構建 5G 智慧交通數位神經中樞－

附錄五、《路口綠燈使用率分析系統》建置指引手冊

交通部運輸研究所

中華民國 115 年 10 月

第一章 前言

1.1 編寫目的

本建置指引手冊的編寫目的，主要係將交通部運輸研究所於「構建 5G 智慧交通數位神經中樞系列計畫」研發之「路口綠燈使用率分析系統」推廣至實務的落地應用，提供各縣市交通管理中心落地實施的建置指南。該系統為智慧交通發展中「號誌效能評估」的核心模組之一，能夠以客觀、即時且可量化的方式，呈現路口各方向綠燈時間的實際使用成效，作為交通號誌優化與時制計畫修正的重要依據。

在過去的交通管理中，號誌調整多依賴人工觀測與經驗判斷，缺乏科學化、持續性的量化指標。本系統的導入，透過 AI 影像辨識技術與資料分析模型，將「綠燈是否被有效利用」轉化為可持續監測的效能數據，實現「數據化交通治理」的政策目標。

本手冊的核心宗旨如下：

一、建立全國統一的綠燈使用率演算法邏輯與評估指標；

二、提供一套可複製、可擴散、可整合的系統架構；

三、協助地方政府在既有交控中心基礎上導入智慧化效能分析；

四、促進交通號誌管理決策的科學化與自動化，提升民眾通行效率與道路服務品質。

本手冊同時具備政策推動面與技術落地面的雙重功能：政策層面上，可作為智慧交通績效監測制度化的參考依據；技術層面上，提供開發廠商與整合商可遵循的架構設計、演算法公式、資料格式與介接規範，確保不同縣市之間能維持一致的技術基準，實現後續跨區域資料整合與成效比較的可行性。

1.2 適用對象與範圍

一、適用對象

本手冊適用於推動智慧交通建設、號誌效能監控與資料整合應用之相關單位與人員，範圍涵蓋政策決策、系統建置及後續維運層面。

(一) 交通主管機關與政策決策者：可藉由本手冊了解綠燈使用率系統之功能定位、技術成熟度及導入效益，作為交通號誌優化計畫與智慧交控政策研擬之參考依據。

(二) 地方政府交控中心及技術承辦人員：作為系統建置、測試與維運時的技術規範與驗收基準，協助落實號誌運作效能的自動化監測與視覺化分析。

(三) 系統整合商與軟體開發廠商：依據本手冊規範，建置具備 AI 辨識與時制資料整合功能的交通監測平台，確保資料格式、演算法邏輯與 API 介接標準一致。

(四) 學術與研究單位：作為智慧交通績效評估或號誌控制策略之研究參考資料，以便進一步驗證與延伸技術應用，促進產學合作與技術推廣。

二、適用範圍

本手冊定義並說明「路口綠燈使用率分析系統」的完整建置與應用範圍，內容涵蓋系統設計原則、資料流程規劃、演算法邏輯與服務介接標準。主要範圍如下：

(一) 系統總體架構設計與建置原則：說明系統整體組成、模組關聯及部署邏輯，作為各單位建置時之參考藍圖。

(二) 綠燈使用率演算法公式與判斷邏輯：定義綠燈使用率之運算方式、統計週期與指標意涵，確保應用一致性。

(三) 資料庫與資料流設計：闡述系統資料來源、傳輸流程、儲存結構與更新機制，確保數據傳輸與整合穩定。

(四) API 介接規範：提供與外部系統（如數位雙生平台、交控中心及其他 ITS 系統）之資料交換格式與服務規範，確保跨系統整合的可行性與擴充性。

(五) 應用範圍與成效分析機制：本系統可獨立部署於單一路口，也可作為城市級智慧交通平台之子模組進行擴充，具備高度模組化與可擴展性。

1.3 相關背景依據

「路口綠燈使用率分析系統」的設計與建置，係基於本計畫中長期於台中地區路口實測所得之研究成果，並結合交通工程理論與實務經驗歸納而成，主要依據如下：

一、AI 影像辨識技術導入交通管理應用：透過 AI CCTV 即時串流影像進行車輛偵測與軌跡分析，統計綠燈期間通過車流數量與間隔，以取代傳統線圈感測方式，達到高準確度、高覆蓋率的交通監測。

二、綠燈效能量化指標建立：系統以「車流量換算綠燈使用」與「綠燈總時長」為基礎計算綠燈使用率，即：綠燈使用率 $=\frac{\text{車流量換算綠燈使用}}{\text{綠燈總時長}}\times 100\%$，若連續多週期綠燈使用率偏低，即視為號誌時制需檢討，提供交控中心作為調整依據。

三、數位雙生平台整合應用：本系統可整合至智慧交通數位雙生平台，並以三維視覺化方式即時呈現各路口的號誌效能。系統每 15 分鐘自動更新各方向的綠燈使用率資

料，於 3D 模擬環境中以立體標註方式顯示百分比數值與方向資訊，直觀呈現每個路口、各行車方向的實際通行效率，此整合應用不僅可協助管理單位即時掌握路口號誌運作效能，亦能作為交通決策、時制調整與系統效能評估的可視化分析工具。

四、標準化資料交換格式：綠燈使用率結果可以 JSON 結構上傳，包含「路口 ID」、「方向」、「綠燈時長」、「使用率」、「通過車輛數」等欄位，支援後端系統串接與統計分析。

第二章 功能簡介

2.1 系統功能說明

「路口綠燈使用率分析系統」為一套整合 AI 影像辨識、交通資料分析、視覺化呈現與決策支援的智慧化系統，具備以下主要功能：

一、自動化影像辨識與資料擷取：系統透過即時影像串流自動辨識車輛行進狀況，精準判定車輛於綠燈期間的通過時點，生成具時間與方向標記的原始交通資料。

二、綠燈使用率即時計算與紀錄：系統依據各方向的車流特性與綠燈時長，自動計算綠燈使用率，反映各路口綠燈時間的實際利用情形。計算結果以百分比形式即時更新，提供交通管理單位掌握各時段號誌效能的量化依據，作為後續號誌時制調整與交通策略分析之參考依據。

