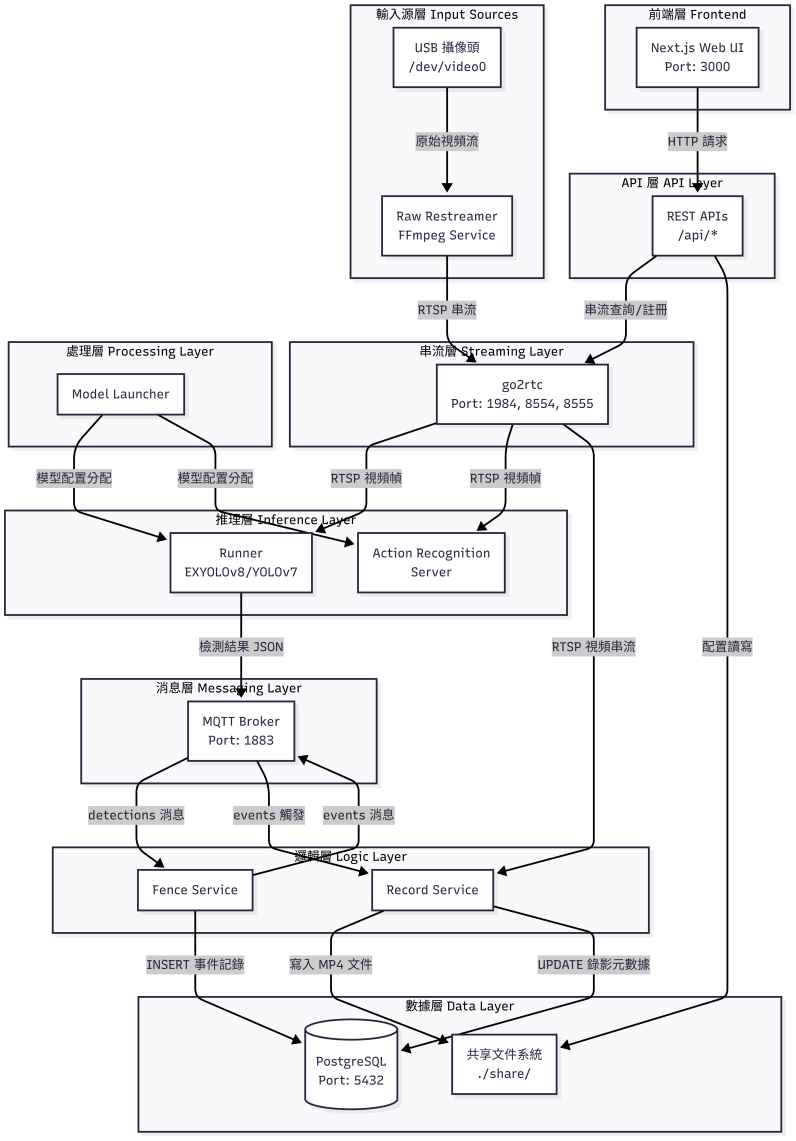
Action Recognition Server - API 文檔

概述

本文檔說明動作辨識伺服器架構中所有可用的 API 接口，包括內部服務通訊協議和外部集成接口。這些 API 可以用於系統嫁接、與外部系統集成，或移植到其他應用中。

系統架構概覽



模組說明

系統由以下 10 個主要模組組成,透過 Docker Compose 協調運行:

1. raw_restreamer

技術: FFmpeg 6.1
端口: 無對外端口
職責:

- 從 USB 攝影機 (`/dev/video0`) 擷取原始影像
- 將 YUYV422 格式轉換為 H.264 編碼
- 透過 RTSP 協議推送至 go2rtc 的 `cam1_raw` 串流
- 支援低延遲配置 (zerolatency tune)

---

2. action_recognition_server

技術: Python + OpenCV (待定) **職責:**

- 從 record 拉取影片並辨識有無行為發生
- （待定）

環境變數:

- `IN_URL`: 輸入 RTSP 來源
- `OUT_URL`: 輸出 RTSP 目標
- （待定）

---

3. model_launcher

技術: Python + YOLOv8/YOLOv7
端口: 無對外端口
職責:

- 根據 `cameras.json` 配置動態啟動 AI 模型推論進程
- 從 `models.json` 匹配對應的模型配置
- 執行物體檢測推論並發布結果至 MQTT
- 支援多攝影機共享同一模型
- 支援 CPU/GPU 推論

