Toc135654474)

[4.6 訓練流程 36](#_Toc135654475)

[4.7 第四章總結 41](#_Toc135654476)

[第五章 訓練優化、模型遷移與路徑規劃導航 42](#_Toc135654477)

[5.1 結合監督學習的訓練優化 42](#_Toc135654478)

[5.2 模型遷移設計與物件偵測 44](#_Toc135654479)

[5.3 路徑規劃與導航 49](#_Toc135654480)

[5.4 神經網路結合路徑規劃與導航 52](#_Toc135654481)

[5.5 第五章總結 53](#_Toc135654482)

[第六章 實驗結果與數據分析 54](#_Toc135654483)

[6.1 模擬世界中強化學習訓練結果 54](#_Toc135654484)

[6.1.1 無障礙物地圖訓練結果 55](#_Toc135654485)

[6.1.2 靜態障礙物與隨機障礙物地圖訓練結果 57](#_Toc135654486)

[6.2 模擬世界神經網路結合路徑規劃與導航測試結果 59](#_Toc135654487)

[6.3 現實世界物件偵測結果 60](#_Toc135654488)

[6.4 現實世界自主導航架構測試結果 62](#_Toc135654489)

[第七章 結論與未來展望 69](#_Toc135654490)

[7.1 結論 69](#_Toc135654491)

[7.2 未來展望 70](#_Toc135654492)

[參考文獻 72](#_Toc135654493)

表目次

表2.1 Q表 11

[表3.1 NVIDIA Jetson Nano規格表[40] 20](#_Toc135654495)

[表3.2 奥比中光DaBai規格表 21](#_Toc135654496)

[表3.3 EAI X2L光學雷達規格表 22](#_Toc135654497)

[表3.4 NVIDIA GeForce GTX 1060顯示卡規格表 23](#_Toc135654498)

[表3.5 系統軟體版本 23](#_Toc135654499)

[表6.1 SAC超參數表 55](#_Toc135654500)

[表6.2 算法架構比對表 60](#_Toc135654501)

圖目次

[圖2.1神經元架構圖 8](#_Toc135654502)

[圖2.2 深度學習網路示意圖 8](#_Toc135654503)

[圖2.3 馬爾科夫決策過程示意圖 9](#_Toc135654504)

[圖2.4 價值轉移示意圖 10](#_Toc135654505)

[圖2.5 DQN網路架構 13](#_Toc135654506)

[圖2.6 訓練架構圖 14](#_Toc135654507)

[圖3.1 系統架構圖 18](#_Toc135654508)

[圖3.2 移動機器人Limo 19](#_Toc135654509)

[圖3.3 NVIDIA Jetson Nano[40] 20](#_Toc135654510)

[圖3.4 奥比中光DaBai[41] 21](#_Toc135654511)

[圖3.5 EAI X2L光學雷達[42] 22](#_Toc135654512)

[圖3.6 ROS註冊節點圖 25](#_Toc135654513)

[圖3.7 ROS節點通訊圖 25](#_Toc135654514)

[圖4.1 模擬環境的代理人(Limo) 27](#_Toc135654515)

[圖4.2 Gazebo ROS節點圖 27](#_Toc135654516)

[圖4.3 模擬環境中的地圖 28](#_Toc135654517)

[圖4.4 目標物狀態空間示意圖 29](#_Toc135654518)

[圖4.5 全局角度示意圖 31](#_Toc135654519)

[圖4.6 角度獎勵示意圖 31](#_Toc135654520)

[圖4.7 策略網絡示意圖(Actor) 32](#_Toc135654521)

[圖4.8 神經元計算圖 33](#_Toc135654522)

[圖4.9 採樣示意圖 34](#_Toc135654523)

[圖4.10 重參數技巧示意圖 34](#_Toc135654524)

[圖4.11 評論網絡示意圖(Critic) 35](#_Toc135654525)

[圖4.12 環境流程圖 36](#_Toc135654526)

[圖4.13 經驗池示意圖 36](#_Toc135654527)

[圖4.14 評論家更新示意圖 37](#_Toc135654528)

[圖4.15 策略對數示意圖 38](#_Toc135654529)

[圖4.16策略網路更新示意圖 39](#_Toc135654530)

[圖4.17 溫度係數更新示意 39](#_Toc135654531)

[圖4.18 神經網路訓練流程 40](#_Toc135654532)

[圖5.1 監督式學習於自主導航示意圖 43](#_Toc135654533)

[圖5.2 以監督式優化強化學習訓練示意圖 43](#_Toc135654534)

[圖5.3 專家數據收集流程圖 44](#_Toc135654535)

[圖5.4 狀態空間模擬世界與真實世界對照圖 45](#_Toc135654536)

[圖5.5 YOLO示意圖 46](#_Toc135654537)

[圖5.6 YOLO-v7-tiny主幹網路示意圖 47](#_Toc135654538)

[圖5.7 細緻物件模塊示意圖 47](#_Toc135654539)

[圖5.8 YOLO-v7-tiny head層示意圖 48](#_Toc135654540)

[圖5.9 SPPCSP、CSP架構圖 49](#_Toc135654541)

[圖5.10 A\*演算法示意圖 51](#_Toc135654542)

[圖5.11 DWA演算法示意圖 51](#_Toc135654543)

[圖5.12 系統流程圖 52](#_Toc135654544)

[圖6.1 強化學習地圖與獎勵 54](#_Toc135654545)

[圖6.2 無障礙物地圖訓練結果 56](#_Toc135654546)

[圖6.3 不同狀態數量測試圖 56](#_Toc135654547)

[圖6.4 SAC結合深度學習結果圖 57](#_Toc135654548)

[圖6.5 靜態障礙物地圖訓練結果 58](#_Toc135654549)

[圖6.6 動態障礙物地圖訓練結果 59](#_Toc135654550)

[圖6.8 導航目標物示意圖 60](#_Toc135654551)

[圖6.9 YOLO-v7 tiny訓練結果 61](#_Toc135654552)

[圖6.10 目標物辨識結果 62](#_Toc135654553)

[圖6.11 傳統導航示意圖 62](#_Toc135654554)

[圖6.12 導航結束示意圖 63](#_Toc135654555)

[圖6.13 強化學習導航示意圖 64](#_Toc135654556)

[圖6.14 網路決策策略分佈及採樣動作圖 65](#_Toc135654557)

[圖6.15 改變tanh斜率示意圖、動作分佈圖 65](#_Toc135654558)

[圖6.16 融合傳統導航及強化學習導航示意圖 68](#_Toc135654559)

# 緒論

## 前言

根據台灣經濟部所公開資訊《2021年中小企業白皮書》表明，臺灣中小企業總數為154萬8,835家，占全體企業 98.93%。人力資源短缺是台灣企業共同面臨的重要問題，若能積極朝向產業自動化邁進，或許是中小型企業的契機與明路[1]。無人移動載具是產業自動化中極為重要的一環，在過去智能工廠的發展中，自動導引車(Automated Guided Vehicle, AGV)在產品運送上不失為一好方法，但AVG行駛時必須使用導線、標記或磁條等引導方式而使得成本大幅提高[2]。自主移動機器人(Autonomous Mobile Robot, AMR)則是依靠光學雷達(light detection and ranging, LiDAR)、攝影機等感測器，完成同時定位與地圖建構(Simultaneous localization and mapping, SLAM)，建構地圖後引入路徑規劃演算法並完成導航[3]。在AMR導航中，對於精準定位的需求極為嚴苛，高精度的定位系統仍是一大挑戰。而端到端的自動駕駛(End-to-End Autonomous Driving, E2E)則希望載具能夠根據其所搭配的感測器，在沒有人工干預或控制的情況下，也能自主決策出適當的動作[4]。相比傳統的AMR，E2E在面對自主導航方面擁有更好的強健性，對精准定位的要求較低，但在實現方面具有更大的挑戰。而傳統的自主導航系統通常基於規則，在導航過程中需要大量的人力成本，同時在處理突發情況時無法有效應對。

近年來隨著電力電子的蓬勃發展，各種硬體性能越來越強，這讓人工智慧技術在生活中得到了越來越廣泛的應用。其中，深度學習(Deep Learning, DL)和強化學習(Reinforcement Learning, RL)是人工智慧領域最為熱門的研究方向之一。深度學習是一種利用神經網路擬合各種函數的做法，可以從大量數據中自動學習特徵並進行分類、識別、預測等任務。深度學習在影像識別、語音識別、自然語言處理等領域有著出色的表現。強化學習則是從環境中學習的機器學習方法，其注重代理人(Agent)與環境互動所產生的經驗學習，在「探索-利用」達到平衡自主學習。2013年DeepMind結合強化學習以及深度學習提出深度Q網路(Deep Q Network, DQN)開啟了深度強化學習(Deep Reinforcement Learning, DRL)的新篇章[5]，深度強化學習在機器人學、電腦視覺、金融、醫療衛生等領域發揮著巨大的作用。其中最為出名的，DeepMind提出一種基於蒙特卡洛樹搜尋(Monte Carlo Tree Search，MCTS)，並結合深度強化學習應用於圍棋中[6]且戰勝數名優秀選手[7]。OpenAI提出一種來自人類反饋的強化學習(Reinforcement Learning from Human Feedback, RLHF)並以此為基底訓練大型自然語言處理網路ChatGPT[8]。

## 文獻探討

在本章節將對移動機器人導航、深度強化學習、端到端的自動駕駛發展現狀，3個方面進行文獻的回顧與簡述。

移動機器人導航方面，研究重點包括機器人定位、環境感知、地圖構建和路徑規劃等方面。例如，Shao-Hung Chan等人引入了一種用於雷射SLAM和視覺 SLAM融合的方法，其實際距離的定位誤差可以小於5%[9]；Guangping Hao等人使用基於超聲相控陣掃描(Phased Array Ultrasonic Testing，PAUT)理論，研製了一種用於焊縫自動檢測的三自由度(DOFs)掃描機器人[10]；陳世安使用視覺SLAM演算法完成地圖構建，採用D-Star演算法完成路徑規劃，實現了輔助肢體殘障者之載具[11]；黃偉哲使用語義分割神經網路實現動態物體過濾功能，降低動態物體干擾[12]；羅一軒使用快速探索隨機樹演算法完成自主地圖構建，採用DWA完成導航[13]。

深度強化學習方面，繼2013年DeepMind首次成功運用深度強化學習在Atari遊戲中超越人類專家，開啟了深度強化學習的熱潮。隨後Hado van Hasselt為解決在DQN在更新網路中存在的高估問題(Bootstrapping)，提出深度雙Q網路(Double Deep Q Network, DDQN)，其想法在於使用一個 DQN 來選擇動作，另一個 DQN 來評估這些動作的Q值。這有助於減少高估並提高Q函數估計的準確性[14]。為解決強化學習能夠決策連續動作，David Silver提出確定性策略梯度(Deterministic Policy Gradient, DPG)[15]。以此為基礎Timothy P. Lillicrap等人將DPG與深度網路相結合提出深度確定性策略梯度(Deep Deterministic Policy Gradient, DDPG)，將深度強化學習有效的運用在連續動作的控制問題中[16]。深度強化學習在與環境的交互中，策略的學習是非常慢的。Volodymyr Mnih等人提出異步優勢演員評論家(Asynchronous Advantage Actor-Critic, A3C)，其在傳統的演員評論家網路中，加入優勢函數以穩定訓練，並使用多執行緒同時對多個環境進行互動，以異步更新神經網絡的方式進行訓練，從而實現快速學習的效果[17]。而Tuomas Haarnoja則認為在強化學習中不應該像DQN一樣，只考慮最大的動作價值。其認為動作價值的分佈應該是一個玻爾茲曼分佈(Boltzmann distribution)，並同時希望最大化獎勵和熵，因此提出Soft Q-learning演算法[18]。此後其原班人馬將Soft Q-learning演算法的融入進演員評論家網路(Actor-Critic, AC)，提出柔性演員評論家(Soft Actor-Critic, SAC)。與Soft Q-learning不同的是，其加入一個專門做出決策的Actor，以最小化與動作價值分佈的KL散度(Kullback-Leibler divergence, KLD)以此更新策略[19]。

在過去許多學者都嘗試將以機器學習的方式完成E2E，其最早可以追溯到1989年，ALVINN 利用攝影機圖像及鐳射測距使用三層全連接層以監督式學習的方式更新全連接網路，利用神經網路的方式決策載具行駛動作[20]。2016年，Mariusz Bojarski 等人使用卷積神經網路(Convolutional Neural Networks, CNN) 將攝影機的原始影像直接映射至轉向動作，藉由來自人類的訓練資料完成決策[21]。隨著深度強化學習的興起，April Yu等人在模擬環境中使用Deep Q-network(DQN)，以圖像作為輸入並從9個離散動作中決策最佳動作[22]。為保證在E2E決策動作有更好的表現，Zhiqing Huang等人運用深度確定性策略梯度(Deep Deterministic Policy Gradient, DDPG)決策連續動作，實現在TORCS(The Open Racing Car Simulator)環境[23]。Siyuan Chen等人提出以一種能夠穩定訓練輔助網路，並結合模仿學習，為DDPG框架提供一組優秀的初始參數，以便強化學習網路更快收斂，其成功的在TORCS及3D機器人模擬器Gazebo[24]環境中駕駛[25]。Junior Costa de Jesus 等人使用SAC，在相同情況下與深度確定性策略梯度 (DDPG) 演算法進行了比較，先前的線速度和角速度，以及移動機器人與目標的相對位置和角度作為網路輸入[26]。

## 論文目標

本論文將結合E2E及AMR的優點完成自主導航系統，以E2E的強健性彌補AMR對於高精度定位的挑戰。使用深度強化學習的方式完成E2E模塊，改善傳統基於規則面對處理突發狀況無法有效處理的問題。

## 論文架構

本文共分七章，各章節內容概括如下：

第一章 緒論：說明研究動機及相關文獻。

第二章 深度強化學習基礎：介紹深度強化學習相關理論，首先說明深度學習中的深度神經網路原理，其次介紹強化學習的基本概念及深度強化學習原理，最後詳細介紹SAC演算法

第三章 系統架構：簡述系統主架構，介紹所使用的硬體設備與較重要的軟體開發環境。

第四章 深度強化學習實施和自主導航環境工作流程：詳細描述深度強化學習實施和自主導航環境工作流程，以及神經網路訓練流程。

第五章 現實世界應用：介紹如何在現實世界應用，並詳細介紹神經網路結合路徑規劃與導航的方法。

第六章 模擬與真實世界實驗結果分析：對模擬世界及真實世界實驗結果進行分析。

第七章 結論與展望：總結全文，並對未來研究工作進行展望。結論及對未來研究工作的展望。

# 背景理論

人工智慧作為一個快速發展的領域，不斷涌現出各種新的理論和技術，對各個領域產生了深遠的影響。從傳統的需要標籤的深度學習到面臨維度災難的表格形式強化學習，再到為解決維度災難而形成的深度強化學習，本章節將逐步介紹這些理論的演進過程。深度學習作為人工智慧領域的核心技術之一，最初需要標籤的訓練數據來實現模型的學習。然而，傳統深度學習在處理大量複雜數據時面臨著標籤收集困難和過度依賴標籤的問題。隨後，強化學習的興起提供了一種從環境交互中自主學習的方式。然而，在處理高維度狀態空間時，傳統的表格形式強化學習受到了維度災難的限制。對於本文的移動機器人導航系統來說，收集大量的標籤是不太現實的，而且在大維度的狀態空間及動作空間中，傳統的表格形式強化學習往往無法實現。因此，本文引入了深度強化學習作為系統的一部分，以克服這些限制，並將在接下來的章節中詳細介紹這一演進過程。

## 深度學習

深度學習的概念是基於人工神經網路的發展而提出的。其核心思想是透過多層神經元，結合非線性函數(激活函數，Activation Function)，並通過反向傳播演算法對權重進行調整，以最小化損失函數，實現從輸入數據到輸出數據的映射，從而實現端到端的變換函數進行擬合。

神經元是深度學習模型中的基本單元，通常由數據輸入層、權重、激活函數和偏差項組成。以圖2.1為例，數據輸入層由A1和A2兩個輸入單元組成，每個輸入單元都分別乘以對應的權重和，這些權重是模型需要學習的參數。這些乘積的總和再加上偏差項，形成了神經元的加權和，即(2.1)所示。

|  |  |
| --- | --- |
|  | (2.1) |

神經元的激活函數會對加權和進行非線性轉換，得到最終的輸出*B*。啟動函數可以是任何非線性函數，例如sigmoid[27]、ReLU[28]等，它們有助於模型學習複雜的非線性關係，如(2.2)所示。

|  |  |
| --- | --- |
|  | (2.2) |

為了能夠讓這個函數竟可能的輸出想要的結果，引入真實目標標籤值*B'*。使用*B'*和神經元的輸出B之間的差異來計算損失(Loss)。常用的損失函數有均方誤差(Mean Squared Error, MSE)等，如(2.3)所示。

|  |  |
| --- | --- |
|  | (2.3) |

在人工神經網路中，目標是通過調整網路中的權重和偏差項來最小化損失函數。這個過程可以通過使用優化演算法來實現，其中梯度下降[29]是最常用的一種。梯度下降演算法可以利用反向傳播來計算每個參數對於損失函數的梯度，然後使用這些梯度來更新每個參數，以使得損失函數最小化，如(2.4)及(2.5)所示，分別對權重及偏差項做偏微分並更新參數。

|  |  |
| --- | --- |
|  | (2.4) |
|  | (2.5) |

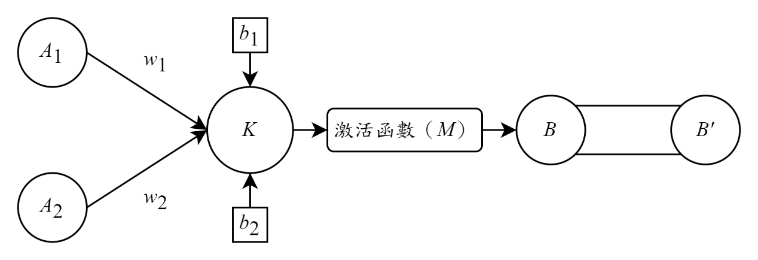


圖2.1神經元架構圖

深度學習模型中由多個神經元組合而成，深度學習模型由多個神經元組合而成，通常以層(layer)的形式排列，每一層都包含了多個神經元，每個神經元接收來自上一層的輸出或者輸入資料，在其內部進行計算和啟動操作，並將結果傳遞到下一層，通過逐層傳遞的方式，深度學習模型可以使用一系列函數逼近各種複雜情況，如圖2.2所示。

