整合物件辨識與自動駕駛功能的小型智慧載具於模擬道路環境中的應用研究

吳建中
南臺科技大學資訊工程系
wucc@ stust.edu.tw

摘 要

隨著無人駕駛技術的普及，其應用範圍不僅限於道路交通，還可擴展至工廠無人搬運車、果園採果搬運車、校園文件配送車等領域。為了促進相關應用的開發，本論文提出一套基於模仿學習與物件辨識技術的智慧載具系統，設計並實現了一個小型智慧載具的自動駕駛系統，目的在透過影像辨識技術提升載具在虛擬道路環境中的自駕能力。系統硬體採用 Raspberry Pi 4 和 NVIDIA Jetson Nano 平台，並結合 Donkey Car 系統和 Tiny-Yolo V7 物件辨識技術，能夠有效識別道路分隔線、交通燈號、交通標誌及行人車輛等物體，並根據辨識結果做出相應的駕駛決策。為了提升駕駛安全性，系統還在車前劃定了危險區域，當物體進入危險區時，系統會立即啟動緊急煞車動作。除此之外，系統可以根據交通燈號的顏色變換、交通標誌的指示等信息調整車輛行駛狀態。測試結果顯示，該系統能夠準確識別交通燈號、交通標誌，並根據辨識到的物體進行適當的停車或啟動操作，達到基本的自駕行駛需求。本研究成功展示了基於低成本嵌入式硬體平台的自動駕駛解決方案，並為未來進一步拓展至真實交通環境提供了有價值的參考。未來可針對物件識別精度、系統運算效能及多場景應用進行進一步優化。

關鍵詞：模仿學習、自動駕駛、無人駕駛、物件辨識、嵌入式系統

Research on the Application of a Small Intelligent Vehicle Integrating Object Recognition and Autonomous Driving Functions in a Simulated Road Environment

Chien-Chung Wu
Department of Computer Science and Information Engineering, Southern Taiwan University of Science and Technology

Abstract

With the widespread adoption of autonomous driving technology, its application scope extends beyond road traffic to areas such as unmanned transport vehicles in factories, fruit-picking transport vehicles in orchards, and campus document delivery vehicles. To promote the development of related applications, this study proposes an intelligent vehicle system based on imitation learning and object recognition technology. A small intelligent vehicle's autonomous driving system was designed and implemented to improve the vehicle's self-driving capability in a virtual road environment using image recognition technology. The system hardware is built on the Raspberry Pi 4 and NVIDIA Jetson Nano platforms, integrating the Donkey Car system with Tiny-Yolo V7 object recognition technology. It can effectively identify objects such as road dividers, traffic lights, traffic signs, pedestrians, and vehicles, making corresponding driving decisions based on the recognition results. To enhance driving safety, the system defines a danger zone in front of the vehicle. When an object enters the danger zone, the system immediately activates an emergency brake. Furthermore, the system adjusts the vehicle's driving state

Received: Feb. 7, 2025; first revised: Mar. 7, 2025; accepted: Mar. 2025.
Corresponding author: C.-C, Wu, Department of Computer Science and Information Engineering, Southern Taiwan University of Science and Technology, Tainan 710301, Taiwan.

based on traffic light color changes and traffic sign instructions. Test results show that the system can accurately identify traffic lights and traffic signs, perform appropriate stop or start operations based on the recognized objects, and meet the basic self-driving requirements. This research successfully demonstrates an autonomous driving solution based on a low-cost embedded hardware platform, providing a valuable reference for future expansion into real-world traffic environments. Future work can further optimize object recognition accuracy, system computational performance, and multi-scenario applications.

Keywords: Imitation learning, Autonomous driving, Autonomous driving, Object recognition, Embedded systems

壹、前言

一、研究目的與背景

（一）研究目的

由於交通事故發生率居高不下，全球各國政府持續努力尋求降低事故的方法。目前，專家預測，透過自動駕駛系統或無人駕駛技術輔助駕駛，約可避免 70% 的交通事故[1]。

無人駕駛技術最早應用於飛機、自動化地鐵及工廠的無人搬運車。然而，若要使其能夠應對複雜的城市道路環境，則須回溯至 2004 年美國莫哈韋沙漠舉辦的 DARPA Grand Challenge[2]，這場比賽開啟了無人駕駛技術的研發競賽。透過高額獎金的激勵，美國各大學與研究機構投入大量人力進行研發。首屆競賽無人能成功完賽，但到了第二屆，多支隊伍成功完成比賽。隨著比賽演進至城市自駕競賽，軟硬體技術不斷進步，最終促成了 Google Driverless Car 以及後來 Tesla 自動駕駛商用車的誕生。

（二）研究背景

無人自動駕駛車輛具備幾個關鍵元素，包括：

1. 感測技術：透過激光雷達、攝影機、超聲波感測器等裝置，車輛能夠捕捉周圍環境的資訊。

2. 資料處理：感測數據需經過高效處理與分析，通常依賴深度學習與計算型智慧技術。

3. 決策與控制：系統需根據感測資訊做出安全的行車決策，例如避開障礙物、遵守交通規則等。基於上述架構設計的自動駕駛車輛，利用先進的感測技術與控制演算法[3]–[4]，能夠在複雜環境中安全導航。然而，面對多變的道路型態，若需調整系統以適應不同環境，則需針對各個元件單獨修改。由於系統內部存在複雜的相互依賴關係，這使得擴充與修正過程極具挑戰性[5]。

4. 學習能力：無人駕駛系統可透過模仿學習（Imitation Learning, IL）及深度強化學習（Deep Reinforcement Learning, DRL）進行自我改進，以提升安全性與效能。隨著晶片技術的進步，例如 NVIDIA GPU 運算能力的提升，使得人工智慧與深度學習技術的應用效率大幅增強，加速了自動駕駛技術的發展。在眾多學習方法中，模仿學習 是一種重要技術，其核心概念是透過觀察專家的示範來學習控制策略[6]–[10]。此方法的一大優勢是可採用 端到端深度學習（End-to-End Learning），透過最終目標對模型的所有參數進行共同優化，從而降低各個元件的調整負擔。

最著名的模仿學習案例之一來自 NVIDIA 工程師團隊。他們在車輛前方左側、中間、右側分別架設三組攝影機，並透過專門的硬體記錄方向盤轉動角度、煞車、油門踩踏訊號及車輛行駛速度。透過專家駕駛示範並收集其駕駛數據，進行模型訓練。訓練完成後，系統依然使用 三組攝影機，並搭配 NVIDIA DRIVE™ PX 硬體與線控裝置（Drive-by-Wire Interface） [11]，根據當前車輛前方影像與行車速度資訊，模仿人類駕駛方式進行 方向盤操控、油門與煞車調整，最終實現自動駕駛車輛的運作。

