OAC-110-002（研究報告）

基於視覺定位與影像辨識之
港灣海漂垃圾清理船

正式研究報告

海洋委員會補助研究

中華民國 110 年 10 月

OAC-110-002（研究報告）

基於視覺定位與影像辨識之
港灣海漂垃圾清理船

正式研究報告

學校：國立高雄科技大學

指導教授：余志成

學生：吳孟懷

研究期程：中華民國110年4月至110年11月

研究經費：新臺幣柒萬元

海洋委員會補助研究

中華民國 110 年 10 月

OAC-110-002

基於視覺定位與影像辨識之港灣海漂垃圾清理船

正式報告

海洋委員會

目次

圖次

摘要

關鍵詞：**海漂垃圾、物件檢測、GPS定位、超音波測距、視覺定位**

近年來，海洋環境保育的議題逐漸被受重視，海漂垃圾對海洋生態與沿岸景觀的危害也被大眾所看見，如何減少這些海漂垃圾成為十分重要的課題。對於海漂垃圾的處理，目前常見的主要方式可分為縮源減量、河口攔截與海面回收三個大方向。本研究將以港灣內的海漂垃圾回收作為主軸，設計並建構一艘清理船，藉由影像辨識中的物件檢測技術搜索海面，配合多種感測器的融合應用，自動針對海漂垃圾進行追蹤與收集。

本計畫研究課題可分為船體定位導航以及海漂垃圾辨識收集兩大部分，為了讓清理船可以更穩定地航行於港灣內，本研究使用 GPS 作為清理船的初級定位依據，配合超音波測距進行現場障礙物的閃避與沿岸的循航，並導入視覺定位技術來修正 GPS 帶來的誤差。海漂垃圾的辨識是從 NVIDIA 開發的人工智慧套件取用 MobileNet 神經網絡作為訓練模型，使用圖像採樣工具生成訓練樣本建立檢測用模型，以寶特瓶取代海漂垃圾作為實測時的檢測對象，驗證影像辨識功能在水上載具運作的可行性。

本計畫於初期建構整合測試用平台作為系統與部分功能測試的載體，用於驗證部分功能與程序結構的可行性，並在以較短時間進行修正與調整。研究中期完成清理船原型機的建置，將系統與功能進行平台轉移，針對水上環境更變驅動方式並進行部分參數的調整，最後於實測時對各項功能進行個別測試與調整，並記錄其運作成效，驗證了本研究對於海漂垃圾清理的可行性。

第一章 前言

第一節 研究緣起

海漂垃圾作為近年來被受重視的議題之一，其危害與影響無論對海洋生態或沿岸經濟皆是十分嚴重的，許多政府與團體的應對方法普遍以縮源減量為主，對於目前仍在漂流的海漂垃圾，則主要以人工或專用船隻進行打撈作業，這些方法十分消耗人力且具有相當高的作業成本與一定危險性，即便能收集到大量的垃圾，其帶來的效益仍十分低落。

隨著自動化技術的發展與人力成本的提高，許多領域開始導入機器人技術來協助作業，又伴隨著機器學習與影像技術的加入，機器人技術的應用領域進一步擴大，從家用服務型機器人到工業用協作機械手臂皆有其蹤跡，這些機器人在取代人力並提高工作效益的同時，可運作於危險環境中，保障人們生命安全之餘，持續推動工作進度。

目前國內外已有少數相關團隊將機器人技術應用於海漂垃圾的收集作業上，然而大多數卻只有單純的收集與移動兩種功能，僅有少數具備障礙物探知的能力，此外則不具有海漂垃圾的辨識功能，這使得這些海漂收集機器人的工作效率難以發揮。本研究將從現有的海漂清理機器人案例中，以工作表現較良好且外型較合適的硬體結構作為參考，為其加入海漂垃圾辨識功能與其他不同的移動模式，使其能夠在工作時自動追蹤海漂垃圾，藉此提升工作的效益。

第二節 問題背景

海洋漂浮垃圾的存在是近年許多人關注的議題之一，因人為活動在各式水域被丟棄的漂浮垃圾，受到海流與風阻效應影響，其分佈將逐漸擴大至各個海域，其中更以太平洋最為嚴重，而地處太平洋東岸的臺灣與亞洲各個區域因此可能面臨巨大的海洋垃圾危害。影響海洋生態之餘，風阻效應可能會將海洋漂浮垃圾推送到沿岸邊，甚至是陸地上，這對於海岸與港灣的環境將造成危害，影響沿岸的生態與經濟活動，因此海漂垃圾的清理成為一項重要的工作。目前常見的主要方式可分為縮源減量、河口攔截與海面回收三個大方向，雖然縮源減量才是治本的作法，然而對現有的海漂垃圾進行清理作業也是不可避免的工作，一旦海漂垃圾

隨著潮流沖刷上岸，由於海岸環境的複雜，使得垃圾的清除只得仰賴人力(圖 1)，這將是非常耗費成本的工作，因此近年很多海漂清理設備的研究與開發集中在河口攔截與海面回收兩方向。

圖 1、大量海漂垃圾囤積於沿岸[1]

圖 2、港灣內的海漂垃圾

第三節 現況分析

本研究以港灣內的海漂垃圾作為主要清理目標，海漂垃圾收集裝置可根據其運作原理分為被動型與主動型兩類，前者往往需依賴水流與海浪等環境媒介，配合裝置本身的機構設計進行收集作業，後者則多具備一定的位移能力，藉由裝置

的自身移動接近目標以進行收集作業。被動型收集裝置不需設置大量感測器，卻十分仰賴周圍的環境要素，因此僅能用於特定環境，而主動型裝置對環境媒介的仰賴降低許多，取而代之則須具備更多的環境感知能力。

由 Andrew Turton 與 Pete Ceglinski 共同發明的 Seabin [2][3]是藉由海面水位差吸入周圍的漂浮垃圾，以達到收集效果。John Kellett 設計開發的 Mr. Trash Wheel [4][5]與 The Ocean Cleanup 設計開發的 THE INTERCEPTOR™ [6]皆是以河水流勢作為推力，配合引導浮標將漂浮垃圾集中並撈起。Chen Su 等人提出的文獻[7]中，收集裝置的前方與側面皆有設置超音波感測器，用於感知裝置與岸邊及障礙物之間的距離，其工作原理是藉由漂浮物容易滯留於岸邊的特性，於岸邊進行規律性的移動以達到清理效果。由湛藍海洋聯盟所設計的第二代湛鬥機 (圖3)以遠端遙控作為主要的控制手段，可透過人為控制針對特定區域進行清理。荷蘭 RanMarine 公司開發的 WasteShark[8]使用 3D 光學雷達感知周圍環境，除了人為遙控操作外，亦可自主定時進行固定路徑的巡航。

目前可見的海漂垃圾清理機器人中，普遍仍以人工遙控作為主要控制手段，只有少數案例附有自動巡航的能力，而海漂垃圾清理機器人不需像掃地機器人一樣進行全面覆蓋式的清理，僅需針對有海漂垃圾的位置進行重點清理即可。

圖 3、由台灣團隊設計開發的第二代湛鬥機[9]

第四節 研究目的

前段提及的各項案例中，無論是國內或國外的海漂清理裝置多以收集方式作為設計重點，移動模式則普遍以人工遙控方式為主，這使得裝置缺乏自主作業的能力，僅有少數案例具備定時巡航功能於固定區域內進行全面覆蓋式清理。

本研究預計將影像辨識技術與相關感測器應用於海漂垃圾收集裝置上，讓收集裝置具備辨識與追蹤海漂垃圾的功能，也令其對周圍環境有一定的感知能力，前者使裝置能針對前方的目標進行重點清理，後者則讓裝置具備障礙物感測的能力，藉此在提升海漂垃圾收集效率與成效的同時，讓裝置自行閃避障礙與規畫路徑，以實現真正的自動運行。相關研究重點分述如下：

一、 VSLAM 定位

VSLAM (Visual Simultaneous Localization and Mapping)技術隨著近年影像辨識技術的普及而一同崛起，藉由影像中的特徵點反向推算相機的姿態變化，讓相機在機器人技術中又增添一項用途，然而相機作為一種新穎的感測器類型，其附帶的資料量與運算量皆十分龐大，這表示 VSLAM 的使用將對處理器的運作造成不可忽視的負荷。

二、 影像辨識

影像辨識技術是基於大量的樣本累積，透過神經網路的迭代運算找出目標物件的圖紋特徵，並於即時影像中快速找出相符特徵的位置，藉此檢測出目標物件。影像辨識的準確性受到樣本有效性、神經網路類型與運作環境影響，因此建立出有效的檢測模型將是本研究的重要課題之一。

三、 避障與循岸功能

港灣作為貨物與漁獲來往的樞紐之一，岸邊時常停靠許多船隻，為避免與船隻發生碰撞以保持清理作業的順利進行，收集裝置須具備更多環境感知能力來實現避障與循邊功能，然而戶外的水上載具在感測器的選用需要面臨較多難題，如日光、水氣、工作範圍等，因此感測器的挑選將對避障與循邊功能的實現產生影響。

四、 **返航功能**

作為一台在海上運作的裝置，能否順利返回出發點是十分重要的，目前常見的海上定位方式多以全球定位系統(GPS)為主，然而大量的誤差卻是其難以避免的問題，這對於小型裝置的影響是難以被忽略的，如何透過軟硬體的調配將其誤差修正則是返航功能的重要因素。

第五節 預期目標

港灣作為船舶停靠的區域，相比外海是較為穩定的水域環境，這導致許多漂浮垃圾在港灣聚集後就難以流出，而港灣同時作為眾多人跡往來的地點，岸上時常殘留許多垃圾被強風吹落而漂浮在水面上。由於水域特性與人為因素的結合，使港灣內容易囤積漂浮垃圾，與外海情況相比，其分佈較為集中，將收集裝置裝設於此可獲得較良好的清理速率與成效。

本研究將設計並製作一艘可用於港灣內自動收集海洋漂浮垃圾的清理船，令其進行漂浮垃圾的追蹤與收集，主動性的清理方式相比其他收集裝置，可以進一步提升清理效率。本研究預計導入影像辨識技術與深度學習演算法檢測水面上的漂浮垃圾位置，藉此進行後續的追蹤與收集作業，同時使用視覺同步定位與地圖建構(VSLAM)技術於岸邊進行視覺定位，令其靠近原點時可以自行前往指定區域進行充電、清潔等後續作業。本研究也將加入 GPS 接收器與超音波感測器，前者用於定位並配合程式推算執行返航功能，讓船體回到原點附近；後者則以陣列方式排列於船體周圍，用於感測周圍障礙物並進行循邊功能，使船體能夠穩定航行在岸邊。