三、歷史資料追蹤與統計分析：系統可依日期、時段、地點等條件查詢歷史使用率趨勢，供交通單位進行時制檢討與方案模擬，支援長期交通效益分析。

四、標準化 API 服務介接：本系統建置完成後可提供 RESTful API 介面，供其他平台（如智慧交控中心、數位雙生系統、交通儀控伺服器等）即時存取分析結果，實現資料共享與跨系統整合。

2.2 適用場域與應用情境

本系統之核心目的在於以數據化方式呈現路口號誌的實際運作效能，並輔助交通管理單位進行時制優化與決策分析。其應用場域並不限於特定型態之交叉口，而是可依據交通型態、道路功能及管理目的，靈活部署於不同層級的交通節點。

一、都市幹道與主要路口

都市幹道多為多車道、多相位號誌控制之交叉路口，車流變化快速且高峰期明顯，過去多依賴人工觀測或歷史流量推估，缺乏動態調整依據。本系統可部署於都市主要幹道（如中正路、民族路、博愛路等）之主要交叉口，透過 AI 辨識即時回傳各方向綠燈使用率。交控中心可根據使用率低於特定值（如 50%）的路口進行時制檢討，調整綠燈配比或相位設計，亦可建立「高峰期綠燈使用率地圖」，協助識別壅塞路段及閒置號誌相位，作為號誌聯控計畫之依據，進而減少綠燈浪費，提高車輛通行效率、縮短高峰期車輛平均等待時間、建立客觀且持續性的時制優化依據，提升城市號誌自適應能力。

二、生活型道路網與次幹道路口

此類路口交通量較小，但受地區性活動、學校上下課、夜市或市場等時段性影響大，號誌多為定時控制或手動調整，易出現綠燈時間配置不均的問題。透過本系統可長期監測低流量路口的綠燈使用情況，分析早晚尖峰與離峰的使用率差異，若長期出現綠燈使用率低於 30%，可建議轉為閃燈控制或改採時段式調整。結合學區交通安全專案，可針對放學時段臨時增設管制並於平常維持節能模式，減少閒置號誌帶來的能源浪費，優化區域號誌設定，提高地區道路通行順暢度，提供鄉鎮級道路精細化交通管理的數據依據。

三、快速道路匝道與轉向匯入路口

快速道路與主幹道路匝道交會處，車流集中且加減速頻繁，容易形成瓶頸，系統可部署於匝道出口與主要幹道交會點，監測匝道放行時的綠燈使用率與匯入效率。若長期偵測匝道綠燈使用率偏低，表示主線壅塞反壓影響放行效率，可提供交控中心啟動「匝道儀控」建議，提升匝道放行效率與安全性，提供事件導向式號誌控制依據，改善快速道路匯入瓶頸與回堵問題。

四、特殊活動或事件型場域

大型活動（演唱會、體育賽事、節慶活動等）期間，特定路段車流劇增，需臨時調整號誌時制，可在活動周邊部署綠燈使用率監測，觀測車流疏散成效。系統可於活動期間以 15 分鐘更新頻率，提供實時使用率趨勢供交控中心參考，如若再結合數位雙生平台，以 3D 介面即時顯示各路口綠燈利用情況，可作為管制策略成效之佐證，快速評估臨時交通管制效果，提供活動後交通管理成效分析報告依據，累積多場域歷史資料，支持後續活動前期預測與規劃。

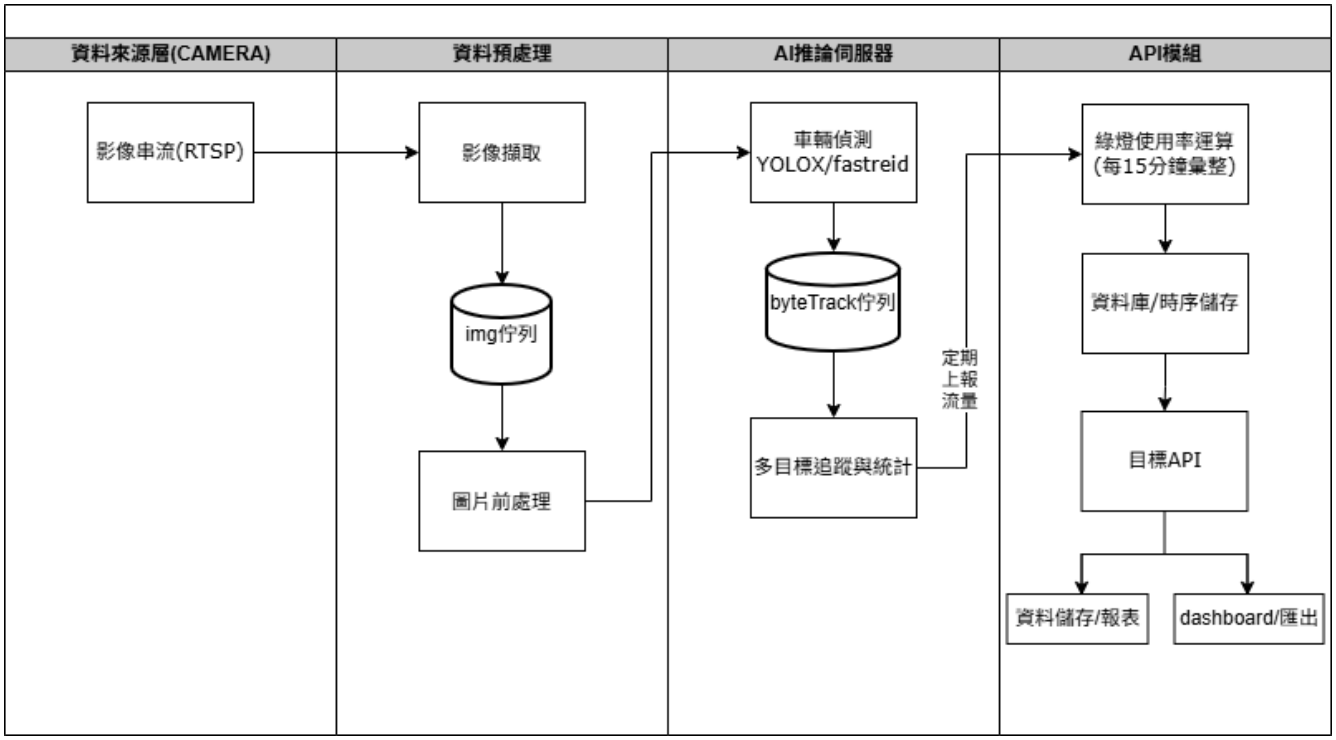
五、智慧城市整合應用(數位雙生平台)