然而，模仿學習技術在自動駕駛中的應用存在一定限制，主要由於模型的行為完全來自專家示範，對於專家未曾教導的情境，系統無法自主推理或舉一反三，可能導致無法預測的結果。此外，因為學習方式來自模仿，系統的最佳狀態僅能等同於專家水準，難以超越，也難以適應不同道路環境的變化。

在端到端學習（End-to-End Learning）的自動駕駛策略中，另一種方法是深度強化學習（Deep

Reinforcement Learning, DRL）。DRL 透過 探索-開發（Exploration-Exploitation）與試錯機制（Trial-and-Error Mechanisms）來建構學習模型[12]–[13]，能夠自主搜尋最佳控制策略並持續優化。然而，進行強化學習訓練時，系統需要一個 代理人（Agent）能夠與環境互動，透過不斷嘗試不同行動，利用獎勵機制（Reward）的回饋來尋找最優解[14]–[16]。

近年來，深度學習 的發展推動了卷積神經網路（Convolutional Neural Networks, CNNs）在電腦視覺領域的廣泛應用，特別是在圖像分類任務中，ResNet[17]被視為最優秀的方法之一。然而，圖像分類的骨幹架構正在經歷變革，逐漸從 CNNs 轉向 Transformers[18]。

Dosovitskiy 等人[19]提出了 Vision Transformer （ViT） 模型，成功將標準 Transformer 架構應用於圖像分類。然而，研究發現，直接將圖像輸入 Transformer 會帶來幾個挑戰，包括：

1. 計算資源消耗過大，導致訓練成本提高；
2. 缺乏局部特徵建模能力，影響圖像細節的學習；
3. 學習效率低，需要大量數據才能達到優秀表現；
4. 信息冗餘問題，增加模型處理負擔。

隨著多模態 Transformer 應用的發展，Prakash 等人[20]將其引入端到端自動駕駛，結合多種感知數據（如 攝影機、光達（LiDAR）等）來提升自動駕駛模型的感知與決策能力。他們利用 ResNet[17]提取影像特徵，並與 Transformer 架構融合，最終透過多層感知器（Multi-Layer Perceptron, MLP）進行多模態數據整合，使模型能夠更準確地理解環境並做出駕駛決策。

本研究目標是透過嵌入式系統，以較經濟的方式建構一個具備物件辨識與自動駕駛功能的小型智慧載具。因此，初期實作階段將不考慮採用 Vision Transformer 模型。

貳、研究方法

為了驗證小型智慧載具能夠根據 交通燈號、交通號誌 及 道路狀況 進行自動駕駛，同時考量嵌入式系統的運算能力，本研究建構了一套小型智慧載具系統，包含 車輛、行人及道路號誌的模擬環境，並結合模仿學習建構自動駕駛技術，並搭配深度學習透過影像建立物件識別系統進行驗證。

系統架構方面，自駕功能的核心控制採用 Raspberry Pi 4 嵌入式開發板，透過攝影機取得前方道路影像，並根據辨識結果預測車速、轉向並進行決策控制。物件識別系統則採用 NVIDIA Jetson Nano，負責分析前方影像，識別行人、車輛、交通燈號與交通號誌等物件。當偵測到行人時，系統將自動啟動緊急煞車以避免事故，並依據識別出的交通號誌與交通燈號遵循交通規則進行駕駛。

本系統主要由三個子系統組成，包括：（a）自動駕駛系統、（b）物件識別系統、（c）行車決策系統。

系統採用兩個攝影機進行影像擷取：

攝影機 A 負責拍攝前方道路影像，並輸入 Keras Linear 神經網路進行運算，網路輸出方向盤轉向角度與油門控制。攝影機 B 瞄準車輛正前方，獲取影像後，使用訓練過的 Tiny-Yolo V7 神經網路進行物件分類，識別人、車輛、交通燈號、交通號誌及其他物件（共五類）。

此外，針對交通號誌種類 與交通燈號狀態，系統額外採用機器視覺技術進行判斷，並將結果輸入行車決策系統，綜合分析後輸出方向盤轉向角度與油門，實現自動駕駛控制，系統架構如圖 1 所示。

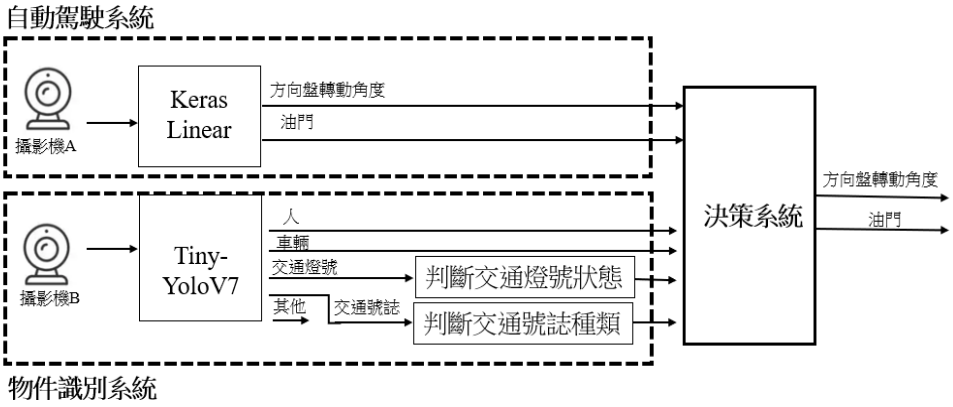


圖 1 *系統方塊圖*

一、自動駕駛系統

自動駕駛系統 採用模仿學習（Imitation Learning）架構，透過專家示範來訓練模型。如圖 2 所示，專家可透過目視或觀看螢幕中載具回傳的前方畫面來監控車輛行動的方向與前方視野，並使用 XBOX Racing Wheel and Pedals 來操控車輛，控制方向盤進行轉向、油門與煞車踏板調整車輛行走方向與車速。