第二章 研究方法及過程

本章將針對本研究的主要功能進行介紹，解釋其運作原理與使用方式，並於最後的過程介紹中說明這些功能是如何應用於本研究中。

第一節 運算核心與控制系統

一、 運算核心系統建置

本研究以 NVIDIA Jetson Nano 開發套件(圖 4)作為運算核心使用，其具有 4GB 64-bit 的記憶體、4 核心的 1.43GHz 處理器(CPU)與 128-core 圖形處理器(GPU)，使其在圖像運算表現上更為強大的。本研究所使用的 Jetson Nano 作業系統是由 NVIDIA 官方提供的 JetPack SDK 系統開發套件進行建置安裝，該套件是以 Linux 架構的 Ubuntu 系統為基礎，加入 CUDA、TensorRT 等功能，讓硬體在電腦視覺、機器學習等相關運算上有更良好的表現。本研究根據 NVIDIA 官方網站提供之教學[10]，將 JetPack SDK 套件之映像檔燒錄至 micro SD 卡中，將其插入 Jetson Nano 主板卡槽後連接電源進入開機程序。Jetson Nano 在開機後將自行讀取 SD 卡的資料進行作業系統與相關套件的建置與安裝，建置完成後便可連接螢幕與鍵盤滑鼠操作使用。

圖 4、NVIDIA Jetson Nano 開發套件[11]

二、 ROS 機器人操作系統

Robot Operating System[12]，簡稱為 ROS，顧名思義為機器人操作系統，其開放式的協作框架，可以更方便地串接各個機器人功能。ROS 具備良好的整合功能，其中便可將感測器與中央處理器進行結合，整併不同感測器的通訊協定，減少繁複的機器人設定。ROS 提供硬體描述、驅動程序管理、消息

傳遞和程序代碼管理等功能，同時集成許多有用的工具與函式庫，以利開發者迅速上手。計算圖(Computation Graph Level)為 ROS 網絡通訊的架構，是使用 P2P(peer-to-peer)網絡傳輸協議，基本的概念包括 Master、Node、Topic、Parameter、Server 等要素，開發者可以利用此架構快速建立各種不同感測器間的通訊，如圖 5 所示。將感測器收集到的資料建立為發佈者(Publisher)，並發送資料形成主題(Topic)，最後由訂閱者(Subscriber)接收資料，此舉可以使不同感測器的資料，在此架構下集中處理所有的數據。

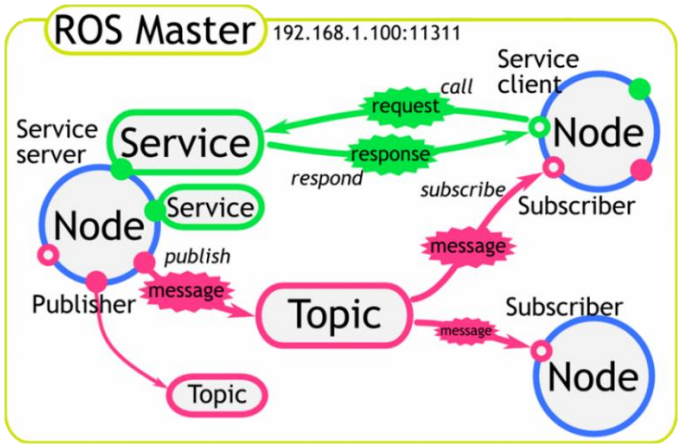

圖 5、ROS 通訊框架構示意圖[13]

三、 控制核心整合

本研究使用 Arduino Uno 作為清理船原型機的驅動控制與感測器接收核心，作為許多開發者常使用的微處理器，其具備成本低廉、操作容易與軟硬體開源等特性，除了可以兼容多種感測器模組，也適用於驅動輸出的訊號控制。本研究於 Jetson Nano 中使用 Python 語言配合 pySerial 函式庫[14]連結 Arduino Uno 單晶片，藉此取得超音波感測器的訊號，並在訊號輸出給馬達的同時反饋當下資料給運算端(Jetson Nano)。

圖 6、Arduino UNO 單晶片開發板[15]

四、 整合測試用平台建置

測試平台以 NVIDIA 推出的 Waveshare JetBot 人工智慧套件作為基礎，在後方加上 Arduino Uno 單晶片開發板，在車體前端三個方向設置超音波感測器，並於前端上半部架設單目相機，以此作為功能整合測試時所使用的移動平台，如圖 7 與圖 8 所示。建構此平台可以用於測試各個功能，讓測試過程不受原型機的建置完成與否影響，及早驗證各個功能的可行性，同時降低驗證難易度與其耗費時間。

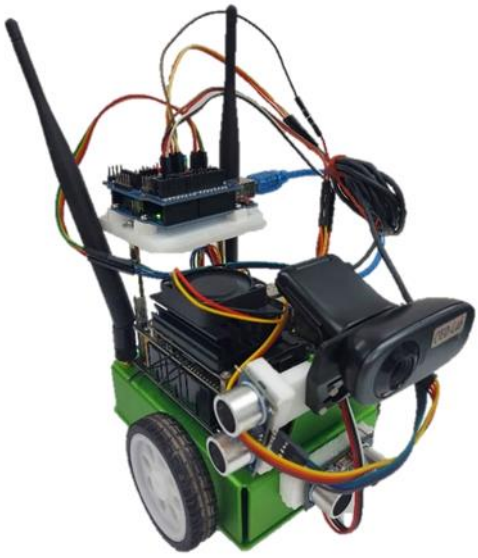

圖 7、整合測試用平台實體圖

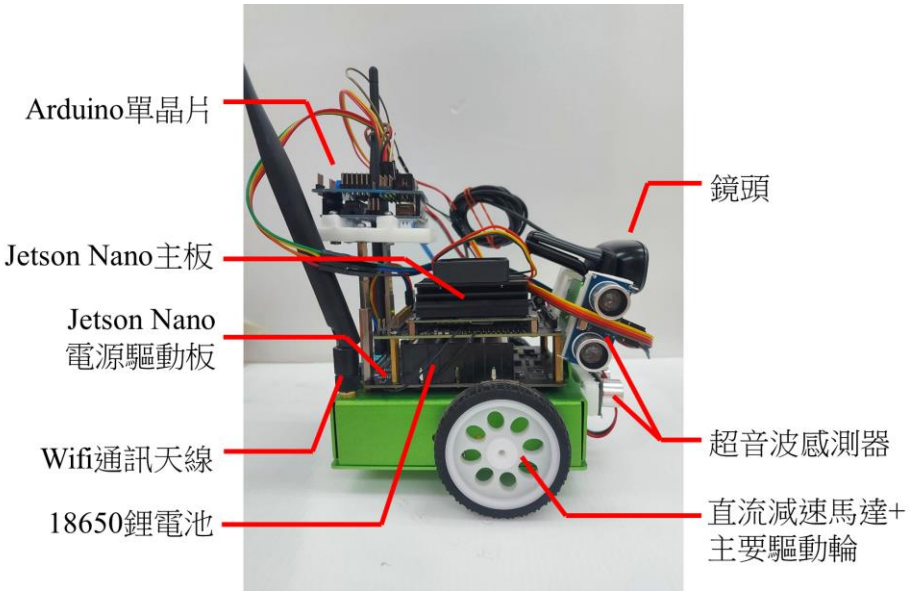

圖 8、整合測試用平台配置圖

第二節 同步定位與地圖建構

一、 視覺同步定位與地圖建構 (VSLAM)

Simultaneous Localization and Mapping，簡稱 SLAM，是藉由雷達、相機等環境感知裝置收集環境資訊，同時進行相機姿態的定位估計的技術，而 VSLAM 則顧名思義是以視覺影像作為資訊進行定位與地圖建構。相機類型主要可分為單目(Monocular)、立體(Stereo)與深度(RGBD)三種，根據不同類型，其背後的演算機制也有所差異。

ORB-SLAM 演算法[16][17][18]作為 VSLAM 可支援上述的三種相機類型，其原理是使用 Oriented FAST and Rotated BRIEF(ORB)演算法[19]提取影像中的特徵點並賦予其描述符，接著與其他影像中的相似特徵點進行匹配(圖 9)，最後由兩特徵點間的差異推算相機姿態的變化。待初始姿態確立後，ORB-SLAM 演算法即可繼續提取新的特徵點擴增點雲地圖，同時藉由特徵點估計相機於點雲地圖中的姿態，並修正新特徵點的位置誤差。如圖 10 所示，紅點代表目前相機照射到的特徵點，黑點代表已保存在地圖環境的點雲，外圍的綠線條則是相機的移動軌跡，而左下角的即時影像可用於觀察目前特徵點提取狀態。

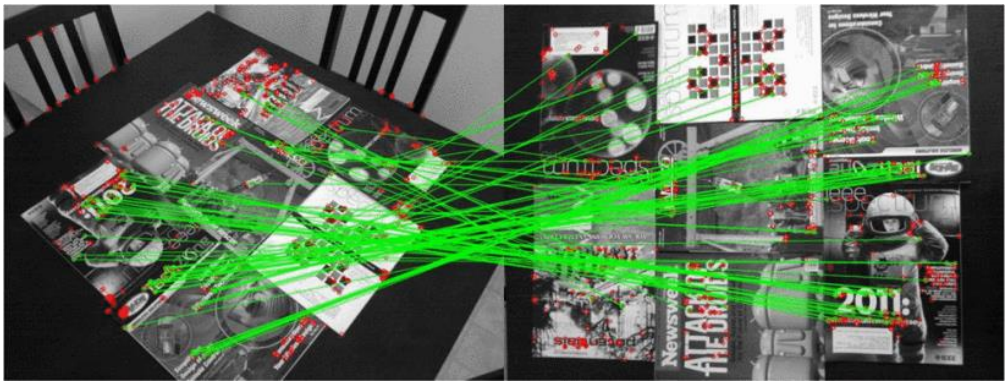

圖 9、ORB 演算法之特徵點匹配[19]

