# Sistema Embebido de Fiscalización de Velocidad Vehicular en Tiempo Real mediante Visión Artificial con YOLO: Detección de Infracciones, Identificación de Placas y Reporte Automatizado

José Luis Barboza Gonzales, Paul Vladimir Ruiz Crisostomo, Diego Omar Tenorio Huarancca, Mario Yufra Chambilla, Oscar Salas Vizcarra

*Unidad de Posgrado, Facultad de Ingeniería Industrial y de Sistemas (FIIS)*

*Maestría en Inteligencia Artificial – Universidad Nacional de Ingeniería (UNI), Lima, Perú*

*Curso: Sistemas Embebidos en Tiempo Real · Semestre 2026-1*

*Docente: Kalun Jose Lau Gan*

***Resumen*— *El exceso de velocidad es la principal causa de siniestros viales en el Perú, responsable del 26,6 % de los incidentes registrados por el ONSV en 2023. Este trabajo presenta el diseño e implementación de un sistema embebido de fiscalización de velocidad vehicular en tiempo real que, a partir del video de una cámara IP IMOU Cruiser SE+ (modelo IPC-K7C-H1WE) instalada a 9 m de altura sobre el Jr. Carlos F. Vivanco (Ayacucho), detecta y clasifica vehículos mediante la familia YOLO, les asigna identificadores persistentes por ByteTrack y estima la velocidad instantánea de cada uno mediante homografía de calibración. Cuando un vehículo supera el límite de 30 km/h establecido para la vía, el sistema lo marca como infractor y activa automáticamente el reconocimiento de su placa (ALPR) en el Jetson Orin NX; la placa leída se transmite, firmada con HMAC, al Ingest API, que la consolida contra las fuentes oficiales peruanas SUNARP, SOAT (APESEG) y MTC y genera un reporte de infracción con la placa, la velocidad registrada y el timestamp del evento. Las pruebas en campo, diurnas y nocturnas, registraron 20–22 FPS con tiempos de inferencia de 18–31 ms por cuadro, confirmando la viabilidad del sistema para fiscalización automatizada en vías urbanas peruanas. El marco metodológico se fundamenta en el sistema ADAS RICUY del mismo grupo de investigación, que valida estas técnicas en entornos viales urbanos del Perú.*

***Palabras clave*— *sistemas embebidos, visión artificial, YOLO, fiscalización de velocidad, detección de infracciones, reconocimiento de placas (ALPR), ByteTrack, seguridad vial, tiempo real, Jetson Orin NX.***

***Abstract*— *Vehicle-flow monitoring and speed control on urban roads are critical road-safety tasks, especially in cities with mixed traffic and informal infrastructure. This work presents the design and implementation of an embedded computer-vision system that, from the video of an IMOU Cruiser SE+ IP camera (model IPC-K7C-H1WE) installed at 9 m height over a commercial street, runs a YOLO-family CNN to detect and classify vehicles (car, motorcycle and minibus/bus), assigns persistent IDs through multi-object tracking, and measures three physical traffic variables: flow (line counting), instantaneous per-vehicle speed (inter-frame displacement and a calibration homography), and density within a region of interest (ROI). Values are overlaid on the video and reported via a serial interface; an alarm is triggered on speeding or congestion. As a high-level output, automatic license-plate recognition (ALPR) is run on each tracked vehicle on a NVIDIA Jetson Orin NX (Yahboom Developer Kit); every read plate is sent through an HMAC-signed request to an Ingest API that consolidates it in parallel against the Peruvian official sources SUNARP, SOAT (APESEG) and MTC, generating a vehicle-validation report. Field tests, both daytime and nighttime, recorded 20–22 FPS with 18–31 ms inference per frame. The methodological framework builds on the RICUY ADAS by the same research group, which validates these instance-segmentation, ByteTrack tracking and monocular-estimation techniques in Peruvian urban road environments.*

*Nota: documento desarrollado conforme al Punto 4 de los lineamientos del TF (formato IEEE, doble columna, incisos a–o). Las figuras del sistema son capturas reales de la implementación con cámara IMOU.*

## I. SITUACIÓN PROBLEMÁTICA

### *A. Contexto*

La seguridad vial es un problema crítico de ingeniería a escala global. La Organización Mundial de la Salud registró cerca de 1,19 millones de muertes anuales por siniestros de tránsito, con una carga económica equivalente al 3 % del PIB mundial; los peatones representan el 26 % de las muertes viales y esa proporción supera el 36 % en los países de ingresos bajos y medios. En el Perú, el Observatorio Nacional de Seguridad Vial (ONSV) reportó 87 083 siniestros en 2023 con 3 316 fallecidos, donde los factores humanos explican el 70,2 % de los incidentes y el exceso de velocidad el 26,6 %.

