國立清華大學 電機工程學系

實作專題研究成果報告

Implementing remote connectivity and path optimization for robot navigation through ROS
透過ROS實現機器人導航的遠端連線與路徑優化

專題領域： 系統組

組　　別： A318

指導教授： 馬席彬

組員姓名： 鄭筑元、陳姿穎、陳昱淳

研究期間： 2022年 7月30日至 2023年 5月 8日止，共 10個月

- Abstract

Given the rise of food delivery platforms, the use of delivery services has become increasingly common. However, behind the convenience and immediacy of these services, there are also a multitude of social issues, such as the stress and danger faced by delivery drivers and concerns over food hygiene. In the aftermath of the pandemic, we can be certain that smart logistics systems will be the future trend. Therefore, in this project, based on the ROS system, we integrate robots and the internet to implement a scenario where robots are used in the food delivery system, including real-time path planning and optimization, and application of visual recognition to enhance interaction.

We set up an order website and store the data in a MySQL database. When new data is received, the system automatically sends the order information to the robot car equipped with Lidar and RGB cameras, and installs Robot Operating System (ROS) on its Linux system. The navigation stack built into ROS can create maps and perform navigation, and we publish target points using our own written node, and solve the TSPPC with a simple algorithm to optimize the path.

After the robot car arrives at the destination, visual recognition is used through the camera on the car to recognize the order number using our trained digit recognition model. This determines whether the restaurant operator has put the food on the robot or if the customer has taken it, and checks whether the order number matches the food, ensuring the accuracy of the delivered food.

■ 摘要

鑑於外送平台的興起，外送服務的使用也越來越普遍，但在帶來便利性與即時性的服務背後，與此相關的社會問題也層出不窮，例如外送員遭受的壓力和危險以及餐點的衛生問題。而在疫情的衝擊後我們更能確信，智慧物流系統將會是未來的趨勢。因此，本次專題我們基於 ROS 系統，透過整合機器人與網路來實現機器人運用於送餐系統的情境，並囊括實時路徑規劃與優化、結合視覺辨識的應用提高互動性。

我們透過架設訂單網站，並將資料存取至 MySQL 資料庫中，當接收到新資料的傳入時，系統會自動將該訂單資料發送給搭載了 Lidar、RGB 鏡頭的機器人小車，並透過其 linux 系統安裝 Robot Operating System (ROS)，而 ROS 內建的 navigation stack 能建立地圖和進行導航，再用自己編寫的 node 發佈目標點，並透過簡單的演算法解決 TSPPC 以用來優化路徑。

小車抵達目的地後，會透過小車上的鏡頭來進行視覺辨識，透過我們訓練的數字辨識模型來辨識訂單數字，藉以判斷餐點是否被餐廳業者放上機器人或是被訂餐者取走，並檢查其訂單編號是否和餐點相符，確保遞送餐點的正確性。

1. 前言

1-1.研究目的和背景簡介

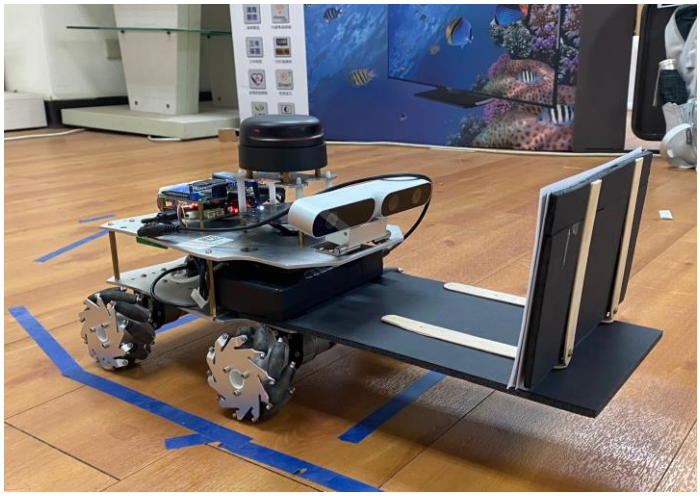
近年來，外送平台的興起蔚為風潮，外送平台的使用也越來越普遍，這三年以來爆發的疫情更凸顯了這項服務的重要性，同時宣告了在這個極其依賴網路而淡薄了人與人之間連結的時代，物流服務將會是不可或缺的一塊版圖。但在便利與即時的背後，與此相關的社會問題也層出不窮：龐大基數導致素質的參差不齊、危險的工作環境和缺乏法律保障的合約等。除此之外，從疫情時期開始，人們也越來越重視食品衛生，減少人與人之間的接觸能使食物更安全。因此我們希望能夠建構出一台可以自動送餐的機器人，透過網路進行下單，再利用演算法做路徑規劃，讓送餐系統可以更有效率的完成工作，並相信自動送餐系統將成為現代城市生活中的一個重要一角。