在訓練過程中，專家駕駛車輛執行以下動作，以確保模型能夠學習正確的駕駛行為：

（一）保持車輛行駛在車道中央，確保穩定駕駛。

（二）正確操控彎道轉向，使車輛順利過彎。

（三）及時修正車輛偏移，避免偏離車道。

車輛在行駛過程中，會記錄與駕駛操作對應的行車資訊，包括：

（一）方向盤轉向角度。

（二）車輛行駛油門的大小。

（三）前方攝影機拍攝的道路影像。

此外，為提升模型的學習能力，訓練時會讓在地圖中依序採用順時針與逆時針方向進行訓練，透過重複上述操作，以獲取更多駕駛數據，使模型學習更加全面。

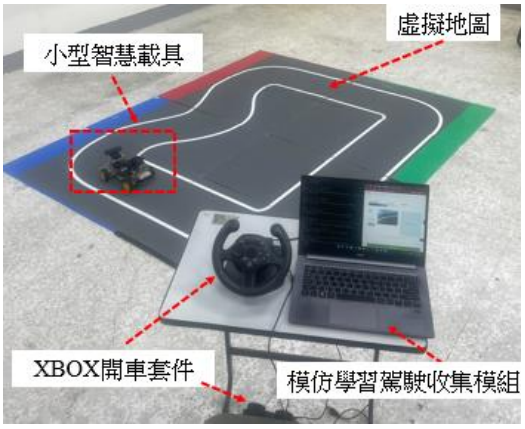


圖 2 *透過模仿學習方式由專家示範來訓練的系統架設照片*

以圖 2 為例，虛擬地圖中直線路段的比例較高，這可能導致自駕車在訓練過程中無法獲取足夠的轉彎操作數據，進而影響轉向能力的學習效果。為了確保模型能夠正確學習彎道駕駛行為，需要進一步對收集到的數據進行 標記與數據平衡（Data Balancing），使不同駕駛情境的數據分布更均勻。

數據處理完成後，即可進行神經網路訓練，並透過驗證與測試來評估模型的準確性與穩定性。整個模仿學習自動駕駛系統的訓練與驗證流程 如圖 3 所示。

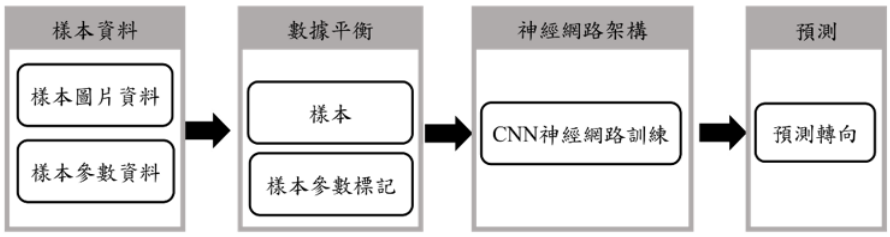


圖 3 *模仿學習自動駕駛系統訓練與驗證流程*

二、訓練資料處理與數據優化

數據處理是人工智慧成功的關鍵，尤其在自駕車的訓練過程中更是不可或缺。以下將詳細說明訓練資料的處理方法，以確保模型的準確性與穩定性。在專家進行示範操作後，系統會逐幀記錄車輛行駛畫面。然而，若直接使用專家駕駛收集到的資料進行訓練，模型通常無法正常運作，主要問題來自以下幾個方面：

（一）數據分佈不均衡

例如，在圖 2 所示的場景中，直線行駛的樣本數量遠多於左轉或右轉的樣本。如果以這種比例訓練模型，轉彎資料過少可能導致模型無法適當收斂。因此，在訓練前須進行數據平衡處理。本研究根據轉向角度範圍，將樣本分為直行、左偏、左轉、右偏、右轉，並確保各類樣本的數量接近，同時移除過多的樣本。

（二）操作失誤的樣本

即使是專家操作，也可能出現誤差，例如駛出車道線或車輛行駛不穩定。為了過濾這些異常樣本，可利用圖 4 所示的訓練資料後處理工具 Donkey UI 進行逐幀檢查並刪除不正確的數據。由於 Donkey UI 原生功能未提供正確刪除樣本的處理方式，本研究在其基礎上進行修正與擴展，使系統能夠根據使用者需求進行逐幀檢查，或透過條件設定進行批次刪除。

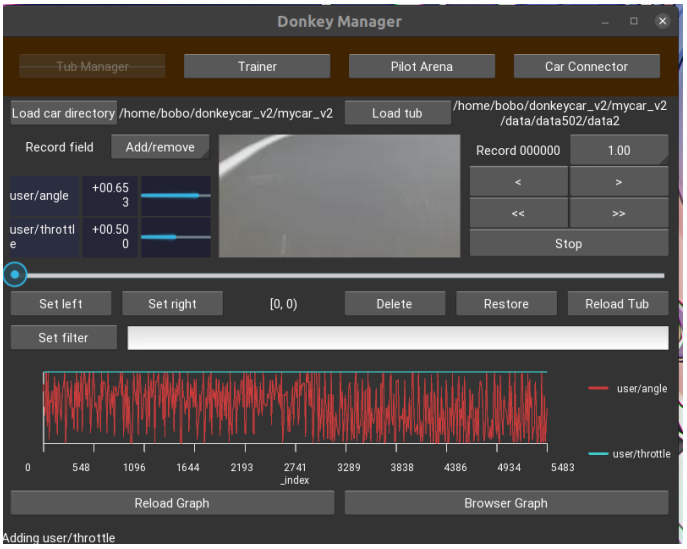


圖 4 *訓練資料後處理 Donkey UI 工具畫面*

（三）容錯能力提升

為了確保模型能夠適應不同光線與環境條件，容錯能力是關鍵訓練項目之一。除了確保車輛能夠保持在車道中央，本研究在資料收集階段，特意讓車輛偏移車道線並立即修正，以強化模型對誤差的適應能力，提升其容錯效果。

（四）速度控制優化

在專家駕駛收集數據時，為避免車輛駛出邊界，通常不會進行明顯的加減速。若直接使用這些資料訓練模型，可能導致自駕車以固定速度行駛，無法根據道路狀況靈活調整。因此，本研究設定不同道路

條件下的速度控制策略，例如：在直線道路上，模型會自動提升車速；在彎道或轉彎區域，模型會自動減速並準備轉向。為了達成此目標，透過修正 Donkey UI 設定將直行樣本中的車速提高，而彎道樣本則維持原速，以確保車輛能夠適應不同路況。