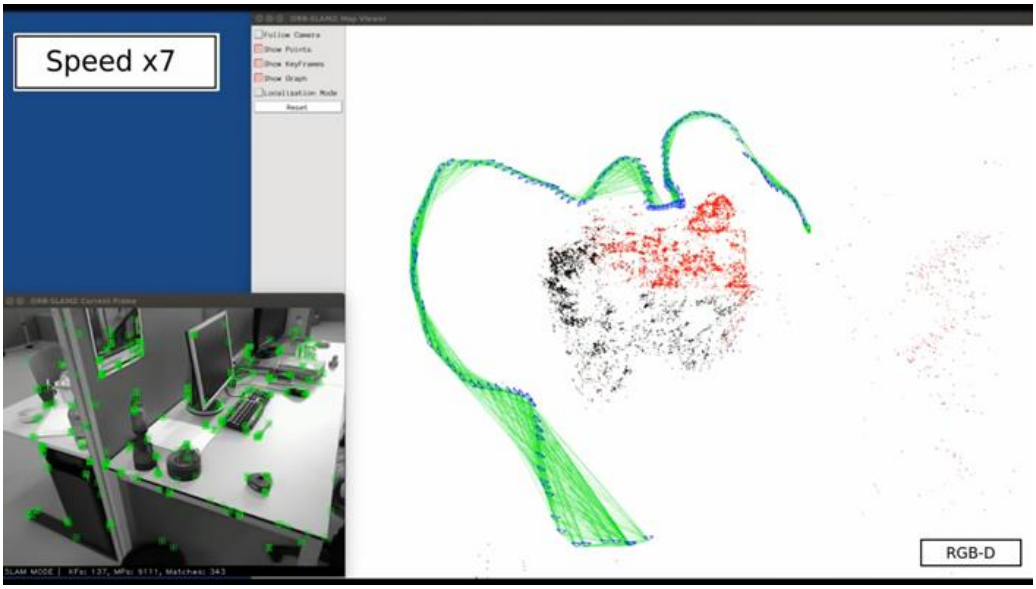

圖 10、ORB-SLAM2 點雲地圖建構示意圖[17]

二、 VLSAM 與 ROS 整合

為使各個功能可以順利整合，必須先完成 ROS 的建置。本研究使用 ROS 官方於 2018 年推出的 Melodic Morenia 長期支援(LTS)版本，相較現有最新的 Noetic Ninjemys 版本，在許多功能上有更良好的相容性。ROS 的建置方法可從官方提供的教學網頁[20]參閱，僅需約二至三個小時便可完成作業，根據不同的硬體配置會有些許差異。完成 ROS 的建置後便可進行 ORB-SLAM3 功能包的安裝與測試，配合 Pangolin 可視化工具，可以針對顯示項目進行調整，亦可選擇工作模式如地圖建構或視覺定位。本研究於實驗室中環繞一圈作為 ORB-SLAM3 之功能測試，如圖 11 示。圖中的紅點代表當前相機捕捉到的特徵，黑點代表過往路徑中已保存的點雲，藍色框體代表紀錄特徵點當下的相機姿態，而穿越各框體中心的綠線則代表相機的移動軌跡。

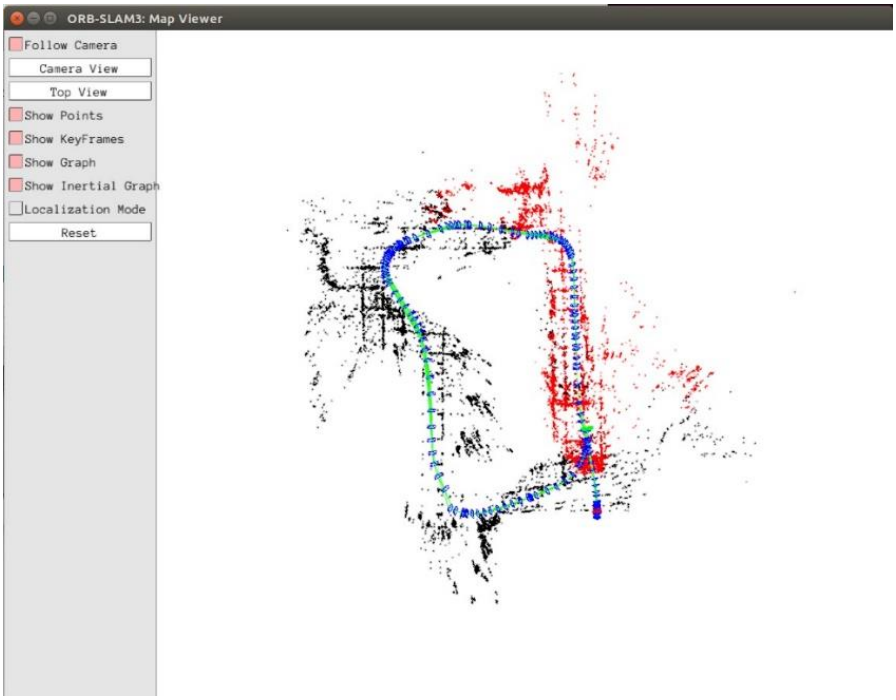

圖 11、使用 ORB-SLAM3 建構實驗室點雲地圖

第三節 影像辨識與物件檢測

一、 深度學習

影像的深度學習主要以深度神經網絡(Deep Neural Networks，簡稱 DNN)為基礎，投入大量圖像樣本進行訓練，藉此產生模型以用於分類、檢測與分割等功能。目前常見的深度學習模型大多以卷積神經網絡（Convolutional Neural Networks，簡稱 CNN）作為核心基礎，藉由卷積層與池化層的反覆堆疊與運算，取得圖像的特徵並降低其資料量，以找出目標物件的特徵排列，如圖 12 所示。

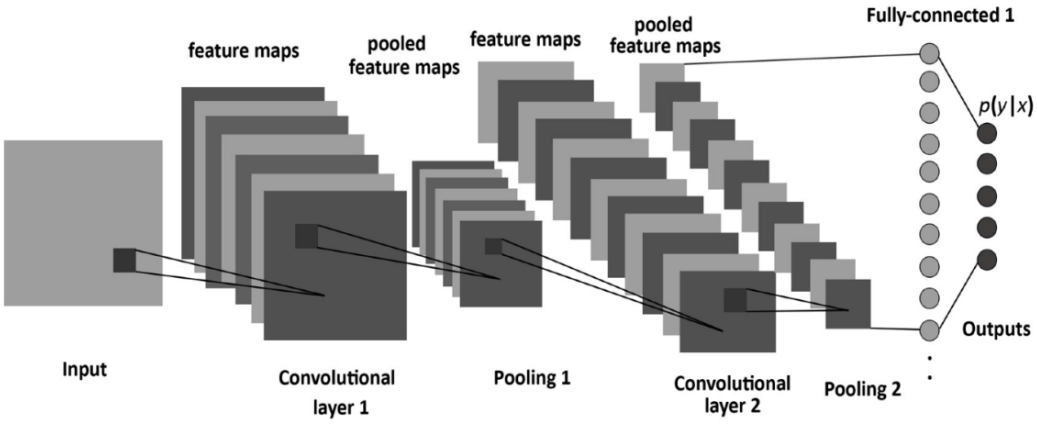

圖 12、CNN 卷積神經網路運作架構[21]

隨著深度學習的發展，針對不同的使用需求與條件，已有許多不同特性的神經網絡模型可供使用，如 AlexNet[22](圖 13)、VGGNet[23]、GoogLeNet[24]與 MobileNet[25]等 CNN 經典模型。同時，許多針對深度學習而建構的函式庫如 Scikit-Learn[26]、TensorFlow[27]與 PyTorch[28]等，皆提供 Python、C++等程式語言快速使用。除了可以令開發者更快速地建立編程環境，更大幅降低深度學習的使用難易度，讓大眾能更輕易入門。

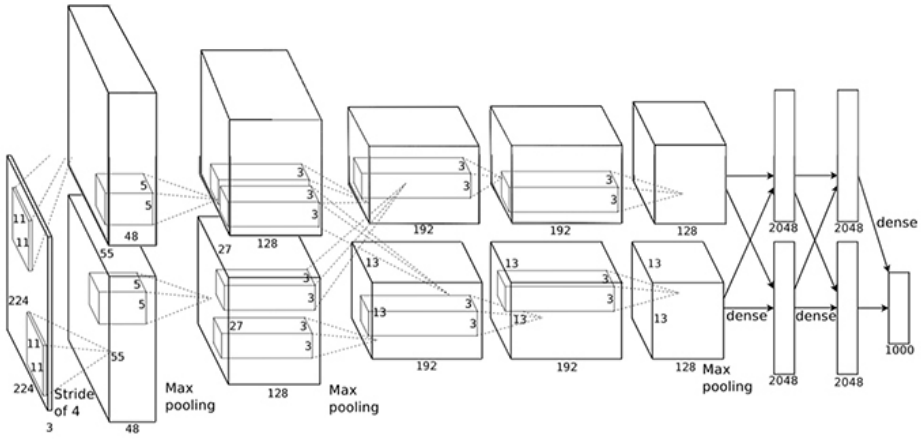

圖 13、AlexNet 神經網絡架構[22]

二、 物件檢測

海漂垃圾之組成成分複雜，難以單獨辨識其中的特定物件，也較難收集大量的樣本進行訓練。因此，本研究以寶特瓶作為檢測目標，除了較容易取得大量訓練樣本外，於後續的實測也較容易進行場地的復原。

本研究使用 NVIDIA 專為 Jetson 系列套件開發的影像辨識模組包作為物件檢測功能的基礎，並使用其中包含的 MobileNet 進行樣本訓練，作為一個輕型神經網絡，除了可以讓 Jetson Nano 即時檢測物件，更可從中選取最合適的影像訓練模型。影像的訓練樣本可從 Microsoft COCO、Open Images 等現成數據集中提取，而本研究是透過模組包內的圖像採樣工具自行製作專用數據集，如圖 14 所示。