1-2.實作成果

本次專題中，我們先用 SLAM Gmapping 建立地圖，再透過 navigation stack 協助避障和實現導航，並透過自己編寫的 node 來作到多個目的地的管理和排程來優化多點導航過程中所遇到的 TSPPC。完成基本使用後，我們使用網路上公開的 mnist 資料庫並利用 Keras 訓練模型，再將訓練好的模型和 tensorflow 結合，讓小車能透過 OpenCV 分析畫面後作到數字辨識的功能。將數字辨識結合導航相關功能，讓小車能在互動上有更多的彈性。最後，我們利用 HTML 和 Node.js 來架設網站，並使用 MySQL 來儲存和傳送訂單資料，實現遠端連線機器人自動化的功能。

2. 研究方法

2-1. 小車相關配備



圖一、小車配備

本次專題使用的小車如圖所示，使用了包含 Lidar、RGB 鏡頭以及四個萬向輪組成，透過系統內的 ubuntu 18.0來安裝 ROS melodic 來完成本專題。

2-2.ROS 系統簡介

我們的小車是透過 ROS 系統(Robot Operating System)來進行操作[1]，是 Linux 發行版 Ubuntu 下的一個用來開發機器人的 Middleware。ROS 系統有以下幾個優點。

1. ROS 系統將各個應用分解為獨立的 Package，因此在擴建和修改功能時較於便利和快速，也不會造成其他 Package 的錯誤。
2. ROS 系統同時支援 Python 和 C++兩種不同的語言，因此可以根據不同的需求來使用不同的語言進行編程。
3. ROS 系統沒有硬件要求，因此可以運用在各種不同的硬件平台上[2]。

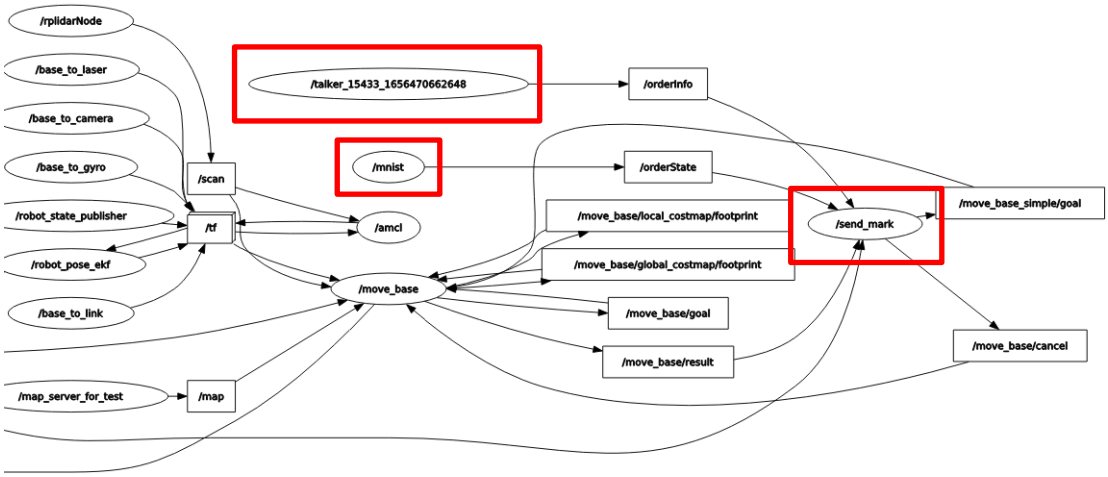
ROS 之間的溝通有多種通訊方式，以下將介紹本次專題主要使用的通訊方法及運作方式。