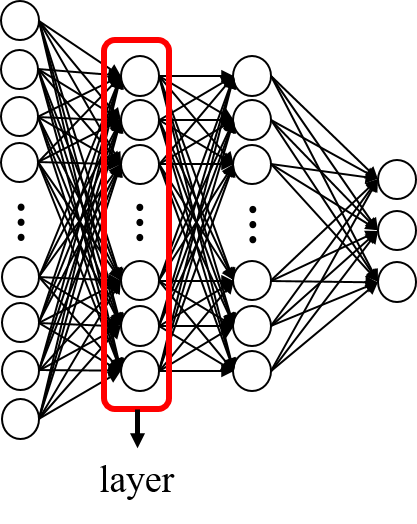


圖2.2 深度學習網路示意圖

## 強化學習

在2.1小節中所提及到的深度學習，在沒有明確的標籤或訓練資料的問題中往往難以進行有效的學習。然而，在現實世界中，很多情況下只能通過與環境交互來學習。

強化學習是一種機器學習方法，它通過代理人(Agent)與環境(Environment)的交互來學習如何選擇最佳行動以獲得最大的累積獎勵。在強化學習中，環境被建模成一個馬爾科夫決策過程(Markov Decision Process，MDP)[30]，如圖2.3所示。

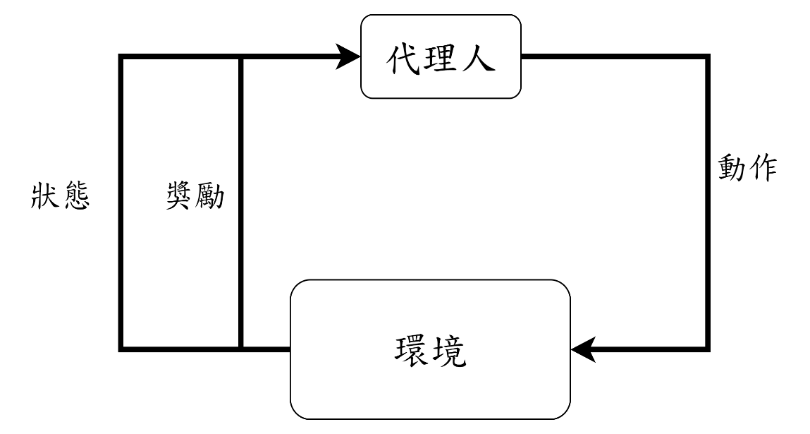


圖2.3 馬爾科夫決策過程示意圖

馬爾科夫決策過程是一種描述強化學習問題的數學框架，它包括狀態(State, *s*)、動作(Action, *a*)、獎勵(Reward, *r*)、策略(Policy, )、價值(Value)等概念。在一個馬爾科夫決策過程中，代理人通過採取不同的動作來影響環境，從而使環境轉移到不同的狀態，並獲得相應的獎勵，這一系列行為被稱之為軌跡(Trajectory, )。每個狀態對應著一個狀態價值函數，表示在該狀態下採取一定策略所能獲得的累積獎勵(Cumulative Future Reward, )的期望值，用於評估在某個狀態下執行某個策略的長期期望回報(Expected Long-term Return)，是一個折扣因數(Discount factor)，用於將未來獎勵的價值折現到當前時間點，的定義如(2.6)所示。

|  |  |
| --- | --- |
|  | (2.6) |

動作價值函數(Action Value Function, )也被稱為Q函數。動作價值函數描述了在某個狀態下，代理人針對每個可用的動作所能獲得的期望累積回報。狀態價值函數可以看作是動作價值函數的期望值，可以通過狀態價值函數來計算動作價值函數。具體而言，動作價值函數等於當前狀態*s*採取動作*a*後的即時獎勵*r*，加上代理人在下一個狀態下根據策略π所能獲得的期望累積回報，其中表示t時刻狀態轉移至t+1時刻狀態之概率，如圖2.4所示。

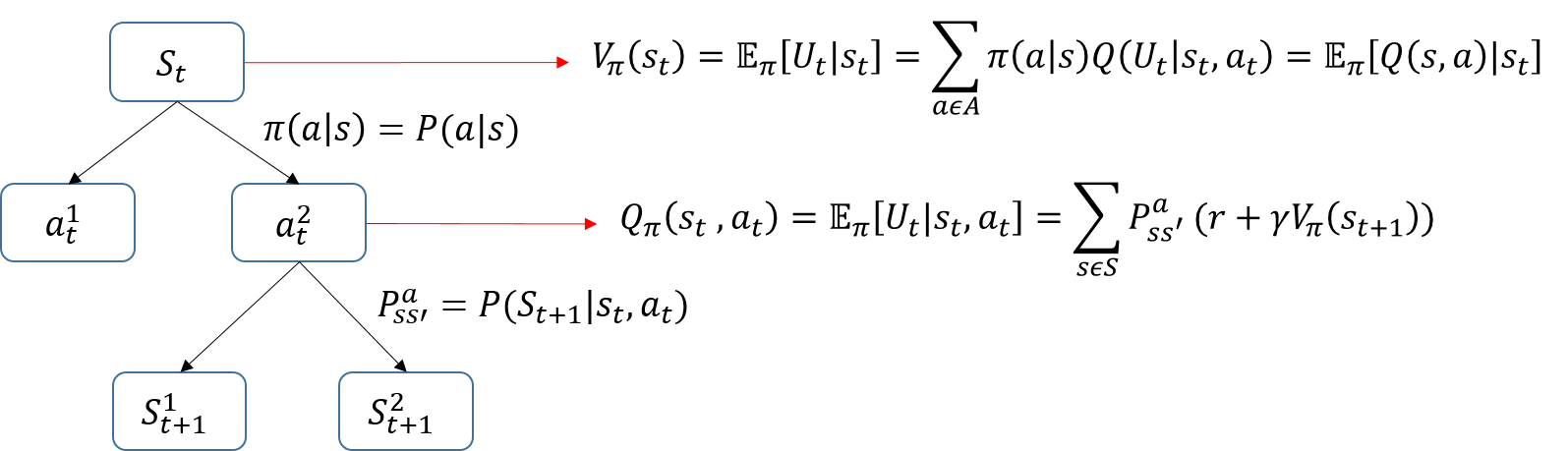


圖2.4 價值轉移示意圖

是否得知，可分為有模型(Model-based)強化學習和無模型(Model-free)強化學習。在有模型強化學習中，代理人能夠獲得環境的完整模型，即它能夠瞭解任何狀態下採取任何行動所導致的下一個狀態和相應的獎勵。這使得代理人能夠直接計算狀態轉移概率，並使用它來計算值函數和策略。在無模型強化學習中，代理人沒有完整的環境模型，並且必須通過與環境的交互來估計狀態轉移概率。代理人通過觀察獎勵信號和下一個狀態來估計狀態轉移概率，這通常需要使用學習演算法和統計技巧來處理雜訊和不確定性。因此，在無模型強化學習中，代理人必須同時學習值函數和策略，以便通過試錯學習最優策略。在本文所要探討的移動機器人研究中，狀態轉移概率是無法得知的，因此將不再討論有模型的強化學習。

在強化學習中，最重要的問題之一是如何學習一個好的策略。為了解決這個問題，人們開發了許多無模型的強化學習演算法，其中最經典和最廣泛使用的三個演算法是Q-learning[31]、REINFORCE[32]和Actor-Critic[33]。

Q-learning是一種基於價值(Value-Based)的強化學習演算法，Q-learning的目標是通過代理人與環境的交互，不斷更新估計的動作值函數，從而學習到最真實的動作價值。Q-learning的更新公式可以表示為(2.7)，該公式使用了增量更新方法，即每次只更新當前狀態下採取的動作的值，而不是更新所有狀態下所有動作的值。這樣可以減少計算量，提高演算法效率。通過不斷反覆運算更新Q值，代理人可以最終學習到最優策略。此類算法亦被稱之為時間差分算法(Temporal Difference algorithm， TD)[34]。

|  |  |
| --- | --- |
|  | (2.7) |

在過去，Q-learning演算法通常使用表格形式來存儲動作值函數。表格的每一行代表一個狀態，每一列代表一個可行的動作，表格中每個元素代表在特定狀態下採取特定動作的值，如表2.1所示。當代理與環境互動時，依照該表格中存儲的值來決定採取哪個動作，並通過更新表格中相應的元素來學習更好的策略。

表2.1 Q表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

REINFORCE演算法是一種基於策略(Policy-Based)的強化學習演算法，它通過學習策略函數*π*來最大化期望累積回報。Reinforce演算法的基本思想是使用蒙特卡洛方法估計累積回報，然後使用策略梯度方法來更新策略函數。其中策略梯度，希望累積獎勵的最大化。為了能夠使用導數的最優化方法求極值，REINFORCE演算法優化目標優化策略參數使得獎勵的期望值最大，如(2.8)所示。

|  |  |
| --- | --- |
|  | (2.8) |

其中，表示根據策略生成的軌跡，表示軌跡的累積獎勵。對求梯度，可以得到策略梯度，如式(2.9)所示：

|  |  |
| --- | --- |
|  | (2.9) |

代理人在環境中運行多個蒙特卡洛模擬，以計算每個狀態下採取每個動作的期望累積回報。然後，代理人使用梯度上升法更新策略函數，從而進一步提高累積回報。

Actor-Critic演算法是一種基於價值函數和策略函數的強化學習演算法，其基本思想是通過值函數評估策略函數的好壞，並使用策略梯度法更新策略函數，同時使用值函數的TD(Temporal Difference)誤差來更新值函數。價值函數更新如(2.10)所示。

|  |  |
| --- | --- |
|  | (2.10) |

其中，表示時間步的TD誤差， 表示時間步的即時獎勵，策略函数更新如(2.11)所示

|  |  |
| --- | --- |
|  | (2.11) |

## 深度強化學習

傳統的強化學習方法通常使用表格的形式來學習策略，例如2.2小節所提到的Q表。然而，在真實環境中，狀態空間往往是高維度和連續的，使用表格形式表示值函數或策略函數的方法會遇到維度災難的問題。因此，現代強化學習的研究趨勢轉向使用基於神經網絡的方法來學習策略，其中代表性的演算法包括DQN、DDPG、TD3[35]和SAC等。

DQN是一種基於神經網絡的演算法，它在Q-learning的基礎上使用神經網絡來近似值函數，解決了高維度和連續狀態空間的問題。其網路架構如圖2.5所示，網路的輸入為與環境交互後得到的狀態，透過網路映射直接預測對每個動作的價值函數。

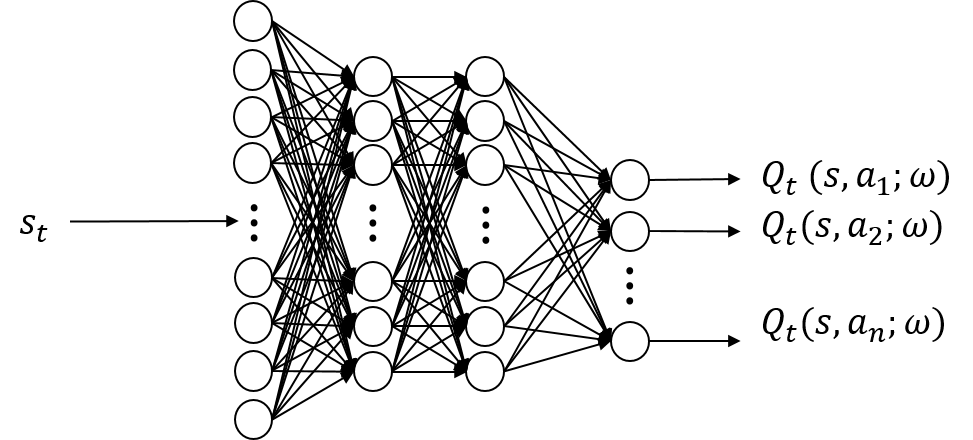


圖2.5 DQN網路架構

選取動作價值最大的索引(index)作為動作與環境互動，得到下一個狀態、獎勵存入經驗池。訓練網路是透過經驗池中隨機抽取部分軌跡，由當前狀態及下一個狀態，計算現在的值與下一個狀態值，當前值與下一具有真實獎勵的最大值相差，作為損失函數如(2.12)所示。

|  |  |
| --- | --- |
|  | (2.12) |

為了提高訓練的效率和穩定性，通常會使用目標網路(Target Network)來進行訓練。目標網路和Q網路結構相同，但目標網路的參數是固定的，不會被即時更新。在每次訓練時，使用目標網路來計算下一個狀態的目標Q值，並根據此目標Q值來更新Q網路的參數。為了降低因為Q值的更新而產生的不穩定性，並提高訓練的效率和穩定性，訓練的過程中通常會固定目標網路的參數，在一定次數的訓練後將Q網路的參數全部複製到目標網路中，這樣做可以避免Q值的震盪，提高訓練的效果。

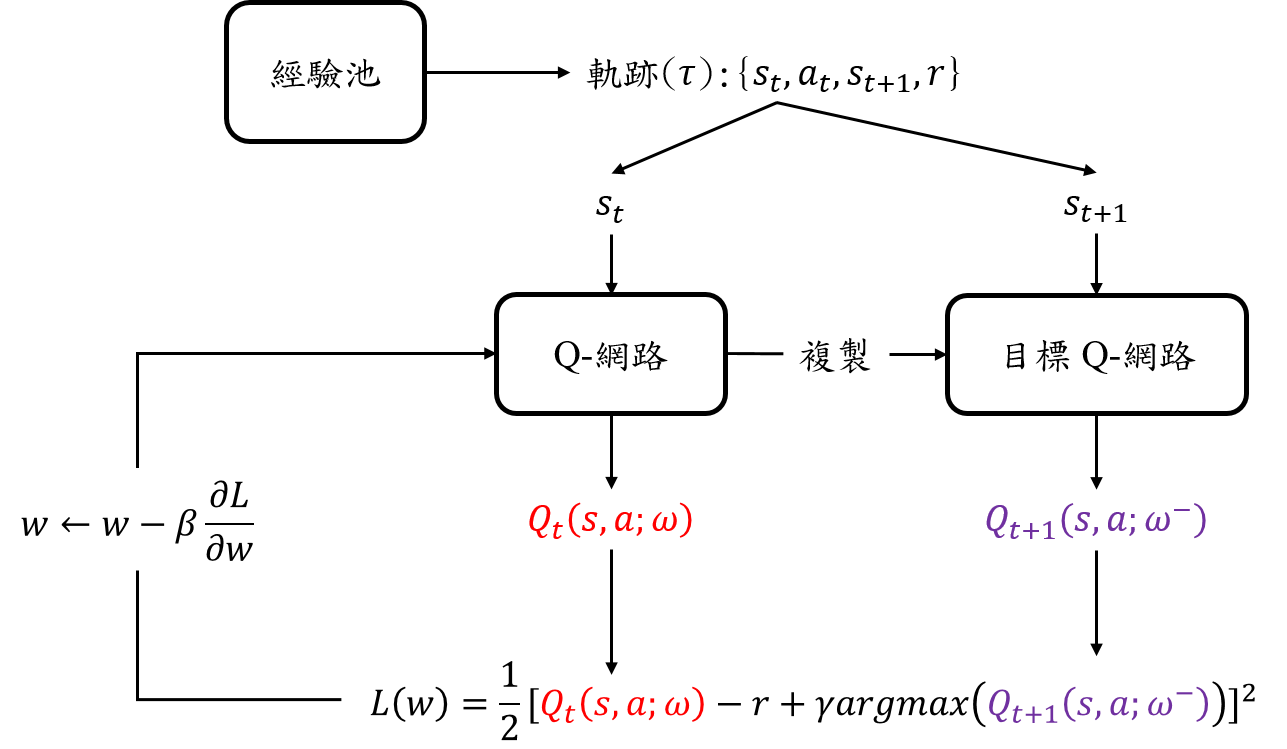


圖2.6 訓練架構圖

在前期探索階段，DQN會採取隨機動作探索環境，以便收集更多的經驗資料。具體來說，DQN在前期使用一個-greedy策略，以一定概率隨機選擇動作，以便探索未知的狀態空間，而以1-的概率選擇當前狀態下具有最大價值的動作。在訓練過程中，隨著經驗的積累，探索率會逐漸減小，模型會更多地依賴於已有的經驗資料進行學習，從而提高訓練效率。

DQN等傳統算法中，探索率是一個重要的超參數(Hyperparameter)，它決定了代理人在執行策略時選擇隨機動作的概率，以增加對未知環境的探索。隨著訓練次數的增加，通常會逐漸減小探索率，使代理人在訓練後更多地遵循已經學習到的最優策略，從而提高其性能。然而，設置過小的探索率可能會導致代理人無法探索新的環境，而僅僅繼續執行已經學習到的策略。為了解決這個問題，一些研究人員引入了最大熵原理，通過增加策略的隨機性來鼓勵探索。