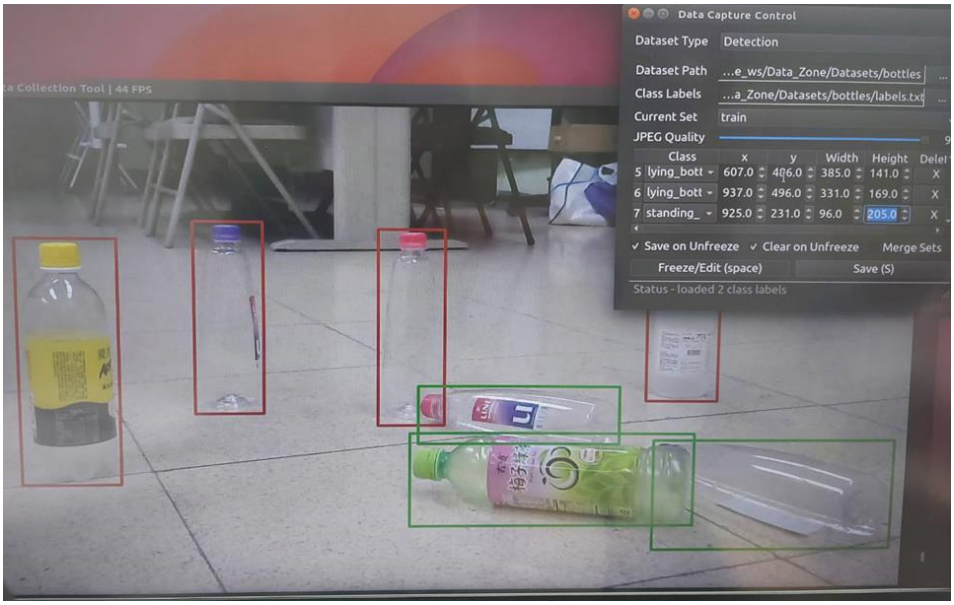

圖 14、使用採樣工具框選目標

三、 物件追蹤

物件追蹤功能是以 Python 語言作為基礎進行編程，使用訓練完成的檢測模型對影像進行即時辨識，藉此取得目標物件於影像中的位置。根據位置的不同，系統將以水平距離最接近影像中心的物件作為追蹤目標(圖 15)，並以其距離大小調控輸出訊號的強弱。根據此方法，隨著影像中的目標越靠近兩側邊界，兩顆馬達間的速差會隨之提升。若無任何目標存在，兩顆馬達則會保持相同轉速，直至目標出現為止。

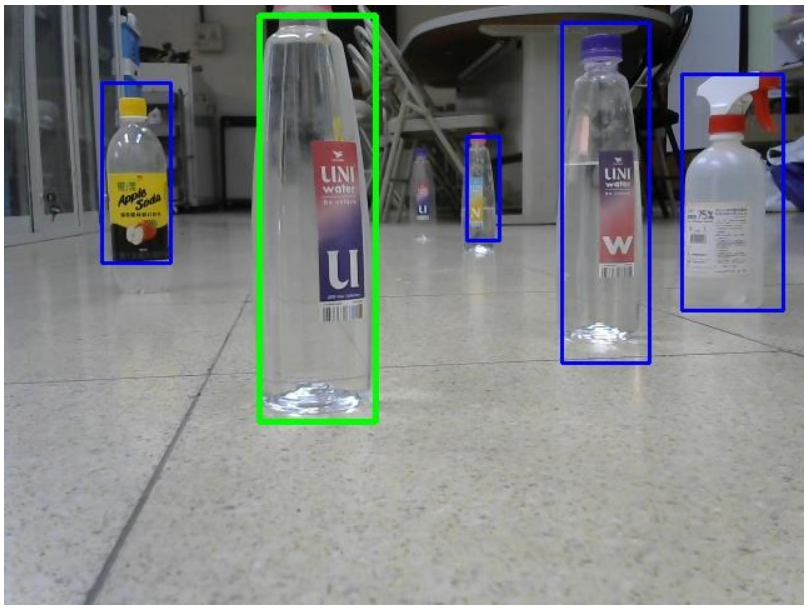

圖 15、多物件之追蹤目標判定(綠框為目標)

第四節 海漂清理船原型機建置

一、 原型機建置

本研究的清理船原型機採用雙體船設計[29]如圖 16 所示，該設計於航行時，對於較大的風浪具有一定的耐受性，於水上移動時具有良好的穩定性與操作性。收集漂浮垃圾時，雙體船設計可將其存放於兩船體間，配合欄柵機構可讓垃圾保持漂浮狀態而不流出，該作法可降低垃圾重量對平台吃水量的影響。移動方式將採用雙驅動設計，並以差速方式進行船體的轉向控制，此舉可提供較大的推進力，避免船體在航行時受到漂浮垃圾的阻擋而無法前進。

原型機的主體是以鋁擠型作為支架，再以 PVC 管作為船體搭建而成，Jetson Nano 與 Arduino 等電路組件放置於密封盒中，並固定於支架上，作為推進動力的電子變速器、無刷馬達、船軸與螺旋槳將固定於 PVC 管中，令其得以接觸水面產生推進力，整體硬體與系統架構如圖 17 與圖 18 所示。

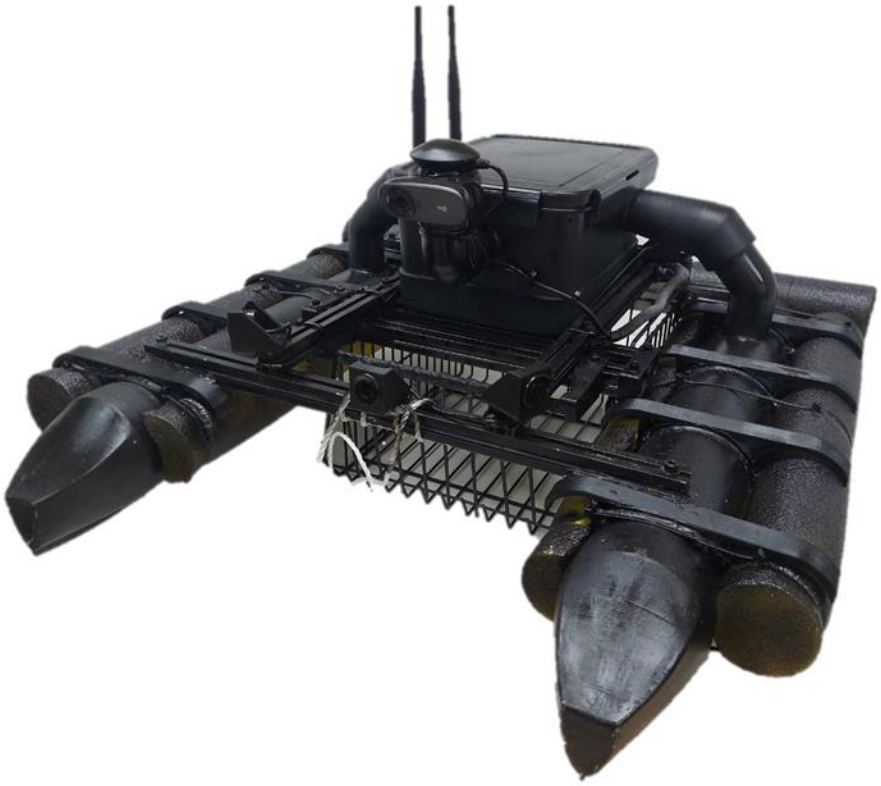

圖 16、清理船原型機採雙體船設計

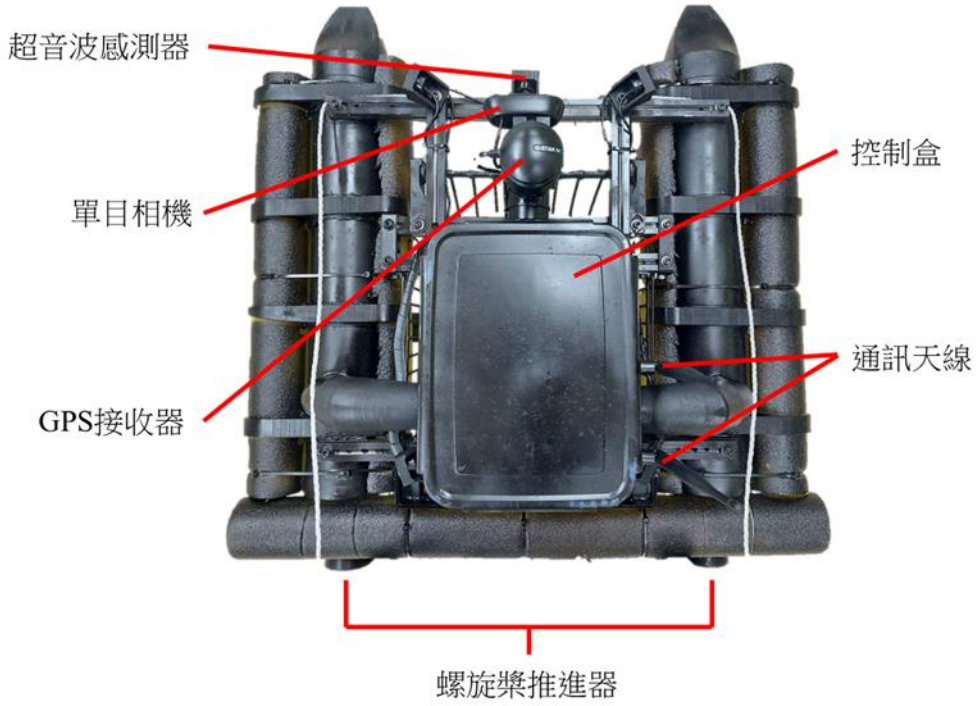

圖 17、清理船原型機硬體配置

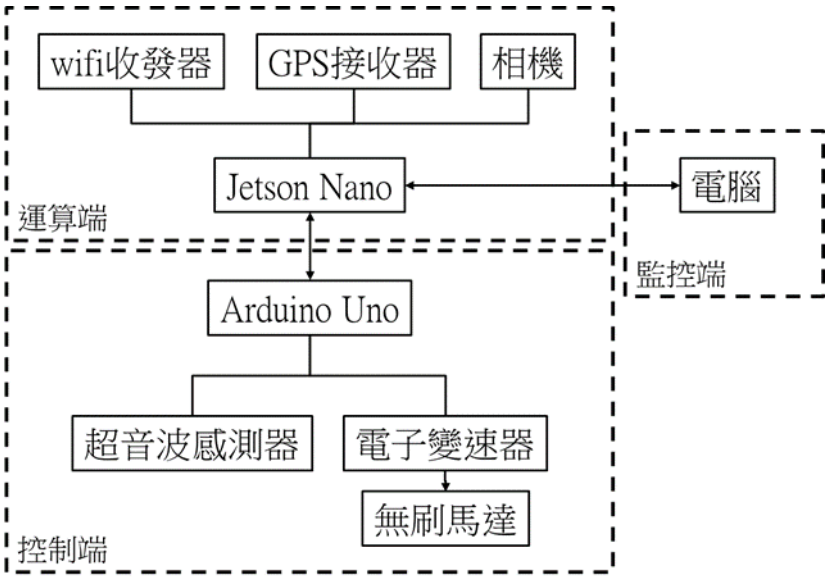

圖 18、清理船原型機系統架構

二、 動力驅動模組

本研究的動力驅動模組包含 2850 無刷馬達、60A 電子變速器、10mm 船軸以及 7500mah 45C 鋰電池，如圖 19 所示。無刷馬達用於帶動船軸，驅動螺旋槳作為原型機位移的動力，馬達轉速的控制是以電子變速器從 Arduino 接收控制訊號並輸出對應電流而完成。電子變速器的使用與伺服馬達相似，