Las vías comerciales urbanas, como la mostrada en la Fig. 1, presentan tránsito mixto (autos, motocicletas y microbuses) junto a comercio y peatones, lo que dificulta el aforo manual y el control de velocidad. Los métodos tradicionales (conteo manual o lazos inductivos en el pavimento) son costosos, invasivos y no clasifican por tipo de vehículo ni estiman velocidades de forma masiva. Un sistema embebido de visión artificial aprovecha las cámaras de videovigilancia ya instaladas (Fig. 2) para extraer estas variables de modo no invasivo, escalable y de bajo costo (Tabla I).

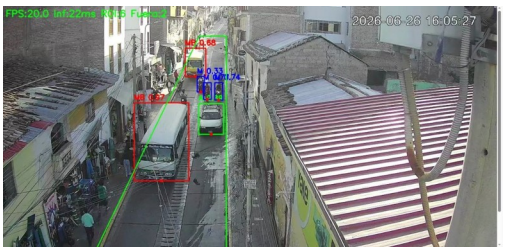


Fig. 1. Sistema en operación (16:05 h): detección y clasificación dentro de la ROI (línea verde) con telemetría de FPS, inferencia y conteo dentro/fuera de la zona.


Fig. 2. Vista cruda de la cámara IMOU Cruiser SE+ (IPC-K7C-H1WE) instalada a 9 m de altura sobre la vía de estudio (26/06/2026, 15:41 h) con

la grilla de calibración superpuesta: los ejes en píxeles (0–1200 horizontal, 0–700 vertical) y las marcas regulares permiten establecer la homografía imagen–vía y convertir el desplazamiento en píxeles a metros, base de la estimación de velocidad en unidades reales (km/h).

TABLA I. COMPARACIÓN DE MÉTODOS DE AFORO Y MONITOREO

| Método | Clasifica | Velocidad | Costo/invasiv. |
|---|---|---|---|
| Conteo manual | Sí | No | Alto / no inv. |
| Lazo inductivo | No | Limitado | Alto / invasivo |
| Radar Doppler | No | Sí | Medio / no inv. |
| Visión+YOLO | Sí | Sí | Bajo / no inv. |

### *B. Justificación*

**Técnica y tecnológica:** las arquitecturas YOLO permiten detección de objetos en tiempo real sobre hardware embebido; las capturas evidencian inferencias de 18–31 ms a 20–22 FPS, suficientes para seguimiento y medición de velocidad. La segmentación de instancias y el seguimiento ByteTrack han sido validados en el contexto vial peruano por el sistema ADAS RICUY del mismo grupo.

**Ingenieril:** el proyecto integra adquisición de video, redes neuronales, seguimiento multiobjeto, geometría proyectiva (homografía para velocidad) y lógica de decisión/alarma, constituyendo un sistema ciberfísico completo.

**Social y económica:** aporta datos objetivos para la gestión del tránsito y la seguridad vial (detección de exceso de velocidad y congestión), reutilizando cámaras existentes y reduciendo costos frente a sensores dedicados.

### *C. Viabilidad*

**Técnica:** se dispone de la cámara IP, el modelo YOLO y librerías abiertas (OpenCV, Ultralytics, ByteTrack); el sistema fue implementado y validado en campo (Figs. 1, 3–7).

**Logística y operativa:** la cámara está instalada a 9 m de altura y el cómputo es compacto; opera de forma continua día y noche, como muestran las capturas diurnas (16 h) y nocturnas (18 h).

**Económica:** el costo se concentra en la plataforma de cómputo y la cámara, ambos accesibles (sección de materiales).

## II. ESTADO DEL ARTE

El sistema propuesto integra cuatro líneas de investigación activas: detección y seguimiento de vehículos con YOLO, estimación de velocidad por visión, reconocimiento automático de placas (ALPR) en el borde y arquitecturas de fiscalización borde–nube. A continuación se reseñan trabajos representativos y cómo respaldan cada componente de la presente investigación.