屬於城市級交通治理與監控平台整合層，聚焦於跨區域、跨路口資料彙整與可視化呈現，將綠燈使用率數據透過 RESTful API 傳入數位雙生平台，以 3D 視覺化介面顯示各路口即時使用率，管理人員可同時檢視多路口效能、趨勢統計及異常警示，搭配其他交通資料（如平均速率、車流密度、事件紀錄）進行多維度交叉分析，強化城市交通全貌掌握能力，提供跨單位資料共享基礎（交控中心、警察局、都發局等），成為智慧交通治理數據決策的核心支撐模組。

第三章 系統功能建置說明

3.1 系統架構概述

本系統以即時影像智慧分析為核心，結合物件偵測、多目標追蹤、跨鏡頭車輛比對與交通指標計算等模組化處理流程，形成可支援多攝影機並行運作的高效非同步處理架構。整體流程(如下圖)由影像擷取開始，經 AI 推論與追蹤演算後，資料彙整與彙報由 API 服務介接至外部分析與展示平台。以下敘述各模組之功能設計與處理邏輯。



3.2 軟體設計／演算法說明

一、ROI 區塊與攝影機設定檔（Setting.yaml）

本系統採用非同步協程（asyncio）架構，以確保多路 RTSP 影像來源在高流量場域下仍能維持即時運算能力。所有攝影機參數、ROI（Region of Interest）區域以及方向性設定，均以設定檔（Setting.yaml）進行集中式管理，以利後續維運與擴充。

(一) 參數說明

1. Index：攝影機索引編號
2. url：RTSP 串流來源
3. roiSettingJson：ROI 與方向性設定檔位置

4. ROI_count：使用單一或多區塊（One / ALL）

(六) ROI 設定檔來源

ROI 設定檔由 roitoolfortraffic 工具產生，用於標註以下內容，此設定檔提供 AI 模組判定車輛通過事件之空間資訊。

1. 停止線位置
2. 每方向 ROI 區塊
3. 車輛方向性（左轉、直行、右轉）
4. 影像校正區域

二、資料蒐集（Data Collection）模組

資料蒐集模組負責自現場路口的多支 AICCTV 攝影機持續接收影像串流，並以影格佇列機制（Frame Queue）管理輸入資料，使影像輸入端與後端運算端得以解耦運作，確保在高流量場域或多路攝影機同時運作時，仍能維持 AI 推論端的即時處理能力。本模組亦為整個綠燈使用率系統之資料入口，其核心功能如下：

(一) 接收即時影像串流

透過 RTSP 協定取得 AICCTV 攝影機之 H.264 / H.265 影像串流。

(二) 影格佇列管理（Frame Queue）

在攝影機輸入端與 AI 運算端之間建立非同步佇列，避免因 GPU 過載而產生影格阻塞或延遲。

(三) 資料流

1. 輸入：AICCTV（RTSP）連續影格（H.264 / H.265 壓縮影像）
2. 輸出：提供給「資料預處理模組」之影格資料（Frame），已通過佇列管理，確保影格順序與時間戳記完整

三、資料預處理

資料預處理模組負責將資料蒐集模組輸入的原始影格轉換為適合 AI 車輛偵測與追蹤運算的格式。此模組的主要目的在於降低運算負擔、提升辨識品質，並過濾非必要區域的影像內容。

(一) 影格抽取（Frame Sampling）：控制每秒處理影格數（如 5–10 FPS），降低 GPU 運算負載。

(二) 資料流

1. 輸入：接收來自「資料蒐集模組」的影像資料。

2. 輸出： 提供給「AI 推論伺服器」之資料格式。

四、AI 推論伺服器

(一) AI 物件偵測與影像理解模組

本模組負責將輸入影像進行前處理後送入深度學習模型，完成車輛、行人等目標物件辨識。輸出資訊包括位置框、分類結果與置信度，作為後續追蹤與行為理解分析之基礎。

(二) 多目標追蹤與交通行為分析模組

本模組將物件偵測結果建立跨時間的追蹤識別，使同一物件在連續影格中維持一致 ID，以利交通行為分析與統計應用。

五、API 服務模組

本模組負責跨模組與外部系統間之資料傳遞與交換。追蹤統計資料採每 15 分鐘週期性彙整方式傳輸，而跨攝影機比對事件以即時發送為原則，以支援交通場景特性。

(一) 觸發機制（Trigger Mechanism）：由外部系統的 API 請求觸發，為被動服務。

(二) 運作流程

1. 接收外部系統的 HTTP 請求，並解析請求中的參數(如路段 id、查詢時間等)。
2. 根據解析後的參數，向 PostgreSQL 資料庫 查詢對應的預測資料。
3. 將從資料庫中檢索到的數據，格式化為通用的 JSON 格式。
4. 將 JSON 格式的數據作為 HTTP Response 回傳給請求方。

(三) 資料流

1. 輸入：外部系統發出的、包含認證金鑰與查詢參數的 HTTP API 請求。
2. 輸出：一個標準的 HTTP Response，其 Body 中包含格式化後的 JSON。

3.3 軟硬體資源需求

本系統所使用之軟體，包括作業系統、開發環境、資料庫與各式 Python 套件均為開源或免費授權軟體。在標準使用條件下，無需額外支出授權費用，能有效降低系統建置及後續擴充之軟體成本。

一、軟體資源需求

本系統所使用之軟體，包括作業系統、開發環境、資料庫與各式 Python 套件均為開源或免費授權軟體，在標準使用條件下，無需額外支出授權費用，能有效降低系統建置及後續擴充之軟體成本。