皆是透過脈波頻率的高低進行調控，在編程上也可使用 Servo 函式來達成。鋰電池可提供最高 45 安培的輸出電流，7500 毫安時的容量也可提供約 2-3 小時的運作時間，其提供馬達的推進力或是續航力對於測試是十分足夠的。

圖 19、動力驅動模組(鋰電池、無刷馬達、電子變速器與船軸)

三、 超音波測距模組與循岸功能

本研究於原型機上設置超音波測距模組用於感知周圍環境，其運作原理是從發射端發出超音波訊號，於接收端感測其反射訊號，從訊號收發的間隔時間推算物件距離，如圖 20 所示。本研究採用 JSN-SR04T-2.0 超音波測距模組作為原型機循岸用的感測器，如圖 21 所示，該模組的有效感測範圍為 20 公分至 6 公尺，並具有 70 度的感測角度，其分離式的設計具備防水性能，適合用於戶外環境。

本研究以陣列排列方式將五顆超音波測距模組固定於船體的左後、左前、前方、右前與右後五個位置，以發散方式向五個方向進行感測，如圖 22 所示。透過兩側測距模組的前、後距離比對，可以推算船體與岸邊的距離及平行與否，藉此讓原型機在靠近岸邊或障礙物時保持一定距離移動，達到循岸移動與障礙物閃避的效果。

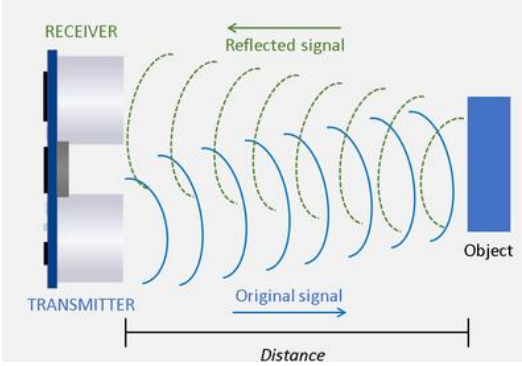

圖 20、超音波感測原理示意圖(HC-SR04 模組)[30]

圖 21、JSN-SR04T-2.0 超音波測距模組

圖 22、超音波測距模組安裝位置

四、 GPS 定位與返航功能

GPS(Global Positioning System)全球定位系統作為現今許多裝置與設備定位用的依據，其原理是透過接收器接收多顆 GPS 人造衛星的訊號後進行解碼，從中獲取接收器當下的位置訊息如經緯度、海拔高度等。GPS 可以提供設備當下的全局位置訊息，在應用上十分方便，但是其精確度卻十分容易受到周圍建築的遮蔽影響，使得 GPS 僅能在戶外有較良好的表現，然而氣候也是影響精確度的要素之一，種種因素導致 GPS 使用上方便，但精確度卻往往需要依賴其他裝置才得以提升。

由於海面環境空曠且遼闊，難以如岸邊一樣使用 VSLAM 與超音波感測進行定位，因此本研究於原型機上加入 GPS 訊號接收器(圖 23)用於取得位置資訊，並藉其作為遠離岸邊時的定位依據。GPS 訊號接收器可藉由 USB 介面連接運算核心，並使用 Python 中的 Serial 函式讀取接收器發出的訊息，以此取得經緯度、海拔高度、位移速率…等資訊。

圖 23、GPS 訊號接收器(BU-353S4)

GPS 作為返航功能的定位依據，為了於戶外實測前確認做法的可行性，本研究先以 Python 程式建構虛擬環境進行模擬測試，由當前位置、前一位置與原點位置三點推算待修正角度與距離，藉此產生下一次移動的指令(圖 24)，並逐步向原點移動，如圖 25 與圖 26 所示，倒三角標示處為返航起始點，中心正三角標示處則為路徑終點，即原點。

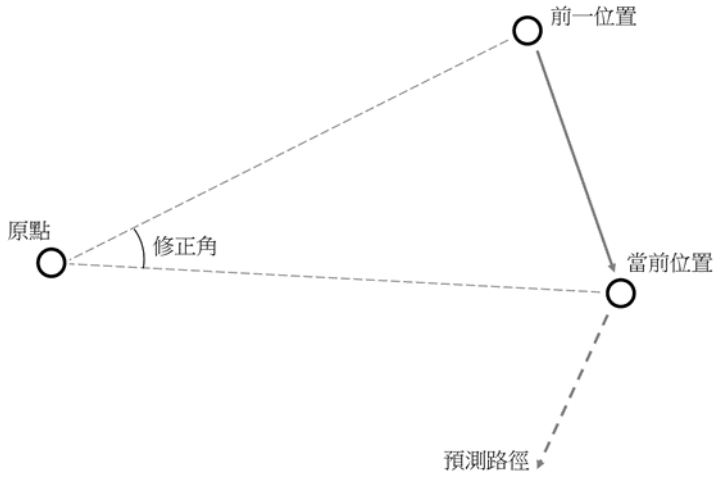

圖 24、返航推算示意圖

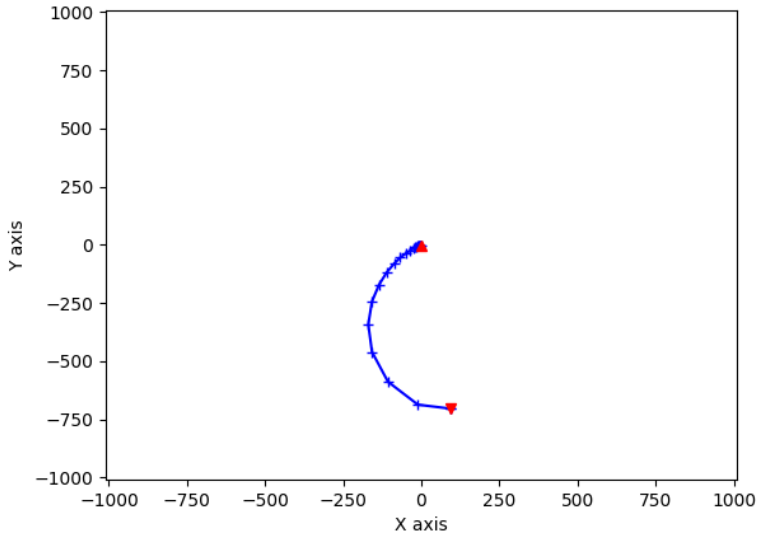

圖 25、返航模擬測試 1

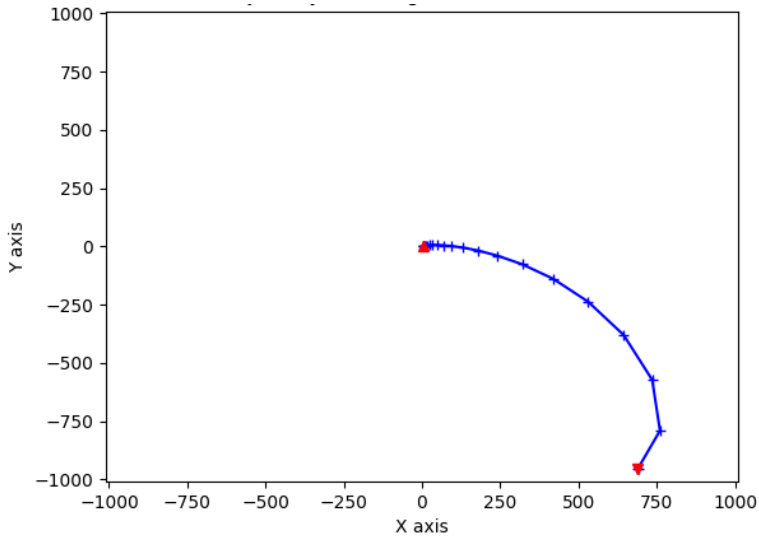

圖 26、返航模擬測試 2

第三章 結果與討論

本章將針對本研究中的主要功能，在試驗場域進行成果測試，並說明其功能成效。

第一節 VSLAM 視覺同步定位與地圖建構

本研究先以實驗室作為 VSLAM 功能在陸上測試時的試驗場地，並於功能驗證完成後將其應用於水上環境。圖 27 為 VSLAM 功能運行時，以桌面遠端連線方式進行監控的畫面，左上視窗為 VSLAM 功能對影像進行特徵點提取的即時狀態，右下視窗則記錄所有特徵點位置，並獲取當下的相機姿態。

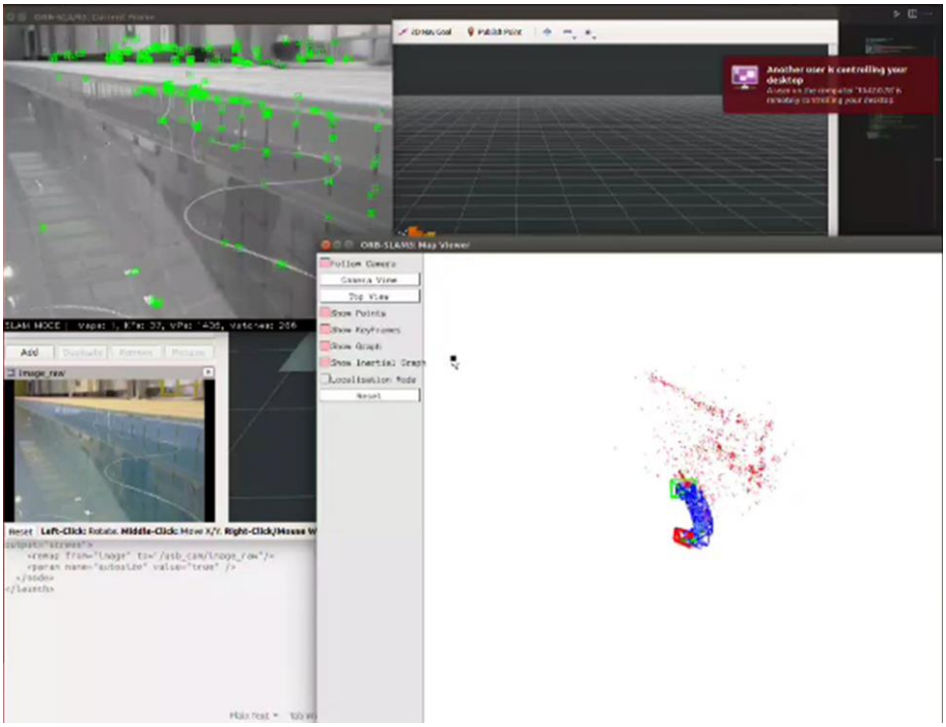

圖 27、VSLAM 功能監控畫面

與陸上測試的成效相同，VSLAM 功能在水面上可藉著岸邊景物的特徵進行定位，並取得相機姿態的變化，然而該成效卻僅限於相機指向岸邊的情況。由於水面在船體移動時會產生波紋，這導致 VSLAM 在相機指向水面時難以從中上取得可靠的特徵，同時過去成功提取的特徵點也難以繼續鎖定，種種要素使得 VSLAM 的定位與地圖建構功能難以在水面上穩定且有效地運行。