（五）車輛行駛方向調整

為進一步提升模型的泛化能力，訓練資料的收集過程會讓車輛分別以順時針與逆時針方向重複上述的數據處理方法，確保模型能夠適應不同行駛方向的變化。

綜合以上方法，本研究透過數據平衡、錯誤樣本刪除、容錯能力提升、速度控制優化及行駛方向調整，以確保訓練資料的品質，進而提升自駕模型的穩定性與準確度。

如圖 5 所示，自駕系統主要由小型智慧載具與資料收集與訓練主機兩部分組成。

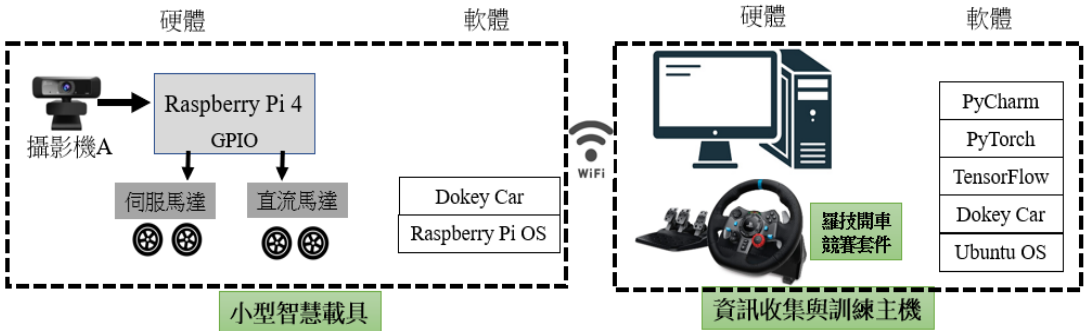


圖 5 *自駕系統系統架構圖*

1. **小型智慧載具：**以 Raspberry Pi 4 為核心，並運行 Raspberry Pi OS。在此硬體上安裝 Donkey Car 4.22 系統，以支援自駕控制與資料收集功能。
2. **資料收集與訓練主機：**採用 NVIDIA GeForce GTX 1080 Ti 顯示卡與 32GB 記憶體的電腦，搭載 Ubuntu 20.04 LTS 作業系統。在此環境下，使用 PyCharm 進行 Donkey Car 的開發，並安裝 TensorFlow 與 PyTorch 來執行模型訓練。

在完成數據收集與前述的數據處理後，模型的訓練工作於資料收集與訓練主機上進行。訓練完成後，將模型與權重檔案透過 WiFi 網路下載至小型智慧載具進行測試與驗證，以確保模型的準確性與穩定性。

三、物件識別系統

經過自動駕駛系統的資料收集、資料後處理與訓練後，小型智慧載具可以透過鏡頭取得的畫面透過神經網路的判斷後，控制載具的轉向與油門，讓載具可以根據道路分隔線進行自動駕駛。然而，單靠這種方式進行自駕模式無法識別交通燈號、交通號誌，也無法偵測行人或其他車輛，因此本系統設計了一套物件識別系統，透過影像辨識來提升自駕能力。

如圖 6 所示，物件辨識系統以 Tiny-YOLO V7 為核心，經過重新收集樣本與調整輸出類別後，目前輸出的物件類別包括：(i) 交通燈號，(ii) 交通號誌，(iii) 行人，(iv) 車輛，(v) 其他物件等五個類別。

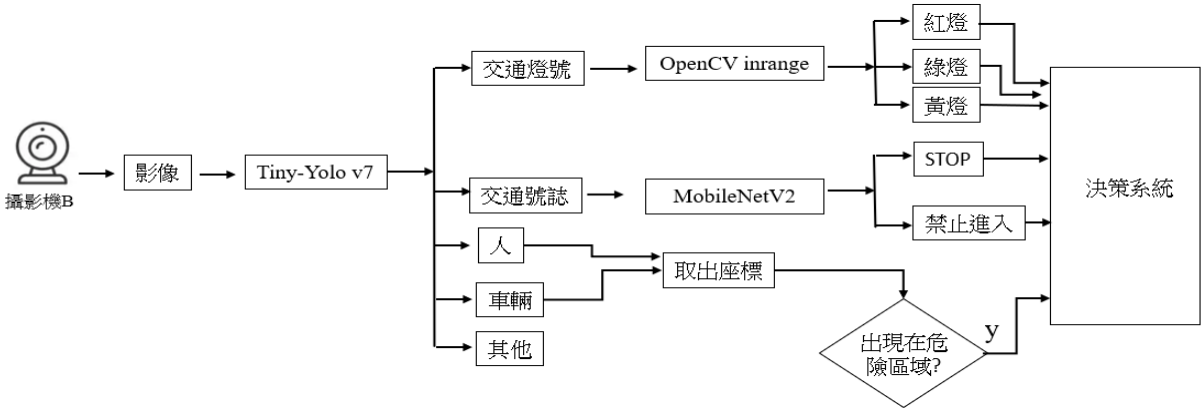


圖 6 *物件識別系統架構圖*

當辨識影像中出現交通燈號時，系統將透過 OpenCV 擷取燈號影像，並使用 inRange（）方法進行燈號顏色解析，以識別紅燈、黃燈或綠燈狀態。

除此之外，若影像中包含交通號誌，系統同樣會透過 OpenCV 擷取號誌影像，並利用 MobileNet V2 進一步識別。本研究主要以 "Stop" 停止標誌 與 "禁止進入" 標誌 作為範例進行辨識與測試。

為了避免小型智慧載具與前方的 行人或車輛發生碰撞，本系統在車輛前方定義了一塊危險區域（如圖 7 紅色區域所示）。當物件識別系統偵測到行人或車輛，並確認其位置位於危險區域內時，決策系統將立即觸發緊急煞車機制，將油門值設為 0，確保小型智慧載具能夠即時停車，避免碰撞事故的發生。

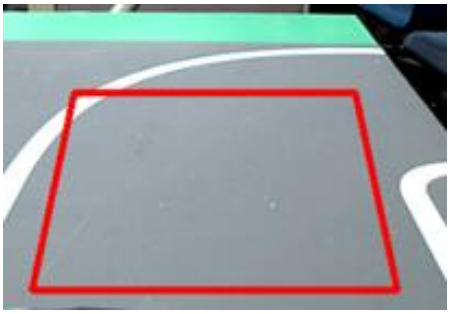


圖 7 *物件識別系統前方危險區域*

如**圖 8** 所示，物件識別系統主要由物件辨識系統與訓練主機兩部分組成。

（一）物件辨識系統：以 NVIDIA Jetson Nano 為核心，運行 Ubuntu Linux OS，並安裝以 Darknet 為核心的 Tiny-YOLO 神經網路，同時搭配 CUDA 與 OpenCV 套件，以提升影像處理與推論效能。