(一) 核心環境

1. 作業系統：Ubuntu 20.04/22.4 LTS Desktop
2. 程式語言：Python 3.9
3. 資料庫環境：MariaDB

(二) 開發與版本控制：Visual Studio Code、Git、Conda

VSCode 為主要編輯器，Git 用於版本控制，Conda 用於管理 Python 環境與套件。

(三) Python 核心函式庫

1. 數據處理與計算
   (1) NumPy
   (2) Scipy
   (3) Mpmath
2. 機器學習與模型訓練
   (1) PyTorch / Torchvision
   (2) YOLOX
   (3) ByteTrack
   (4) OpenCV-Python
3. 資料庫連接
   (1) psycopg2
   (2) sshtunnel

(四) API 服務開發

1. Web 框架：.NET Core MVC
2. 後端開發：.NET 6

(五) 授權與費用：本系統所規劃使用的上述所有軟體元件，均為廣泛使用的開源軟體，在標準使用條款下無需支付任何商業授權費用。

二、硬體資源需求

本系統之硬體效能需求，將取決於以下條件：1.監測路口數量、2.AI 推論頻率(FPS)、3.同時處理之攝影機數量、4.API 查詢量、5.是否啟用 ReID、ByteTrack 等高負載模組。以下區分三種情境，提供政府單位作為建置規劃之參考。

(一) 情境一：開發與測試環境，此規格適用於開發人員進行功能開發、單元測試或小規模數據驗證，預計處理路口數量在 100 個以下。

1. 架構：建議採用單機架構，所有服務（應用程式、資料庫）運行在同一台主機上。
2. CPU：現代 x86-64 架構，至少 8 核心處理器。
3. 記憶體：至少 32 GB RAM。
4. 儲存空間：至少 1 TB NVMe SSD，以確保快速的資料讀寫與模型載入速度。
5. 網路：標準 1 Gbps 網路介面。

(二) 情境二：中小型正式環境，此規格適用於正式部署，預計處理數百至一千個路口，資料量與使用者請求量適中。

1. 架構：建議採用主從式架構，將應用程式主伺服器與資料庫伺服器分離，避免資源競爭。
2. 主伺服器：
   (1) CPU：32 核心以上伺服器級 CPU。
   (2) 記憶體：至少 128 GB RAM。
   (3) 儲存空間：2 TB 以上企業級 NVMe SSD。
   (4) GPU：NVIDIA A4000 / RTX 4080（依路口數）
3. 資料庫伺服器：
   (1) CPU：16 核心以上伺服器級 CPU。
   (2) 記憶體：至少 64 GB RAM。
   (3) 儲存空間：4 TB 以上企業級 SSD（可搭配高容量 HDD 進行備份）。
4. 網路：10 Gbps 網路介面卡與對應交換器。

(三) 情境三：大型正式環境（大規模部署），此規格為高運算效能的配置，適用於處理數千個以上路口的即時預測，需同時處理大量影像串流、AI 偵測與高併發 API 請求，通常適用於直轄市或大型跨區部署。

1. 架構：獨立主伺服器與資料庫伺服器，並可根據需求擴展為叢集架構。
2. 主伺服器（應用程式與模型運算）：
   (1) CPU：AMD Ryzen TR PRO 9995WX (96 核 128 緒)或同等級高核心數處理器。
   (2) 記憶體：128 GB (32GB*4) DDR5 3200 MHz 或更高。
   (3) 儲存空間：系統碟使用 2TB M.2 SSD；快取與工作區使用 4TB SSD*2；長期資料儲存使用 20TB HDD*2；GPU：多張 A4000（視部署規模）。
   (4) 網路頻寬：500 Mbps 以上。

3. 資料庫伺服器：

(1) CPU：Intel Core i7-13700 (16 核 24 緒) 或同等級高時脈處理器。

(2) 記憶體：128 GB (32GB*4) DDR4 3200 MHz 或更高。

(3) 儲存空間：主資料庫使用 2TB M.2 SSD；熱資料使用 4TB SSD*2；冷資料與備份使用 20TB HDD*2。

(4) 網路頻寬：500 Mbps 以上。

4. 網路設備：需配備對應吞吐量的路由器（Router）與交換器（Switch）。

表 1 硬體資源需求彙整表

硬體規格	情境一： 開發與測試	情境二： 中小型正式環境	情境三：大型正式環境 （大規模部署）
適用情境	功能開發、單元測試、小規模數據驗證（處理路口數量 < 100）	正式部署，處理數百至一千個路口綠燈使用率計算（資料量與請求量適中）	正式部署，處理數千個以上路路口綠燈使用率計算（需應對高併發與龐大資料吞吐量）
架構	單機架構（所有服務運行於同一主機)	主從式架構（應用程式與資料庫伺服器分離）	獨立主伺服器與資料庫伺服器（可根據需求擴展為叢集架構）
主伺服器 CPU	現代 x86-64 架構，至少 8 核心	32 核心以上伺服器級 CPU	AMD Ryzen TR PRO 9995WX（96 核 128 緒）或同等級
主伺服器 記憶體	至少 32 GB RAM	至少 128 GB RAM	128 GB（32GB*4） DDR5 3200 MHz 或更高
主伺服器 儲存空間	至少 1 TB NVMe SSD	2 TB 以上企業級 NVMe SSD	● 系統碟：2TB M.2 SSD 快取/工作區: 4TB SSD*2 ● 長期儲存：20TB HDD*2
資料庫伺服器 CPU	與主伺服器共用	16 核心以上伺服器級 CPU	Intel Core i7-13700 （16 核 24 緒）或同等級
資料庫伺服器 記憶體	與主伺服器共用	至少 64 GB RAM	128 GB (32GB*4) DDR4 3200 MHz 或更高
資料庫伺服器 儲存空間	與主伺服器共用	4 TB 以上企業級 SSD（可搭配高容量 HDD 進行備份）	● 主資料庫：2TB M.2 SSD 熱資料: 4TB SSD*2 ● 冷資料/備份：20TB HDD*2
網路	標準 1 Gbps 網路介面	10 Gbps 網路介面卡與對應交換器	● 頻寬：500 Mbps 以上 ● 設備：需配備對應吞吐量的路由器與交換器
硬體購置成本	預估至少 8 萬元	預估至少 40 萬元	預估至少 150 萬元