1. ROS topics：此通訊方式是透過 Node (節點)來運作，要傳輸的訊息以 Topic 的形式進行傳輸。一個 Package 中可以包含多個 Node，各個 Node 中可以包含多個 Publisher 或 Subscriber。Publisher 會負責發送 topic，而 Subscriber 會負責接收它訂閱的 topic。
2. ROS services：此通訊方法是雙向溝通的方式。由 server 和 client 組成。client 會發送 request 給 server，server 再回傳對應的 response 給 client。此通訊方法的缺點是即時性較不好，且只能進行一對一的訊息傳輸。

2-3. 系統設計

我們專題的系統設計流程如圖三所示。使用者會先從網站下訂單，下單之後頁面會切換到訂單完成的畫面，同時將訂單資料存進資料庫並傳給小車。當新資料加入資料庫時，會從電腦發送以下訂單訊息給小車：餐廳、住址以及餐點完成時間，小車接收到後會將這些資訊透過 topic /orderInfo 發佈，負責目的地管理與導航相關的 node send_mark 會訂閱這個 topic，並藉由貪婪演算法對後續導航的目標點做出排序與優化。小車抵達目的地後會先停下，藉由 node mnist 的數字辨識偵測使用者是否把某個訂單放上或取下來並用 topic /orderState 回報給 send_mark，同時確保餐點的訂單編號與小車目前的排程相符。

[1]我們使用的 Ubuntu 版本是 Ubuntu18、使用的 ROS 版本是 Melodic
[2]這次專題中，我們使用的是 Raspberry Pi



圖二、rqt 圖

2-4.網站架設和資料庫建立

我們利用 HTML 和 CSS 製作簡易的前端網頁，如圖五所示，當使用者送出訂單後，介面會切換至訂單完成的畫面，並顯示使用者輸入的內容。

基於上述方法所完成的網頁僅能在本地伺服器使用，或在同網域下才能連線到此伺服器，因此透過 ngrok 來讓外部網路可以連線到本地伺服器。ngrok 會在本地伺服器和 ngrok 服務器之間建立一個基於 TLS 加密協議的通道，從而將本地服務器暴露到公開互聯網中。每當有一個請求到達時，它都會被轉發到 ngrok 服務器，通過一個隨機生成的子域名和公共端口號將其映射到用戶端的本地端口，再轉發回用戶端。

後端的部分我們使用 Node.js 來完成，它的特點在於它是事件驅動、非阻塞 I/O 模型。這種設計模式使得 Node.js 能夠以非阻塞的方式處理大量的並發請求，並且能夠快速地響應事件。因為訂單平台通常需要處理大量的並發請求，所以選用 Node.js 來完成後端。

為了減少小車的運算負擔，我們將網站架設和資料庫的部分放在自己的電腦上，因此需要再將資料庫的內容傳給小車，才能進行後續的導航。我們使用 Python 的 Socket 模組來完成此功能。

本地電腦的 Client 會連線到 MySQL 資料庫中，當有資料更新時，會將該資料以 Socket 發送給小車上的 Server，為了減少發送的信息量及因為小車導航功能包中希望接收目標點的 topic 是以 Int16MultiArray 的資料格式來進行傳輸，因此在用 Socket 傳輸前，會先將資料作數據化的轉換[3]。當 Server

[3] 本次專題中，設定店家數量有3家，使用者地點有3處不同的地方。

接收到新資料後，會將該資料以 topic 的形式發出，並等待導航的節點來訂閱。圖九為執行的結果，當 topic 發出時，將發出的內容輸出[4]。

2-5.建立導航功能

(a) Navigation stack 簡介

Navigation stack 是 ROS 中用於自主導航的功能套件，提供了一套完整的導航系統，負責了包括、定位、地圖建構、路徑規劃、避障等功能，讓機器人能在已知或未知環境中自主導航，並用 node "move_base"來與整個系統互動。