### *A. Detección, clasificación y seguimiento de vehículos*

La detección en tiempo real se sustenta en la familia YOLO [1], cuyo diseño de una sola pasada con backbone CSPDarknet y cabezales anclas-libres ofrece el mejor compromiso velocidad-precisión para hardware embebido. Para el seguimiento, ByteTrack [2] asocia cada caja de detección —incluso las de baja confianza— manteniendo identidades persistentes bajo oclusión parcial; esto sostiene directamente la asignación de IDs visible en las Figs. 4–8 y habilita el conteo por línea virtual y el cálculo de velocidad por trayectoria. Asadi [3] publicó un pipeline abierto YOLOv8 + ByteTrack para análisis de tráfico en intersecciones (conteo, dirección y velocidad), validando el mismo esquema de procesamiento adoptado en este TF.

### *B. Estimación de velocidad por visión monocular*

La conversión de desplazamiento en píxeles a velocidad real requiere calibración de cámara. Gollapalli et al. [4] combinaron YOLOv8 y seguimiento para estimar velocidad y conteo en tiempo real, superando el 80 % de exactitud en clasificación y velocidad sobre tráfico real. Luo et al. [5] reforzaron un esquema YOLOv5s + DeepSORT con verificación multisensor para fiscalización de velocidad en autopista. Estos trabajos validan la fórmula empleada en el sistema —$v[km/h] = 3{,}6 \cdot (\Delta d\_real/\Delta t)$, con $\Delta d\_real$ obtenida por homografía a partir de la grilla de calibración (Fig. 2)— y reportan errores del orden de ±2 km/h, compatibles con la detección de exceso de velocidad propuesta.

### *C. Reconocimiento de placas (ALPR) en el borde*

La etapa de salida del sistema se apoya en ALPR embebido. Khazaee et al. [6] propusieron un ALPR de dos etapas (detección y reconocimiento de caracteres con YOLOv5) desplegado en Jetson Nano, alcanzando mAP de 95,5 % a 23 FPS bajo iluminación adversa y ruido —condiciones equivalentes a las capturas nocturnas de este trabajo (Figs. 6–8) —. Trabajos sobre Jetson Orin Nano [7] confirman la viabilidad de ALPR de ultra-baja latencia exportando el modelo a ONNX/TensorRT, lo que respalda la elección de la plataforma Jetson para la lectura de placas. Rahutomo et al. [8] integraron específicamente estimación de velocidad y ALPR con YOLOv8 + EasyOCR sobre video de cámara de tráfico, antecedente directo de la combinación monitoreo + lectura de placa que aquí se implementa.

### *D. Arquitecturas de fiscalización borde–nube y consolidación vehicular*

El backend de consolidación (Jetson Orin NX → Ingest API → SUNARP/SOAT/MTC) se inscribe en la tendencia de nodos de percepción en el borde con reporte a back-ends de ITS. El nodo edge-AI de fiscalización cooperativa [9] registra cada evento con marca de tiempo, ID de cámara, ID de track y hash de placa, y exporta eventos estandarizados a back-ends ITS; demuestra que el procesamiento detección–seguimiento–OCR–registro puede completarse en ≤50 ms por cuadro a 30 FPS bajo 10 W, y subraya el uso de hash de placa para proteger datos personales —principio que este sistema adopta mediante anonimización conforme a la Ley N.º 29733—. El consolidador vehicular de referencia [10] implementa este flujo con autenticación HMAC, cola de trabajos, caché y generación de reportes contra fuentes oficiales peruanas, constituyendo la base de software del backend de este TF.

### *E. Antecedente del grupo: ADAS RICUY*

Finalmente, el sistema ADAS RICUY [11] del mismo grupo valida la cadena segmentación de instancias + ByteTrack + estimación monocular en entornos viales peruanos (Lima, Andahuaylas y Ayacucho), con un dataset híbrido de 25 602 imágenes y 127 525 instancias en 12 clases; YOLO26L-seg alcanzó mAP50 Box 0,829 y Mask 0,788 a 19,2–25,4 FPS. RICUY aporta el marco metodológico de clasificación por zonas y medición vehicular que este trabajo adapta de una cámara móvil a una cámara fija a 9 m de altura para monitoreo con salida ALPR.