3.4 功能驗證與效果評估方法

一、功能驗證方法

功能驗證旨在確認系統的每一個模組與整體流程，是否都按照設計規格正確無誤地運行。建議採用分層的測試策略，從最小的程式單元到完整的端到端流程進行驗證。

(一) 單元測試 (Unit Testing)

單元測試針對系統內最小的功能單元（函式、模組）進行驗證，確保輸入與輸出邏輯正確，避免後續整合階段出現連鎖錯誤。建議測試對象清單包含：

1. 資料蒐集模組
   (1) RTSP 解析與影格擷取函式
   - 驗證 RTSP URL 是否可成功連線
   - 模擬影像中斷、URL 錯誤時是否能正確例外處理
   - 檢查擷取影格 FPS 是否符合設定（如 sampling=5FPS）

   (2) ROI/方向性設定檔解析函式
   - 讀取 YAML（Setting.yaml）並正確解析 index、url、ROI_count 等欄位
   - 驗證 ROI JSON（由 roitoolfortraffic 產生）是否包含：
     - ROI polygon
     - direction 設定
     - stop line index
   - 模擬缺欄位、路徑錯誤之例外測試

2. 資料預處理模組
   (1) 影格標準化函式
   - 驗證 resize、BGR → RGB、tensor 格式轉換是否正確
   - 測試在不同光源、夜間、雨天影格下仍能正常輸出

   (2) ROI 遮罩與方向性判斷
   - 給定一個包含多車輛的影格
   - 檢查是否只擷取 ROI 區域
   - 驗證方向性標籤是否正確套用

3. AI 偵測與追蹤模組（YOLOX + ByteTrack + ReID）

(1) YOLOX 偵測函式

- 使用標記影像測試是否能正確抓到車輛 bounding boxes
- 測試光線差、遮擋、部分畫面不完整時偵測是否正常

(2) ByteTrack 多目標追蹤

- 模擬車輛被遮擋後再出現，驗證 Track ID 是否不重置
- 驗證同一車輛停止後再移動，ID 是否保持一致

(3) ReID 特徵比對

- 給定兩張相同車輛影像，驗證 cosine similarity > 閾值
- 給定不同車輛影像，驗證 similarity < 閾值
- 測試 embedding 向量維度與計算正確性

4. API 服務模組

(1) Query String 參數驗證：驗證 Controller 是否能正確解析請求中的 Query 參數（如 window_start、window_end），並對無效或缺失的參數回傳正確的錯誤狀態碼。

(2) 回應格式化函式：給予一個從資料庫查詢出的原始數據（如字典列表），驗證此函式能否將其轉換為 API 規格所定義的最終 JSON 格式。

有關以上各單元測試作業，綜合以上建議可使用.NET 單元測試框架進行測試。

(二) 整合測試 (Integration Testing)

整合測試用以驗證各模組之間的資料傳遞是否順暢，確保資料流沒有斷裂、格式不符或延遲。

1. 影像 → AI 偵測 → ByteTrack → ReID 整合測試

(1) 驗證流程：

- 使用 RTSP 模擬器／錄影檔輸入影像
- 檢查 YOLOX、ByteTrack 是否能正確串接
- 驗證 Track ID 一致性與 ReID 效果

(2) 檢核點：

- 模組之間 JSON / dict 資料格式一致
- 追蹤事件可正確產生「car_id、direction、timestamp」

2. 通過事件 → Green Used 計算整合測試

(1) 驗證流程：

- 從 trackdemo_asyncio 產生 crossing event
- 傳入綠燈使用率計算模組
- 檢查 PCU 累積與 green_used 輸出

(2) 檢核點：

- 相同車輛不重複累加
- 方向性對應正確
- 15 分鐘統計邊界正確

3. 資料庫寫入與讀取

(1) 驗證流程：

- 寫入綠燈使用率資料（green_total、green_used、usage）
- API 讀取同一區間資料

(2) 檢核點：

- 欄位型態正確
- 時間戳記無跨日、越界
- DB → API 回傳內容一致性

(三) 端到端測試 (End-to-End Testing)

端到端測試之目的為模擬真實世界的使用情境，驗證整個系統從起點到終點的完整自動化流程是否順暢。

1. 完整使用率流程

(1) 流程：

- RTSP → 偵測 → 追蹤 → ReID → Line Crossing
- 15 分鐘統計（PCU / 綠燈使用率）
- 寫入資料庫
- 使用 API 查詢結果

(2) 檢核點：

- 與資料庫資料一致
- 15 分鐘時間窗精確
- 各方向 usage 計算正確

2. 多攝影機併行負載測試

測試項目：

- 同時處理 4 / 8 / 12 支攝影機

- FPS 是否下降
- asyncio 的 imgQ、byteTrackQ queue 深度是否爆滿

3. ROI 設定錯誤恢復能力

模擬錯誤：

- ROI JSON 缺欄位
- polygon 格式錯誤
- RTSP 中斷

需驗證：

- 系統能否 log 錯誤
- 自動跳過該路口不造成整體中斷

(四) API 測試

API 測試的目的為確保 API 服務模組的穩定性與可靠性，驗證其是否能正確處理各種請求。

建議測試案例清單包含：

1. 針對端點：GET /api/usage-window

(1) 功能與正確性測試

- 測試案例 1：有效參數
  - 操作：使用一個有效的 apiKey 發送請求。
  - 預期結果：伺服器應回傳 200 OK 狀態碼，回傳 JSON 包含：Intersection、slots[]、summary（min/max/mean/std）
- 測試案例 2：日期格式正確但查無資料
  - 操作：使用一個有效的 apiKey 並加上 format=csv 參數發送請求。
  - 預期結果：伺服器應回傳 404 Not Found 及回傳 { "error": "..." }

(2) 參數錯誤與例外測試測試

- 測試案例 3：
  - 操作：發送一個缺少 apiKey 參數的請求。
  - 預期結果：伺服器應回傳客戶端錯誤狀態碼（如 400 Bad Request），並附帶說明錯誤原因的訊息。
- 測試案例 4：
  - 操作：使用一個格式正確但日期為未來時間發送請求。