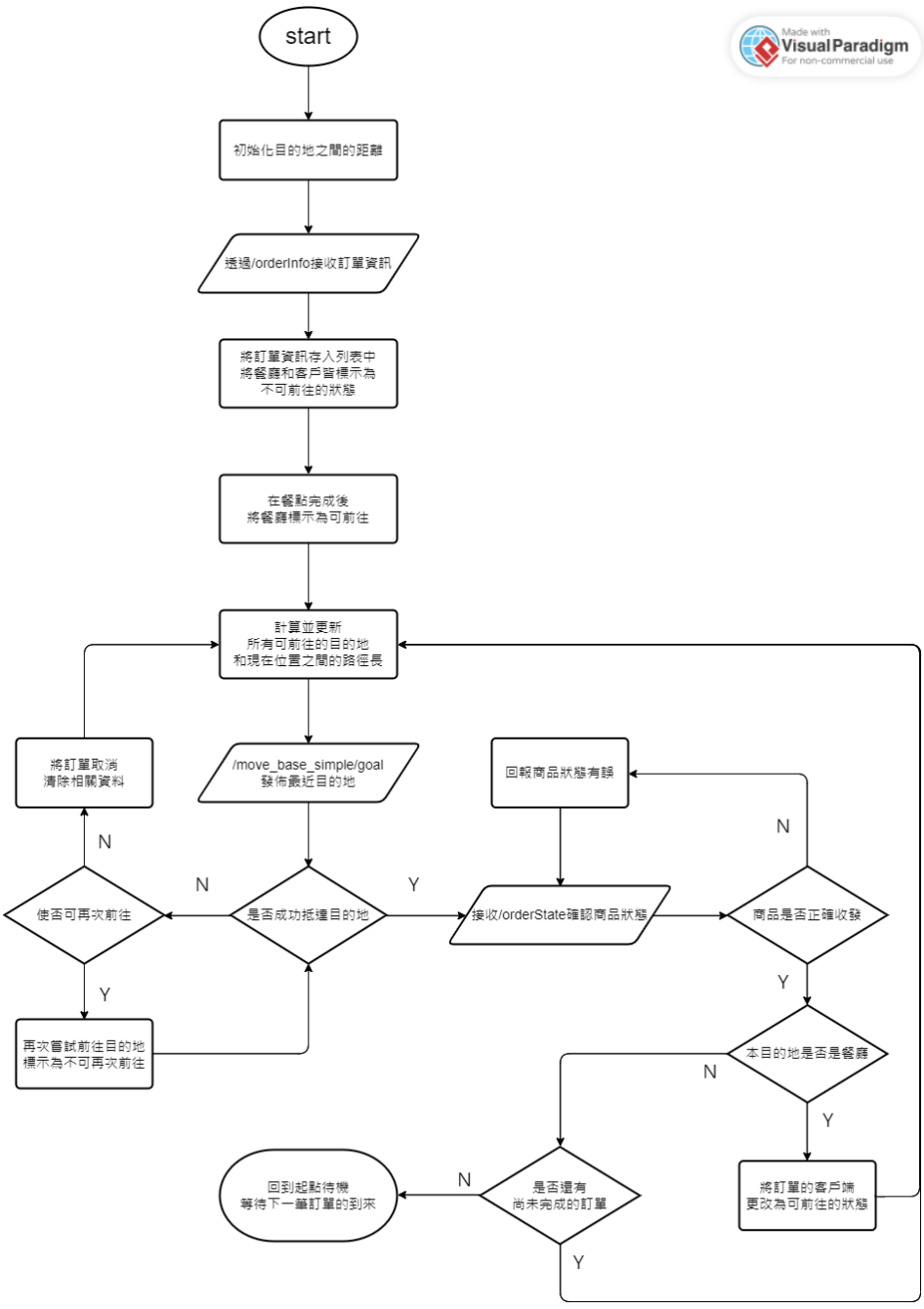
本次專題中，我們使用 SLAM 先建立一張地圖，讓小車在已知環境中行動，隨後透過 global planner 規畫路徑，最後利用 lidar 和 local planner 做到小車避障並確保小車能安全到達終點。為了符合題目需求，我們針對其中一些功能做了擴展，以下為詳細的功能說明和流程圖。

