LWOPAN

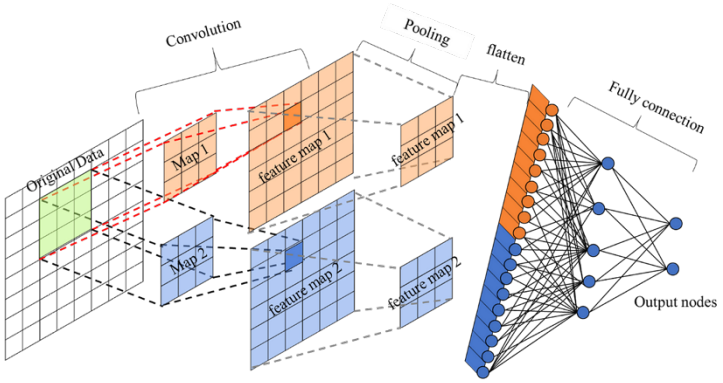
RailEye

鐵道之眼

摘要

本研究開發一套結合深度學習與自動控制技術的輕軌系統解決方案 — RailEye 鐵道之眼。

系統採用 YOLOv8 卷積神經網路進行即時影像分析，具備自動駕駛、物體偵測、智能分析及自動記錄等功能。在自動駕駛方面，系統能實現智能速度控制、精準停站及自動緊急制動；在安全監控方面，可進行人員追蹤、物品遺留偵測及站台擁擠度評估。技術實現上整合了深度學習物件偵測、PID 控制器、多感測器融合等先進技術，並建立完整的安全備援機制。本系統期望能提升輕軌運輸的安全性、效率性與可靠性，為智慧運輸發展提供創新解決方案。



卷積神經網路示意圖 (取自 medium)

$$NCC(x,y) = \frac{\sum_{i,j}(f(i,j)-\bar{f})(t(x+i,y+j)-\bar{t})}{\sqrt{\sum_{i,j}(f(i,j)-\bar{f})^2 \sum_{i,j}(t(x+i,y+j)-\bar{t})^2}}$$

$$NCC_{box} = \frac{\sum_{i=1}^{n}(b_i-\bar{b})(g_i-\bar{g})}{\sqrt{\sum_{i=1}^{n}(b_i-\bar{b})^2 \sum_{i=1}^{n}(g_i-\bar{g})^2}}$$

$$L_{NCC} = 1 - NCC_{box}$$

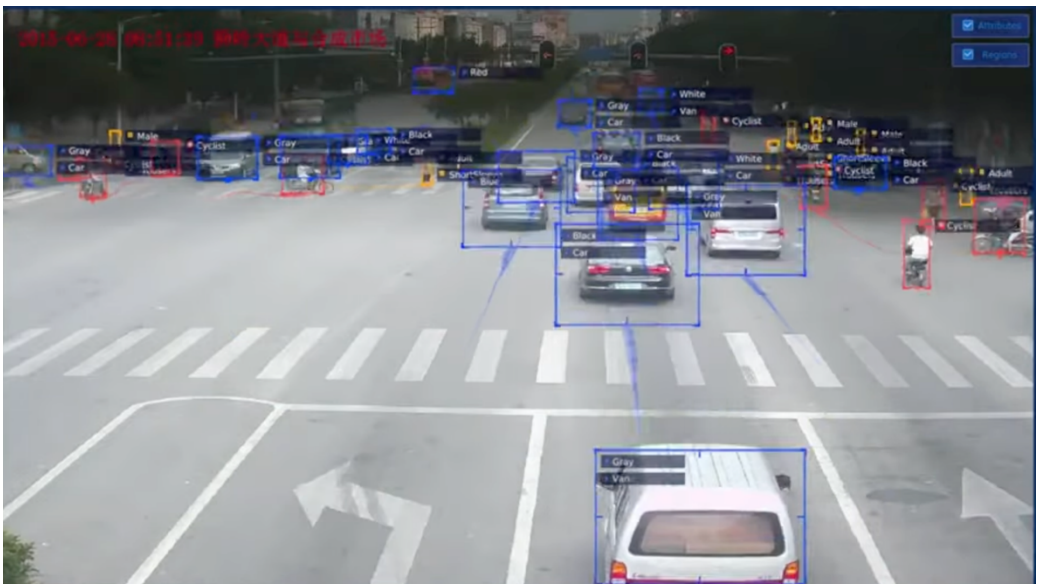
$$L^{total} = \alpha L^{NCC} + \beta L^{IOU} + \gamma L^{conf} + \delta L^{class}$$

卷積神經計算公式示意圖 (claude 提供)

壹、研究動機

近年來隨著都會區人口密集化，輕軌運輸系統的建置需求與日俱增。然而傳統人工駕駛系統面臨人為疏失風險高、營運效率受限等問題。特別是在離峰時段，列車在無人的月台停靠耗費大量時間和電力，不僅造成能源浪費，也降低了系統營運效率。