在SAC中，評論家希望能夠評估策略的價值，而價值不僅僅來自於獎勵，還包括策略的熵，即策略的不確定性。因此，SAC的評論家會將熵作為一個額外的項目納入價值的計算中，從而鼓勵策略保持多樣性和探索性。在傳統的DRL中，與環境互動的策略都希望最大化期望累積獎勵，並在未知環境中進行探索以獲得更多的回報信息，這是RL中探索和利用之間的基本折衷。例如DQN中探索性來自探索率的大小。最大熵強化學習是一種基於最大熵原理的強化學習方法，它的目標是學習一個最大熵策略，使得在當前狀態下，所有可能的行動都具有相同的概率，並且在該策略下期望累積獎勵最大。其目標函数如(2.13)所示。

|  |  |
| --- | --- |
|  | (2.13) |

其中，表示t時刻的即時獎勵，表示狀態和動作的分佈，溫度係數(Temperature Coefficient)是熵的權重參數，是當前狀態下策略的熵，其中表示給定狀態下採取的行動。表示一個在狀態下的策略，它為每個可能的行動給出概率 。在資訊理論中，熵是一種度量不確定性或隨機性的概念。它是由Shannon在1948年提出的，通常用來描述資訊源中包含的信息量的多少，也可以用來描述一個隨機變數的不確定性大小[36]。熵定義如(2.14)所示。

|  |  |
| --- | --- |
|  | (2.14) |

其中，X表示一個隨機變數，表示X取到的概率，log表示以2為底的對數。因此， 表示在狀態下根據策略採取行動的不確定性或隨機性的度量，也稱為在狀態下的策略的熵，如(2.15)所示。最大熵強化學習的優點是它能夠在探索和利用之間取得平衡。

|  |  |
| --- | --- |
|  | (2.15) |

圖2.4中所提到的價值函數轉移中動作價值的轉移如(2.16)所示，SAC中不僅僅考慮獎勵還要考慮策略的熵，將熵嵌入計算可得(2.17)。

|  |  |
| --- | --- |
|  | (2.16) |
|  | (2.17) |

也因此SAC中評論家網路的更新，同樣基於最小化均方誤差的目標，目標是使預測的行動價值和實際觀測到的行動價值之間的差距最小化，優化目標如(2.18)所示。其中，是評論家網路的參數。與常規的評論家相比，常規評論家只能評估當前狀態-動作對的價值，無法根據策略資訊評估下一個狀態-動作對的價值。因此，它只能通過在環境中採集更多的資料來學習更準確的價值函數。在沒有最大熵的情況下，策略的學習可能會被困在一個局部最優解中，因為策略在執行動作時執行的動作概率會越來越高，而其他動作的概率則會越來越低，因此探索率變低。最大熵強化學習通過將熵(即不確定性)作為一個額外的目標，使策略保持較高的探索性，即使一個動作的期望報酬較低，也可能被選中，從而更好地探索整個狀態空間。因此，加入最大熵可以使策略更具有探索性和強健性。

|  |  |
| --- | --- |
|  | (2.18) |

演員的優化目標是最大化策略分佈與價值函數分佈之間的相似性，使用KL散度(KL divergence)[37]來度量策略分佈和價值函數分佈之間的距離。動作價值分佈被建模成玻爾茲曼分佈，演員所要決策的策略需要盡可能的與動作價值分佈相似如(2.19)所示，展開後如(2.20)所示。以最小化(2.20)為目標作為演員網路loss。

|  |  |
| --- | --- |
|  | (2.19) |
|  | (2.20) |

溫度係數(Temperature Coefficient)用來控制演員策略的探索程度。當趨近於0時，演員策略會趨近於對應的最大動作價值，即策略會變得越來越保守。相反地，當趨近於正無窮時，演員策略會變得越來越隨機，從而增加了探索的可能性。因此，的值需要自我調整地變化，以便在訓練的不同階段中平衡探索和利用的需求。首先定義目標熵通過最小化相同的目標函數來學習每一步的最佳溫度係數如(2.21)所示，以此更新溫度係數。

|  |  |
| --- | --- |
|  | (2.21) |

# 系統架構與軟硬體介紹

在本章節中，將介紹系統架構與軟硬體介紹，系統架構部分將闡述整體系統的設計結構和模塊之間的關聯，以實現機器人的自主導航能力。而硬體介紹則會介紹用於系統運行的重要硬體組件，包括邊緣運算平台、影像擷取設備、2D LiDAR以及電腦端設備。這些硬體設備的選擇和配置對系統性能和功能起著重要作用。除了硬體介紹，本章節還將介紹開發環境及各種軟體工具。深度學習框架是實現深度強化學習算法的關鍵，將介紹所選擇的深度學習框架，以及其在系統中的應用。同時，機器人作業系統（Robot Operating System, ROS）也是機器人開發中常用的軟體平台，本章節將介紹ROS的基本概念和使用方式。

## 系統架構

本論文提出了一種結合傳統路徑規劃和深度學習的自主導航系統。系統架構如圖3.1所示。在此架構中，機器人端會負責地圖建構，同時獲取攝影機、2D LIDAR以及里程計等資訊，這些資訊將被傳送到電腦端。在電腦端，系統會根據攝影機所回傳的影像判斷是否觀測到目標物，並根據傳統演算法或神經網路來決策動作。該系統旨在提供平穩高效的導航。

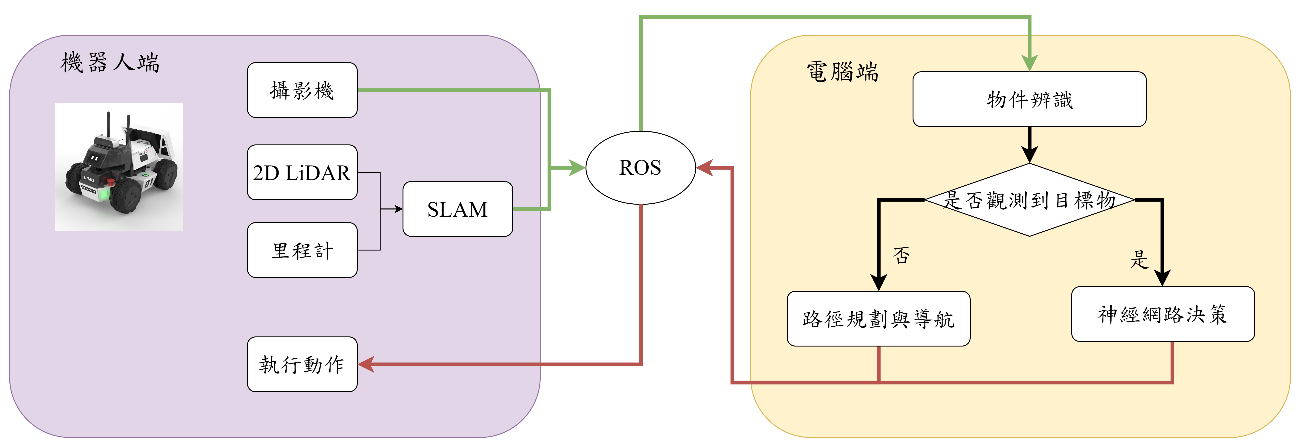


圖3.1 系統架構圖

## 硬體介紹

硬件方面，本論文採用Agilex[38]所開發的移動機器人Limo[39]作為系統的導航載體，其搭載了奧比中光 DaBai攝影機、EAI X2L光學雷達等感測器。使用四個14.4W輪轂馬達作為Limo動力來源，以Jetson Nano作為邊雲運算平台，如圖3.2所示。

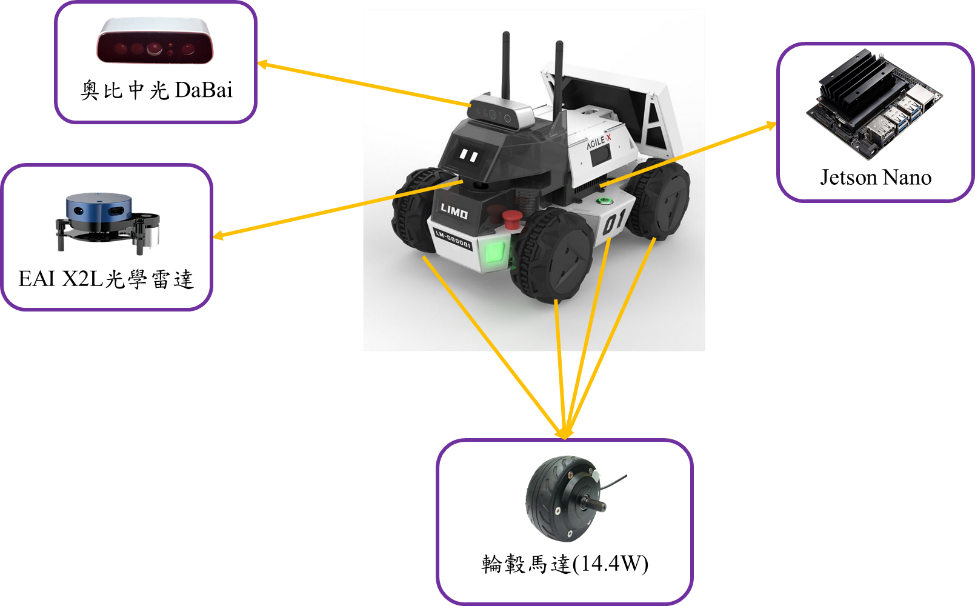


圖3.2 移動機器人Limo

### 邊緣運算平台

Jetson Nano是NVIDIA邊緣運算平台如圖3.3所示，該平台可以完成地圖建構以及與電腦端的溝通任務。



圖3.3 NVIDIA Jetson Nano[40]

在系統中，機器人端使用Jetson Nano來構建地圖，並獲取攝影機、2D LIDAR和里程計等資訊，這些資訊會被傳送到電腦端進行導航動作的決策。詳細規格如表3.1所示。

表3.1 NVIDIA Jetson Nano規格表[40]

|  |  |
| --- | --- |
| 項目 | 規格 |
| GPU | 128-Core Maxwell |
| CPU | Quad-core ARM57 @1.43Ghz |
| 記憶體 | 4GB 64Bit LPDDR4 25.6GB/s |
| 儲存空間 | 16 GB Emmc 5.1 |
| 電源 | 5V/4A 直流輸入  5V 直流輸出(GPIO) |
| 尺寸規格 | 100 mm80 mm29 mm |
| 作業系統 | Ubuntu 18.04 LTS |

### 影像擷取設備

DaBai是奥比中光旗下的一款深度攝影機，作為本系統中的視覺感測器作為物件辨識的主要元件，如圖3.4所示。



圖3.4 奥比中光DaBai[41]

其規格如表3.2所示[41]。

表3.2 奥比中光DaBai規格表

|  |  |
| --- | --- |
| 項目 | 規格 |
| 深度距離 | 0.3-3m |
| 功耗 | 整機工作平均功耗<2W，  鐳射開啟瞬間峰值 <5W(持續時間 3ms),  待機功耗典型值為<0.7W |
| 深度圖解析度 | 640\*400@30FPS  320\*200@30FPS |
| 彩色圖分辨率 | 1920\*1080@30FPS  1280\*720@30FPS  640\*480@30FPS |
| 電源 | 5V/4A 直流輸入  5V 直流輸出(GPIO) |

### 2D LiDAR

EAI X2L是EAI公司推出光學雷達作為建圖及神經網路導航設備，如圖3.5所示。

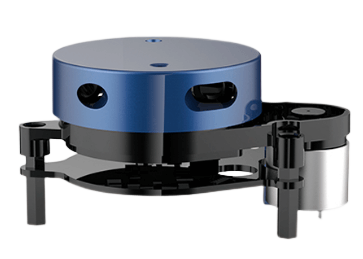


圖3.5 EAI X2L光學雷達[42]

其規格如表3.3所示[42]。

表3.3 EAI X2L光學雷達規格表

|  |  |
| --- | --- |
| 項目 | 規格 |
| 測距頻率 | 3000 Hz |
| 掃描頻率 | Min: 5; Max: 8 Hz |
| 測距範圍 | Min: 0.12; Max: 8 m |
| 掃描角度 | 0-360 Deg |
| 絕對誤差 | 2 cm |
| 相對誤差 | 3.5% |
| 角度解析度 | 5 Hz: 0.60; 6 Hz: 0.72; 8Hz: 0.96 |

### 電腦端

本論文中，物件辨識的神經網路、路徑規劃與導航以及神經網路決策的運算皆由算力更強大的電腦端進行，包括神經網路的訓練。在訓練神經網路的過程中，主要使用NVIDIA GeForce GTX 1060顯示卡，其具體規格可參考表3.4[43]。

表3.4 NVIDIA GeForce GTX 1060顯示卡規格表

|  |  |
| --- | --- |
| 項目 | 規格 |
| CUDA核心 | 1280個 |
| 記憶體容量 | 6GB GDDR5 |
| 記憶體介面 | 192-bit |
| 基本時脈 | 1506 MHz |
| 加速時脈 | 1708 MHz |
| 記憶體時脈 | 8008 MHz |
| 功耗 | 120W |

## 開發環境及各種軟體介紹

本文所使用的操作系統、深度學習框架、軟體等如表3.5所示。

表3.5 系統軟體版本

|  |  |
| --- | --- |
| 套件 | 版本 |
| Python | 3.9.12 |
| Numpy | 1.23.1 |
| Torch | 1.13.1 |
| Ubuntu | 18.04 |
| Ros | Melodic |
| Gazebo | 9.0.0 |

### 深度學習框架

深度學習框架Torch[44]是一個開源的深度學習框架，提供了一個強大的數據處理能力和建模工具，旨在幫助開發者構建高效、靈活的神經網路模型。在深度學習領域中，Torch框架被廣泛應用於計算機視覺、自然語言處理和強化學習等多個領域。可以通過組合不同的模塊構建神經網路，並使用自適應優化算法進行模型訓練，如隨機梯度下降、Adam[45]等。Torch支持多種常用的神經網路層，如全連接層、卷積層、循環神經網路等，並提供了豐富的數據處理工具，如圖像處理、文本處理、時間序列等，以及多種常用的損失函數和指標評估方法，如均方誤差、交叉熵等。總之，Torch框架具有簡單易用、高效靈活、可擴展性強等優點，在深度學習領域中得到了廣泛的應用和推崇。

### 機器人作業系統(Robot Operating System, ROS)

機器人的控制涉及到多個感測器和執行器，需要對這些元件進行即時的資料獲取、處理和控制，因此通常是一個非常複雜的問題。ROS[46]作為一種機器人軟體發展平臺，提供了一系列的工具和框架，可簡化機器人控制的實現。ROS提供了消息傳遞機制，其中ROS Master作為中心節點協調ROS節點之間的通信。所有ROS節點都需要向ROS Master進行註冊，才能進行通信。當節點需要與其他節點通信時，它們會向ROS Master發送消息以查找所需的資訊，例如主題名稱、發佈者和訂閱者等。以圖3.6為例，感應器A和感應器B希望向控制器發佈消息，控制器需要接收感應器A和感應器B的消息並做出控制信號，然後控制信號被發送到馬達。為了實現這一過程，所有節點必須在ROS Master上進行註冊，以便它們可以相互通信和交換資訊。因此，ROS Master在ROS消息傳遞機制中扮演著重要的角色，起到協調和管理的作用，但並不參與具體的消息傳遞過程。

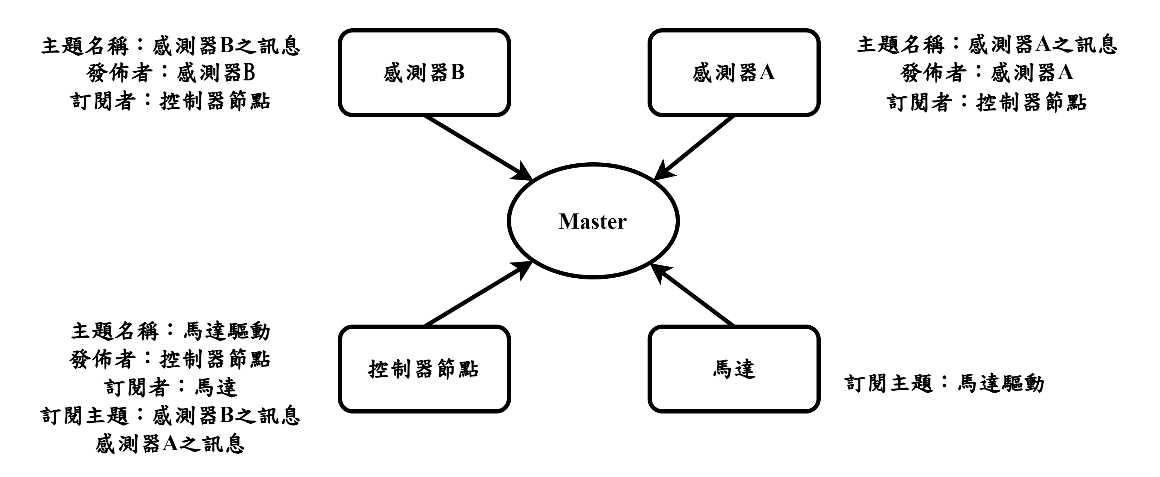


圖3.6 ROS註冊節點圖

在圖3.6中，節點向Master註冊後，即可進行通訊，如圖3.7所示。各節點可以通過ROS的發佈和訂閱機制進行消息傳遞。

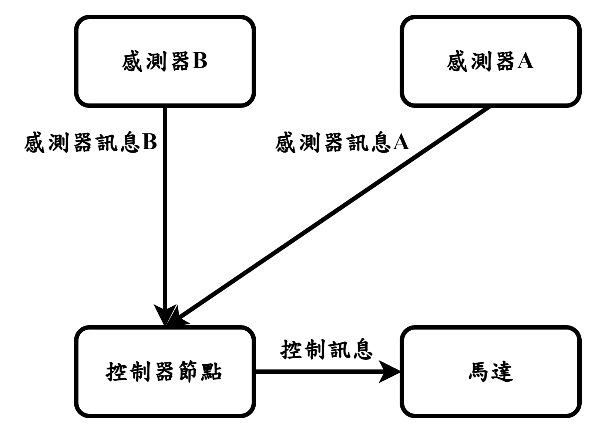


圖3.7 ROS節點通訊圖

# 基於強化學習的導航避障

這一章中，將討論如何搭建模擬環境、設計適合的動作空間和狀態空間、制定適當的獎勵設計、以及構建適用於導航避障任務的網路架構。此外，還將介紹環境流程設計和訓練流程，以實現有效的學習過程。

## 模擬環境搭建

正如2.2小節所提到的，強化學習通過與環境交互來學習。但在真實的環境中，與環境交互的代價可能會非常昂貴，例如在實際的機器人上進行實驗需要花費大量的時間和金錢。為了解決這個問題，模擬環境被廣泛地應用於強化學習中。通過在模擬環境中進行訓練，可以更快地獲得大量的經驗，並進一步優化策略，同時也可以減少實際實驗的成本。此外，模擬環境還可以提供更多的自由度和靈活性，以便於設計更多樣化的實驗場景。因此，模擬環境在強化學習研究和應用中扮演著非常重要的角色。

Gazebo是一款模擬真實世界的機器人模擬器，其盡可能的模擬各種物理效應，如碰撞、重力、摩擦等。在本文將Limo架設於模擬環境，訓練用於決策的神經網路，並結合ROS設計符合馬爾科夫鏈的強化學習環境。

本論文中Limo即是代理人，為了得知代理人在是否發生碰撞行為，在代理人四周加裝虛擬的碰撞感測器(Collision Sensor)，除了碰撞感測器，代理人在模擬環境中還設置了其他感測器，如攝影機、2D LIDAR如圖4.1所示。這些感測器能夠幫助代理人感知周圍環境的信息，並利用這些信息進行決策。代理人的目標是在模擬環境中順利地行駛到指定的目的地，同時避免碰撞和其他潛在的危險。

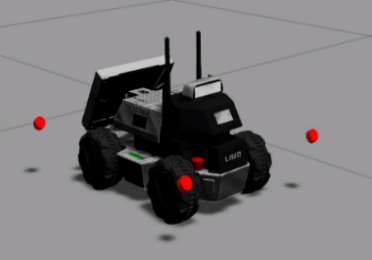


圖4.1 模擬環境的代理人(Limo)