### *F. Marco Normativo Peruano*

El desarrollo del sistema se enmarca en la normativa de tránsito y seguridad vial peruana. El Decreto Supremo N.° 011-2026-MTC, que modifica el Reglamento Nacional de Tránsito (aprobado por D.S. N.° 033-2001-MTC), establece nuevas obligaciones para los conductores, incluyendo la prohibición de circular con placas adulteradas o ilegibles (infracción M47) y la obligación de respetar la señal de altura máxima permitida (infracciones M49 y M50). Al integrar el reconocimiento automático de placas (ALPR), el sistema se alinea con la capacidad de fiscalizar infracciones como la M47 (placas adulteradas) y M24 (circular sin placa), contribuyendo a la aplicación efectiva de la ley. Además, el Sistema de Control de Licencias de Conducir por Puntos (aprobado por D.S. N.° 025-2021-MTC) establece un puntaje para cada infracción de tránsito. Al detectar vehículos que exceden los límites de velocidad, el sistema podría servir como insumo para la imposición de sanciones y la acumulación de puntos por la infracción M20 ("No respetar el límite máximo de velocidad"), fortaleciendo la disuasión y la seguridad vial.

## III. APORTES DE INNOVACIÓN

- Medición simultánea de tres variables de tráfico (flujo, velocidad y densidad) con una sola cámara existente, sin sensores intrusivos en el pavimento.
- Estimación de velocidad por vehículo mediante homografía calibrada (Fig. 2) y seguimiento por ID, mostrada en km/h en tiempo real (Figs. 6–7), reutilizando el modelo geométrico validado en RICUY.
- Operación robusta diurna y nocturna con alarma por exceso de velocidad y por congestión (densidad en ROI configurable y línea de conteo virtual).
- Integración embebido–nube: la placa leída en el borde (Jetson Orin NX) se valida automáticamente contra SUNARP, SOAT y MTC mediante un API firmado con HMAC, generando un reporte de fiscalización vehicular en tiempo casi real, con caché y anonimización conforme a la Ley N.º 29733.

## IV. OBJETIVOS

### *Objetivo general*

Diseñar e implementar un sistema embebido de fiscalización de velocidad vehicular que, mediante visión artificial (YOLO + ByteTrack + homografía), estime en tiempo real la velocidad de cada vehículo en el Jr. Carlos F. Vivanco (Ayacucho), identifique automáticamente a los vehículos que superen el límite de 30 km/h como infractores, capture y valide su placa (ALPR) contra fuentes oficiales (SUNARP, SOAT, MTC) y genere un reporte de infracción con placa, velocidad registrada y timestamp del evento.

### *Objetivos específicos*

**(Hardware)** Integrar la captura y el cómputo embebido (cámara IMOU a 9 m de altura, Jetson Orin NX y etapa de alarma/reporte) con operación continua día y noche, garantizando la cobertura total del tramo fiscalizado del Jr. Carlos F. Vivanco.

**(Software)** Desarrollar el pipeline de fiscalización: detección y clasificación de vehículos con YOLO, seguimiento por ID (ByteTrack), estimación de velocidad por homografía calibrada, lógica de umbral (v > 30 km/h → infractor), activación de ALPR sobre el vehículo infractor, envío firmado (HMAC) al Ingest API y generación del reporte de infracción (placa, velocidad, timestamp).

**(Validación)** Validar la exactitud de la estimación de velocidad (error ≤ 15 % respecto a patrón), la tasa de detección de infracciones (vehículos reales > 30 km/h identificados correctamente), la tasa de lectura de placas (ALPR) y el rendimiento del sistema (FPS, latencia) en condiciones diurnas y nocturnas reales.