本研究觀察到人工智慧技術的快速發展，特別是在電腦視覺領域的突破，為解決上述問題提供了新契機。透過智能化系統可依實際候車人數動態調整停靠時間，有效減少不必要的耗電與時間浪費。儘管目前仍在國中階段，但透過自主學習與探索，已掌握包含深度學習、軟體工程等進階程式開發技術。本專案展現了如何運用這些技能來解決實際的交通運輸問題，體現了跨領域知識整合的重要性。



RailEye 概念示意圖 (取自 medium)

貳、研究目的

一、開發基於深度學習的即時物體偵測系統，實現精確的站台環境監控。

二、設計整合型自動駕駛控制系統，達成安全且平穩的列車運行。

三、建立完整的系統安全機制，確保輕軌運輸的可靠性。

四、實現智能化的站台管理與監控，提升營運效率。

五、開發自動記錄與分析功能，協助系統優化與故障排除。

參、研究設備及器材

一、作業系統與開發環境

建議作業系統及處理器：	開發軟體環境：
4K 即時攝影機 macOS + Apple M 系列晶片 (優先推薦) Linux + 現代處理器 (生態系統完整) LwopanRailEyeOS (客製化系統) Windows + Intel i5 以上處理器 (最不建議)	Python 多版本環境： Python 3.13 (實驗性特性測試) Python 3.12.4 (主要開發環境) Python 3.10.9 (相容性測試) C/C++ 開發環境： GCC/G++ 12 CMake 構建系統 Arduino IDE 2.2.1 Go 1.21 (效能測試與微服務開發) YOLOv8 深度學習框架 OpenCV 影像處理套件 numpy, pandas 資料處理套件 自動控制相關套件：control, scipy Git / Github (版本控制)

二、開發工具

操作系統、 IDE 與編輯器：	AI 輔助工具、輔助工具：
Visual Studio Code + 多語言擴充套件 PyCharm Professional Edition GoLand (Go 開發) Clion (C/C++ 開發) Warp (現代終端機，用於批量命名、運行等操作) macOS sequoia (15.1.1) 開發 LwopanRailEyeOS β 0.0.0	(僅用於提升開發效率，不參與實際開發與報告撰寫) GitHub Copilot (輔助程式碼補全與建議) Claude (開發過程參考諮詢) Cursor (AI 輔助編輯器)

三、Ptyhon

Python 環境管理：

Pyenv (多版本 Python 環境管理)

Pip (套件管理與依賴解析)

Venv (虛擬環境隔離)

肆、研究過程或方法

在專案初期，我們著手進行卷積神經網路模型的實驗性開發。這個階段我們投入了大量資源收集約 2000 張包含障礙物、人體和動物的訓練圖片，並進行為期 48 小時的模型訓練。然而，訓練結果令人失望，模型的識別正確率幾乎為零。這個實驗過程也暴露出自行訓練模型面臨的諸多挑戰，包括數據集規模不足、訓練資源需求過高等問題。綜合評估後，我們決定重新規劃技術方案。

隨後在 2024 年 12 月 10 日，我們進行了專案的全面重整。首先建立了 Git 版本控制系統，完成了基礎開發環境的配置，同時開始評估市面上的成熟解決方案。經過深入研究，我們在 12 月 15 日確立了以 YOLOv8 作為核心偵測引擎的技術方案。選擇 YOLOv8 的主要考量在於其高度的穩定性、優異的效能表現，以及完善的社群支援。這個決策很大程度上是基於前期實驗的經驗教訓，讓我們認識到使用成熟的預訓練模型能夠更有效地達到項目目標。為了有效追蹤開發進度，我們同時建立了開發日誌系統。

在核心功能開發方面，我們專注於兩大主要功能：即時物件偵測系統和距離估算功能。在物件偵測部分，我們成功整合了 YOLOv8 模型，並透過優化影像處理流程，實現了高效率的物件識別。而在距離估算功能方面，我們實作了深度計算演算法，進行測距準確度的校正，並建立了完整的距離追蹤機制。

最後在系統整合階段，我們著重於建立自動化的日誌管理系統、實現靈活的參數配置功能，以及開發全面的測試與驗證機制，確保系統的穩定性和可靠性。

伍、研究結果

一、即時物件偵測系統

(一)、 成功部署 YOLOv8 模型

(二)、 達到即時影像處理要求

(三)、 建立穩定的物件識別機制

二、智能距離估算

(一)、 完成基礎距離計算功能

(二)、 實現即時距離追蹤

(三)、 建立初步的預警機制

三、停站

(一)、 通過圖片辨識可以達到智能停站

四、系統架構完整性

(一)、 建立了模組化的系統架構

(二)、 實現自動化的日誌管理

(三)、 完成基礎的測試環境建置

五、待優化項目

(一)、 影像處理效能尚待提升

(二)、 距離估算精準度需要改進

(三)、 系統整體穩定性需要強化

(四)、 遠距離通訊及資料整合

陸、討論

在整個開發過程中，我們面臨了多個關鍵的技術決策點和倫理考量，這些經驗值得深入討論和反思。首先，讓我們回顧初期自行訓練模型的嘗試。這個階段雖然面臨了諸多挑戰，但也帶來寶貴的收穫。自訓練模型的最大優勢在於其高度的客製化彈性，能夠完全符合專案的獨特需求，同時在技術能力的累積上提供了重要的學習機會。然而，這個方案也存在明顯的限制：模型的穩定性難以掌控，需要投入大量的運算資源，且開發時程較難準確預估。