結合ROS設計符合馬爾科夫鏈的節點，如圖4.2所示。其中Gazebo會發出模擬環境的感測器訊息，由DRL節點監聽及決策然後再透過ROS將決策結果傳遞給Gazebo模擬器，控制代理人的動作。

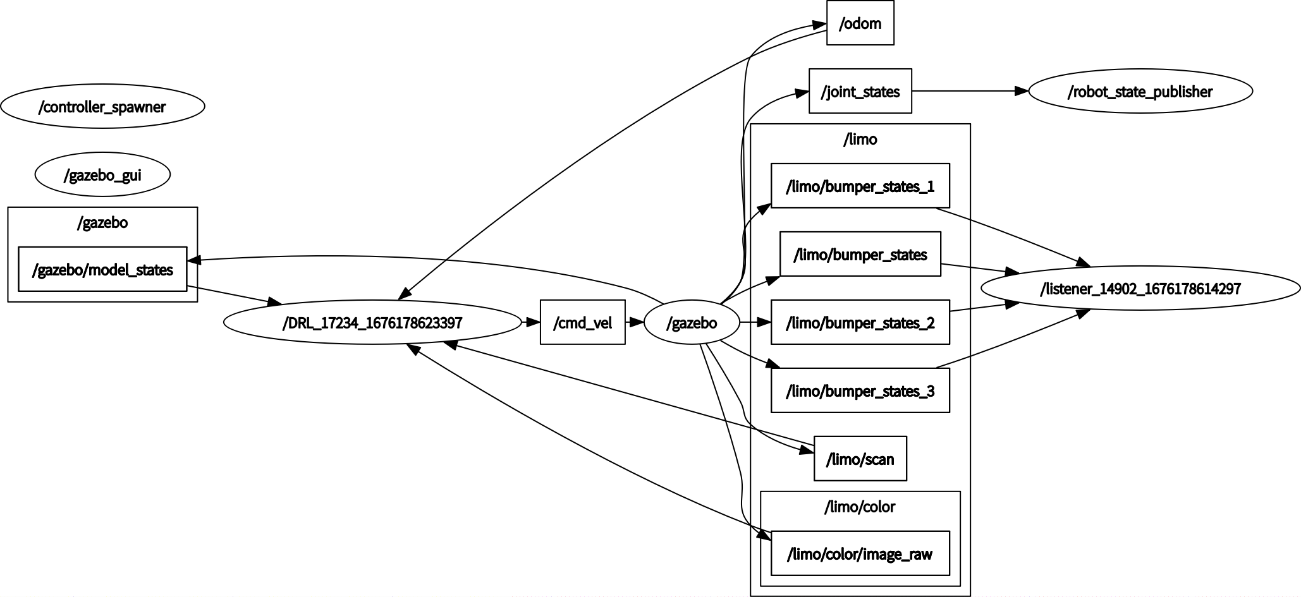


圖4.2 Gazebo ROS節點圖

為了迎合真實世界可能遇到的各種不同情況和環境，本論文在模擬世界搭建了三個不同的地圖，包括4\*5m無障礙物、4\*5m 靜態障礙物和4\*5m 隨機移動障礙物。在無障礙物區域中，代理人可以自由移動，沒有任何阻礙需要通過機器視覺到達指定地點，如圖4.3(a)所示；在靜態障礙物區域中，設置了一些不可移動的障礙物，代理人需要避免撞上它們的同時尋找到目標物，如圖4.3(b)所示；在隨機移動障礙物區域中，障礙物可以隨機移動，代理人需要根據實時感知的信息進行快速反應，避免碰撞，如圖4.3(c)所示。其中圖4.3中以紅色方塊當做目標物，藍色區塊為代理人(Limo)所搭載的2D LIDAR可視化的點雲。圖4.3(b)中的白色方塊是不可移動的靜態障礙物，圖4.3(c)的紫色小方塊是會隨機移動的障礙物。

|  |  |
| --- | --- |
| (a) | |
| (b) | (c) |

圖4.3 模擬環境中的地圖

## 動作空間與狀態空間設計

在代理人的控制中，動作空間和狀態空間是兩個核心概念。動作空間是指代理人可進行的所有動作的集合，而狀態空間是指機器人可能處於的所有狀態的集合。在本論文的導航與避障任務中，其中動作空間被定義為一個包含線速度()和角速度()的元組集合，即(4.1)所示。

|  |  |
| --- | --- |
|  | (4.1) |

考慮到模擬世界和現實世界之間的差距，在狀態的定義中不能夠只用影像，因此，本論文中的狀態空間被定義為包含激光雷達掃描點和二值化影像中目標物體的中心點的集合。如圖4.4所示，左圖為代理人在模擬環境中攝影機所拍攝之影像，將影像二值化後如右圖所示。其中，找到二值化後最靠左和最靠右的點(index)，並分別命名為和，從而找到目標物中心點，如(4.2)所示。

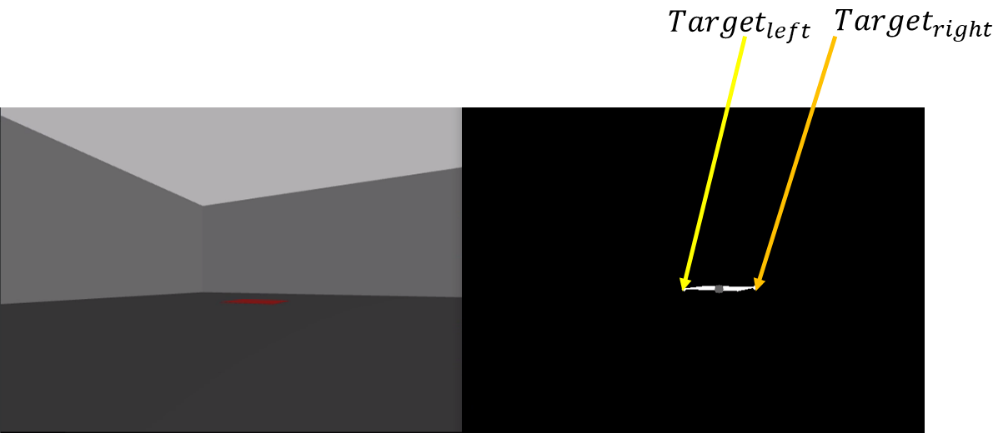


圖4.4 目標物狀態空間示意圖

|  |  |
| --- | --- |
|  | (4.2) |

因此，狀態在本文被定義為(4.3)

|  |  |
| --- | --- |
|  | (4.3) |

## 獎勵設計

獎勵作為深度強化學習過程中唯一的回饋信號起著至關重要的作用，獎勵的好壞直接決定了代理人學習效果的優劣。本論文在獎勵的設計方面分為距離獎勵、角度獎勵、動作獎勵、與障礙物距離及結束獎勵。距離獎勵如(4.4)所示，其中和是分別代表代理人和目標在空間中的位置向量，分母為代理人和目標之間的歐式距離。當代理人距離目標越近時，獲得的獎勵越高，以這樣的獎勵設計鼓勵代理人向目標移動。為避免分母為0的情況在(4.4)尾項加入，在本文中，被設置為一個極小值，從而避免產生不可解或錯誤的結果。。

|  |  |
| --- | --- |
|  | (4.4) |

角度獎勵是基於目標位置和代理人方向之間的夾角設計的，為了引導代理人能夠更快的學會如何抵達目標物，定義角度獎勵如(4.5)所示，希望代理人車頭方向越朝向目標物獎勵越大。其中，表示目標物坐標與代理人坐標之間的夾角，為代理人車頭在世界坐標系的方向。。

|  |  |
| --- | --- |
|  | (4.5) |

在公式(4.5)中，表示目標物坐標與代理人坐標之間的夾角，計算過程如(4.6)所示。其中和表示目標在空間中的位置向量，和表示代理人在空間中的位置向量。

|  |  |
| --- | --- |
|  | (4.6) |

[47]是加了約束的 arctan，解決了arctan在第二、第三象限時，無法得到正確的結果，如(4.7)所示其中表示圓周率。

|  |  |
| --- | --- |
|  | (4.7) |

如圖4.5所示，將坐標點帶入(4.6)計算出目標位置和代理人方向之間的夾角。

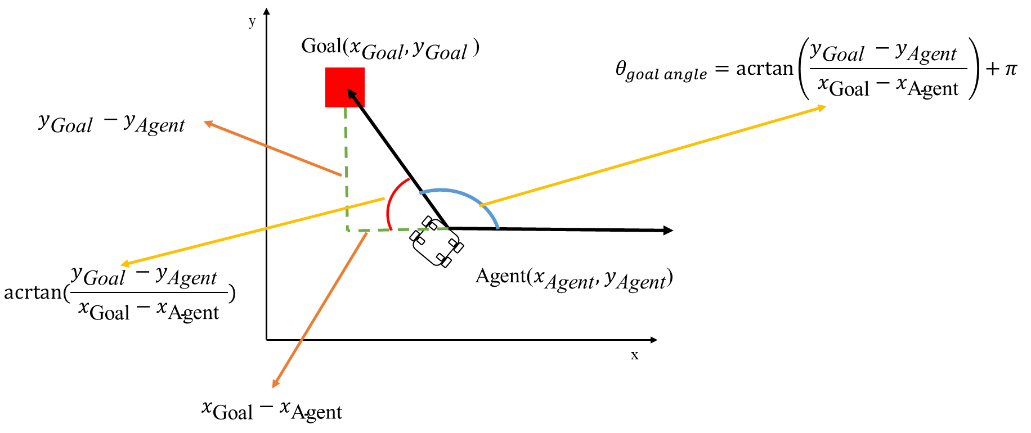


圖4.5 全局角度示意圖

(4.5)中的，為車頭在世界坐標系的方向。4.2小節所提到的動作空間的設計中，透過線速度()和角速度()決定代理人的動作。假設當前代理人的朝向為，控制系統決定讓代理人的角速度為，控制週期為。則代理人在時間內的朝向變化量為(4.8)

|  |  |
| --- | --- |
|  | (4.8) |

將加到當前的朝向上，即可得到代理人更新後的朝向，如(4.9)所示。

|  |  |
| --- | --- |
|  | (4.9) |

因此透過角速度與控制週期的控制，可以隨著時間不斷更新代理人的朝向。角度獎勵的設計如圖4.6所示，代理人越指向目標物獎勵也就越好，引導代理人盡快抵達目標物。

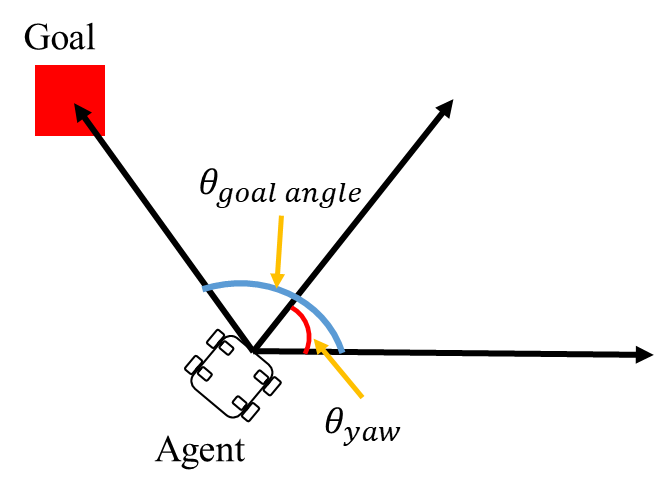


圖4.6 角度獎勵示意圖

為了鼓勵代理人盡可能快速到達目標物，即希望代理人以最快的速度到達目標物並且移動的時候盡可能的走直線，動作獎勵定義為(4.10)

|  |  |
| --- | --- |
|  | (4.10) |

此外，鼓勵代理人能夠躲避障礙物，將代理人與障礙物的距離作為懲罰因素，即代理人距離障礙物越近，懲罰越嚴重。這樣可以使得代理人更加努力避免與障礙物發生碰撞。如果2D LiDAR最小的點小於一定閥值，判定為與障礙物靠的太近而進行扣分，這個懲罰項的大小根據實際情況進行調整，定義為。當代理人成功到達目標物時，會獲得高分獎勵；而如果代理人發生碰撞，則會受到扣分懲罰。

## 網路架構

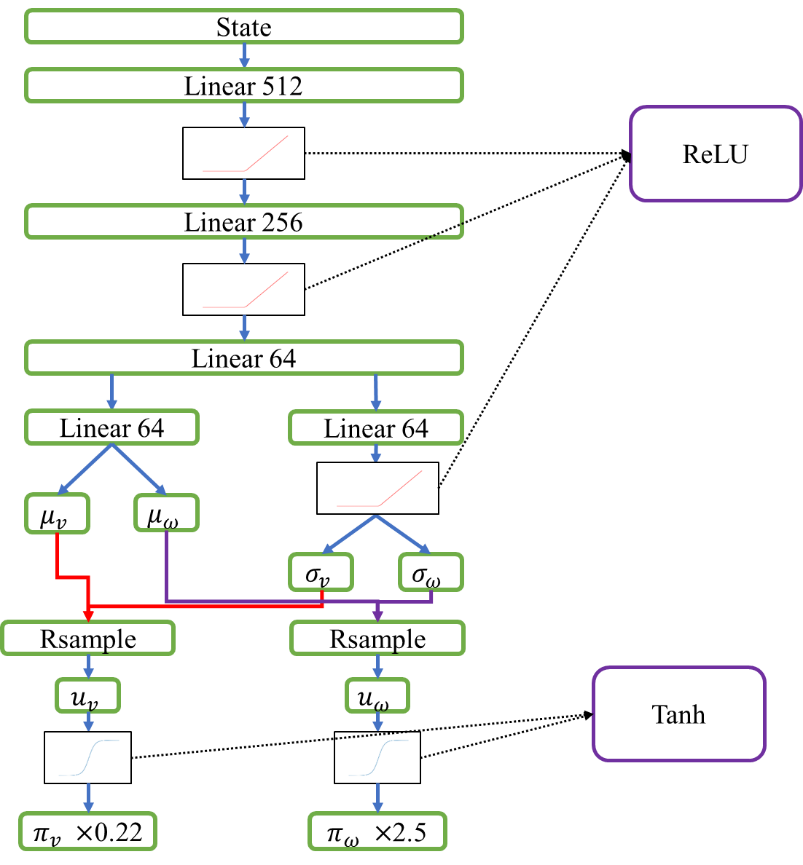


圖4.7 策略網絡示意圖(Actor)

本論文使用神經網路進行代理人動作決策，神經網路的架設如圖4.7所示，由(4.3)所定義的狀態輸入網路，經過多層神經元進行處理，輸出結果為動作的平均值()和標準差()。標準差的前一層有ReLU使得標準差的值不會為負的，根據輸出的平均值和標準差從高斯分布中抽取一個隨機變量，通過tanh函數壓縮到[-1,1]之間，分別乘以動作空間邊界符合4.2小節所定義的動作空間(4.1)。

但在採樣的過程中是會失去梯度的，以圖2.1的神經元為例，在反向傳播時會先計算每個節點的梯度，使用鏈式法則將其與上一層的梯度相乘，從而得到當前層對上一層的梯度。這樣反向傳播可以有效地計算出每個參數對損失函數的梯度，進而進行梯度下降優化參數。

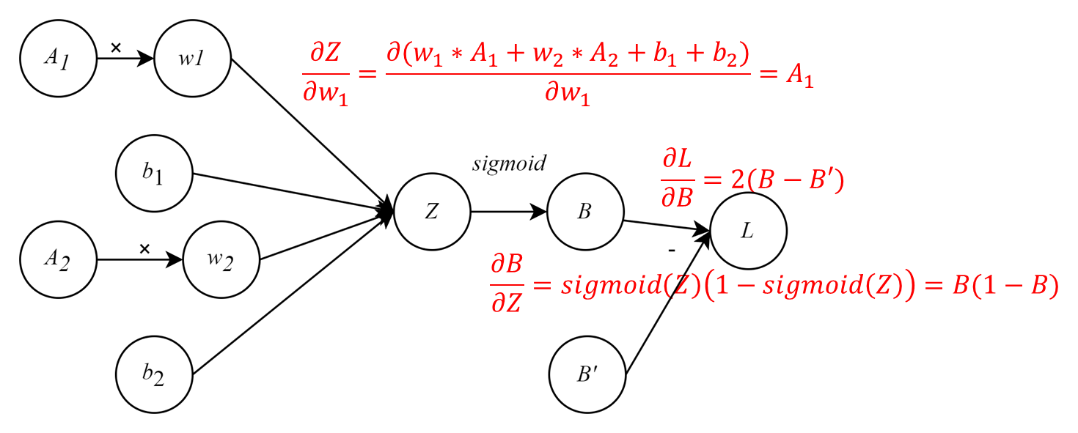


圖4.8 神經元計算圖

圖4.8是圖2.1神經元的計算圖，計算每個節點的梯度以為例，損失被定義為輸出值與標籤值的均方誤差。根據計算圖可以輕易的計算出L對的偏導如(4.11)所示。

|  |  |
| --- | --- |
|  | (4.11) |

在採樣的過程中，生成的隨機值是獨立於和的，因此它不會對和的梯度產生影響。由於是從無法微分的隨機分布中生成的，因此在反向傳播過程中，無法建立計算圖將的梯度傳遞回模型進行更新。如圖4.9所示，通過採樣使得和、的計算圖斷裂。

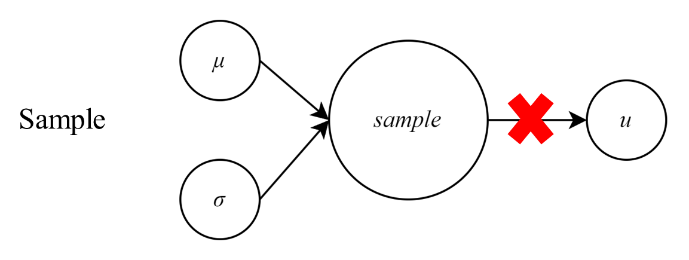


圖4.9 採樣示意圖

本文在網路架構的採樣處使用重參數化技巧(Reparameterization trick)[48]解決採樣的過程中失去梯度的問題。首先從一個是0，是1的標準正態分布中採樣一個值，這個採樣的值和神經網路輸出正態分布採樣的值之間只存在這縮放和平移之間的關係，因此可以表示為(4.12)。如圖4.10所示，這樣的操作使得輸出值與、產生關聯性使得採樣操作是可微的。

|  |  |
| --- | --- |
|  | (4.12) |

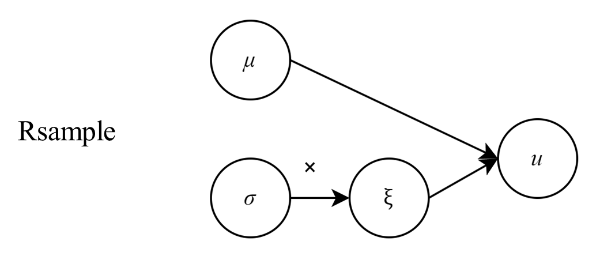


圖4.10 重參數技巧示意圖

為了方便後續訓練流程的設計，需要設計一個網路負責評估代理人做出某個動作的好壞程度。這個網路的目的是根據當前的狀態輸入和代理人決策出的動作，評估這個動作的優劣程度，並返回一個評估值，以便代理人進行下一步動作的決策。網路架構如圖4.11所示，輸入為(4.3)所定義的狀態和策略網絡所決策出的動作，輸出為動作價值、，分別對應代理人選擇向量和的優劣程度評估。