- 預期結果：伺服器應回傳 400 Unauthorized 狀態碼，拒絕此次請求。

2. 非功能性測試(壓力測試)：
   (1) 操作：使用 JMeter、Locust 等壓力測試工具，模擬大量使用者在短時間內同時對 API 端點發送請求。
   (2) 檢核點：觀察在高併發情境下，API 的平均回應時間、錯誤率及伺服器的 CPU、記憶體使用率是否維持在可接受的範圍內，以確保服務的穩定性。

3.5 常見問題與風險因應

本系統在長期運作中將面臨影像來源、AI 模型效能、追蹤穩定性、事件判斷準確性、資料庫寫入效能與 API 服務等多面向的挑戰。本節旨在說明系統於實際部署與維運時可能遭遇的典型風險，並提供具體且可執行之技術因應策略，協助維運單位建立長期穩定之監測能力。

一、影像來源品質不穩定

(一) 風險描述：路口 AI-CCTV 影像可能因設備老化、網路波動、逆光、雨霧氣候、夜間低照度或供電問題導致品質下降，甚至影像中斷。這會造成 AI 偵測率下降或無法產生通過事件。

(二) 潛在影響

1. 偵測準確度下降：影像模糊、偏暗或高反光會造成車輛偵測誤判或漏判。
2. 追蹤中斷：畫面跳動或卡頓會使 ByteTrack 重置 track_id。
3. 綠燈使用率失準：車輛事件不足，導致使用率被低估。

(三) 因應對策

1. 影像品質監控機制
   (1) 建議每日檢查影像亮度、對比度、FPS、丟幀率並建立影像健康指標(Quality Score)。
   (2) 對異常影像自動發出警示。
2. 自動重連與緩衝策略
   (1) RTSP 採 retry/backoff 機制，影像中斷時自動重連。
   (2) 對高延遲影像建立 frame skip 邏輯避免排程阻塞。
3. ROI 輔助修正：在極端逆光或夜間環境，可透過 ROI 裁切與 gamma 校正

輔助改善辨識品質。

二、AI 偵測與追蹤模型效能衰退（Model & Tracking Drift）

(一) 風險描述：路口環境常因季節（光線角度）、施工（場景遮擋）、大型車流量變化等因素改變。AI 模型長期未更新或追蹤演算法遇到複雜遮擋場景，將導致偵測與追蹤精度下降。

(二) 潛在影響

1. Tracking ID 失真：ByteTrack <-> ReID 無法維持同車一致 ID。
2. 事件錯計：若 track_id 不一致，將使同一台車被計算多次或漏算。
3. 綠燈使用率偏差：車流量錯誤直接影響使用率計算。

(三) 因應對策

1. 模型效能監控
   (1) 定期抽樣影像並以人工比對 YOLOX 偵測結果、ByteTrack 追蹤率。
   (2) 設置 ReID Similarity Score 閾值監控飄移。
2. 季節性模型更新：對夜間、雨天、逆光等高風險場景建議每 6–12 個月重新訓練或微調模型。
3. Tracking + ReID 整合優化
   (1) 針對大型車(公車、卡車)遮擋情境調整 IoU 與 ReID similarity threshold。
   (2) 於 ROI 內外使用不同權重避免場景變動造成干擾。

三、追蹤錯亂與重複計算風險

(一) 風險描述：多車併行、機車群聚、遮擋、車輛停等、突然加速等情況可能造成追蹤跳號，導致「同一車被算兩次」或「不同車被誤認為同一車」。

(二) 潛在影響

1. 綠燈使用率偏高或偏低
2. 方向性錯誤（錯誤方向被累計）
3. 一段時間的統計資料失真

(三) 因應對策

1. 建立 Line Crossing 去重機制：同一 track_id 在 1.5–3 秒內不得重複觸發事件。
2. 結合 ReID embedding，避免 ByteTrack 重置 ID。
3. 對停在停止線上的車輛建立「靜止狀態偵測」，避免計算多次事件。
4. 增加虛擬線前後緩衝區（buffer zone），提升跨線判斷精度。

四、資料庫效能瓶頸

(一) 風險描述：綠燈使用率每 15 分鐘會產生多筆事件資料；長期累計會產生大量記錄（例如一年可達百萬以上筆資料，視路口數量而定）。

(二) 潛在影響

1. 查詢延遲，使 API 回應時間增加。
2. 批次統計或報表產生時間過長。
3. 數位雙生平台查詢即時性受影響。

(三) 因應對策

1. 建立複合索引（id + timestamp）
2. 每月/每季資料封存機制（保留 12 個月資料於主庫）
3. 使用 MariaDB Partition（依時間分區）
4. API 查詢範圍限制（避免一次查詢超過 30 天）

五、API 服務中斷或效能風險

(一) 風險描述：API 層可能因流量暴增、資料庫延遲、外部系統攻擊或程式錯誤導致無法即時回應。

(二) 潛在影響

1. 數位雙生平台無法取得最新使用率
2. 系統之間無法交換資訊
3. API 超時造成服務品質下降

(三) 因應對策

1. API 層錯誤處理：所有查詢錯誤需回傳標準錯誤 JSON（依 API 文件）。
2. Rate Limiting（速率限制）：限制每個 API Key 每分鐘存取次數。
3. 負載平衡（Load Balancer）：多台 API 伺服器分擔查詢流量。
4. 快取（Caching）：固定統計結果可加上 Redis 1 分鐘快取，加速 API 回應速度。
5. 錯誤告警：API latency > 500ms 持續 3 分鐘自動告警。