(b) 導航功能說明

在 navigation stack 中，小車能透過/move_base_simple/goal 這個 topic 發佈目的地座標來讓小車前往指定地點，而在本次專題中，我們需要小車能夠做到對多個目標點的路徑比較與排序來優化送餐過程所需時間，因此在收到新的訂單資訊時，我們先透過 send_mark 這一 node 整理目的地，再由 send_mark 決定要透過/move_base_simple/goal 發佈的位置和時機。

首先讓 send_mark 從 topic /orderInfo 中取得訂單資訊，包括[餐廳代號、客戶代號、餐點完成所需時間]，在 node 中分別轉換為兩個目的地，每個目的地都會記錄以下幾項資料：1. 座標、2. 訂單編號、3. 餐廳 or 客戶、4. 能否前往、5. 餐廳或客戶的代號、6. 與小車當前距離。

[4] 發出的 Int16MultiArray 中包含3個整數，第一位是店家代號，第二位是使用者地點的代號，第三位為等待的時間。



圖二、send_mark 流程圖

同一筆訂單的餐廳和客戶會共享一個訂單編號，而目的地為餐廳或客戶則分別需要滿足不同的條件才能前往：餐廳需要等待餐點完成的倒數計時結束後才能前往，客戶需要等餐點上車後才會改為可前往目的地。

在決定了要前往的目的地後，send_mark 透過/move_base_simple/goal 發佈目的地，交給 move_base 來導航到目的地。在過程中如果有新的可前往目的地出現的話，send_mark 會取消當前目的地的導航，更新各可前往目的地與小車當前的距離，並依此規劃出新的優化路徑，在重新透過/move_base_simple/goal 發佈目的地。在接下來的部分將會解釋小車如何優化路徑。

```
[INFO] [1656472372.505071]: the path length have been initialize.
[INFO] [1656472411.718597]: A new order, No.1.
[INFO] [1656472411.724307]: Going to the restaurant of order 1.
[INFO] [1656472413.724928]: A new order, No.2.
[INFO] [1656472414.936192]: Goal changed! A batter route has found.
[INFO] [1656472414.940350]: Going to the restaurant of order 2.
[INFO] [1656472415.731279]: A new order, No.3.
[INFO] [1656472417.959958]: Goal changed! A batter route has found.
[INFO] [1656472417.964087]: Going to the restaurant of order 3.
[ INFO] [1656472437.561150071]: GOAL Reached!
[INFO] [1656472437.562352]: Have reach the restaurant of order 3, waiting for
the loading.
```

圖三、send_mark 在短時間收到三筆訂單並更改了路徑

(c) 路線最佳化問題

在最佳化問題中，這種經過每一個頂點最後回到起點並求最短路徑的問題被稱為 Travelling Salesman Problem (TSP)，在本次專題之中，餐廳和客戶之間還存在「先去餐廳取餐，才能送餐到客戶端」的前置制約條件，因此這是一個 Travelling Salesman Problem with Precedence Constraints (TSPPC)。

我們選擇使用最簡單但絕對不會錯的方式來實現：前往當前可以前往的目的地之中距離小車最近的一個點，並在一開始只比較餐廳和小車間的距離，等到取到餐點後再將客戶納入考量，每到一個目的地或有新的目的地加入時就更新距離並重新比較。

這種利用貪婪演算法的方式雖然能取得一個快速的解，但因為沒有考慮每一個區域最佳解對於全域最佳解的影響，因此也與正確性相差一段距離，這也是為什麼在這裡我們選擇使用「優化」而非「最佳化」一詞。