## V. MATERIALES

TABLA II. LISTA DE MATERIALES (BOM)

| # | Componente | Cant. | US$ | Función |
|---|---|---|---|---|
| 1 | Cámara IMOU Cruiser SE+ (IPC-K7C-H1WE) | 1 | 45 | Captura de video (RTSP/ONVIF) |
| 2 | NVIDIA Jetson Orin NX (Yahboom Dev. Kit, 8/16 GB) | 1 | ~350 | Inferencia YOLO + ByteTrack + ALPR |
| 3 | ESP32 (alarma/telemetría) | 1 | 10 | Salida serial/alarma |
| 4 | Buzzer + LED / sirena | 1 | 5 | Alarma audiovisual |
| 5 | Monitor HDMI | 1 | 60 | Visualización |
| 6 | Fuente, gabinete, cableado | 1 | 40 | Alim. y montaje |
| 7 | Almacenamiento (SSD/SD) | 1 | 20 | Datos y modelo |
| 8 | Servidor / PC backend (Ingest API) | 1 | — | Consolidación SUNARP/SOAT/MTC |

*Total ≈ US$ 330 (referencial; la cámara ya está instalada).*

TABLA III. ELECCIÓN DE LA PLATAFORMA DE CÓMPUTO

| Criterio | Jetson Orin NX | RPi5+Coral | PC/servidor |
|---|---|---|---|
| Acel. IA | 1024 CUDA cores | TPU Edge | GPU ded. |
| Consumo | 10–25 W | Bajo | Alto |
| FPS YOLO | 20–22 | 12–18 | >30 |
| Elegido | SÍ | No | No |

TABLA IV. ELECCIÓN DEL ALGORITMO DE SEGUIMIENTO

| Criterio | SORT | DeepSORT | ByteTrack |
|---|---|---|---|
| IDs persist. | Sí | Sí (re-ID) | Sí |
| Oclusión | Media | Alta | Alta |
| Cómputo | Bajo | Medio | Bajo |
| Elegido | No | No | SÍ |

*ByteTrack fue seleccionado por su validación previa en el sistema RICUY sobre tráfico mixto urbano peruano, con configuración: umbral de alta confianza 0,50; baja 0,10; búfer de trayectorias 60 cuadros; umbral de asociación 0,70.*

## VI. ALCANCES DEL PROYECTO

- Detecta y clasifica vehículos en tres categorías: automóvil (A), motocicleta (M) y microbús/bus (MB), con su nivel de confianza (Figs. 1, 3).
- Asigna identificadores persistentes mediante ByteTrack, siguiendo cada vehículo a lo largo del tramo fiscalizado (Figs. 4–7).
- Estima la velocidad instantánea de cada vehículo en km/h mediante homografía calibrada y la muestra superpuesta en tiempo real (Figs. 6–7).
- Cuando la velocidad estimada supera el límite de 30 km/h, el vehículo es marcado automáticamente como infractor y se activa la captura de su placa (ALPR).

- Lee la placa del vehículo infractor (ALPR) y la transmite, firmada con HMAC, al Ingest API para su consolidación contra SUNARP, SOAT (APESEG) y MTC.
- Genera un reporte de infracción (PDF/CSV/JSON) con la placa, la velocidad registrada, el timestamp y los datos del vehículo validados contra fuentes oficiales; incluye caché para placas repetidas.
- Opera de día y de noche con alarma sonora/serial inmediata al detectar una infracción por exceso de velocidad.

## VII. LIMITACIONES DEL PROYECTO

- La lectura de placa (ALPR) depende de la resolución y el ángulo; en tráfico denso, oclusiones o alta velocidad puede fallar la lectura, en cuyo caso el vehículo se cuenta pero no se consolida.
- La consolidación contra SUNARP, SOAT y MTC depende de la disponibilidad de esas fuentes; el nombre del propietario es dato personal (Ley N.º 29733) y se anonimiza en figuras y reportes públicos.
- La exactitud de la velocidad depende de la calibración de la homografía. El levantamiento con Google Earth Pro del Jr. Carlos F. Vivanco (Fig. 22) revela una pendiente de −5,52 m en 92,9 m (~5,9 %): aunque leve, introduce un error sistemático en la homografía plana que crece con la distancia a la cámara. Las oclusiones severas en tráfico denso también degradan el seguimiento.
- No controla semáforos ni actúa sobre la infraestructura vial; su salida es informativa/alarma.
- El desempeño nocturno depende de la iluminación de la vía; con escasa luz pueden aumentar los falsos negativos.