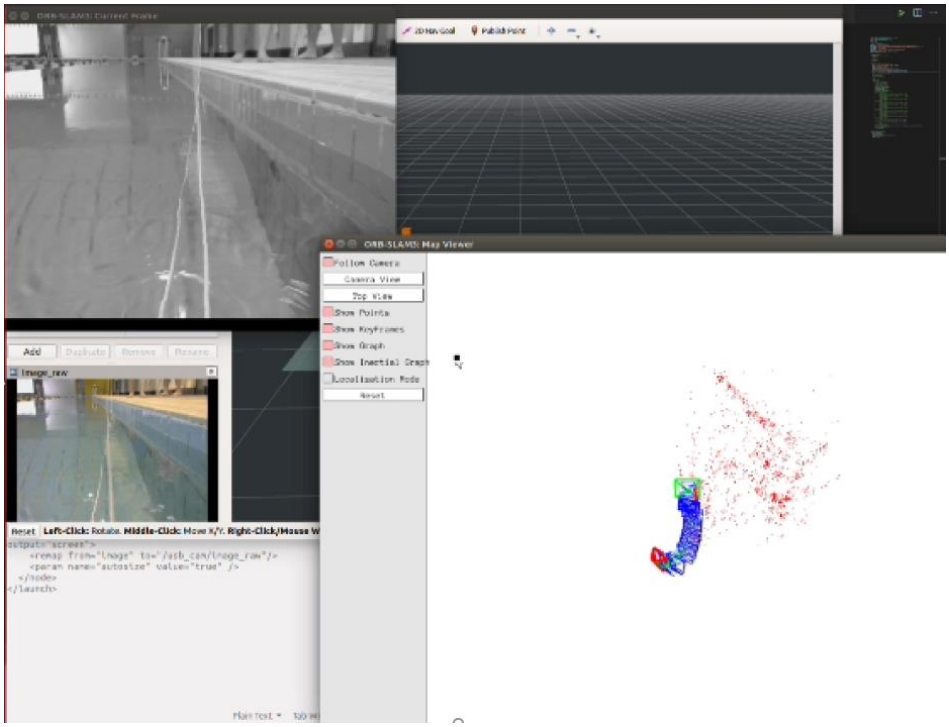

圖 28、VSLAM 功能丟失

第二節 物件檢測與追蹤

為提高驗證過程的效率，同時簡化測試場地的復原流程，本研究以寶特瓶取代漂浮垃圾作為檢測與追蹤的目標，並分別從陸上與水上兩種環境進行取樣，建立對應的檢測模型，用於不同環境的測試流程中。

本研究在水上測試過程中，將寶特瓶隨機丟入於泳池各處，並於其後放入清理船進行追蹤作業。圖 29 為清理船進行檢測作業，紅圈處為船體前方的寶特瓶。圖 30 為清理船檢測完成後執行追蹤程序，從船體方向可見其針對目標寶特瓶向右修正移動路徑，並將其收入下方收集籃內。圖 31 為清理船以水上檢測模型執行物件追蹤作業時的畫面，從中可見檢測的準確度良好，僅有少數受到遮蔽而較不完整的寶特瓶未被檢測到。

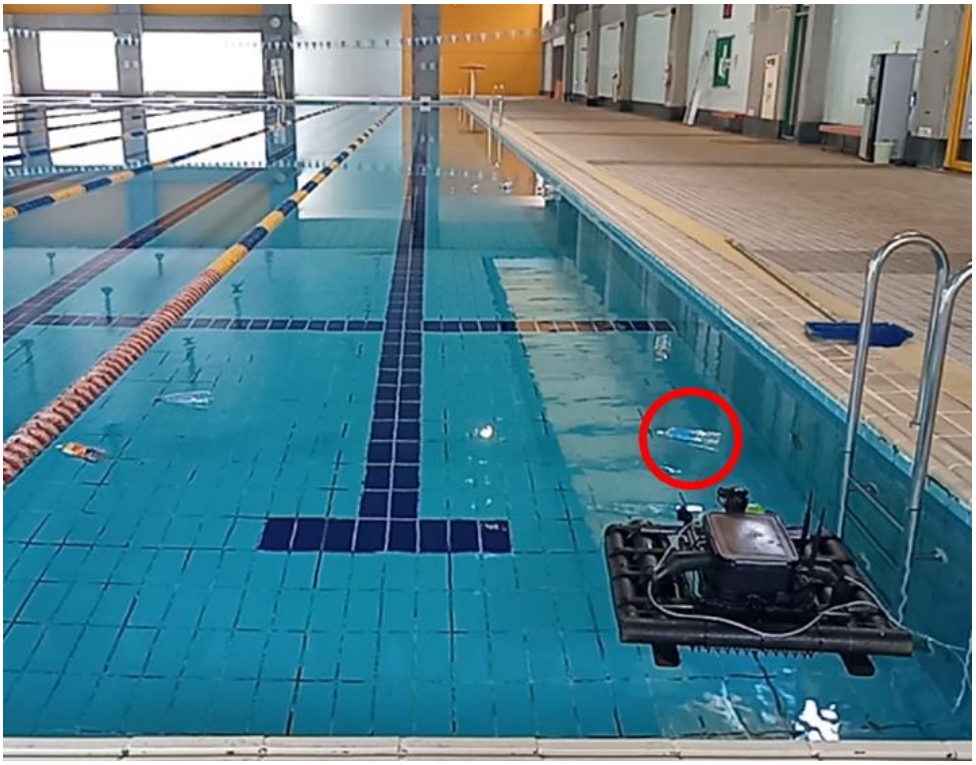

圖 29、物件追蹤水上測試-檢測流程

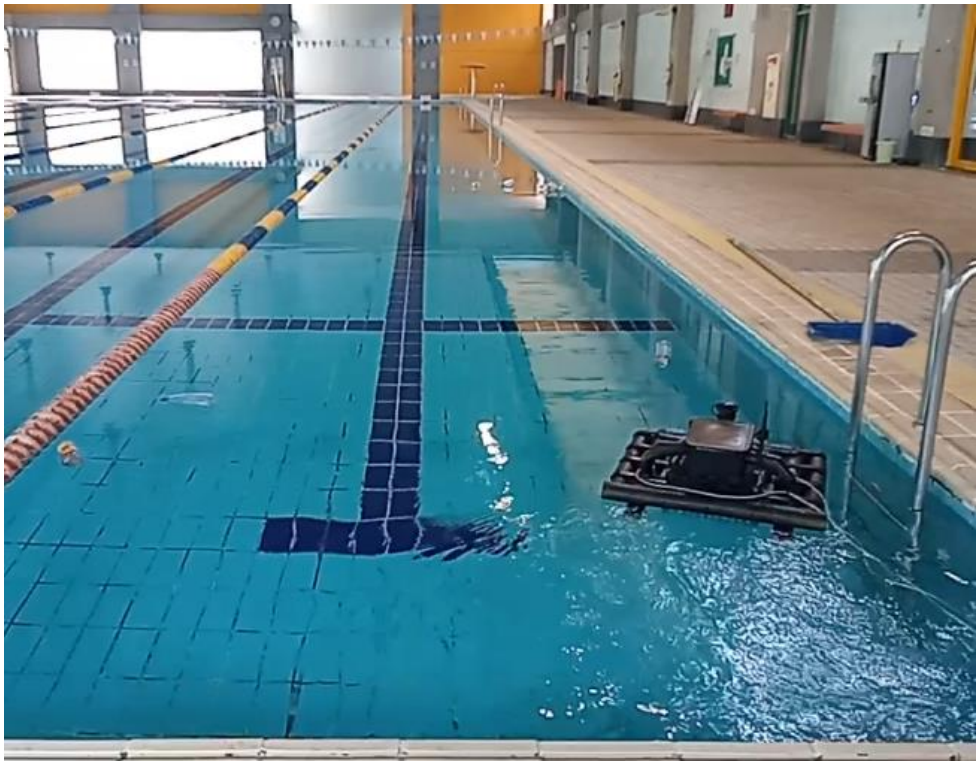

圖 30、物件追蹤水上測試-追蹤流程

圖 31、水面物件檢測成效

第三節 循岸功能

本研究藉由超音波測距模組感測周圍是否有沿岸或障礙物，並與其保持一定約 1 公尺至 1.5 公尺的距離前行。如圖 32 與圖 33 所示，當船體靠近沿岸時會觸發循岸功能進行方向調整，使船體的移動方向與沿岸保持水平。經實際測試，在循岸功能的運作下，清理船可以穩定航行於長 50 公尺，寬 3 公尺的泳道內，不會與沿岸發生碰撞。

圖 32、船體靠近沿岸準備觸發循岸功能

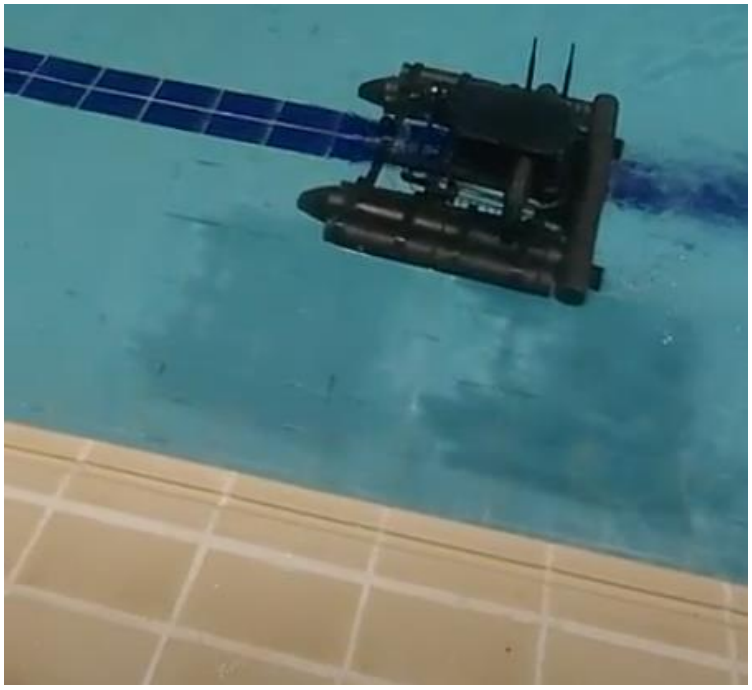

圖 33、船體與沿岸保持一定距離

第四節 返航功能

本研究以 GPS 定位技術配合 Python 的程式推算，將移動指令發送給 Arduino，使其進行驅動控制。受限於細長的水域環境與清理船較大的迴轉半徑，在水上測試容易受到沿岸影響而難以確認其成效，同時水域上方的遮蔽物將嚴重影響 GPS 定位精度，因此返航功能的驗證主要採用推車乘載方式，由遠端連線監控程式發出的移動指令，以人工方式依照指令行動，藉此驗證返航功能的成效，如圖 34 所示。