MQTT 發布:

- Topic: `vision/{cameraId}/detections`
- Payload: JSON 格式的檢測結果 (bbox, class, confidence)

支援模型:

- YOLOv8 系列 (Nano, Small, Medium 等)
- YOLOv7 系列
- 可擴展其他物體檢測模型

---

4. fence

技術: Python + Paho MQTT
端口: 無對外端口
職責:

- 訂閱 MQTT 的物體檢測結果
- 根據 `cameras.json` 中的虛擬圍欄配置判斷入侵
- 使用 Ray Casting 演算法判斷物體是否進入圍欄區域
- 產生事件並儲存至 PostgreSQL
- 透過 MQTT 發布事件通知
- 實作冷卻機制避免重複觸發

MQTT 訂閱/發布:

- 訂閱: `vision/+/detections`
- 發布: `vision/{cameraId}/events`

資料庫操作:

- 插入事件至 `events` 表

---

5. record

技術: Python + FFmpeg
端口: 無對外端口
職責:

- 全時段持續錄影 (按時間分段)
- 維護短期循環緩衝 (Buffer Recording)
- 監聽事件 MQTT 並裁剪事件片段
- 自動將 `.ts` 格式轉檔為 `.mp4`
- 更新資料庫中的事件縮圖路徑

錄影模式:

1. **持續錄影**: 每 5 分鐘分段,儲存於 `recordings/`
2. **緩衝錄影**: 1 秒分段,保留最近 10 秒,儲存於 `recordings_buffer/`
3. **事件裁剪**: 從緩衝或持續錄影中裁剪事件前後片段

環境變數:

- `SEGMENT_SECONDS`: 持續錄影分段長度 (預設 300 秒)
- `EVENT_PRE_SECONDS`: 事件前秒數 (預設 10 秒)
- `EVENT_POST_SECONDS`: 事件後秒數 (預設 10 秒)
- `EVENT_BUFFER_ENABLED`: 是否啟用緩衝錄影 (預設 1)

---

6. go2rtc

技術: Go
端口: 1984 (HTTP API), 8554 (RTSP), 8555 (WebRTC)
職責:

- 多協議串流伺服器 (RTSP/WebRTC)
- 管理所有攝影機串流的註冊與分發
- 提供低延遲 WebRTC 串流給瀏覽器
- 提供 RESTful API 管理串流
- 支援串流轉碼與協議轉換

API 端點:

- `GET /api/streams`: 列出所有串流
- `PUT /api/streams?name={name}&src={url}`: 註冊新串流
- `GET /api/webrtc?src={name}`: WebRTC 播放端點

配置檔: `go2rtc/go2rtc-config.yaml`

---

7. frontend

技術: Next.js 16 + React
端口: 3000 (HTTP)
職責:

- 提供 Web 儀表板界面
- 即時串流播放 (WebRTC)
- 攝影機配置管理
- 事件瀏覽與篩選
- 錄影管理與下載
- 虛擬圍欄視覺化配置
- 系統狀態監控

Server Actions:

- 讀寫 `cameras.json`、`events.json`、`recordings.json`
- 直接操作檔案系統 (新增/刪除事件與錄影)
- 與 go2rtc、PostgreSQL 整合

環境變數:

- `GO2RTC_API_URL`: go2rtc 內部 API 地址
- `NEXT_PUBLIC_GO2RTC_URL`: 給瀏覽器使用的 go2rtc 公開地址

---

8. mqtt

技術: Eclipse Mosquitto 2
端口: 1883 (MQTT)
職責:

- MQTT Broker 提供訊息中介服務
- 支援發布/訂閱模式
- 連接 model_launcher、fence、record 模組
- 提供事件驅動架構的核心通訊層
- 支援帳密認證 (可選)

主要 Topics:

- `vision/{cameraId}/detections`: 檢測結果
- `vision/{cameraId}/events`: 圍欄事件

配置檔: `MQTT/mosquitto.conf`

---

9. postgres

技術: PostgreSQL 16
端口: 5432
職責:

- 儲存事件資料 (`events` 表)
- 支援時區感知的時間戳
- 提供索引優化查詢效能
- 資料持久化與 ACID 保證

資料表:

- `events`: 事件記錄 (id, camera_id, class_name, ts, thumbnail, score)

索引:

- `idx_events_camera_ts`: 按攝影機與時間查詢
- `idx_events_class_ts`: 按類別與時間查詢

認證:

- 使用者: `vision_user`
- 密碼: `vision_pass`
- 資料庫: `vision`

---

10. pgadmin

技術: pgAdmin 4
端口: 5050 (HTTP)
職責:

- PostgreSQL 資料庫管理界面
- 提供視覺化查詢工具
- 資料表瀏覽與編輯
- SQL 查詢執行
- 資料庫監控與維護

登入資訊:

- Email: `admin@example.com`
- 密碼: `admin_pass`

使用場景:

- 開發階段資料庫調試
- 手動查詢事件資料
- 資料庫結構檢視

API 快速參考

go2rtc API 端點（Port 1984）

HTTP方法	API端點	功能說明	主要用途
GET	`/api/streams`	獲取所有串流列表	發現可用串流、獲取串流源信息
PUT	`/api/streams?name={name}&src={url}`	註冊新串流源	動態添加串流、遠程攝像頭集成
GET	`/api/webrtc?src={name}`	WebRTC 播放端點	低延遲瀏覽器播放
GET	`/api/ws?src={name}`	WebSocket 端點	實時串流數據

MQTT 主題（Port 1883）

主題模式	消息類型	訊息內容	發布者	訂閱者	主要用途
`vision/{cameraId}/detections`	發布/訂閱	物體檢測結果（bbox, class, confidence）	Runner	Fence	實時檢測結果分發
`vision/{cameraId}/events`	發布/訂閱	虛擬圍欄事件（event_id, fence_name, timestamp）	Fence	Record	觸發事件錄影

RTSP 串流端點（Port 8554）

串流格式	RTSP端點	功能說明	主要用途
RTSP	`rtsp://localhost:8554/{streamName}`	標準 RTSP 串流	VLC/ffplay 播放、第三方 NVR 集成

串流格式	RTSP端點	功能說明	主要用途
範例	`rtsp://localhost:8554/cam1_raw`	原始攝像頭串流	未處理的原始視頻
範例	`rtsp://localhost:8554/cam1overlay`	疊加處理後串流	帶有檢測框和標籤的視頻

PostgreSQL 數據表（Port 5432）

events 表結構

欄位名	數據類型	約束條件	默認值	功能說明	使用範例
`id`	TEXT	PRIMARY KEY	-	事件唯一標識符	`"evt_003"`
`camera_id`	TEXT	NOT NULL	-	攝像頭 ID	`"cam1_raw"`
`class_name`	TEXT	NOT NULL	-	檢測物體類別	`"person"`, `"falling_down"`
`ts`	TIMESTAMPTZ	NOT NULL	-	事件發生時間戳（帶時區）	`2026-01-02T12:10:45Z`
`thumbnail`	TEXT	NULLABLE	NULL	事件縮圖文件名或路徑	`"evt_003.mp4"`
`score`	DOUBLE PRECISION	NULLABLE	NULL	檢測置信度分數 (0.0-1.0)	`0.88`
`created_at`	TIMESTAMPTZ	NOT NULL	NOW()	記錄創建時間（自動生成）	`2026-01-02T12:10:46Z`

索引:

- `idx_events_camera_ts`: 複合索引 (`camera_id`, `ts DESC`) - 用於按攝像頭查詢最近事件
- `idx_events_class_ts`: 複合索引 (`class_name`, `ts DESC`) - 用於按類別查詢最近事件

配置文件接口

配置文件	文件路徑	數據格式	功能說明	訪問方式
`cameras.json`	`/app/share/cameras.json`	JSON	攝像頭配置、虛擬圍欄	文件系統讀寫、REST API
`models.json`	`/app/share/models.json`	JSON	AI 模型配置	文件系統讀取

API Payload 範例

PostgreSQL 事件管理（Port 5432）

查詢事件範例

查詢特定攝像頭最近事件

```
SELECT * FROM events
WHERE camera_id = 'cam1_raw'
ORDER BY ts DESC
LIMIT 10;
```

按類別統計事件數量（過去24小時）

```
SELECT class_name, COUNT(*) as count
FROM events
WHERE ts > NOW() - INTERVAL '24 hours'
GROUP BY class_name
ORDER BY count DESC;
```