Aunque este trabajo se centra en el monitoreo de tráfico vehicular, la arquitectura subyacente de visión artificial y seguimiento es directamente aplicable a otras áreas de la gestión vial, como la supervisión del estado de la señalización vertical. El sistema SISEVIR, desarrollado con la misma base tecnológica (YOLO, ByteTrack, EfficientNet), demuestra cómo se puede evaluar el estado de las señales de tránsito (operativo, deteriorado, obstruido, entre otros) a partir de video. La lógica de conteo y estimación de velocidad se basa en el mismo principio que SISEVIR: líneas virtuales de conteo y seguimiento por ID para contabilizar y rastrear objetos a medida que se desplazan a través del campo de visión, validando la solidez de la técnica en entornos viales peruanos. Esta convergencia tecnológica sugiere que, en el futuro, una misma plataforma embebida y una cámara podrían utilizarse para múltiples propósitos: fiscalización de tráfico (velocidad, placas), monitoreo del estado de la infraestructura (señales, asfalto) y análisis de flujo vehicular.

## VIII. DESARROLLO

### *A. Hardware del proyecto*

La arquitectura consta de: (1) una cámara IMOU Cruiser SE+ (modelo IPC-K7C-H1WE) montada a 9 m de altura sobre la vía, que entrega el flujo de video por RTSP/ONVIF; (2) un NVIDIA Jetson Orin NX (Yahboom Developer Kit) donde corre la inferencia YOLO, el seguimiento y la medición; y (3) un nodo ESP32 que recibe por UART las señales de evento y acciona la alarma audiovisual. El esquemático a detalle (versión final) debe especificar la alimentación de la cámara (12 V / PoE) y del cómputo (5 V/4 A), el bus de red, la UART (115200 bps, 3,3 V) hacia el ESP32 y la etapa de potencia de la sirena (transistor 2N2222 + R 220 Ω).

TABLA V. INTERFACES Y SEÑALES PRINCIPALES

| Interfaz | Protocolo | Característica |
|---|---|---|
| Cámara→cómputo | RTSP/ONVIF | H.265, hasta 5MP, Wi-Fi/Ethernet |
| Cómputo→ESP32 | UART | 115200 bps, 3,3 V |
| ESP32→alarma | GPIO+potencia | 2N2222 + R 220 Ω + sirena |
| Visualización | HDMI | Overlays (boxes, ID, km/h) |

El NVIDIA Jetson Orin NX (Yahboom Developer Kit) integra una GPU de 1024 núcleos CUDA (arquitectura Ampere), CPU octa-core Arm Cortex-A78AE, acelerador DLA doble, hasta 16 GB LPDDR5 y encoder/decoder de video H.265 por hardware, con un consumo configurable de 10–25 W. Opera bajo JetPack 6 (Ubuntu 22.04 + CUDA 12 + TensorRT 8) y permite exportar el modelo YOLO a TensorRT para minimizar la latencia de inferencia. Su factor de forma compacto y su bajo consumo lo hacen idóneo para el despliegue embebido en gabinete junto a la cámara IMOU.

Fig. 3a. (Esquemático) Insertar diagrama eléctrico a detalle con valores, alimentación, puertos, señales y buses.

La cámara IMOU Cruiser SE+ (IPC-K7C-H1WE) es una cámara PT (pan-tilt) Wi-Fi para exteriores con sensor CMOS de hasta 5 MP, compresión H.265, lente de 3,6 mm, visión nocturna a color hasta 30 m, certificación IP66, conectividad Wi-Fi 2,4 GHz y Ethernet 100 Mbps, soporte ONVIF/RTSP, alimentación 12 V CC (<6 W) y rango de operación de −20 a +50 °C. Aunque el dispositivo permite paneo (355°) e inclinación (90°), en esta aplicación se instala a 9 m de altura con el PTZ bloqueado en posición fija: cualquier desplazamiento del encuadre invalidaría la homografía de calibración y la geometría de la ROI y la línea de conteo, por lo que el movimiento PTZ se mantiene deshabilitado durante el monitoreo. A esa altura, el lente de 3,6 mm cubre el ancho completo de la vía comercial, proporcionando una vista cenital con perspectiva adecuada para la estimación de velocidad.