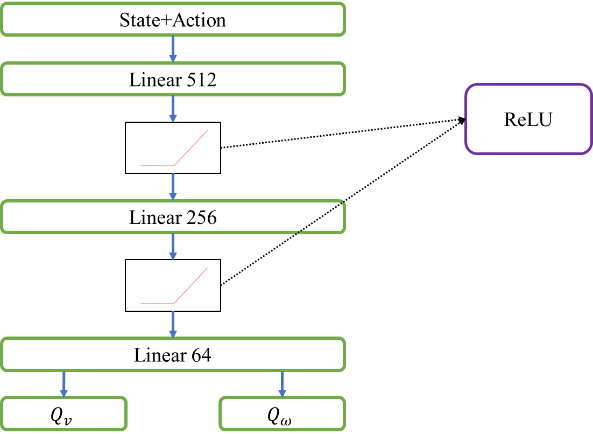


圖4.11 評論網絡示意圖(Critic)

## 環境流程設計

DRL演算法在應用的時候需要滿足MDP條件，即環境需要具有狀態轉移、即時反饋、獎勵函數等基本特徵。本文使用Gazebo和Ros模擬平台配合完成MDP條件，如圖4.12所示。在模擬環境中，首先需要初始化代理人的位置、速度和感測器的資訊，將感測器回傳的值整理為狀態空間的形式，並輸入至神經網路進行決策動作。同時，模擬環境會即時回饋代理人的位置和障礙物位置等資訊，用於判斷是否發生碰撞和是否抵達目標物。當代理人完成目標任務時，系統會給予相應的獎勵，反之則會給予懲罰。此外，在每個代理人完成一次任務後，系統會重新初始化模擬世界以讓代理人進一步學習。這樣重複操作可以完成MDP條件。



圖4.12 環境流程圖

## 訓練流程

代理人的訓練是通過與環境的交互來實現的。在本文使用SAC演算法進行神經網路的訓練，與圖4.12一樣，先由模擬環境中代理人的感測器獲得狀態，交由神經網路決策出動作，獲取下一時刻的狀態，計算獎勵及觀察回合是否結束，把每條軌跡存入經驗池，如圖4.13所示。

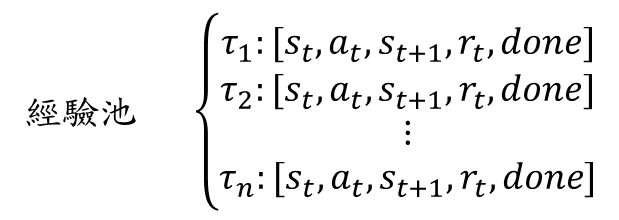


圖4.13 經驗池示意圖

從經驗池中抽樣取得一個批次的經驗，透過離線更新的方式進行Q值和策略網路的訓練。首先從經驗池一個批次輸入值Actor網路會得到動作和策略概率的對數。

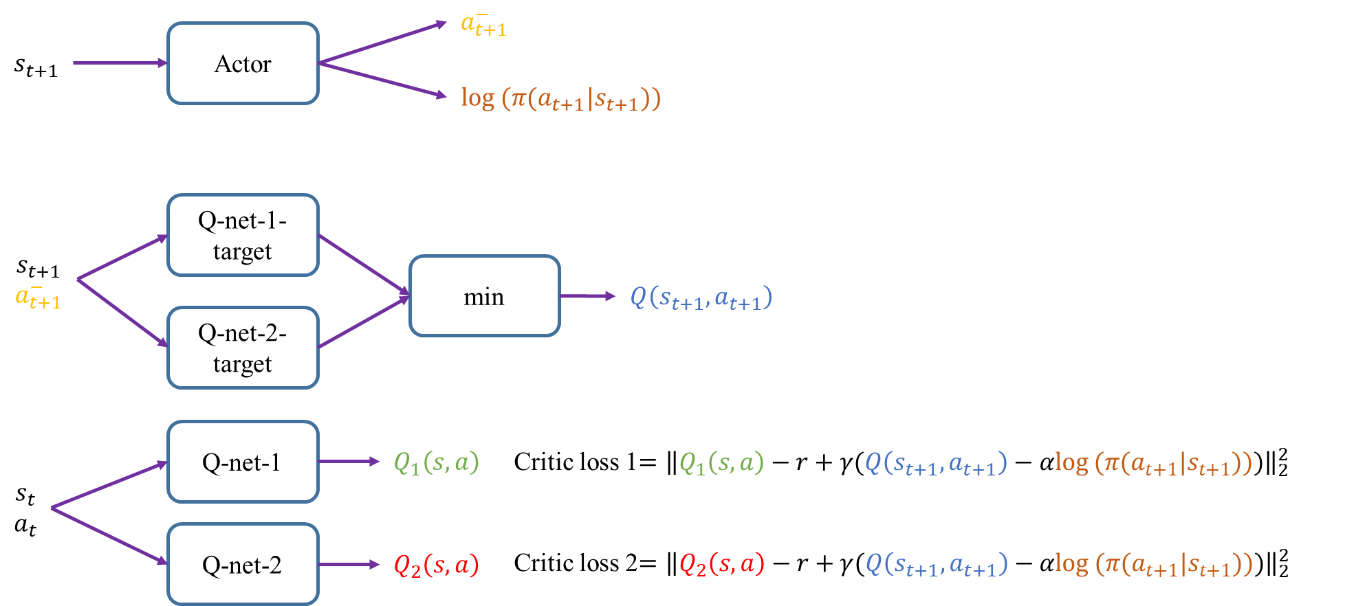


圖4.14 評論家更新示意圖

其中對數的運算如圖4.15所示，網路輸出的高斯分佈通過重參數化技巧採樣得到，而的概率可以直接帶入高斯分佈的概率密度函數計算。而策略是由通過tanh壓縮並且縮放得到的，所以策略的概率會等同於的概率通過變數替換[49]得到如(4.13)所示。也因此策略概率的對數如(4.14)所示，因為log(0)是未定義的，為了保證運算的正常，在中加入了一個極小值。

|  |  |
| --- | --- |
|  | (4.13) |
|  | (4.14) |

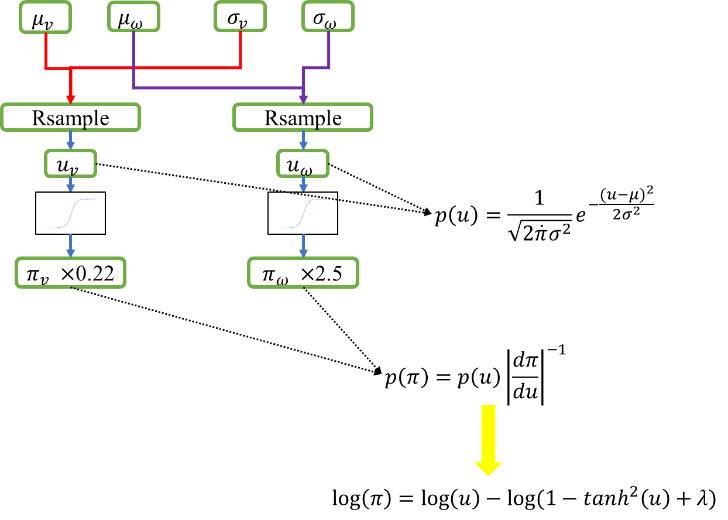


圖4.15 策略對數示意圖

緊接著，將和輸入至兩個目標評論網路，分別計算出t+1時刻Q值的估計值，並選擇其中較小的一個作為的估計值，將經驗池抽取的和輸入給兩個評論網路，分別計算t時刻的Q值：、，並且利用最小二乘法來分別更新兩個評論家網路，如(4.15)所示。其中是從經驗池抽取軌跡的獎勵，、是超參數分別代表2.3章節所提及的折扣率和溫度係數。

|  |  |
| --- | --- |
|  | (4.15) |

兩個目標評論家網路由兩個評論家網路分別按照一定比例更新，如(4.16)所示。其中表示圖4.14中Q-net-1-target網路的參數，為Q-net-1網路的參數，、分別為Q-net-1-target、Q-net-1網路的參數。表示更新參數的比例，其是一個0到1之間的浮點數。

|  |  |
| --- | --- |
|  | (4.16) |

決策網路的參數更新如圖4.16所示，將經驗池抽取的輸入至決策網路計算得到及，將、輸入給兩個評價網路估計動作價值Q，選取比較小的作為。計算優化目標]，運用梯度下降算法更新決策網路參數。

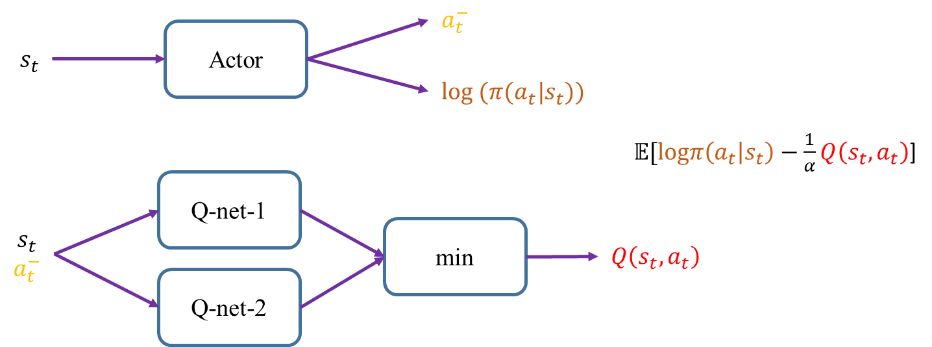


圖4.16策略網路更新示意圖

溫度係數是用來衡量獎勵與熵之間重要性的比值，會變動的會顯得比較合理，的更新如圖4.17所示。同樣的從經驗池抽取的軌跡中，將輸入給策略網路計算及，計算的優化目標]，其中是一個超參數，在本文被定義為負的動作空間大小-2，運用梯度下降算法更新。

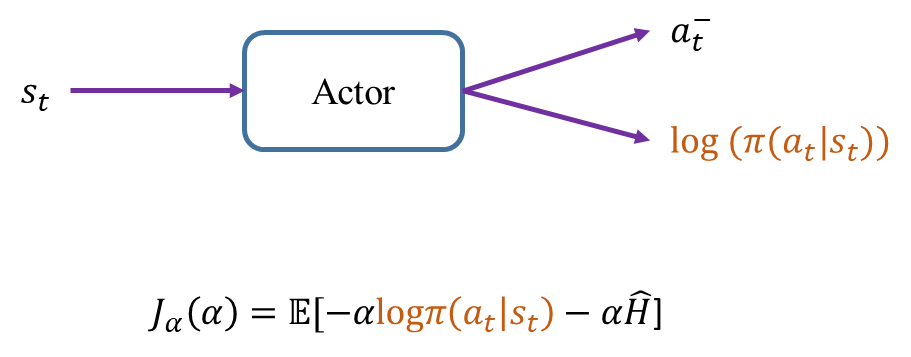


圖4.17 溫度係數更新示意

在模擬環境中，代理人依靠感測器回傳的值進行動作決策，並將與環境互動的軌跡存入經驗池。當經驗池中的經驗數量達到批次大小時，進行經驗抽取並依次更新四個神經網路：策略網路、兩個評論家網路、兩個目標評論家網路和一個溫度係數。若經驗池中的經驗數量不足批次大小，代理人會繼續與環境互動以收集更多經驗。當代理人到達訓練次數的限制時，則結束訓練。以上是神經網路在模擬環境中的更新流程，如圖4.18所示。

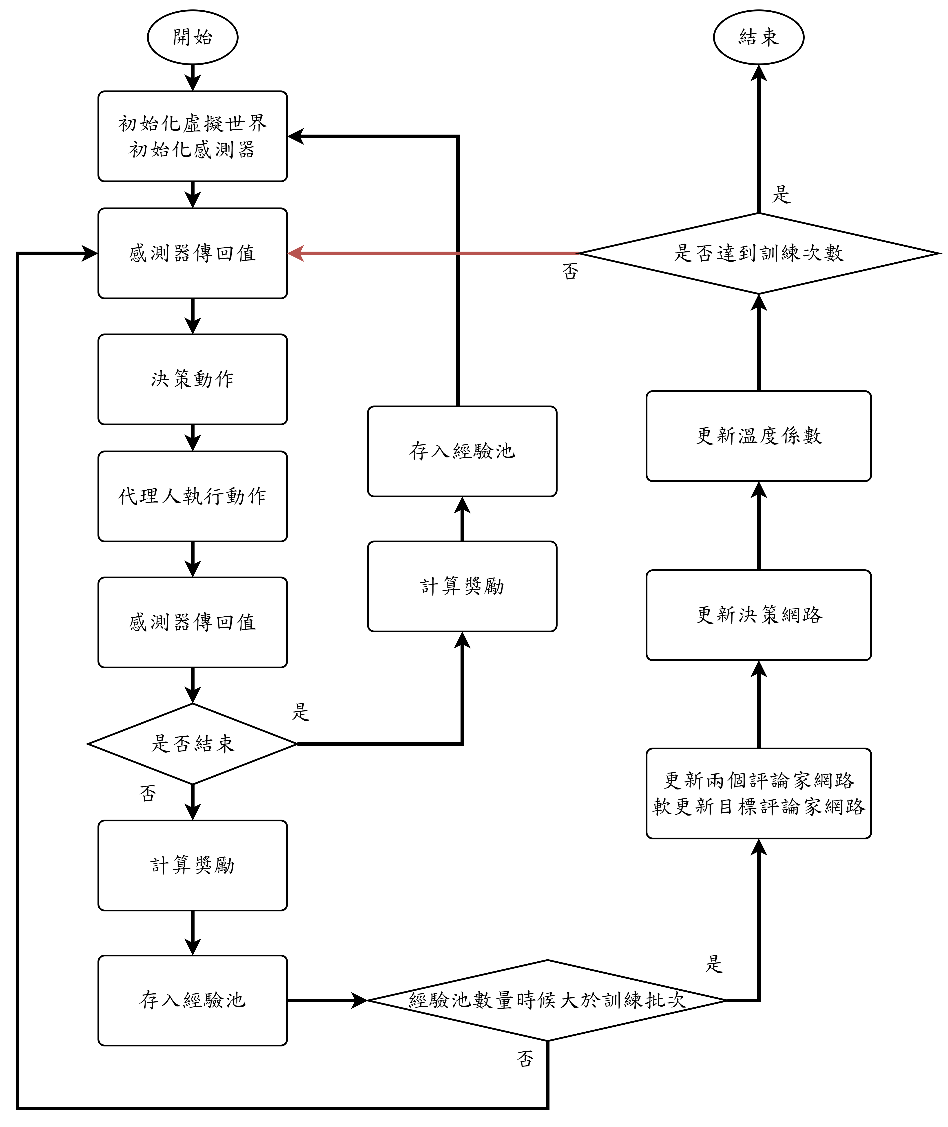


圖4.18 神經網路訓練流程

## 第四章總結

本章節介紹了在系統架構圖中的神經網路決策，運用模擬環境對神經網路進行訓練、實現代理人與環境的互動學習的方法。首先，在模擬環境中建立了符合MDP框架的流程，並分別設計了獲取狀態空間和動作空間的方法，以及獎勵函數和神經網路的設計。接著，本文設計了如何使用Soft Actor-Critic演算法在模擬環境訓練神經網路的方法，並且對各個神經網路的更新進行了詳細的說明。最後，我們給出了在模擬環境中進行神經網路訓練的流程圖。

# 訓練優化、模型遷移與路徑規劃導航

本章將探討基於強化學習的導航避障方法的進一步優化和應用。在前面的章節中，介紹了導航避障系統的基本架構和訓練流程。然而，為了進一步提升系統的性能和應用能力，本章將探討幾個關鍵問題。首先，將探討結合監督學習的訓練優化方法，以加快系統的學習速度和提高導航避障策略的效果。其次，將討論模擬環境轉移到現實世界所面臨的困難，並探討模型遷移設計和物件偵測的相關技術，以提高系統在真實環境中的適應能力。使代理人能夠智能地規劃最佳路徑並達到預定目標。這些優化方法和應用技術將有助於進一步提升本系統的性能和實用性。

## 結合監督學習的訓練優化

在強化學習中，模擬環境是訓練代理人的常用工具。然而，由於模擬環境本身的限制，模型的表現常常不能符合預期。為了解決這個問題，本節介紹一種結合監督學習的方法，用於優化強化學習訓練的過程，以提高訓練效果。本文嘗試在模擬環境中使用監督學習的方法來優化強化學習的訓練，以便在短時間內使代理人學習復雜的任務。

以監督式學習的方法作用於自主導航中，將輸入的狀態直接映射到輸出的動作。這可以被視為一個回歸問題，因為試圖預測一個連續的輸出值(動作)。在這種情況下，使用許多已標記的訓練示例來訓練模型，以學習輸入狀態和輸出動作之間的映射關係，這種做法又被稱為行為複製(Behavior Cloning)。如圖5.1所示，但監督式學習往往需要大量的標記訓練數據，這對於自主導航等實時應用來說可能是一個挑戰。