圖 34、以推車乘載清理船進行返航功能測試

如圖 35 所示，測試區域為長約 45 公尺，寬約 40 公尺的空地，圖中標示的倒三角為清理船的出發點位置，方形為清理船完成任務後準備返航的位置，而正三角則是清理船在返航功能中判斷已經抵達出發點的位置。測試時返航功能的完成判定機制是以出發點周圍，方圓約 10 公尺的範圍為準，而兩三角形的間距約為 5 至 6 公尺，這表示以 GPS 作為初級返航功能的運作是正常的。

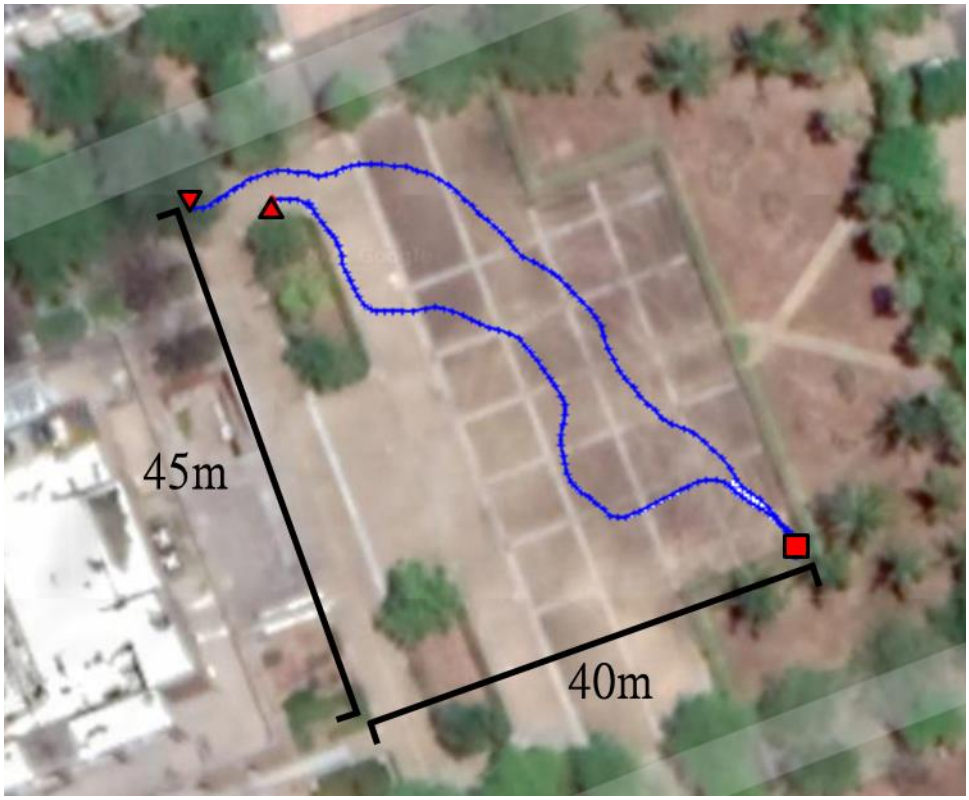

圖 35、返航功能測試成果

第四章 結論

港灣作為各個行業在海上作業的進出口據點，特殊的地形與頻繁的人跡活動使其容易囤積漂浮垃圾在海面上，破壞附近生態之餘更影響沿岸的環境衛生與市容。為解決上述問題，本研究以港灣內的漂浮垃圾作為主要目標，設計並製作一台專用的清理船。如圖 36 所示，清理船平時可停駐在港灣旁的充電站內進行保養與充電，並於指定時間到 GPS 指定地點執行清理作業。工作地點可藉由座標指定方式在路徑上自動設置中繼點，使清理船依照指定路徑前往目的地執行任務，根據環境的不同可設置多個區域讓清理船依序前往。當清理船進入清理區域後將開啟物件追蹤功能以執行海漂垃圾收集任務，並於目標丟失時以隨機游走方式等待下一個目標進入視野。若清理船在移動的過程中靠近沿岸或障礙物，將啟動循岸模式直到脫離為止。當清理船以 GPS 返航模式進入充電站方圓 5 至 10 公尺的範圍時，將啟用視覺定位或影像辨識鎖定充電站位置，以更精準的局部定位進入充電站。

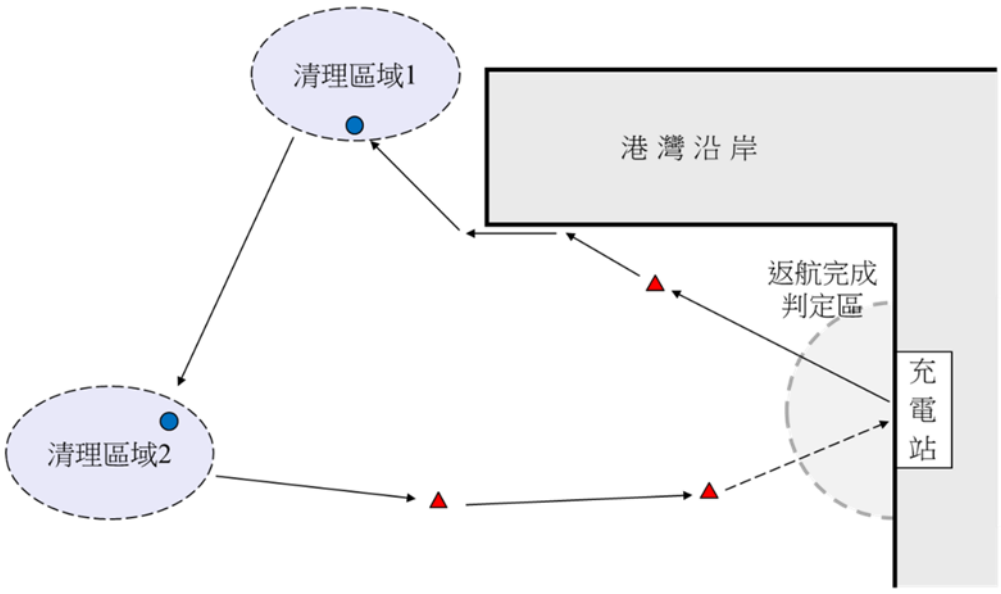

圖 36、清理船工作流程示意圖

第一節 執行研究中所遭遇之問題與困難

一、 硬體設備對視覺定位的限制

視覺定位作為一種定位方式，其原理是記錄影像中的特徵並賦予其描述，再投射至地圖空間後反向推算相機目前的姿態，藉此達到定位效果。為了提升視覺定位的準確度與穩健性，可透過參數的調整增加單幀影像中的特徵提取量，然而這意味著運算負荷將隨之提升，這對於本研究所使用的 ORB-SLAM 也不例外。由於本研究的重點在於影像辨識於海漂垃圾清理船的應用，其對於處理器的運算負荷十分龐大，視覺定位的加入將造成處理器難以兼顧兩者，並在運算時產生高溫導致系統效能降低。

二、 影像辨識取樣

本研究為了確認物件檢測在清理船上的可行性，同時降低測試時的場地復原難易度，因此以海漂垃圾中常見的寶特瓶替代作為功能驗證的對象。本研究分別從陸上與水上兩種環境的寶特瓶圖片中取樣進行檢測模型的訓練，然而寶特瓶的外型種類繁多，而在水上漂浮的角度更是難以完全記錄，這使得訓練完成的模型仍舊無法完全應付實際應用狀況，導致測試時偶爾會出現追蹤功能失常的情況。

第二節 未來展望

本研究嘗試將影像辨識技術應用於海漂垃圾的追蹤清理上，並製作簡易的船體作為功能驗證時的原型機，同時為其加入循岸與返航兩種功能。由於本研究的重點為各個功能的可行性驗證，對於隨機因素較多的實際海域應對方法並未多著墨，同時在測試過程中僅以寶特瓶作為清理目標，並無針對其他海漂垃圾類型進行檢測模型。在未來可針對本研究的硬體進行優化，同時讓軟體方面可以應付更多隨機變因，藉此可以實際於海上運行的海漂清理船。

一、 硬體設計

本研究使用的船體僅參考雙體船的架構設計，對於船體本身並未針對船舶、流體等原理進行設計，同時也縮小尺寸雖已能達到功能測試目的，未來可結合相關專業人士進行設計，藉此做出更加符合實際環境的應用船體設計。

研究中船體的收集空間是以斜口籃配合被動式欄柵機構製作而成，其收納量與阻擋效果十分有限。未來可專為船體設計特有收納裝置，內部具有較大的空間與更密集的間隙，同時設計特殊結構讓囤積垃圾對航行的影響降低，藉此提升整體收集效率與航行時間。

推進船體的馬達可選用馬力更強的型號，配合更大片的螺旋槳提升推進力，在未來船體容量提升的同時，可以支援其進行清理作業。

二、 **提高辨識成效**

未來可針對海面上的漂浮垃圾進行拍攝並取樣，藉此獲取大量的實際樣本，讓清理船得以針對實際海域進行作業。同時也可改用運算表現更良好的高階處理器，讓影像辨識使用更穩健的神經網絡，提高辨識的穩定性與精確性。

三、 **加入 RTK 即時動態定位技術**

目前的研究中僅以 GPS 作為返航時的定位依據，而 GPS 的誤差仍舊是無法忽視的問題，加入 RTK 即時動態定位技術可以有效修正 GPS 的誤差，提高清理船的定位精度。

四、 **運算能力更高的處理器**

本研究目前是以 Jetson Nano 作為運算用處理器，而其僅是 NVIDIA Jetson 系列中的初階型號，未來可選擇使用更高階版本的型號，或是開發專用的運算晶片，針對本研究中的功能進行優化，讓本研究中的影像辨識與視覺定位得以被同時應用。