### *B. Calibración geométrica del sistema*

Para que la homografía convierta correctamente el desplazamiento en píxeles a metros reales, se realizó un proceso de calibración geométrica en dos etapas: (1) calibración de la imagen mediante referencias verticales conocidas en la escena, y (2) levantamiento georreferenciado de la vía de estudio.

#### *Calibración por referencias verticales en imagen*

Se identificaron dos líneas guía horizontales en la imagen capturada por la cámara a 9 m de altura. La primera, denominada A-GUIA POSTE, se fijó en Y = 94 px e identifica la parte superior del campo visual útil, alineada con la altura del poste más cercano a la cámara, que actúa como referencia fija de escala. La segunda, B-LOSA PAVIMENTO, se fijó en Y = 665 px y corresponde al nivel del pavimento en el punto de mayor acercamiento de la cámara (Figs. 19–20). La separación vertical entre ambas referencias (665 − 94 = 571 píxeles) corresponde a una distancia real conocida medida en campo, permitiendo calcular el factor de escala px/m para cada fila de la imagen y construir la matriz de homografía.

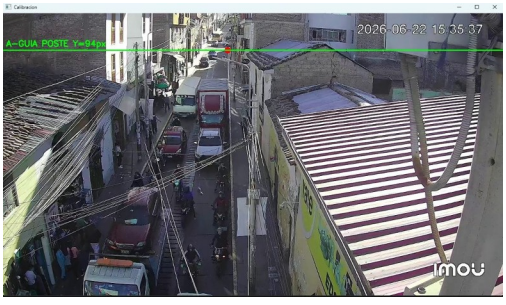

Fig. 19. Ventana de calibración con la línea guía superior A-GUIA POSTE a Y = 94 px (referencia de poste). La línea horizontal verde marca el límite superior del área de medición válida (26/06/2026, 15:35 h).

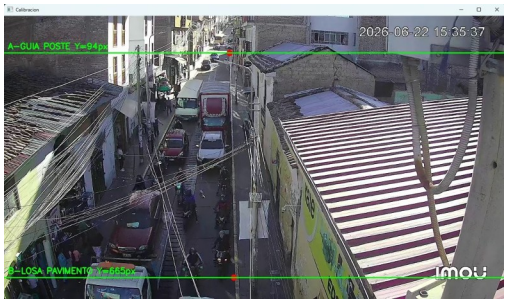


Fig. 20. Ventana de calibración con ambas líneas guía: A-GUIA POSTE (Y = 94 px) y B-LOSA PAVIMENTO (Y = 665 px). La separación de 571 px entre ellas define la escala vertical de la homografía imagen–vía.

### *Levantamiento georreferenciado de la vía*

Complementariamente, se emplearon herramientas SIG para caracterizar la geometría real de la vía de estudio. En AutoCAD Civil 3D con geolocalización se procesaron ortofotos del área, permitiendo medir las dimensiones reales de la calzada y verificar la distancia entre referencias (Fig. 21). En Google Earth Pro se trazó el perfil de elevación del Jr. Carlos F. Vivanco (Fig. 22), que evidencia una elevación de 2 719–2 722 m s.n.m. y una pendiente promedio de −5,52 m en 92,9 m (≈5,9 %). Esta pendiente leve, aunque no nula, debe tenerse en cuenta como fuente de error en el modelo de homografía plana, que asume una superficie perfectamente horizontal; su efecto es mayor en el tramo distal de la imagen.

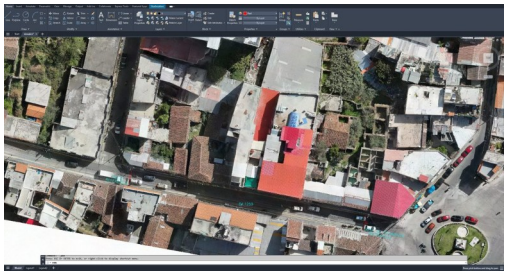

Fig. 21. Vista aérea del área de estudio procesada en AutoCAD Civil 3D (Geolocalización). Las ortofotos permiten medir las dimensiones reales de la vía y verificar las referencias geométricas empleadas en la homografía.