(d) 導航路徑長計算的實現

為了能夠比較目的地和小車間的路徑長，我們透過 navigation stack 中原本就有的功能包 global planner 來規劃並計算路徑長。

Global planner 能透過已經建立的地圖利用 Dijkstra's Algorithm 來規劃兩點之間的距離，平常透過/move_base_simple/goal 來接收目的地，並讀取小車在地圖中的位置來規劃出路徑。經由 request 將想要計算距離的兩個點傳給 global_planner 的 service /move_base/make_plan，得到回傳的 response，而其內容為一個 PoseStamped()的陣列，可以想成由好幾個小箭頭組成一條路徑，因此透過迴圈計算出箭頭長度總和就能得出路徑長。

(e) 距離計算功能優化

為了降低小車的負擔，我們利用動態規劃來降低路徑的計算量。在 send_mark 啟動時，我們先計算出所有記載的目的地之間的距離，並儲存起來，當後續小車到達目的地要規劃下一個目標時，就不用再計算這個目的地到各點之間的距離為

何而是直接從已經儲存的資料直接存取。

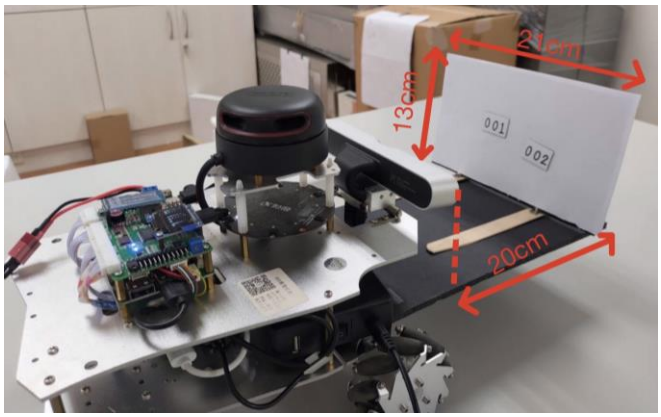
如此一來，就只有當小車在移動途中接收到新的目的地而要規劃時才需要計算路徑長，不過因為 move_base 在移動途中是不提供 global _planner 規劃路徑的功能，這也是為什麼小車必須要取消當前目的地的導航才更新各點之間的距離。

(f) 餐點收發確認

在到達目的地之後，send_mark 會等待 topic /orderState 發佈的訊息來確認相應訂單編號的餐點是否上車或是已經取餐，訊息內容包含一組訂單編號 xxx 和1/0，來說明訂單編號為 xxx 的餐點是上車還是取餐，等到餐點確認收發正確之後才會前往下一個目的地，如果餐點的收發跟預期的情況不同，會回報錯誤並等到餐點狀態正確才會出發。在下一個小節，將會說明小車是如何透過視覺辨識來模擬餐點收發的情境。

2-6.建立餐點辨識功能

由於小車無法裝載實際的餐點，因此我們利用小車是否有辨識到訂單編號的字卡來模擬餐點是否放置在小車上的情境。



圖四、小車與屏幕尺寸

(a) 視覺辨識模型訓練

視覺辨識的部分是選用在機器學習領域中被廣泛使用的 MNIST 手寫數字資料集作為訓練數據，並使用 AutoKeras 訓練模型。選用 AutoKeras 的原因除了其有撰寫較簡便的特性外，它還可以透過調整神經網路架構來改進表現。

(b) 數字辨識方法

使用 OpenCV 來捕捉鏡頭的畫面並進行處理，將其轉為灰階後再二值化，然後透過 cv2.morphologyEx( )過濾 noise，最後用 cv2.findContours( )出各個可能存在數字的輪廓，對輪廓加以檢測是否符合預定的大小及位置條件。接著，我們會將輪廓擴展成方形並調整大小用於後續的預測，並讓每個數字皆有它固定的座標。