（二）訓練主機:採用 NVIDIA GeForce GTX 1080 Ti 顯示卡與 32GB 記憶體的電腦，運行 Ubuntu 20.04 LTS 作業系統，同樣以 Darknet 為核心，進行 Tiny-YOLO 神經網路的開發與訓練。

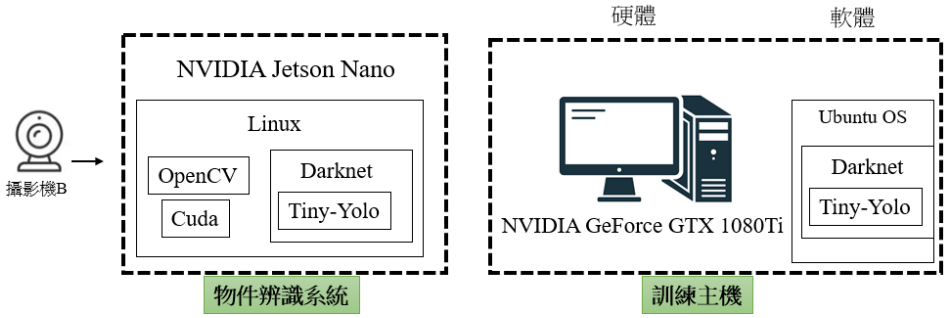


圖 8 *物件辨識系統架構圖*

四、訓練流程

（一）資料集準備

以 Tiny-YOLO 資料集為基礎，額外加入交通燈號與交通號誌的影像資料，使用 LabelImg 工具進行標註與微調，並調整輸出類別。

（二）模型訓練與部署

在訓練主機上執行 Tiny-YOLO 訓練，完成模型訓練與最佳化。訓練完成後，將權重檔與神經網路模型透過 USB 隨身碟傳輸至 NVIDIA Jetson Nano，進行測試與驗證，以確保模型識別的準確性與穩定性。

五、決策系統

如圖 9 所示，決策系統的運作流程如下：

決策系統負責整合 自動駕駛系統與物件識別系統的輸出資訊，根據優先權進行判斷，並決定小型智慧載具的行駛狀態。

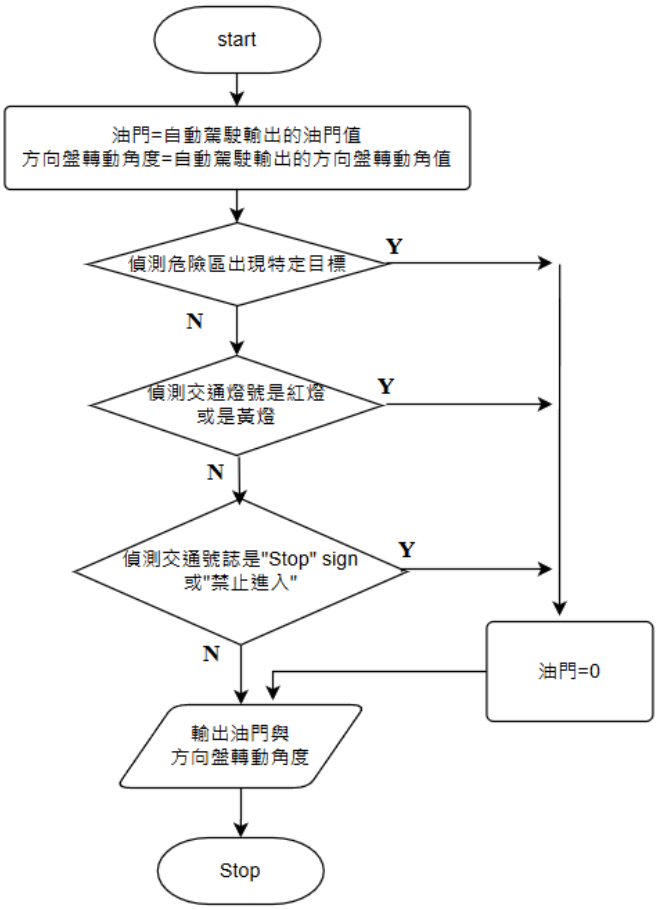


圖 9 *決策系統判斷的流程圖*

六、決策優先權規則

主要分成特殊駕駛狀態與正常駕駛狀態兩種。

當下列任一條件被滿足時，系統會進入特殊駕駛狀態，將油門輸出調整為 0，讓小型智慧載具停止前進：

（一）危險區域內偵測到行人或車輛。

（二）交通燈號顯示紅燈或黃燈。

（三）交通號誌為"Stop" 停止標誌或"禁止進入"標誌。

若上述條件皆不滿足，系統將恢復正常駕駛狀態，將根據自駕系統輸出的油門值與方向盤轉動角度，透過 Raspberry Pi 4 的 3 組 GPIO 接腳傳送適當的 PWM 訊號控制轉向伺服馬達與 RC-motor 讓小型智慧載具進行自動駕駛。

決策系統運作於無窮迴圈（infinite loop）中，會持續即時監測自駕系統與物件辨識系統的輸出資訊，並即時送出正確的控制訊號，確保小型智慧載具的安全與穩定運行。

參、研究結果

一、系統測試與部件驗證

在進行系統整合測試之前，將針對各個核心部件進行獨立測試，最後再進行整合測試。測試程序如下所述。

（一）自動駕駛系統測試

如**圖 10** 所示，本系統的小型智慧載具實體結構如下：

1. 核心硬體整合:

Raspberry Pi 4 安裝於車輛下方，負責自動駕駛系統的運算與控制。

攝影機 A 安裝在車輛最前方，朝向前下方，主要用於偵測行駛道路資訊。

物件識別系統 由 NVIDIA Jetson Nano 與攝影機 B 組成。

攝影機 B 角度不同於攝影機 A，安裝位置較高、平視前方，可提早偵測 交通燈號、交通號誌、行人與車輛。

2. 驅動控制機制:

Raspberry Pi 4 透過 兩個 GPIO 接腳 連接 L298N 馬達驅動模組，並輸出兩組 PWM 訊號控制 RC-motor 的轉速與方向，其中馬達控制邏輯:

當 PWM1 為正值，PWM2 為 0→RC 馬達正轉（載具往前方移動），PWM 值越大，轉速越快。

當 PWM1 為 0，PWM2 為正值→RC 馬達反轉（載具往後方移動）。

當 PWM1、PWM2 均為 0→載具停止運行。

3. 方向控制機制:

Raspberry Pi 4 透過第三組 GPIO PWM 控制伺服馬達，調整車輛的轉向角度。

伺服馬達負責轉向輪的角度調整，確保載具能夠正確轉向。

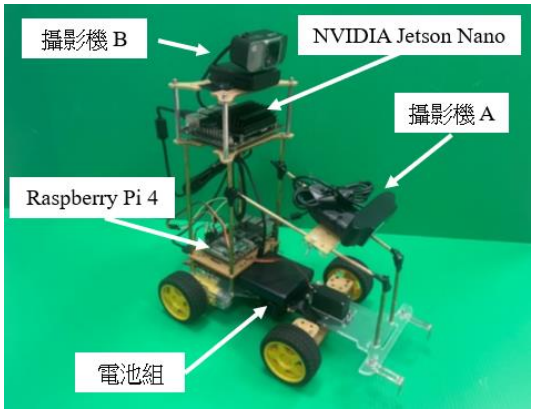


圖 10 *小型智慧載具實體照片*

二、物件識別系統

本系統在物件識別系統的實驗環境中，架設了一個 ESP32 控制的模擬交通燈號，能夠自動模擬一般路口的綠燈、黃燈、紅燈 切換行為。此外，在測試環境中，會不定期放置"STOP"交通標誌立牌或人形立牌，進行靜態測試，以驗證系統的辨識能力與反應機制。

（一）測試場景與結果

1. 綠燈狀態測試（圖 11 (a)）

測試時，ESP32 交通燈號切換至綠燈，並安裝於載具前方。

執行時，系統螢幕顯示 偵測到的燈號 以及 估算的距離，以確保辨識結果正確。

2. 紅燈 + "STOP" 標誌測試（圖 11 (b)）

當交通燈號切換至紅燈時，在測試環境中額外放置 "STOP"交通標誌。

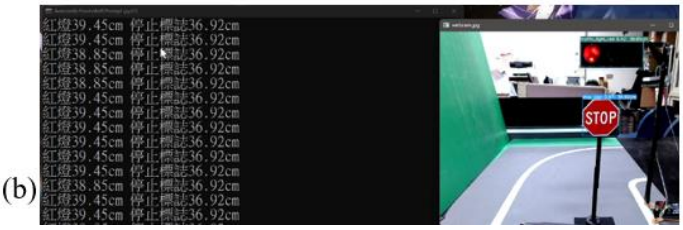
系統成功偵測到 紅燈 與 "STOP" 標誌，並在螢幕左側顯示辨識資訊與估算距離。

3. 紅燈 + "STOP" 標誌 + 行人測試（圖 11 (c)）

在交通燈號切換至紅燈時，於危險區域內放置人形立牌，並在後方放置 "STOP" 交通標誌。

系統成功偵測 紅燈、"STOP" 標誌，並辨識到 行人進入危險區域，同時在螢幕左側顯示警告資訊與估計距離。

(a)交通燈號綠燈測試



(b) 交通燈號紅燈與交通號誌"Stop"Sign 搭配測試

(c)人出現在危險區域與交通燈號紅燈與交通號誌"Stop"Sign 搭配測試

圖 11 *物件識別系統靜態測試*

三、整合測試

在整合測試環境中，首先實驗中分別在**圖 12** 標記 A、B、C、D 安置交通燈號、交通號誌。除此之外，分別在**圖 12** 標記 1、2、3、4 安置人形立牌。

其中交通燈號是指設置一個 ESP32 控制的交通燈號，能夠模擬一般路口燈號，並根據預設時間自動切換綠燈、黃燈、紅燈。交通號誌是指小型 "STOP"交通標誌，而 人形立牌 則為印有人形圖像的立牌。

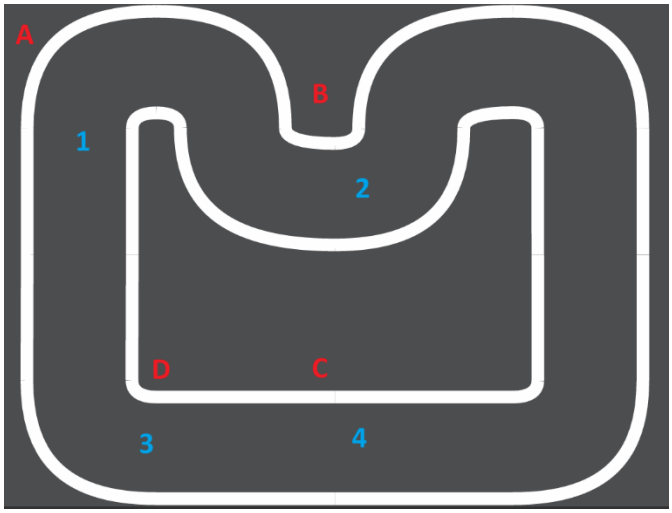


圖 12 *實驗設置位置*

實驗方式

本實驗在不同位置進行測試，每次測試均進行順時針與逆時針各 2 圈的行駛。

實驗一：基本測試

在此測試中，於座標 A、C 放置交通燈號與交通號誌，並於座標 1 至 4 放置人形立牌，使自駕車進行測試。測試結果如表 1 所示，自駕車能夠依照交通燈號與號誌行駛，並在遇到人形立牌時即時煞車。

表 1 *實驗一結果*

座標 A	座標 B	座標 C	座標 D	座標 1	座標 2	座標 3	座標 4	結果
交通燈號		交通號誌						正確
交通燈號		交通號誌		人形立牌				正確
交通燈號		交通號誌			人形立牌			正確
交通燈號		交通號誌				人形立牌		正確
交通燈號		交通號誌					人形立牌	正確

實驗二：交通燈號與號誌下的人形立牌測試

本實驗測試當交通燈號與交通號誌前後同時出現人形立牌時，自駕車的反應狀況。具體設置如下：

在人形立牌的測試中，我們將其放置於交通燈號與交通號誌的前後位置，並透過順時針與逆時針方向的行駛進行測試，以評估不同放置位置對自駕車行為的影響。

測試結果如表 2 所示，自駕車均能依照交通燈號與號誌行駛，並在遇到人形立牌時即時做出正確反應，結果符合測試規範。

表 2 *實驗二結果*

座標 A	座標 B	座標 C	座標 D	座標 1	座標 2	座標 3	座標 4	結果
交通燈號		交通號誌		人形立牌				正確
交通燈號		交通號誌					人形立牌	正確
交通燈號		交通號誌			人形立牌			正確
交通號誌		交通燈號		人形立牌				正確
交通號誌		交通燈號					人形立牌	正確
交通號誌		交通燈號				人形立牌		正確