查詢高置信度事件

```
SELECT * FROM events
WHERE ts BETWEEN '2026-01-02 00:00:00' AND '2026-01-02 23:59:59'
  AND score > 0.8
ORDER BY ts;
```

插入新事件

```
INSERT INTO events (id, camera_id, class_name, ts, thumbnail, score)
VALUES ('evt_004', 'cam1_raw', 'falling_down', NOW(), 'evt_004.mp4', 0.92);
```

刪除舊事件（保留最近30天）

```
DELETE FROM events
WHERE ts < NOW() - INTERVAL '30 days';
```

---

go2rtc連線檢查 API

GET /api/go2rtc/health - 檢查 go2rtc 狀態

```
// Response
{
  "status": "ok",
  "message": "go2rtc is reachable",
  "url": "http://go2rtc:1984",
  "streamCount": 2,
  "streams": ["cam1_raw", "cam1overlay"]
}
```

攝像頭配置 API

camera-config - 獲取攝像頭配置

```
// Response
{
  "webrtcServerUrl": "http://localhost:1984",
  "availableDetectionObjects": ["person", "book", "tv", "falling_down"],
  "cameras": [
    {
      "id": "cam1_raw",
      "name": "cam1_raw",
      "resolution": "1920x1080",
      "fps": 30,
      "modelId": "YOLOv8_V1",
      "detectObjects": ["person"],
      "virtualFences": [
        {
          "name": "Zone 1",
          "points": [
            {"x": 0.223, "y": 0.206},
            {"x": 0.857, "y": 0.381}
          ],
          "enabled": true,
          "detectObjects": ["person", "tv"]
        }
      ]
    }
  ]
}
```

模型配置文件

models.json - AI 模型配置

```
{
  "models": [
    {
      "type": "object_detection",
      "name": "YOLOv8_V1",
      "runner": "/runner/yolov8_inference.py",
      "weights": "/models/yolov8_coco.pt",
      "class_file": "/class/yolov8_coco.txt",
      "inputSize": [640, 640],
      "classes": [
        { "id": 0, "name": "person" }
      ],
      "defaultConfidence": 0.5,
      "nmsThreshold": 0.45,
      "device": ""
    },
    {
      "type": "object_detection",
      "name": "YOLOv7 coco",
      "runner": "/runner/yolov7_inference.py",
      "weights": "/models/yolov7_coco.pt",
      "class_file": "/class/yolov7_coco.txt",
      "inputSize": [640, 640],
      "classes": [
        { "id": 1, "name": "mobile_phone" }
      ],
      "defaultConfidence": 0.5,
      "nmsThreshold": 0.45,
      "device": "cuda:0"
    }
  ]
}
```

MQTT Message Payloads (Port 1883)

Topic: vision/{cameraId}/detections - 檢測結果

```
{
  "camera_id": "cam1_raw",
  "timestamp": "2026-01-02T03:15:30.123Z",
  "detections": [
    {
      "class_id": 0,
      "class_name": "person",
```

```
      "confidence": 0.95,
      "bbox": {
        "x1": 0.25,
        "y1": 0.30,
        "x2": 0.45,
        "y2": 0.80
      }
    }
  ],
  "frame_width": 1920,
  "frame_height": 1080
}
```

Topic: vision/{cameraId}/events - 虛擬圍欄事件

```
{
  "event_id": "evt_003",
  "camera_id": "cam1_raw",
  "class_name": "person",
  "fence_name": "Zone 1",
  "timestamp": "2026-01-02T03:15:30Z",
  "score": 0.88,
  "action": "fence_enter"
}
```

附錄

連接信息總覽

服務名稱	通信協議	連接地址	認證憑證
Next.js Frontend	HTTP	`http://localhost:3000`	-
go2rtc API	HTTP	`http://localhost:1984`	-
go2rtc RTSP	RTSP	`rtsp://localhost:8554`	部分串流需要認證
go2rtc WebRTC	WebRTC	`http://localhost:8555`	-
MQTT Broker	MQTT	`mqtt://localhost:1883`	-
PostgreSQL	PostgreSQL	`localhost:5432`	`vision_user` / `vision_pass`

文檔版本: 1.0
最後更新: 2026-01-02
維護者: Action Recognition Server Team