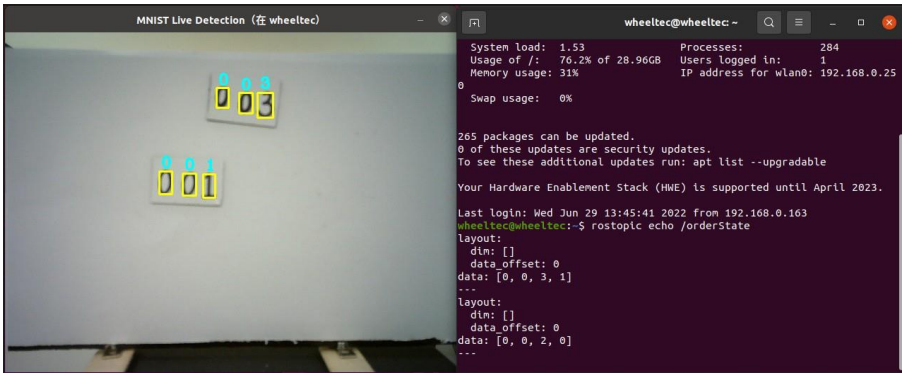
待標示完這些 contour 後，用訓練的模型辨識所有 contour 中的數字。

(c) 訂單編號辨識

到目前為止，小車只能辨識到畫面上所有的數字，無法找出新出現的數字，更遑論找出有排列順序的訂單編號，加上當有其他東西進入畫面時，模型可能會將其誤判為數字，因此我們對讀取到的畫面做近一步分析。

我們透過 Time.sleep(0.5)控制每秒擷取兩幀畫面，並記錄該畫面中所有數字的數值及座標，當連續兩秒的畫面的數值和座標都相同時，我們就稱這樣的畫面為一個「穩定狀態」。透過比較兩個穩定狀態之間的變化，小車就可以知道消失或新增的數字是哪幾個，又因為訂單編號一定是由左到右排成一組數字，所以依照 x 座標做排序，就可以知道新增或消失的訂單編號是多少。而辨識到字卡就相當於餐點是否有在小車上，也就是說當辨識到屏幕上有新增的訂單編號即表示店家已將餐點放上小車；同理，若訂單編號從屏幕上移除，則代表餐點已被客戶取走。舉例來說，當畫面上新增0、0、3這三個數後，會發出[0,0,3,1]，代表訂單編號003上車；0、0、2從畫面上消失後發佈[0,0,2,0]代表訂單編號002已取餐。為了與其他小車上的 ROS 功能做結合，這裡會建立 node mnist，透過將這些訊息發佈 topic /orderState 使 send_mark 得以確認當前的小車上的餐點狀態。

為了提高系統對於取餐與送餐的準確率，我們針對以下兩部分進行錯誤偵測的處理。首先，由於我們設定的訂單編號皆為三位數，因此若畫面中的數字數量不為三的倍數，表示可能出現數字未被偵測到或是部分字卡被遮擋的情形。而若兩次穩定狀態的數字數量相同，但在相同位置卻有不同的數字，則表示有字卡被完全覆蓋。上述兩種狀況皆會透過/mnist 發佈[-1,-1,-1,2]，用以提示使用者餐點有誤需要人為排錯，系統會在排錯後繼續正常運作。



圖五、視覺辨識展示

(d)視覺辨識對導航表現的影響

在模擬的過程中，我們發現導航時有使用視覺辨識的同時，似乎會對導航的表現有影響，因此設計了簡單的實驗來驗證是否視覺辨識對導航的表現帶來影

響。

首先讓小車在與所儲存的地圖一致的場景做導航並記錄完成秒數，分別進行四次，實驗結果如下表紀錄：

無障礙物	第一次	第二次	第三次	第四次
有視覺辨識	26.9	25.6	26.9	25.0
無視覺辨識	26.2	25.5	28.4	25.3

從記錄中可以發現，在地圖簡單且沒有太多外來干擾的時候，兩者的表現並不會有明顯的差距。

接下來我們在場景中置入障礙物，讓小車在比地圖所記錄的還要更加複雜的場景中嘗試導航並記錄秒數，且特意將障礙物放置在小車和終點之間提高計算難度，實驗結果如附表所示：

有障礙物	第一次	第二次	第三次	第四次
有視覺辨識	Failed	25.2	60.2	failed
無視覺辨識	22.6	41.0	25.3	29.7

這裡的 Failed 指的是小車判斷無法到達目的地兩次，因為在設計中，小車若兩次判斷無法到達目的地就會取消該目的地的導航。整體的結果中可以窺見小車在開啟視覺辨識後在複雜場景中的導航能力大幅度下降。