實驗三：轉彎處人形立牌測試

本實驗的主要目的是測試自駕車在轉彎前後遇到人形立牌時的反應情況。我們在轉彎處的座標 A 與座標 B 分別設置交通燈號與交通號誌，並於轉彎處前後（順時針與逆時針方向）放置人形立牌進行測試。測試結果如表 3 所示，在 S 形轉彎處，若交通燈號為變化狀態，當人形立牌出現時，自駕車可能來不及停下，或偏離車道線。然而，在後續實驗中，當我們將行駛速度減半，上述錯誤情況得以改善，車輛能夠正確應對人形立牌的出現。

表 3 *實驗三結果*

座標 A	座標 B	座標 C	座標 D	座標 1	座標 2	座標 3	座標 4	結果
交通號誌	交通燈號			人形立牌				正確
交通號誌	交通燈號				人形立牌			錯誤
交通燈號	交通號誌			人形立牌				正確
交通燈號	交通號誌				人形立牌			正確

實驗四：不同路段交通燈號與號誌對人形立牌識別的影響

本實驗主要測試自駕車在直線與轉彎路段，面對不同交通燈號與交通號誌時，人形立牌對系統判斷與行駛的影響。我們在轉彎處（座標 D）與直線路段（座標 C）分別設置交通燈號與交通號誌，並於轉彎處前後（順時針與逆時針方向）放置人形立牌進行測試。

實驗結果顯示，如果行駛方向先遇到轉彎處（座標 D）的交通燈號，若交通燈號處於變化狀態，當人形立牌出現在座標 3 時，自駕車有時無法及時停下。

然而，當我們降低行駛速度進行測試後，錯誤情況得以解決，車輛能夠正確應對人形立牌的出現。

表 4 *實驗四結果*

座標 A	座標 B	座標 C	座標 D	座標 1	座標 2	座標 3	座標 4	結果
		交通燈號	交通號誌			人形立牌		**正確**
		交通燈號	交通號誌				人形立牌	**正確**
		交通號誌	交通燈號			人形立牌		**錯誤**
		交通號誌	交通燈號				人形立牌	**正確**

實驗五:整合隨機測試與展示

在整合測試環境中，我們設置了一個 ESP32 控制的交通燈號，能夠模擬一般路口燈號，並根據預設時間自動切換綠燈、黃燈、紅燈。每次測試時，交通燈號會隨機放置在虛擬地圖道路的不同位置，增加測試的隨機性與真實性。

此外，為了進行動態測試，系統會在不同時間點隨機放置小型 "STOP"交通標誌或人形立牌於虛擬地圖的不同位置，模擬行駛過程中的各種可能情境。同時，每次測試時，小型智慧載具的初始位置也會隨機設定，並透過 遠端啟動 進行測試。

這個實驗，為了讓測試更明確，在測試時另外開啟幾個視窗呈現測試結果，主要多出自駕系統攝影機 A 的監控畫面、自駕系統的決策結果與物件識別系統識別結。圖 13 為小型智慧載具在模擬道路環境中測試的影片截圖，畫面中包含以下資訊：

（一）左下角：顯示 自駕系統的前方攝影機（鏡頭 A） 所拍攝的畫面。

（二）左上角：顯示 自駕系統的油門速度，即載具的行駛狀態。

（三）右側畫面：展示虛擬地圖與小型智慧載具的即時測試畫面。

（四）右下角：顯示物件識別系統的判斷結果，包含交通燈號、交通標誌及行人偵測資訊。

四、載具自駕反應與決策邏輯

（一）交通燈號識別與反應

當遇到紅燈或黃燈 → 小型智慧載具 自動停車。

當綠燈亮起 → 載具 自動起步並通過 路口。

當載具因紅燈停車後 → 綠燈出現時會 自動恢復行駛，確保符合交通規則。

（二）交通標誌 "STOP" Sign 的識別與反應

當載具偵測到 "STOP"交通標誌 → 會先停止，然後再起步，確保模擬駕駛遵循交通規範。

（三）行人辨識與應對

當小型智慧載具 前方出現人形立牌（模擬行人）時，物件識別系統會即時判斷並 將油門降為 0，使載具緊急煞車，避免碰撞。

由**圖 13** 由左下角自駕系統攝影機畫面可見，雖然部分行人立牌的影像出現在視野中，但實際決策是由物件識別系統負責處理，並在右下角顯示相應的識別資訊。

肆、結論

本研究成功設計並實現了一套小型智慧載具的自動駕駛系統，該系統整合 Raspberry Pi 4、NVIDIA Jetson Nano、Donkey Car 平台與 Tiny-Yolo V7 物件辨識技術，能夠在製作的模擬車道中進行自動駕駛，並透過影像辨識技術識別 交通燈號、交通標誌、行人與車輛，進而做出相應的決策。