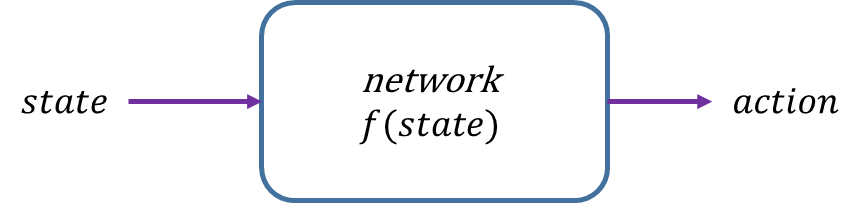


圖5.1 監督式學習於自主導航示意圖

因此，為了克服監督式學習需要大量標記數據的限制，將監督式學習與強化學習相結合。使用監督式學習來訓練一組初始化參數，然後使用強化學習來對這些初始化參數進行進一步的優化。這樣的結合方法通常稱為混合式學習，可以在不使用大量標記數據的情況下實現代理人的快速訓練和優化。如圖5.2所示，首先透過模擬環境中感測器的回傳值整理狀態，然後使用人為的決策動作作為監督式學習的標籤(label)，將相同的狀態輸入至決策網路(Actor)的初始網路。接著，利用均方差作為網路的損失(loss)進行訓練，以更新SAC的決策網路。其中表示SAC的決策網路，是監督式學習的學習率。



圖5.2 以監督式優化強化學習訓練示意圖



圖5.3 專家數據收集流程圖

人為的決策動作稱之為專家數據，在模擬環境中專家數據的收集流程，如圖5.3所示。首先，初始化虛擬世界和感測器數據，並監聽感測器的回傳值進行人工判斷動作。為確保人工判斷有足夠的思考時間，只有當確定動作後，才會控制代理人進行運動。同時，檢查代理人的線速度和角速度是否都為0，如果都為0則表示沒有人工操作，不會記錄狀態和動作，否則將其記錄以供學習。然後，檢查是否繼續收集專家數據，並且判斷代理人是否撞墻或到達目標物。

## 模型遷移設計與物件偵測

模型遷移指的是將在模擬環境中訓練好的神經網路模型應用於現實世界的場景中。儘管模型在模擬環境中表現出色，但實際應用中可能會遇到許多挑戰和困難。例如在4.2章節中提到的狀態空間設計中，目標物的中心點Target在現實世界中很難像在模擬世界中一樣通過二值化方法找到。

|  |  |
| --- | --- |
| (a)模擬環境代理人攝取影像 | (b)真實世界代理人攝取影像 |
| (c)模擬環境圖像的二值化 | (d)真實世界圖像的二值化 |

圖5.4 狀態空間模擬世界與真實世界對照圖

如圖5.4所示，(a)和(b)分別是模擬世界和真實世界透過攝影機攝取的影像，(c)和(d)分別對應(a)和(b)圖像的二值化結果。可以發現，在模擬世界中攝取的影像(a)經過二值化處理後(c)，目標物非常清晰規整，很容易找到目標物的中心點。然而，在真實世界中攝取的影像(b)經過二值化處理後(d)，由於存在許多環境噪音，因此無法找到中心點。

因此，在本文將使用YOLO-v7-tiny[50]來尋找狀態空間中的目標物的中心點Target。YOLO(You Only Look Once)系列是一種基於深度學習的物件偵測技術，以其快速的檢測速度聞名於世。如圖5.5所示，經過訓練的YOLO會尋找出所要找的物件，以bounding box(bbox)的形式做為網路的輸出。bbox表示物件的框，以長h，寬w及中心點x，y進行物件的定位。

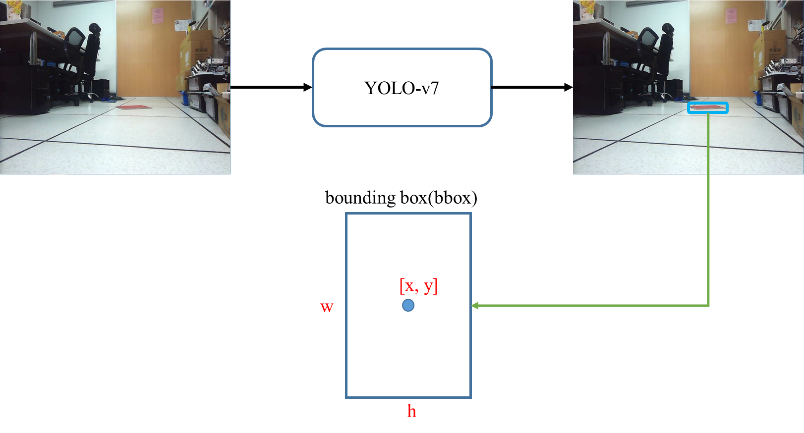


圖5.5 YOLO示意圖

YOLO-v7-tiny分為主幹網路、Head層，其中主幹網路的架構如圖5.6所示， 其中CBL(-1, 32, 33, s=2)表示一個卷積層接續一個Batch Normalization[51]再接續激活函數LeakyReLU[52]。(-1, 32, 33, s=2)則是這個CBL的具體參數，假設輸入圖像大小為[1,3,600,480] (批次數為1，3通道的RGB圖像，長寬分別為600、480)，CBL參數中-1表示上一層網路輸出的深度，也就是3，通過CBL(-1, 32, 33, s=2)會與32個大小為3\*3的卷積核進行點積運算，卷積核一次移動兩格，最後輸出大小為[1, 32, 300, 240]的特徵圖。Concat(dimension=1)表示特徵圖會在第一個維度進行拼接，也就是深度的維度。

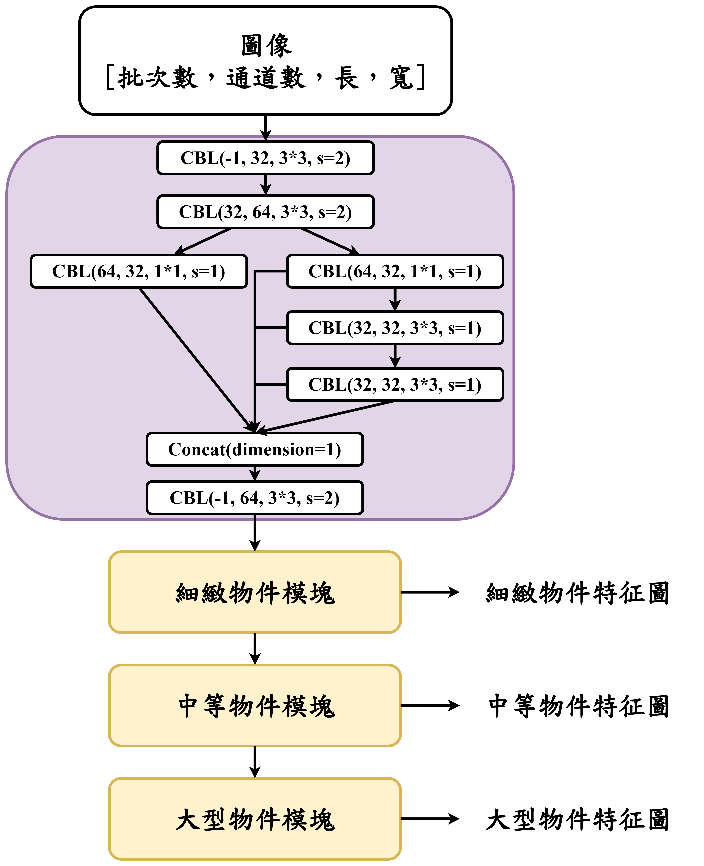


圖5.6 YOLO-v7-tiny主幹網路示意圖

為了在 YOLOv7-tiny 中能夠有效地檢測各種不同大小的物體，該模型會返回3種不同大小的特徵圖，供後續的head層進行特徵提取和bounding box計算。

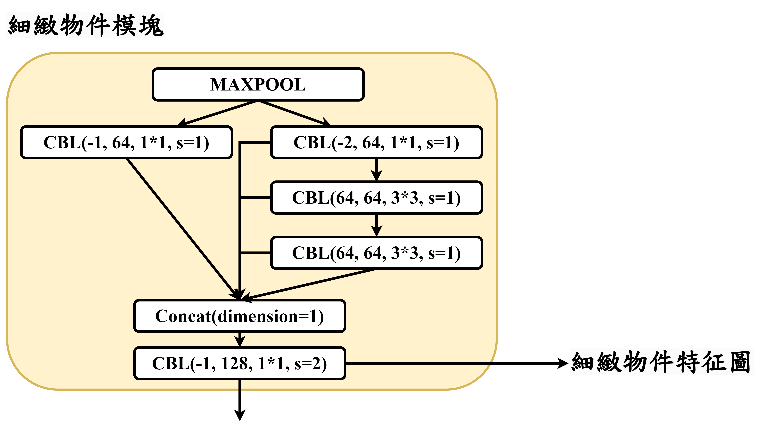


圖5.7 細緻物件模塊示意圖

以細小物件模塊為例，如圖5.7所示，其中使用MAXPOOL[53]進行下採樣操作，並透過兩條殘差分支提取特徵，然後在深度維度進行拼接，接著再通過一個CBL模塊進一步提取特徵，最後將所得的細緻物件特徵圖傳遞到Head層進行更進一步的特徵提取。中等物件模塊和大型物件模塊的架構與細緻的物件模塊完全相同，但中間CBL模塊的輸出通道分別為128和256。

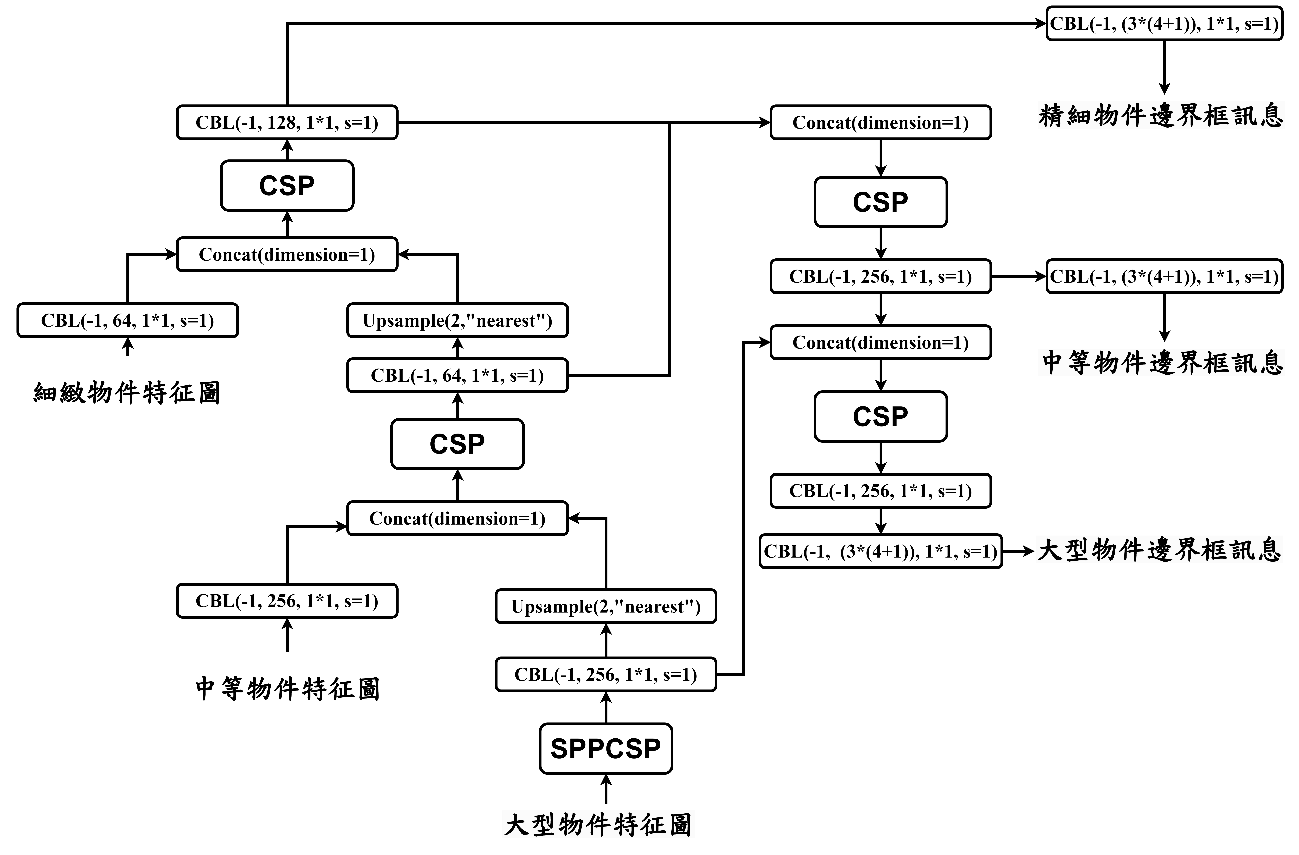


圖5.8 YOLO-v7-tiny head層示意圖

在YOLO-v7-tiny中，主幹網路會提取三種不同大小的特徵圖並傳遞到head層，如圖5.8所示。這三種特徵圖是由主幹網路的不同層提取而來，例如最後一層提取的特徵圖專門用於檢測大型物件，因此比前兩個特徵圖包含更多的特徵。在head層中，這三種特徵圖會被上採樣到相同的尺寸，然後進行特徵提取和融合。其中，Upsample(2,"nearest")層表示上採樣的比例為2，採用最近鄰插值的方法進行上採樣。SPPCSP[54]、CSP[55]模塊如圖5.9所示，CSP模塊的延續來自於ResNet[56]中的殘差連接思想，將特徵圖在通道維度進行分割，將一部分分支作為跨級連接部分，另一部分作為本級處理部分。這種分割的方式避免了在特徵圖通道數量過大時出現的梯度消失和記憶體佔用過大的問題。SPPCSP模塊則是在CSP模塊的基礎上加入SPP，SPP會把輸入特徵圖分成不同的池化區域，每個區域池化成一個固定大小的特徵向量。這樣可以獲得不同大小的特徵向量，用於表達輸入特徵圖中的局部和全域資訊。

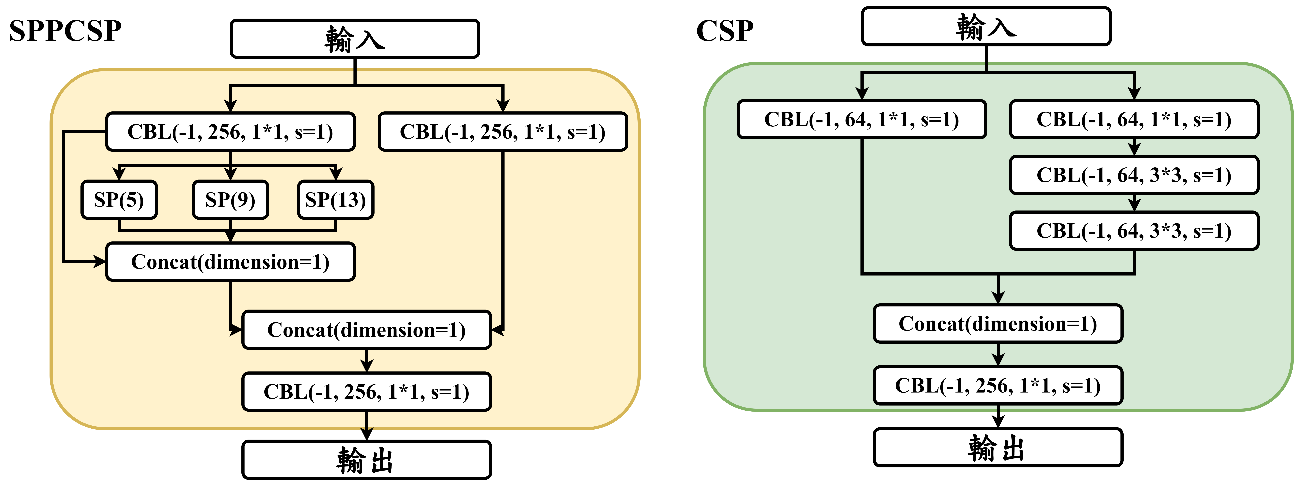


圖5.9 SPPCSP、CSP架構圖

head層最後的輸出的會是3個大小尺寸的bbox的訊息，以大型物件邊界框訊息為例，其會是一個[20\*20\*(3\*(4+1))]， 每個特徵圖上的每個網格單元預測3個bbox，每個bbox包含4個座標值和1個置信度分數，因此每個網格單元的預測輸出深度為(3\*(4+1))=15。隨後通過非極大抑制(Non-Maximum Suppression, NMS)[57]消除不必要的框，只保留機率最高的邊界框作為最後的預測框。

## 路徑規劃與導航

在本章中，將完成系統架構圖中路徑規劃導航的設計。當代理人無法看到目標物時，神經網路難以做出明智的決策。因此，本文將使用路徑規劃導航算法來引導代理人遵循預先設定的路徑行駛，直到能夠看到目標物為止。將使用A\*( A Search Algorithm)[58]+DWA(Dynamic Window Approach)[59]算法來實現當代理人無法看到目標物時的路徑規劃導航。這種方法是基於事先建立的地圖和定位系統的導航方式。

首先對環境進行地圖建構，使用Gmapping[60]建構地圖。在完成地圖建構後，通過角速度及線速度估計出代理人自身位置，使用A\*+DWA算法來計算代理人沿著地圖的最佳路徑，使用Ros的Navigation工具包實現[61]，其中包含定位包ACML(Adaptive Monte Carlo Localization)[62]，ACML算法使用機器人的移動模型和上一時刻的位置估計來產生一個機器人的可能位置集合，即粒子集。在估計步驟中，ACML算法通過感測器提供的測量信息對每個粒子進行位置估計和權重更新，從而得到機器人的最終位置估計。A\*算法是一種廣泛使用的路徑規劃算法，其使用啟發式搜索來尋找最短路徑。用於計算每個節點的價值如(5.1)所示。其中，表示從起始節點到節點的估計代價；表示起始節點到節點的實際代價；表示節點到目標節點的估計代價，不斷地搜索尋找最佳路徑。

|  |  |
| --- | --- |
|  | (5.1) |

以圖5.10為例，圖中的數字表示估計代價，其中起點到當前節點的實際代價由起點到當前節點的曼哈頓距離計算，即起點到當前節點所走的步數。節點到目標節點的估計代價由歐氏距離計算。首先對相鄰的節點進行探索，計算估計代價，並選擇估計代價小的節點進行探索，直到到達終點。從而生成黃色的最佳路徑。