六、ROI 設定錯誤與場景變動風險

(一) 風險描述：ROI JSON 由人工透過工具繪製，若錯誤設定或現場施工導致車道位置變化，會造成事件判斷錯誤。

(二) 潛在影響

1. 車輛無法正確觸發 crossing event
2. 方向性判斷錯誤

3. 綠燈使用率計算失真

(三) 因應對策

1. 每半年重新驗證 ROI 與停止線位置。

2. 當 ByteTrack 出現與 ROI 不一致的軌跡時，自動標記為「場景變動可能」。

3. 當現場施工或道路改道時需立即更新 ROI 設定。

七、RTSP 來源中斷與多攝影機併發風險

(一) 風險描述：同一主機需要同時處理多組 RTSP，若 RTSP 中斷或延遲，asyncio Queue（imgQ / byteTrackQ）可能塞滿造成整體運算延遲。

(二) 影響

1. 整體 FPS 降低

2. 影像延遲，事件判斷時間錯位

3. 不同攝影機之統計時間窗不同步

(三) 因應對策

1. 設定 Queue 最大長度與 overflow handling。

2. 設定 RTSP reconnect/backoff。

3. 每支攝影機設定個別 timeout，避免拖垮全部管線。

八、AI 辨識風險管理措施

「針對影像辨識引發之治理風險，建置單位應採取以下措施：

(一) 邏輯校核：建立「AI 事件檢預機制」，若單一方向綠燈使用率連續 5 週期出現 100% 或 0% 之極端數值，系統應自動擷取關鍵影格供人工複核，避免因影像遮擋導致之決策誤判。

(二) 透明監控：維運介面應提供 AI 辨識置信度（Confidence Score）即時呈現，符合《AI 基本法》之可解釋性要求。

第四章 整合與落地應用作法

4.1 整合應用模式

為使本路口綠燈使用率分析系統的成果能順利融入交通管理單位既有作業流程與資訊系統，建議在系統建置完成後，可採用以下整合模式，並結合實際情境進行應用。

一、API 介接模式

此模式為最具彈性、易於部署且耦合度最低的整合方式。交通管理單位的既有交控平台、數位雙生平台或道路管理系統，可透過 API 向綠燈使用率系統查詢各路口各方向的使用率、車流量與統計結果。

(一) 整合架構：交控中心/數位雙生平台→API Gateway→綠燈使用率系統後端 (.NET 6 Web API)

(二) 優勢

1. 技術解耦：兩系統可獨立開發維護，不需共享資料庫或部署環境。
2. 安全性高：可經由 API Gateway 管控 API Key、授權、流量管制、存取紀錄。
3. 標準化與通用性：採 RESTful + JSON，符合政府系統介接標準，跨語言、跨平台皆可調用。

(三) 技術重點

1. 認證與加密：認證機制建議採用 API Key(本系統目前使用)或可升級為 JWT / OAuth2，傳輸通道必須使用 HTTPS/TLS 加密。
2. 資料查詢與回傳格式
   (1) API 呼叫範例：使用 GET 方法，並透過 Query String 傳遞參數。
   (2) 回傳格式範例 (JSON Schema)：回傳的 JSON 應具備清楚且固定的結構，例如：


```
{
  "intersection": {
    "id": 1,
    "name": "中清路二段_黎明路三段",
    "coordinates": [
      24.196733354865565,
      120.65739373679033
    ]
  },
  "window_start": "2025-09-23T16:30:00.000+08:00",
  "window_end": "2025-09-23T16:44:59.000+08:00",
```

```
  "interval_minutes": 15,
  "lcm_total": 900,
  "slots": [
    {
      "start": "2025-09-23T16:30:00.000+08:00",
      "end": "2025-09-23T16:44:59.000+08:00",
      "weekday": "Tue",
      "results": [
        {
          "approach": 1,
          "cycle": 180,
          "green_per_cycle": 138,
          "cycles": 5,
          "green_total": 690,
          "pcu_count": 343,
          "green_used": 686,
          "usage": 0.99
        },
        {
          "approach": 2,
          "cycle": 180,
          "green_per_cycle": 138,
          "cycles": 5,
          "green_total": 690,
          "pcu_count": 140,
          "green_used": 280,
          "usage": 0.41
        },
        {
          "approach": 3,
          "cycle": 180,
          "green_per_cycle": 27,
          "cycles": 5,
          "green_total": 135,
          "pcu_count": 58,
          "green_used": 116,
          "usage": 0.86
        },
        {
          "approach": 4,
          "cycle": 180,
          "green_per_cycle": 27,
          "cycles": 5,
          "green_total": 135,
          "pcu_count": 25,
          "green_used": 50,
```

```
                "usage": 0.37
            }
          ]
        }
    ],
    "summary": {
        "min": 0.37,
        "max": 0.99,
        "mean": 0.6575,
        "std": 0.2718
    }
}
```


3. 流量管理與錯誤處理

(1) 速率限制 (Rate Limit)：應在 API Gateway 或服務本身設置速率限制（如本系統設計為每分鐘 600 次請求），避免因客戶端程式錯誤或惡意請求導致服務過載。

(2) 錯誤碼設計：定義標準的 HTTP 錯誤碼，便於客戶端進行錯誤處理。

- 400 Bad Request：一個或多個請求參數遺失或無效 (例如：日期格式錯誤、endDate 早於 startDate)。
- 403 Forbidden：已通過認證，但沒有存取此資源的權限。請確認您的 apiKey 是否有足夠權限。
- 404 Not Found：請求的 roadId 不存在。
- 405 Method Not Allowed：請求的方法不被允許，或此資源不支援該 HTTP 方法（例如僅支援 GET，但使用 POST）。
- 429 Too Many Requests：超過請求限制。
- 500 Internal Server Error：伺服器內部發生錯誤，無法完成請求。請稍後再試，或聯絡系統管理員。

4. 版本管理：建議在 API 路徑中加入版號（如 /api/v1/usage-window），當未來 API 規格有重大變更時，可透過升級版號（/api/v2/...）的方式，確保舊有系統的相容性。