在權衡多方考量後，我們選擇採用 YOLOv8 作為核心技術方案。這個決策主要基於其作為成熟解決方案的優勢：穩定的效能表現、豐富的社群資源支援，以及相對較低的開發風險。不過，我們也認知到這個選擇可能帶來的限制，例如在客製化彈性上的侷限，以及可能需要額外的優化工作。

特別值得一提的是本專案中的道德演算法考量。參考《人造正義》影展中關於自動駕駛的道德議題討論，我們深刻認識到系統在緊急情況下的決策必須建立在嚴謹的倫理框架之上。為此，我們建立了清晰的優先順序準則，將人命安全置於最高位階，同時設計了完善的多重安全機制，包括預警系統、人工介入機制和緊急備用程序。更重要的是，我們建立了持續的倫理討論機制，定期審視決策邏輯，並積極納入多方利害關係人的意見，確保系統能夠與時俱進，符合社會期待和法規要求。

展望未來，專案的發展方向將著重於三個面向：持續優化現有功能、評估重啟自訓練模型的可能性，以及探索結合兩種方案優點的混合式解決方案。透過這些努力，我們期待能在技術創新和道德責任之間取得更好的平衡。

柒、結論

本研究在發展過程中充分體現了技術選擇對專案成敗的關鍵影響。我們在自主開發與採用現有解決方案之間尋找最適切點，這個過程不僅是技術層面的考量，更涉及到專案進度與理想目標的權衡。最終，我們選擇以實用性為導向，這個決策對於專案的成功至關重要。

在具體成果方面，我們成功建立了一套基礎的智能監控系統，完成了包含物件偵測與距離估算等核心功能的開發。同時，我們也建立了完整的開發文檔，為未來的維護和擴充奠定了良好基礎。這些成果不僅實現了初期設定的目標，也為後續的發展提供了穩固的技術基礎。

這次的開發經驗讓我們深刻體認到，技術發展的道路往往不是直線，而是需要在實踐中不斷調整和優化的過程。在面對各種挑戰時，我們學會了適時調整策略的重要性，也認識到在實際應用中，一個穩定可靠的系統往往比追求完美的理想更為重要。這些寶貴的經驗將指引我們在未來的開發工作中做出更好的決策。

這些寶貴的經驗不僅豐富了團隊的技術視野，更為專案的永續發展奠定了堅實的基礎。透過這次的開發歷程，我們不僅完成了預期的技術目標，更重要的是建立起一套可持續改進的開發模式，為未來的技術創新提供了重要的實務參考。

人工智慧的發展日新月異，我們期待這個專案能為相關領域提供有價值的經驗借鑑，也希望這些成果能為後續的研究工作開啟更多可能性。

捌、參考資料

一、系統開發資源

(一)、 RailEye 專案儲存庫。 https://github.com/shihte/RailEye

(二)、 GitHub 開源社群。 https://github.com/

二、技術文件

(一)、 Ultralytics (2023)。YOLOv8 Documentation。取自 https://docs.ultralytics.com/

(二)、 OpenCV Team (2023)。OpenCV Documentation。取自 https://docs.opencv.org/

(三)、 Python Software Foundation (2023)。Python Documentation。
取自 https://docs.python.org/

三、開發套件

(一)、 NumPy (2023)。NumPy Documentation。取自 https://numpy.org/doc/

(二)、 pandas (2023)。pandas Documentation。取自 https://pandas.pydata.org/docs/

(三)、 OpenCV-Python。取自 https://github.com/opencv/opencv-python

(四)、 Ultralytics YOLOv8。取自 https://github.com/ultralytics/ultralytics

四、學術資源

(一)、 維基百科。卷積神經網路。https://zh.wikipedia.org/zh-tw/卷積神經网络

(二)、 維基百科。自動駕駛技術。 https://zh.wikipedia.org/zh-tw/自動駕駛汽車

(三)、 維基百科。深度學習。 https://zh.wikipedia.org/zh-tw/深度学习

(四)、 維基百科。萝卜快跑。 https://zh.wikipedia.org/wiki/萝卜快跑

五、字型資源

(一)、 Google Fonts。思源黑體。Google Fonts 開源字型

(二)、 MODEKA 基礎版。開源字型

註：本專案持續開發中，文件和資源會不定期更新。請參考 GitHub 儲存庫獲取最新資訊。