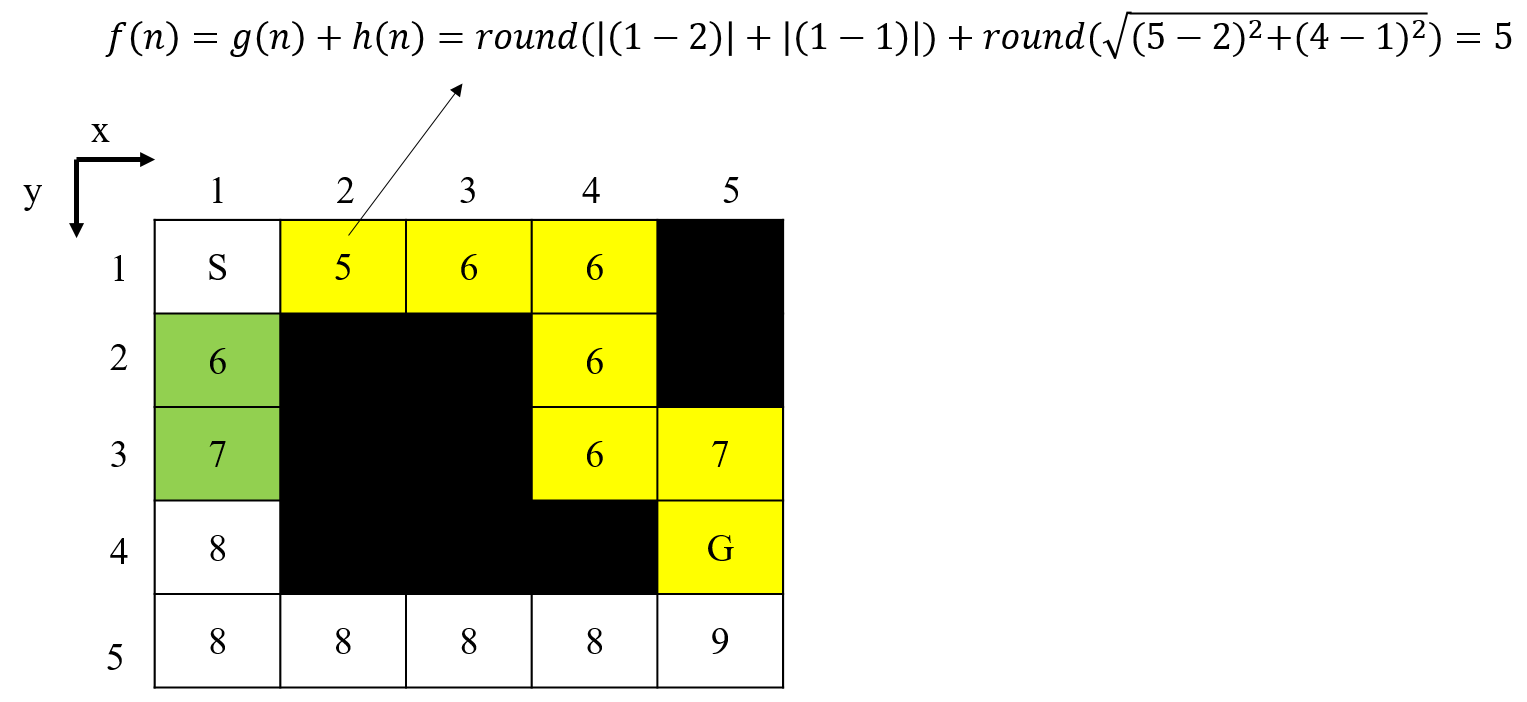


圖5.10 A\*演算法示意圖

在路徑規劃好之後，交由DWA演算法規劃動作。首先將A\*規劃的路徑劃分為若干個線段，然後針對每一段生成很多可以運動的動作，如圖5.11所示。接着，使用(5.2)對運動線段進行評估，其中：該軌跡的方向與目標方向的夾角，越接近目標方向，分數越高。：該軌跡與障礙物的最小距離，越大，分數越高。：該軌跡的速度越快，分數越高。：該軌跡的轉向角度，越小，分數越高。：軌跡與障礙物的相對方向，越接近與障礙物相反方向，分數越高。、分別代表調整各項參數的權重。最後，選擇得分最高的運動軌跡作為最佳動作。

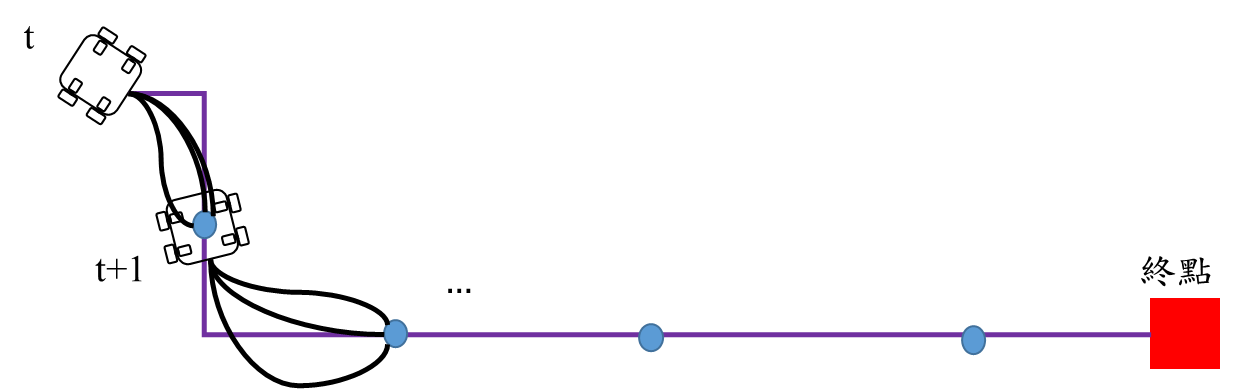


圖5.11 DWA演算法示意圖

|  |  |
| --- | --- |
|  | (5.2) |

## 神經網路結合路徑規劃與導航

神經網路及傳統的路徑規劃與導航的算法中，在移動載具的控制中可謂各有各的優點。在本文中，神經網路依靠攝影機影像、2D LiDAR的回傳值決策路徑完成導航，不會依賴定位的需求，但在看不到目標物的情況下難以決策優秀的動作。而傳統的A\*+DWA算法則需依靠建構的地圖和精準的定位完成。

本文提出的系統流程如圖5.12所示：系統開始時，攝影機會不斷地拍攝環境，並透過訓練好的YOLO-v7 tiny物件偵測，如果偵測到目標物，就會提供狀態給神經網路進行動作決策。如果未偵測到目標物，則使用A\*規劃全局路徑，並透過DWA算法進行導航。重複這個過程，直到到達目標物。

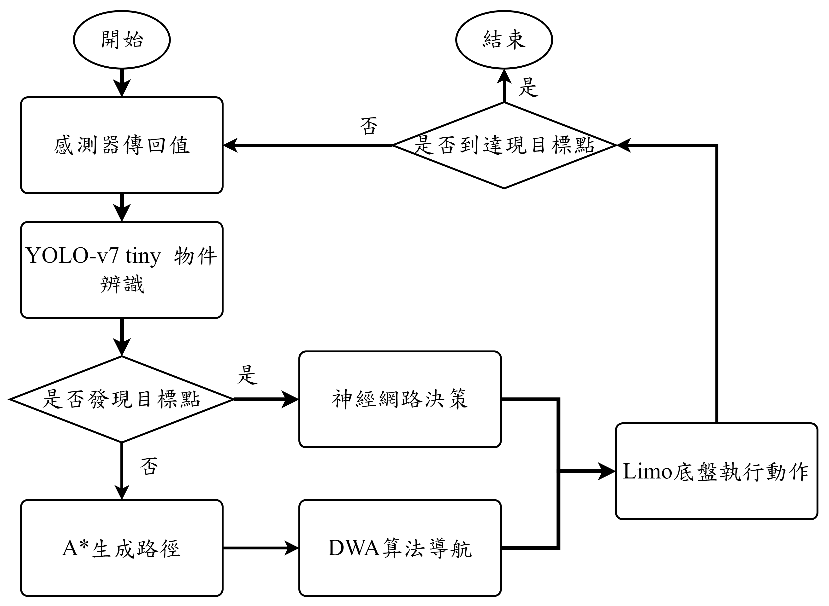


圖5.12 系統流程圖

## 第五章總結

在本章節中，提出了結合監督學習的訓練優化方法，以加快強化學習的收斂速度和提高表現。使用監督學習的方法來為強化學習提供初始化參數，同時採用物件偵測技術來為強化學習網路在現實世界提供狀態資訊。此外，在代理人無法看到目標物時，使用A\*算法+DWA算法來引導代理人遵循預先設定的路徑行駛，直到能夠看到目標物為止，以實現圖3.1中的系統架構。這些方法的應用使得強化學習在實際應用中更加可靠和高效。

# 實驗結果與數據分析

在本章節中，將對本論文提出的自主導航架構進行詳細的實驗評估。首先，將在模擬環境中測試神經網路訓練的穩定性，並探討結合監督學習的訓練優化方法對強化學習效果的提升，同時比較強化學習算法、路徑規劃與導航以及本文提出的將兩者結合的方法之性能優劣。接下來，把訓練好的網路遷移至現實世界進行評估，通過實驗評估來驗證本論文所提出自主導航架構的有效性和實用性。

## 模擬世界中強化學習訓練結果

|  |  |
| --- | --- |
| (a) | |
| (b) | (c) |

圖6.1 強化學習地圖與獎勵

在4.1節中建立了三個用於訓練的模擬世界環境，並使用SAC算法進行訓練。這三個環境分別對應三種不同的獎勵方式。如圖6.1所示，當代理人在沒有障礙物的環境中時，它應該盡可能到達目標物並避免碰到四周的墻壁。因此，獎勵包括定位距離獎勵、角度獎勵和動作獎勵的總和，這是為了鼓勵代理人達成移動到目標物。在具有靜態障礙物的地圖(b)中，代理人需要注意躲避障礙物並盡快到達目標物。此時，提供一個引導代理人躲避障礙物的獎勵是非常必要的。因此，基於(a)獎勵，加入了一個障礙物獎勵。在具有動態障礙物的地圖(c)中，獎勵也是相同的。代理人需要注意躲避動態障礙物並盡快到達目標物，同時也需要考慮到障礙物的移動。用SAC算法進行訓練，還需要設置一些超參數，本文所設置的超參數，如表6.1所示。

表6.1 SAC超參數表

|  |  |
| --- | --- |
| 批次大小(batch size) | 512 |
| 決策網路學習率(Actor Learning Rate) |  |
| 評論網路1&2學習率(Critic Learning Rate) |  |
| 溫度係數學習率(- Learning Rate) |  |
| 溫度係數() |  |
| 目標熵() | -2 |
| 折扣率() | 0.99 |
| 軟更新率() |  |
| 經驗池大小 | 1 |
| 狀態空間(scan)輸入網路數量 | 24 |

### 無障礙物地圖訓練結果

本文使用SAC算法在無障礙物的地圖中進行強化學習訓練，以每回合獎勵加總作為評估標準，如圖6.2(a)所示。為了使得收斂曲線看起來平滑，收斂曲線的繪製過程中使用了移動平均(moving average)技巧，即對每一個時間點取其前面若干個時間點的平均值作為該時間點的平均回報，如圖6.2(b)所示。為了評估訓練的穩定性，本文進行了多次實驗，在線圖的上下方填充一個陰影區域，以表示訓練的穩定性。經過實驗結果的分析，證明了在無障礙物的地圖中，使用SAC算法可以有效地實現訓練收斂。

|  |  |
| --- | --- |
| (a) | (b) |

圖6.2 無障礙物地圖訓練結果

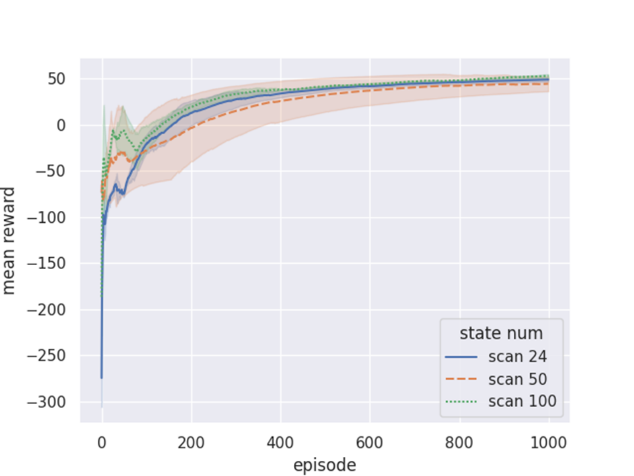


圖6.3 不同狀態數量測試圖

在表6.1中，使用了24個2D LiDAR所傳回的值作為神經網路的輸入，但這不一定是最佳的狀態輸入。因此，在同一個無障礙物的地圖中，分別使用了20、50和100個2D LiDAR所傳回的值作為神經網路的輸入，並測試了它們的性能。如圖6.3所示，在訓練方面，使用24個點的性能明顯比使用50個和100個點的性能略好一些。在該實驗環境下使用24個2D LiDAR傳回值的效果優於50個和100個2D LiDAR，即使用較少的輸入能夠達到更好的訓練效果。

在5.1小節中提到了一種結合強化學習和深度學習的方法。本文首先人工控制代理人在無障礙物地圖中行駛20次，收集了869筆狀態和動作數據。然後使用深度學習進行預訓練，訓練500次，學習率為。接下來，使用SAC算法進行進一步訓練。結果如圖6.4所示，其中(a)明顯顯示出結合深度學習的SAC算法在平均獎勵上遠優於普通的SAC算法，在(b)中也可以看到訓練後期更加穩定的收斂趨勢。

|  |  |
| --- | --- |
| (a) | (b) |

圖6.4 SAC結合深度學習結果圖

### 靜態障礙物與隨機障礙物地圖訓練結果

在靜態障礙物地圖中，同樣分為24、50和100個2D LiDAR所傳回的值作為神經網路的輸入。其結果如圖6.5所示，可以發現在此地圖中100個2D LiDAR所傳回的值作為神經網路的輸入相比24、50略勝一籌，選取收斂效果最穩定的100的回傳值進行深度學習預訓練，

在靜態障礙物地圖中，同樣將2D LiDAR傳回的值分為24、50和100個點作為神經網路的輸入。根據圖6.5(a)的結果，可以觀察到在這個地圖中，以24個2D LiDAR點作為輸入的神經網路稍微優於50和100個點的效果。因此，選擇使用24個點的輸入進行深度學習的預訓練，如圖6.5(b)，使其在收斂效果上更為穩定。

|  |  |
| --- | --- |
| Z:\109_林栩永\实验结果图\静态障碍物_scan.png  (a) | C:\Users\user\AppData\Local\Microsoft\Windows\INetCache\Content.Word\静态障碍物与deep.png  (b) |

圖6.5 靜態障礙物地圖訓練結果

而在動態障礙物地圖中，神經網路的輸入狀態在50個LiDAR資料點時基本不會收斂，而在24個資料點時雖然可以達到收斂，但波動較大，在100個資料點時表現最為穩定，如圖6.6(a)所示。可以發現，這種輸入狀態使整體性能表現更好。同樣地，在這種最佳狀態下，使用基於專家數據的深度學習預訓練，反而會出現更加不穩定的收斂情況，如圖6.6(b)所示。這可能是由於專家數據在特定的輸入狀態下對模型的預訓練造成了一定程度的干擾或不適應性。

|  |  |
| --- | --- |
| Z:\109_林栩永\实验结果图\动态.png  (a) | (b) |

圖6.6 動態障礙物地圖訓練結果

## 模擬世界神經網路結合路徑規劃與導航測試結果

本章節旨在比較多種不同的算法在不同地圖上的表現，評估指標為事先設計好的獎勵。比較的算法包括強化學習、傳統算法、使用深度學習預訓練的強化學習、深度學習以及本文所提出的神經網路結合路徑規劃與導航的方法。在之前的實驗中，使用了三張不同的地圖，分別是沒有障礙物、有靜態障礙物和有隨機移動障礙物的地圖(Map 1、Map 2、Map 3)。

表6.2展示了各個算法在各種地圖上的比較結果，其中SAC代表強化學習算法，Navigation代表傳統算法，Sac with DL代表使用深度學習預訓練的強化學習算法，DL代表深度學習算法，Navigation with SAC代表本文所提出的神經網路結合路徑規劃與導航的方法。為了模擬出真實世界定位的偏差，在Navigation算法及本文所提出Navigation with SAC在定位方面加入一定程度噪音，SAC算法選用上一小節訓練效果最好的網路，深度學習網路使用先前訓練好的預訓練網路。每種算法中代理人都需要抵達10個目標點，將每局的獎勵平均，以此評斷算法好壞。可以看出本文提出的方法在Map 2、Map 3優於其他4種，而傳統算法在無障礙物的地圖中優於其他算法。

表6.2 算法架構比對表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Method  Map | SAC | Navigation | Sac with DL | DL | Navigation with SAC (ours) |
| Map 1 | 105 | **126** | 110 | -9 | 119 |
| Map 2 | 40 | 105 | 42 | -53 | **117** |
| Map 3 | 114 | 94 | 13 | -12 | **119** |

## 現實世界物件偵測結果

在5.2小節中，分析了如何在現實世界中獲取跟模擬世界一樣的環境，本文將以不同的材料作為導航的目標物，如圖6.7所示，其中(a)為紅色木板，(b)為獎杯。代理人需要自主導航到目標物。

|  |  |
| --- | --- |
| (a) | (b) |

圖6.7 導航目標物示意圖

為了訓練YOLO-v7 tiny，首先控制代理人在實驗室環境中拍攝。為了更好地辨識目標物，以動作空間相同的速度邊行駛邊拍照。分別收集紅色木板和獎杯資料集：500張訓練集、50張驗證集、100張測試集。訓練300次後，結果如圖6.8所示，其中(a)為紅色木板物件偵測的損失，藍色為測試集的、黃色為驗證集的；(b)為獎杯資料集的損失。紅色木板的測試集精確度為94.87%，獎杯資料集的測試集平均精確度為98.11%。這表明YOLO-v7 tiny模型可以準確地檢測不同材料的導航目標物。在本實驗設備，每次檢測時間為0.0120秒為，這是非常快速的，足以支持在實時環境中應用。圖6.9，(a)、(b)展現了紅色木板的預測結果，其中藍色框為YOLO預測的框，藍色為標籤框。(c)、(d)展現了獎杯的預測結果。

|  |  |
| --- | --- |
| (a) | C:\Users\user\AppData\Local\Microsoft\Windows\INetCache\Content.Word\獎杯訓練.png  (b) |

圖6.8 YOLO-v7 tiny訓練結果

|  |  |
| --- | --- |
| (a) | (b) |
| C:\Users\user\AppData\Local\Microsoft\Windows\INetCache\Content.Word\example_saved97.png  (c) | (d) |