附錄

研究編程之部分開發程式碼

```python
class Arduino:
    def __init__(self, port, baudrate, time_out):
        self.ser = serial.Serial(port, baudrate, timeout = time_out)

    def sonarInit(self):…

    def sonar(self):
        self.ser.write(b'sonar\n')
        # while self.ser.in_waiting:
        distance = self.ser.readline().decode('utf-8').split(' ')[:5]
        dist = [float(i) for i in distance]
        return dist

    def motorInit(self):…

    def motorCtrl(self, action, spdValue):   #action[mode,deviation] spdValue[stop,low,high]
        self.ser.write(b'motor\n')
        if action[0] == 'forward':
            lspd = spdValue[1]
            rspd = spdValue[1]
        elif action[0] == 'left':
            lspd = spdValue[0]
            rspd = spdValue[0] + round((spdValue[2]-spdValue[1])*min(abs(action[1]), 1.0))
        elif action[0] == 'right':
            lspd = spdValue[0] + round((spdValue[2]-spdValue[1])*min(abs(action[1]), 1.0))
            rspd = spdValue[0]
        else:
            lspd = spdValue[0]
            rspd = spdValue[0]
        self.spdMsgSend([lspd, rspd])
        mspd = self.ser.readline().decode('utf-8').split(' ')[:2]
        return lspd, rspd, mspd

    def spdMsgSend(self, spd):
        spd_msg = str(spd[0]) + str(spd[1])
        self.ser.write(bytes('%s\n'%(spd_msg), 'utf-8'))
```

```python
class Objectdetection:
    def __init__(self, net, model, label, input, output, thres):…

    def dist_cal(self, center_vec):…

    def center(self, detection, imgsize):…

    def closest(self, detections,imgsize):…

    def detect(self):
        self.image = self.input.Capture()
        self.imgSize = [self.input.GetWidth(),self.input.GetHeight()]
        detections = self.net.Detect(self.image)
        self.object_num = len(detections)
        # print("detected {:d} objects in image".format(len(detections)))
        detection = [d for d in detections if d.ClassID == 2]
        det = self.closest(detection, self.imgSize)
        if det is None :
            result = 'forward'
            deviation = 0.0
        else:
            deviation = self.center(det, self.imgSize)[0]
            if deviation == 0.0:
                result = 'forward'
            elif deviation > 0.0:
                result = 'right'
            elif deviation < 0.0:
                result = 'left'
        if self.input.IsStreaming() and self.output.IsStreaming():
            self.output.Render(self.image)
        return result, deviation
```

```python
class Rehome:
    posi_counter = 1
    previous_posi = [0, 0]
    correction_ang = 0
    current_dist = 0
    def __init__(self, homeGpsData, homeSize):…

    def dist_cal(self, pt0, pt1):   # Calculate the distance between two points.…

    def ang_cal(self, pt0, pt1):    # Calculate the angle of the line which is connected by two points.…

    def correction_ang_cal(self, ang0, ang1):   # Calculate the correction angle between two angles.…

    def posi2ang(self, posi1, posi2):   # Input two position data to get angle and correction angle.
        ang_home2posi1 = self.ang_cal(self.home_posi,posi1)
        posi2 = [float(posi2[0]), float(posi2[1])]
        ang_posi12posi2 = self.ang_cal(posi1,posi2)
        correctangle = self.correction_ang_cal(ang_home2posi1, ang_posi12posi2)
        return ang_posi12posi2, correctangle

    def point_recoder(self, posi_counter, position, distance, angle, correction):   # Record the point d
        if position != None:
            self.posiPoint.record(position)
        if distance != None:
            self.distPoint.record([posi_counter, abs(distance)])
        if angle != None:
            self.angPoint.record([posi_counter, abs(angle)])
        if correction != None:
            self.correctionPoint.record([posi_counter, abs(correction)])

    def initCheck(self, location):  # Get the first point when the rehoming process started.
        current_posi = location
        self.current_dist = self.dist_cal(self.home_posi, current_posi)
        self.point_recoder(self.posi_counter, current_posi, self.current_dist, None, None)
        self.posi_counter += 1
        self.previous_posi = current_posi

    def nextCheck(self, location):  # Get other points before the boat arrives home.
```

```python
    def nextCheck(self, location): # Get other points before the boat arrives home.
        current_posi = location
        self.current_dist = self.dist_cal(self.home_posi, current_posi)
        ang, self.correction_ang = self.posi2ang(self.previous_posi, current_posi)
        self.point_recoder(self.posi_counter, current_posi, self.current_dist, ang, self.correction_ang)
        if self.homeCheck(current_posi) == False:
            self.posi_counter += 1
            self.previous_posi = current_posi
            return True
        else:
            print('Arrived!!')
            return False

    def actionCheck(self):  # Make choice between three action mode
        if self.correction_ang <= 3 and self.correction_ang >= -3:
            return 'forward'
        elif self.correction_ang > 3:
            return 'left'
        elif self.correction_ang < -3:
            return 'right'

    def homeCheck(self, current_posi):  # Stop the process when the boat arrived home
        if abs(current_posi[0]-self.home_posi[0]) <= self.home_size and \
                abs(current_posi[1]-self.home_posi[1]) <= self.home_size:
            return True
        else:
            return False

    def figSave(self):
        self.posiPoint.show('Position-{}'.format(time.time()), True, True)
        self.distPoint.show('Distance-{}'.format(time.time()), True, True)
        self.angPoint.show('Angle-{}'.format(time.time()), True, True)
        self.correctionPoint.show('Correction Angle-{}'.format(time.time()), True, True)
```

參考文獻

[1] iT 邦幫忙〈進擊的巨大污染！解析海漂垃圾的全球分布：專訪中研院鄭明修〉(民國 110 年 5 月 2 日)。<https://reurl.cc/lRvOVA> (15 Jun. 2021)

[2] “Seabin Project - Cleaner Oceans for a Brighter Future,” Seabin Project, 2021, <https://seabinproject.com/> (16 Jun. 2021)

[3] “Introducing the Seabin project,” Australian National Maritime Museum, 8 Jun 2018, <https://reurl.cc/Q94yZZ> (16 Jun. 2021)

[4] “Mr. Trash Wheel,” Wikipedia, 6 May 2021, <https://en.wikipedia.org/wiki/Mr._Trash_Wheel> (16 Jun. 2021)

[5] “MR. TRASH WHEEL®: A PROVEN SOLUTION TO OCEAN PLASTICS,” Mr. Trash Wheel official website, <https://www.mrtrashwheel.com/> (16 Jun. 2021)

[6] “THE INTERCEPTOR,” THE OCEANCLEANUP official website, 2021, <https://theoceancleanup.com/rivers/> (16 Jun. 2021)

[7] C. Su, W. Dongxing, L. Tiansong, R. Weichong and Z. Yachao, “An autonomous ship for cleaning the garbage floating on a lake,” 2009 Second International Conference on Intelligent Computation Technology and Automation, vol. 3, (2009): 471-474.

[8] “WasteShark,” Ranmarine official website, 2016-2021, <https://www.ranmarine.io/products/wasteshark/> (8 Apr. 2021)

[9] Medium-湛・Azure〈【湛況報導】2021 年 7 月號 — 近況分享〉(民國 110 年 8 月 6 日)。< https://reurl.cc/q18x1D > (15 Sep. 2021)

[10] “Getting Started with Jetson Nano Developer Kit,” NVIDIA DEVELOPER, 2021, <https://reurl.cc/MkZZRX> (4 Aug 2021)

[11] NVIDIA 官方網站，〈邊緣運算-JETSON NANO〉。<https://reurl.cc/Q94Ax9> (14 Apr. 2021)

[12] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. B. Foote, J. Leibs, et al., “ROS: An open-source robot operating system,” Proc. ICRA Open-Source Softw. Workshop, (2009).

[13] iT 邦幫忙〈[ROS#2]系統架構及重要觀念介紹〉(民國 108 年 09 月 17 日)。<https://ithelp.ithome.com.tw/articles/10216423> (14 Apr. 2021)

[14] “pySerial API,” pySerial, 2021, <https://pyserial.readthedocs.io/en/latest/pyserial_api.html> (10 Aug. 2021)

[15] “Arduino Uno Rev3,” Arduino, 2021, <https://store.arduino.cc/usa/arduino-uno-rev3> (16 Jun. 2021)

[16] R. Mur-Artal, J. M. M. Montiel and J. D. Tardós, "ORB-SLAM: A Versatile and Accurate Monocular SLAM System," in IEEE Transactions on Robotics, vol. 31, no. 5, (Oct. 2015): 1147-1163

[17] R. Mur-Artal and J. D. Tardós, “ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras,” IEEE Transactions on Robotics, vol. 33, no. 5, (Oct. 2017): 1255-1262.

[18] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. M. Montiel and J. D. Tardós, "ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual–Inertial, and Multimap SLAM," in IEEE Transactions on Robotics, (May 2021): 1-17.

[19] E. Rublee, V. Rabaud, K. Konolige and G. Bradski, “ORB: An efficient alternative to SIFT or SURF,” 2011 International Conference on Computer Vision, (2011): 2564-2571.

[20] “Debian install of ROS Melodic,” ROS Wiki, 23 May 2018, <http://wiki.ros.org/melodic/Installation/Debian> (12 Apr. 2021)

[21] WordPress CH.Tseng 個人頁面，〈初探卷積神經網路〉(民國 106 年 09 月 12 日)。<https://reurl.cc/R0vRRG> (6 May 2021)

[22] A. Krizhevsky, I. Sutskever, and G. E. Hinton. “Imagenet classification with deep convolutional neural networks.” Advances in neural information processing systems 25 (2012): 1097-1105.

[23] K. Simonyan, and A. Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556 (2014).

[24] C. Szegedy , et al. "Going deeper with convolutions." Proceedings of the IEEE conference on computer vision and pattern recognition, (2015): 1-9.

[25] A. G. Howard, et al. “Mobilenets: Efficient convolutional neural networks for mobile vision applications.” arXiv preprint arXiv: 1704.04861 (2017).

[26] “scikit-learn,” Scikit-Learn official website, <https://scikit-learn.org/stable/> (1 Jun. 2021)

[27] “TensorFlow,” TensorFlow official website <https://www.tensorflow.org/> (1 Jun. 2021)

[28] “PyTorch,” PyTorch official website, <https://pytorch.org/> (2 Jun. 2021)

[29] “Catamaran,” Wikipedia, 22 Apr. 2021, <https://en.wikipedia.org/wiki/Catamaran> (5 May 2021)

[30] TW511 教學網，〈超聲波感測器 HC-SR04 和 Arduino 進行距離計算〉(民國 109 年)。<http://www.tw511.com/24/274/9922.html> (14 Jun. 2021)