透過測試結果可知，本系統能夠：

一、 基於道路分隔線 進行自動駕駛。

二、 準確識別交通燈號，並在 紅燈與黃燈時停車，綠燈時通行。

三、 識別 "STOP" Sign 或 "禁止進入" 交通標誌，並執行相應的停止決策。

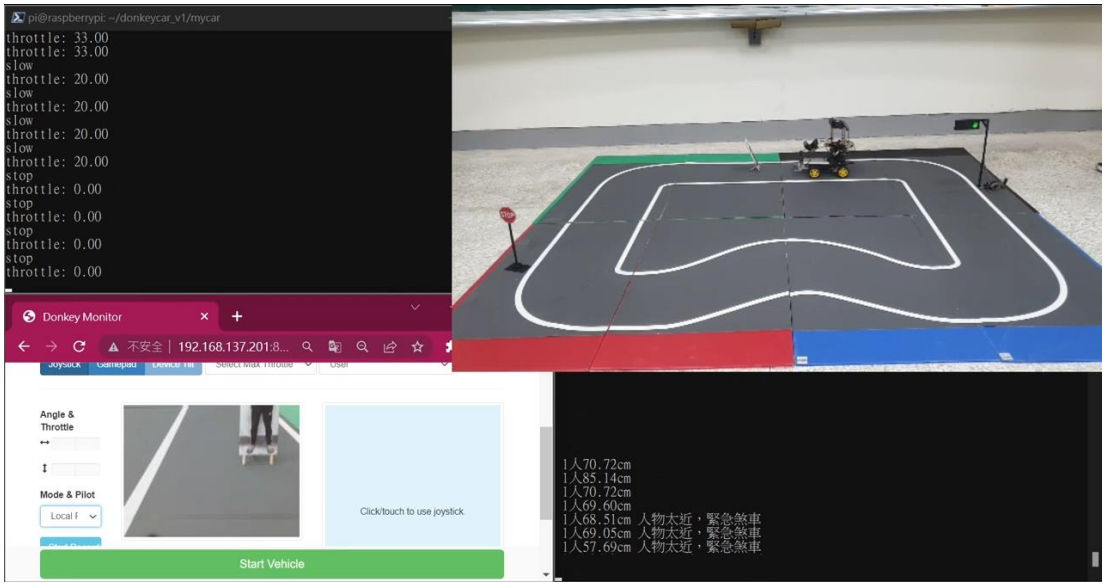


圖 13 *小型智慧載具於模擬道路環境中測試影片的截圖*

四、 偵測行人與車輛，當物件出現在 危險區域 時，自動進行緊急煞車，確保行駛安全。

整體而言，本系統透過深度學習物件辨識技術提升了自駕載具的決策能力，使其能夠更符合現實交通環境的需求，並驗證了以 Raspberry Pi 4 與 NVIDIA Jetson Nano 為核心的低成本自駕車系統的可行性，實驗中也發現，在轉彎幅度較大的路段，必須透過放慢行車速度才可以及時處理突發狀況。

然而，本系統仍有以下幾點限制與未來發展方向：

一、 物件識別的準確度仍有提升空間

目前主要基於 Tiny-Yolo V7 進行物件辨識，未來可考慮採用 更高效能的模型來提升準確度。

二、 測試環境仍以模擬環境為主

本研究主要在 製作的模擬車道中 進行測試，未來可將系統部署到 實際場景，如開放式場地或小型道路，驗證其適用性。

三、 可加入更多環境因素的考量

目前系統尚未考量 天候變化（如雨天、夜間光線變化） 及 複雜交通情境（如多車道、多行人場景），未來可進一步擴展測試範圍。

綜合而言，本研究採用低成本嵌入式平台，透過模仿學習與深度學習技術成功開發了一款能夠執行基本自駕功能的智慧載具，未來可進一步提升辨識精度、優化決策邏輯，並拓展至更真實的交通場景，以提升其應用價值。

參考文獻

[1] National Highway Traffic Safety Administration (NHTSA). (n.d.). *Road safety topics*. https://www.nsc.org/road-safety/safety-topics

[2] Behringer, R., Gregory, B., Sundareswaran, V., Addison, R., Elsley, R., Guthmiller, W., & deMarchi, J. (2004, June 14–17). The DARPA grand challenge-Development of an autonomous vehicle. *Proceedings of the IEEE Intelligent Vehicles Symposium*, Parma, Italy.

[3] Montemerlo, M., Becker, J., Bhat, S., Dahlkamp, H., Dolgov, D., Ettinger, S., Haehnel, D., Hilden, T., Hoffmann, G., Huhnke, B., Johnston, D., Klumpp, S., Langer, D., Levandowski, A., Levinson, J., Marcil,

J., Orenstein, D., Paefgen, J., Penny, I., ... Thrun, S. (2008). Junior: The Stanford entry in the urban challenge. *J. Field Robot., 25*(9), 569–597.

[4] Levinson, J., Askeland, J., Becker, J., Dolson, J., Held, D., Kammel, S., Kolter, J. Z., Langer, D., Pink, O., Pratt, V., Sokolsky, M., Stanek, G., Stavens, D., Teichman, A., Werling, M., & Thrun, S. (2011). Towards fully autonomous driving: Systems and algorithms. *IEEE Intell. Veh. Symp.*, 163–168.

[5] Kiran, B. R., Sobh, I., Talpaert, V., Mannion, P., Sallab, A. A. A., Yogamani, S., & Pérez, P. (2020). Deep reinforcement learning for autonomous vehicles: A survey. *IEEE Trans. Intell. Veh., 5*(4), 691–715.

[6] Cai, P., Sun, Y., Chen, Y., & Liu, M. (2019). Vision-based trajectory planning via imitation learning for autonomous vehicles. *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*.

[7] Zhang, J., & Cho, K. (2017). Query-efficient imitation learning for end-to-end simulated driving. *Proceedings of the AAAI Conf. Artif. Intell.*, 2891–2897.

[8] Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., & Meger, D. (2017). Deep reinforcement learning that matters. *arXiv preprint arXiv:1709.06560*.

[9] Codevilla, F., Muller, M., Dosovitskiy, A., López, A., & Koltun, V. (2017). End-to-end driving via conditional imitation learning. *arXiv preprint arXiv:1710.02410*.

[10] Liang, X., Wang, T., Yang, L., & Xing, E. P. (2018). CIRL: Controllable imitative reinforcement learning for vision-based self-driving. *arXiv preprint arXiv:1807.03776*.

[11] Bojarski, M., Testa, D. D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L. D., Monfort, M., Muller, U., Zhang, J., Zhang, X., Zhao, J., & Zieba, K. (2016). End-to-end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*.

[12] Codevilla, F., Santana, E., López, A. M., & Gaidon, A. (2019). Exploring the limitations of behavior cloning for autonomous driving. *Proceedings of the IEEE/CVF International Conference on Computer Vision*.

[13] Cota, J.L., Rodríguez, J.A.T., Alonso, B.G., & Hurtado, C.V. (2022, March 28–31). Roadmap for development of skills in artificial intelligence by means of a reinforcement learning model using a DeepRacer autonomous vehicle. *Proceedings of the 2022 IEEE Global Engineering Education Conference (EDUCON)*, Tunis, Tunisia.

[14] Óscar, P.G., Rafael, B., Elena, L.G., Luis, M.B., Carlos, G.H., Rodrigo, G., & Alejandro, D.D. (2022). Deep reinforcement learning-based control for autonomous vehicles in CARLA. *Multimed. Tools Appl., 81*, 3553–3576.

[15] Terapaptommakol, W., Phaoharuhansa, D., Koowattanasuchat, P., & Rajruangrabin, J. (2022). Design of obstacle avoidance for autonomous vehicles using Deep Q-Network and CARLA simulator. *World Electr. Veh. J., 13*, 239.

[16] Zhu, Y., Mottaghi, R., Kolve, E., Lim, J.J., Gupta, A., Fei-Fei, L., & Farhadi, A. (2017). Target-driven visual navigation in indoor scenes using deep reinforcement learning. *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 3357–3364.

[17] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–778.

[18] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS), 30*, 5998–6008.

[19] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., & Houlsby, N. (2021). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

[20] Prakash, A., Chitta, K., & Geiger, A. (2021). Multi-modal fusion transformer for end-to-end autonomous driving. *Proceedings of the 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 7073–7083.