圖6.9 目標物辨識結果

## 現實世界自主導航架構測試結果

|  |  |
| --- | --- |
| (a) | C:\Users\user\Desktop\畢業門檻物件\論文初稿\圖\frame_81.jpg  (b) |
| C:\Users\user\AppData\Local\Microsoft\Windows\INetCache\Content.Word\20230508_190919.jpg  (c) | C:\Users\user\Desktop\畢業門檻物件\論文初稿\圖\20230508_191718.jpg  (d) |

圖6.10 傳統導航示意圖

在傳統算法中，會先進行地圖建構緊接著進行路徑規劃和導航，如圖6.10(a)、(b)所示。代理人希望最終能夠到達目標物(c)、(d)處，這種基於地圖的導航只能給出目標物大概的坐標點，這也是為什麼本文會在表6.2中的Navigation算法及所提出Navigation with SAC在定位方面加入一定程度噪音。圖6.11中(a)表示了代理人已經到達目標物位置，但在真實環境(b)中，代理人並沒有真正抵達目標物位置。

|  |  |
| --- | --- |
| C:\Users\user\AppData\Local\Microsoft\Windows\INetCache\Content.Word\frame_369.jpg  (a) | C:\Users\user\AppData\Local\Microsoft\Windows\INetCache\Content.Word\frame_398.jpg  (b) |

圖6.11 導航結束示意圖

當將以強化學習訓練好的神經網路應用於真實世界時，將其與YOLO-v7相結合，以實現對目標物的感知和辨識。結果顯示，當Limo檢測到目標物時，它能夠自主地朝著目標物移動，如圖6.12(a)、(b)所示。然而，當Limo無法看到目標物時，它會進行自我旋轉以搜索目標物，如圖6.12(c)、(d)所示，但在目標物較遠的時候無法很好的找到目標物。

|  |  |
| --- | --- |
| C:\Users\user\Desktop\畢業門檻物件\論文初稿\影片\rl\onrl\ub\ub_726.jpg  (a) | C:\Users\user\Desktop\畢業門檻物件\論文初稿\影片\rl\onrl\world\ub_726.jpg  (b) |
| C:\Users\user\Desktop\畢業門檻物件\論文初稿\影片\rl\onrl\ub\ub_258.jpg  (c) | C:\Users\user\Desktop\畢業門檻物件\論文初稿\影片\rl\onrl\world\ub_258.jpg  (d) |

圖6.12 強化學習導航示意圖

強化學習網路在遷移至現實世界，在偵測到目標物時並移動時，由於時間序列的影響。假設在時間時刻，神經網路根據環境狀態做出了動作決策和，此時Limo應該位於坐標。按照原始設計，時間間隔到的差異應該趨近於0。然而，在真實世界中，Limo實際上位於坐標，這導致時刻觀測到的環境狀態與理想情況有所偏差，進而使得Limo出現劇烈擺動的現象。圖6.15顯示了這一現象，其中(a)展示了神經網路在真實世界中行駛至目標物時所做出的多個值的高斯分佈。根據之前提到的內容，策略網路傾向於解析動作價值分佈，並選擇透過tanh函數對這些分佈的均值進行壓縮以獲得實際動作。然而，由於時間序列的影響，的分佈往往會出現在左右兩極值的情況，這就解釋了為何Limo會出現劇烈擺動的現象。

|  |  |
| --- | --- |
| (a) | (b) |

圖6.13 網路決策策略分佈及採樣動作圖

因此為了解決上訴問題，本文在最後通過tanh壓縮中改變tanh的斜率，在tanh的輸入乘上一個0到1之間的值，如圖6.14(a)所示。這樣使得壓縮後的動作分佈平滑從而解決劇烈擺動的問題，如圖6.14(b)所示。

|  |  |
| --- | --- |
| C:\Users\user\Desktop\畢業門檻物件\論文初稿\圖\tenh改.png  (a) | (b) |

圖6.14 改變tanh斜率示意圖、動作分佈圖

在模擬環境中使Limo在無障礙物的地圖中尋找三次目標物，軌跡圖如圖6.15所示，其中(a)採用完全不更改tanh斜率的軌跡，(b)為更改tanh斜率的軌跡，可以發現通過更改更改tanh斜率可以有效地解決Limo車頭劇烈擺動的問題。將兩條軌跡映射在地圖中，紅色軌跡為更改斜率，黑色為不更改，可以觀察到紅色軌跡更佳順滑流暢。

|  |  |
| --- | --- |
| (a) | (b) |
| (c) | |

圖6.15 Limo軌跡圖

為了解決上述兩種算法的缺陷，本文提出了一種結合兩者的架構。圖6.16顯示了實驗結果，其中(a)、(c)、(e)、(g)是以Limo的第一視角觀察到的結果，而(b)、(d)、(f)、(h)則是第三視角拍攝結果。

這種結合兩種算法的方法很好地解決了距離目標物較遠以及無法定位目標物的問題。該方法對於不同種類的目標物都表現良好，並且只需要訓練YOLO-v7模型來處理不同的目標物。

|  |  |
| --- | --- |
| C:\Users\user\AppData\Local\Microsoft\Windows\INetCache\Content.Word\c1_379.jpg  (a) | C:\Users\user\AppData\Local\Microsoft\Windows\INetCache\Content.Word\c1_868.jpg  (b) |
| C:\Users\user\AppData\Local\Microsoft\Windows\INetCache\Content.Word\d1_132.jpg  (c) | (d) |
| C:\Users\user\AppData\Local\Microsoft\Windows\INetCache\Content.Word\ub_3.jpg  (e) | C:\Users\user\AppData\Local\Microsoft\Windows\INetCache\Content.Word\world_3.jpg  (f) |
| C:\Users\user\AppData\Local\Microsoft\Windows\INetCache\Content.Word\ub_504.jpg  (g) | (h) |

圖6.16 融合傳統導航及強化學習導航示意圖

# 結論與未來展望

## 結論

本論文提出室內移動機器人的自主導航系統的架構，成功結合了端到端的自動駕駛和傳統導航演算法的優點，以克服傳統導航演算法過度依賴高精度定位的挑戰。採用深度強化學習方法設計了E2E模塊，並運用2D-LiDAR和攝影機結合物件偵測技術YOLO-v7 tiny，實現自主導航系統。透過行為複製穩定訓練，成功將深度強化學習網路遷移至現實世界，提升了系統在真實環境中的適應性和穩定性。同時，運用A\*演算法進行全域最優路徑規劃，並結合DWA演算法完成局部路徑規劃，有效解決了強化學習神經網路在無法偵測目標物時難以做出優良決策的問題。

在神經網路的訓練方面，首先在模擬環境中建立了一個允許代理人不斷試錯的環境，以培養其學習和適應能力。為了提供代理人預訓練的初始參數，使用人為控制的方式，讓代理人在模擬環境中進行數次遊玩。這些遊玩過程中收集的專家數據將作為代理人神經網路的初始參數，以幫助代理人快速開始學習。為了進一步優化代理人的學習效果，採用了SAC演算法進行與環境的互動學習。SAC是一種基於深度強化學習的演算法，它結合了最大熵理論和軟更新策略，能夠有效地提升代理人的學習效能和探索能力。代理人通過與環境的互動，不斷更新其神經網路模型的參數，以最大化預期的累積回報。透過這樣的訓練方法，能夠使代理人在模擬環境中不斷學習、試錯和優化自身的行為策略。最終，代理人能夠在僅依靠2D-LiDAR和攝影機就能夠決策動作。

為有效將模型遷移至現實世界，在真實環境的收集目標物影像進行標註，訓練YOLO-v7 tiny模型，為強化學習訓練的神經網路提供狀態。而目標物在較遠時，採用傳統演算法A\*演算法進行全域最優路徑規劃，並結合DWA演算法完成局部路徑規劃，完成本文所提出的系統。

## 未來展望

本論文提出來一套作用於室內移動機器人的自主導航系統的架構，但在未來仍存在許多潛力和發展方向，本小節將分為改進方向、發展方向進行討論。

發展方向:

* 增加YOLO-v7目標物件樣本使物件偵測更加穩定：在未來的研究中，可以進一步增加YOLO-v7目標物件的樣本數量，包括各種不同形狀、大小和角度的物件，以使物件偵測更加準確和穩定。通過收集更多的樣本數據，並使用先進的數據增強技術，可以提高系統對於各種場景和物件的識別能力。
* 在強化學習網路中加入LSTM等模塊加強時間序之間的聯繫：為了更好地處理時間序列之間的關聯性，未來的研究可以將LSTM(長短期記憶)等模塊引入強化學習網路中。LSTM具有記憶單元，可以捕捉到時間序列數據中的長期依賴關係，這有助於機器人更好地理解過去的狀態和行為對當前和未來決策的影響。透過引入LSTM等模塊，系統的學習能力和決策能力可以得到進一步的提升。

發展方向:

* 向真實自動駕駛邁進：室內移動機器人的自主導航系統有望向真實自動駕駛領域發展。未來的研究可以將系統擴展到更複雜和多樣化的環境中，如辦公樓、商場或醫院等。通過在這些場景中實現自主導航，機器人可以提供更廣泛的服務，如送貨、尋找特定物品等，從而實現真正的自動化。
* 增加視覺感測範圍或融合不同類型的感測器：為了更好地感知和理解環境，未來的研究可以考慮增加機器人的視覺感測範圍，或者融合多種不同類型的感測器，如雷射雷達、深度相機、超聲波等。這樣的感測器組合可以提供更全面和精確的環境感知能力，使機器人能夠更好地避免障礙物、感知目標物體並做出更優的導航決策。

# 參考文獻

1. 經濟部中小企業處(2021)，“2021年中小企業白皮書”，臺北：經濟部中小企業處。
2. M. Shneier, R. Bostelman, “Literature Review of Mobile Robots for Manufacturing”, NIST Interagency/Internal Report (NISTIR), 2015.
3. C. Cadena et al., “Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age”, IEEE Transactions on Robotics, vol. 32, no. 6, pp. 1309-1332, 2016.
4. A. Tampuu, T. Matiisen, M. Semikin, D. Fishman and N. Muhammad, “A Survey of End-to-End Driving: Architectures and Training Methods”, IEEE Transactions on Neural Networks and Learning Systems, vol. 33, no. 4, pp. 1364-1384, 2022.
5. V. Mnih et al., “Playing Atari with Deep Reinforcement Learning” ArXiv abs/1312.5602, 2013
6. D. Silver et al., “Mastering the game of Go with deep neural networks and tree search” Nature 529, pp. 484–489, 2016.
7. Google AlphaGo 的勝利：不是電腦打敗人類，而是人類打敗人類: https://technews.tw/2016/01/30/google-alphago-2/ (Accessed October, 2023)
8. Y. Bai et al., “Training a Helpful and Harmless Assistant with Reinforcement Learning from Human Feedback”, ArXiv, abs/2204.05862, 2022
9. S. H. Chan, P. T. Wu and L. -C. Fu, “Robust 2D Indoor Localization Through Laser SLAM and Visual SLAM Fusion” 2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Miyazaki, Japan, 2018, pp. 1263-1268 2018
10. G. P. Hao and Z. Q. Deng, “Path Planning of Scanning Robot for Tubular Joint,” 2006 6th World Congress on Intelligent Control and Automation, Dalian, China, 2006, pp. 8178-8182, 2006
11. 陳世安，“結合SLAM與路徑規劃技術於肢體殘障者行動輔具系統開發” 國立勤益科技大學電機工程系碩士論文，2020.
12. 黃偉哲，“適用於多人動態空間之視覺SLAM系統與人員智慧追蹤功能開發” 國立勤益科技大學電機工程系碩士論文，2022.
13. 羅一軒，“以2D-LiDAR與深度相機異質感測融合應用於機器人自主探索” 國立勤益科技大學電機工程系碩士論文，2022.
14. H. V. Hasselt et al. “Deep Reinforcement Learning with Double Q-Learning.” ArXiv abs/1509.06461, 2015.
15. D. Silver et al. “Deterministic policy gradient algorithms.” In ICML, 2014.
16. T. Lillicrap et al. “Continuous control with deep reinforcement learning” CoRR abs/1509.02971 2015
17. V. Mnih et al. “Asynchronous Methods for Deep Reinforcement Learning” ArXiv, abs/1602.01783. 2016
18. T. Haarnoja et al. “Reinforcement Learning with Deep Energy-Based Policies” International Conference on Machine Learning 2017.
19. T. Haarnoja et al. “Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor” International Conference on Machine Learning 2018.
20. D. A. Pomerleau, “Alvinn: An autonomous land vehicle in a neural network”, Proc. Adv. Neural Inf. Process. Syst., pp. 305-313, 1989.
21. M. Bojarski et al. “End to End Learning for Self-Driving Cars.” ArXiv abs/1604.07316 2016
22. Y. April et al. “Deep Reinforcement Learning for Simulated Autonomous Vehicle Control” 2016.
23. Z. Huang, J. Zhang, R. Tian and Y. Zhang, “End-to-End Autonomous Driving Decision Based on Deep Reinforcement Learning” International Conference on Control, Automation and Robotics (ICCAR), pp. 658-662, 2019.
24. Gazebo, Retrieved March 18, 2023, from <https://gazebosim.org/home>.
25. S. Chen, M. Wang, W. Song, Y. Yang, Y. Li and M. Fu, “Stabilization Approaches for Reinforcement Learning-Based End-to-End Autonomous Driving” in IEEE Transactions on Vehicular Technology, vol. 69, no. 5, pp. 4740-4750, 2020.
26. J. d. Jesus et al. “Soft Actor-Critic for Navigation of Mobile Robots” J Intell Robot Syst 102, 31, 2021.
27. D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors,” Nature, vol. 323, no. 6088, pp. 533-536, Oct. 1986.
28. V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in Proceedings of the 27th International Conference on Machine Learning (ICML-10), 2010, pp. 807-814.
29. Y. LeCun, L. Bottou, Y. Bengio, & P. Haffner, (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278-2324.
30. R. S. Sutton and A. G. Barto, “Reinforcement learning: An introduction”. MIT press, 2018.
31. M. Watkins and P. Dayan, “Q-Learning”. Machine Learning, vol. 8, no. 3-4, pp. 279-292, 1992.
32. R. J. Williams, “Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning” Machine Learning, vol. 8, no. 3-4, pp. 229-256, 1992.
33. K. Konda and J. N. Tsitsiklis, “Actor-critic algorithms," Advances in Neural Information Processing Systems” vol. 14, pp. 1008-1014, 2002.
34. R. S. Sutton, “Learning to predict by the methods of temporal differences. Machine learning” 3(1), 9-44, 1988.
35. T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, “Continuous control with deep reinforcement learning” Proceedings of the International Conference on Learning Representations (ICLR), 2019.
36. C. E. Shannon, “A mathematical theory of communication.” The Bell System Technical Journal, 27(3), 379-423, 1948.
37. S. Kullback, & R. A. Leibler, “On information and sufficiency” The annals of mathematical statistics, 22(1), 79-86, (1951).
38. Agilex. (n.d.). Agilex Robotics. Retrieved from <https://www.agilex.ai/?lang=en>.
39. Agilex Robotics. (n.d.). Limo: A ROS-based intelligent mobile robot. GitHub. Retrieved from https://github.com/agilexrobotics/Limo-doc.
40. NVIDIA Jetson Nano Developer Kit. (2019). Retrieved from <https://developer.nvidia.com/embedded/jetson-nano-developer-kit>.
41. Orbbec. (n.d.). Orbbec Dabai. Retrieved 2023, from <https://www.orbbec.com/en/products/dabai/>.
42. EAI X2L 2D LiDAR. (n.d.). EAI. Retrieved April 6, 2023, from <https://www.eaibot.com/x2l.html>.
43. NVIDIA Corporation. (n.d.). GeForce GTX 1060. NVIDIA. Retrieved April 6, 2023, from <https://www.nvidia.com/zh-tw/geforce/graphics-cards/geforce-gtx-1060/specifications/>.
44. R. Collobert, K. Kavukcuoglu, and C. Farabet, “Torch7: A Matlab-like environment for machine learning” in BigLearn, NIPS Workshop, vol. 4, 2011.
45. D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” arXiv preprint arXiv:1412.6980, 2014.
46. M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, “ROS: An open-source Robot Operating System” in ICRA Workshop on Open Source Software, 2009.
47. D. E. Knuth, “The Art of Computer Programming, Volume 1, Fundamental Algorithms” Addison-Wesley, 1968.
48. D. P. Kingma and M. Welling, “Auto-encoding variational bayes” arXiv preprint arXiv:1312.6114, 2014.
49. A. Papoulis & S. U. Pillai (2002). Probability, random variables, and stochastic processes. McGraw-Hill.
50. C. Y. Wang, A. Bochkovskiy, H. Y. M. Liao, “YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors”, arXiv, vol. abs/2207.02696, 2022.
51. S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," arXiv preprint arXiv:1502.03167, 2015.
52. A. L. Maas, A. Y. Hannun, and A. Y. Ng, “Rectifier nonlinearities improve neural network acoustic models,” in Proc. icml, vol. 30, no. 1, 2013, p. 3.
53. Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, 1998.
54. K. He, X. Zhang, S. Ren, and J. Sun, “Spatial pyramid pooling in deep convolutional networks for visual recognition.” ECCV 2014.
55. R. Zhang, X. Wang, & L. Li, “CSPNet: A New Backbone That Can Enhance Learning Capability of CNN.” IEEE Transactions on Circuits and Systems for Video Technology, 31(6), 2386-2397, 2020.
56. K. He, X. Zhang, S.Ren, & J. Sun, “Deep residual learning for image recognition.” In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778), 2016.
57. A. Neubeck and L. Van Gool, “Efficient non-maximum suppression.” in Proceedings of the 18th International Conference on Pattern Recognition, vol. 3, pp. 850-855, 2006.
58. P. E. Hart, N. J. Nilsson, B. Raphael, “A Formal Basis for the Heuristic Determination of Minimum Cost Paths”. IEEE Transactions on Systems Science and Cybernetics. 4 (2): 100–107, 1968.
59. D. Fox, W. Burgard, & S. Thrun, “The dynamic window approach to collision avoidance.” IEEE Robotics & Automation Magazine, vol. 4, no. 1, pp. 23-33, 1997.
60. G. Grisetti, C. Stachniss, and W. Burgard, “Improved Techniques for Grid Mapping With Rao-Blackwellized Particle Filters.” IEEE Transactions on Robotics, vol. 23, no. 1, pp. 34-46, 2007.
61. “navigation - ROS Wiki.”2021. ROS Wiki. Retrieved September 20, 2021, <https://wiki.ros.org/navigation#References>.
62. “AMCL - ROS Wiki”2021. ROS Wiki. Retrieved May 10, 2023, from https://wiki.ros.org/amcl#References.