二、資料庫層級整合模式

此模式適用於政府單位內部已存在大型資料處理平台或需進行大量批次查詢的情境。

(一) 優勢

1. 大量資料查詢時延遲最低

2. 適合離線報表、統計分析、AI 模型研究等用途

(二) 技術重點

1. 連線方式：

(1) 建議提供 唯讀資料庫副本（Read-Only Replica）

(2) 所有查詢應透過 SSH Tunnel + SSL 加密

(3) 主資料庫不得直接提供外部查詢，以避免影響寫入效能

2. 權限控管

(1) 採 RBAC 模型：

(2) 建立專屬唯讀帳號

(3) 僅允許查詢固定資料表

(4) 建議提供 View 而非直接讀取基礎資料表

3. 資料表結構與維運：若本系統的資料庫 schema 需更新，建議採用 Entity Framework Core Migration 或其他資料庫版本管理工具（如 Flyway / Liquibase）進行控管。所有 schema 變更應版本化管理，並提前通知所有依賴資料庫的外部系統，確保相容性與整體服務穩定性。

4. 備援與風險處理

(1) 若唯讀副本失效，可啟用備援 Replica

(2) 緊急情況下可降級為 API 介接模式（fallback）

三、資料交換平台(DEP)模式：若交通單位已有資料交換平台(如市級資料中樞、TDX、交通雲)，可採此模式。

(一) 整合架構：路口綠燈使用率分析系統→(SFTP / MQ / API Push)→資料交換平台→其他單位系統

(二) 優勢

1. 單一平台多系統共享

2. 政府單位之間更容易使用統一資料集

3. 數據授權、版本管理與流量管理均由平台統一處理

(三) 技術重點

1. 傳輸方式：透過 FTP/SFTP、Message Queue（如 Kafka、RabbitMQ）

2. 傳輸格式：JSON / CSV / Parquet

3. 傳輸頻率：每 15 分鐘或每 5 分鐘

4.2 系統維運建議

為確保綠燈使用率系統在部署後能長期穩定運作，維運流程需涵蓋影像來源、AI 推論模組、事件判斷、資料庫效率、API 服務與硬體設備等層面的監控與調校。本節提出一套完整且可落地執行的維運策略，協助交通管理單位於實務環境中持續掌握系統健康狀態，並及早因應可能的異常情形。

一、影像來源監控

綠燈使用率系統的最核心基礎為穩定且可辨識的影像來源。因此，維運單位需建立每日執行的影像品質檢測程序，透過自動化腳本對各攝影機來源檢查影格輸入狀況，包括 FPS 是否異常下降、RTSP 串流是否出現中斷、夜間影像亮度是否低於可辨識門檻，或畫面是否出現模糊、閃爍與大量噪點等現象。透過此自動化監測，可在問題尚未擴大前即時發現異常來源。例如，若某攝影機在連續 10 分鐘內均未輸入有效影格，系統應立即觸發告警，以便維運人員能迅速查明設備故障、網路異常或其他外在原因，避免因影像品質不良而影響綠燈使用率的判讀品質。

二、系統功能檢測（Health Check）

為確保整體系統流程皆能正常運作，應每日於離峰時段執行自動化健康檢查流程。此檢查應包含從影像讀取到最終 API 回應的完整端到端流程，內容包括：啟動影像讀取程序、執行一次 YOLOX 偵測、利用 ByteTrack 追蹤生成至少一筆事件、將事件寫入測試資料庫，並透過 API 查詢是否能成功取得該筆資料。此流程能有效驗證 AI 模組、事件判斷模組、資料庫寫入模組與 API 提供服務的整體通暢性。若健康檢查中的任一環節出現異常，例如偵測模組無法啟動、事件未寫入資料庫或 API 無法回應，系統應立即通知維運人員，以便迅速進行問題排除。

三、AI 模型效能追蹤

AI 模組（YOLOX 偵測、ByteTrack 追蹤與 ReID 特徵比對）是綠燈使用率判讀的關鍵，因此需建立定期的模型效能監測機制。建議每週針對部分路口進行抽樣檢查，包含偵測結果是否與人工標記接近、ByteTrack 是否能維持車輛 ID 的一致性，以及 ReID similarity score 是否仍符合模型建置時的標準。此外，可透過觀察遮擋場景下的錯誤率（Shadow Rate）變化，判斷模型在現場環境變動下是否逐漸失效。若模型的偵測準確度或追蹤穩定性下降幅度超過預設標準（例如 20%），維運單位應啟動模型重訓、調整參數或更新 ReID 特徵庫，以維持系統效能。

四、設備保養與巡檢

為避免因硬體故障而影響系統運作，建議採取自動化監控與人工巡檢並行的保

養策略。在自動化部分，可透過監控代理程式持續記錄 GPU 與 CPU 的溫度狀況、磁碟的 SMART 健康狀態、RTSP 串流丟幀率與伺服器資源（CPU/RAM）使用率。在人工巡檢方面，建議每季由維運人員進行機房巡查，檢查伺服器運作環境（如溫度、濕度與通風情形）、線路連接是否鬆動、攝影機鏡頭是否受污損或偏移，以及現場道路環境是否因施工或調整而影響 ROI 的正確性。透過雙軌保養，可以在問題擴大前提早發現硬體異常，減少長期運作中的風險。

五、異常即時通報機制

為提升系統維運效率，建議建立統一的異常告警平台，並整合所有資料來源異常、AI 模組錯誤、API 服務延遲、資料庫負載過高以及伺服器硬體資源異常等訊息。當偵測到重要指標超出閾值，例如 API 錯誤率突然上升、資料庫查詢延遲明顯增加，或 GPU 使用率長時間維持在高負載，系統應立即透過 Email、Line、Teams 等通訊工具向指定維運人員推送告警訊息。此即時通報機制有助於縮短問題反應時間，降低服務中斷風險。

六、版本更新管理

為確保系統在功能更新與安全修補過程中的穩定性，應建立完整的版本控管與部署流程。建議採用 Git Flow 等標準化版本管理模式，將所有程式碼分為 feature、develop 與 main 等清晰分支，並在合併前進行程式碼審查（Code Review）。系統更新前應透過 CI/CD 自動化流程於預備環境進行完整的單元測試、整合測試與端到端測試，以驗證所有模組與 API 的運作無誤後，才部署至正式環境。此外，AI 模型（如 YOLOX、ReID 模型）更新時，也應採用相同的版本控管流程，記錄模型版本、訓練資料來源與更新原因。正式環境必須保留至少一個可立即回滾的版本，以避免更新失敗時造成服務長時間中斷。