3. 結論

1. 導航優先級的權重目前只以地圖上的最短路徑長做為標準，在區域發生的變化導致當前規劃效果不如預期的情況沒有辦法及時改善導航規劃。
2. 視覺辨識會影響導航能力，進一步導致預期之外的延遲與錯誤。或許能選擇避免導航全程都開啟視覺辨識以節省算力。
3. 目前只利用網站做出單方面的訊息傳送，如果可能希望能做到小車和網站間的雙向溝通。

4. 參考文獻

[1] ROS Wiki https://www.ros.org/

5. 專題製作規劃和團隊合作方式

5-1.專題製作規劃

我們規劃7~9月間進行基礎知識學習，這段期間中，我們學習了 ROS 系統的運作方式、小車的導航、定位、建圖方式等等。10月開始我們討論了專題的方向，並進行分工。

教授規劃我們每週報告一次進度，報告時會給予我們建議的方向，並和學長一起針對我們提出的問題提出可能的解決方向。

5-2.團隊合作方式

我們將工作分成網頁和資料庫、導航規劃和數字辨識三大部分，一人負責一部分，最後再將所有功能整合在一起。每個禮拜我們會集合討論一至兩次，討論目前的進度並互相協助解決遇到的問題，其餘時間則各自完成該週的進度。

6. 實作專題心得

陳姿穎

經過這一年的實作專題，我學到了許多東西，不但學到了許多 ROS 系統相關的知識，本身對硬體設備不甚瞭解的我也開始有了基礎的認知。在實作專題的這段期間，我認為我最大的收穫是在查詢資訊的能力上有了很大的進步，很多時候小車發生的 bug 很難自行解決，因此從網路上查詢其他人的經驗和其他背景相關知識就相當重要。另外，當需要添加一些新功能時，也常常需要透過網路來獲得相關知識，像是我先前沒有架設網頁相關的知識，但透過在網路上學習，最後有成功完成我們專題所需的功能。在這一年中，也要特別感謝學長和教授每周抽空和我們開會，並給予問題解決的方向，幫助我們完成我們的目標。

陳昱淳

這次實作專題是第一次接觸 ROS 這個實作系統，因此花了較多的時間去探索與熟悉，起初碰到了許多挫折，除了遇到版本不相容且無法更新的問題外，也時常碰到需要花比較多功夫才能搜尋到相關解決方案的難關。儘管如此，我們卻也藉此過程學習在面對挑戰時應該如何應對及瞭解團隊合作的重要性，我想這也是專題中的一大收穫。雖然因為小車尺寸構造的限制以及環境因素等等使得我們無法更貼近的模擬外送自走車有些可惜，但過程中也習得了不少新知識，更汲取了很多難能可貴的經驗，希望以後有機會可以改善我們專題的不足之處，在未來也可以將這些所學及經驗用於各個領域。最後，很感謝教授及學長每周花時間跟我們開會討論並給予許多寶貴建議。

鄭筑元

透過這個專題，我得到一次寶貴的經驗去實現平時在課堂上學習到的知識，並有機會為了自己的興趣去學習新的內容，而非為了單單應付的課業內容，讓我在這段時間對於學習的意義有了不同的見解。雖然需要在課業之餘花費額外的時間來完成專題，但同時也填補了我許多空閒的時間，整體的生活也過得更為充實且充滿目標。比較可惜的是在專題的過程中與其他課業的應衡讓我有時無法做到兩全其美，也在逐步了解新的知識的同時對更前方的領域有了嚮往，只能在有限時間的專題最後割愛，也希望在競賽之後能利用剩餘的時間把專題的內容自己盡善盡美，做到不虛此行的目標。