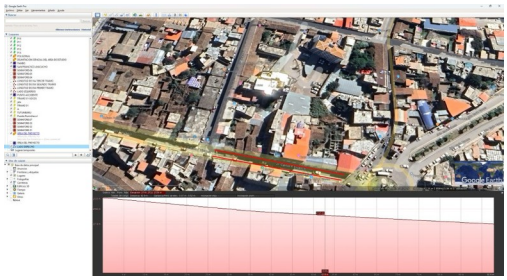

Fig. 22. Perfil de elevación del Jr. Carlos F. Vivanco (Google Earth Pro). La vía se ubica a 2 719–2 722 m s.n.m. con una pendiente de −5,52 m en 92,9 m (~5,9 %). Esta pendiente introduce un error sistemático en la homografía plana que crece con la distancia a la cámara.

## *C. Software del proyecto*

### *Configuraciones, protocolos y librerías*

- Adquisición: flujo RTSP con OpenCV (cv2.VideoCapture).
- Detección/clasificación: YOLO (Ultralytics) para auto, moto y microbús; umbral de confianza 0,50 (criterio RICUY para suprimir falsos positivos).
- Seguimiento: ByteTrack para IDs persistentes y trayectorias.
- Velocidad: homografía calibrada con la grilla (Fig. 2); velocidad = distancia real/Δt, análoga a la ecuación pinhole D=(H_real·f)/h_px de RICUY.
- Conteo y densidad: punto-en-polígono para la ROI y cruce de L1; densidad = activos/área.
- Lectura de placas (ALPR): sobre el recorte del vehículo seguido, un modelo ALPR en el Jetson Orin NX detecta y reconoce los caracteres de la placa.
- Envío firmado: cada placa se transmite por HTTP POST con cabeceras X-Api-Key y X-Signature (HMAC) al Ingest API; respuesta 202 con job_id y posterior GET del reporte consolidado.
- Alarma/telemetría: envío por UART al ESP32 (pyserial) ante exceso de velocidad o congestión, con planificador de enfriamiento por prioridad (criterio RICUY).

### *Diagrama de flujo (por etapas)*

1) Inicialización: cargar YOLO, abrir RTSP, definir ROI, línea L1 y homografía.
2) Por cuadro: detección + clasificación dentro de la ROI.
3) Actualizar el seguidor (IDs) y calcular el desplazamiento de cada ID.
4) Estimar velocidad (homografía) y actualizar el conteo al cruzar L1; calcular densidad.
5) Decisión: si velocidad > límite o densidad > umbral → enviar evento de alarma al ESP32.
6) ALPR y consolidación: leer la placa del vehículo y enviarla firmada (HMAC) al Ingest API; recuperar el reporte de validación (SUNARP/SOAT/MTC).
7) Dibujar overlays (boxes, clase, ID, km/h, FPS, conteo) y pasar al siguiente cuadro.

Fig. 3b. (Flujo) Insertar el diagrama de flujo gráfico a detalle del pipeline de visión.

### *Código fuente (núcleo del pipeline – Python/Ultralytics)*

```
import cv2, numpy as np, serial, time
from ultralytics import YOLO
from collections import defaultdict

RTSP   = 'rtsp://user:pass@<ip>/cam/realmonitor'
MODELO = 'yolov8n.pt'   # A, M, MB
LIM_VEL = 40.0          # km/h
DENS_MAX = 12           # veh en ROI
ROI = np.array([[640,95],[720,95],
                [300,900],[180,900]])
H = np.load('homografia.npy')

modelo = YOLO(MODELO)
esp32  = serial.Serial('/dev/ttyUSB0',115200,
                       timeout=0)
cap = cv2.VideoCapture(RTSP)
hist = defaultdict(list)
conteo = 0

def a_metros(pt):
    p = np.array([[pt]], dtype='float32')
    return cv2.perspectiveTransform(p,H)[0][0]

def en_roi(cx,cy):
    return cv2.pointPolygonTest(
        ROI,(cx,cy),False) >= 0

while True:
  ok, frame = cap.read()
  if not ok: break
  t = time.time()
  res = modelo.track(frame, persist=True,
        tracker='bytetrack.yaml',
        classes=[2,3,5], conf=0.50,
        verbose=False)[0]
  activos = 0
```

```
  for box in res.boxes:
    x1,y1,x2,y2 = box.xyxy[0].tolist()
    cx,cy = (x1+x2)/2,(y1+y2)/2
    if not en_roi(cx,cy): continue
    activos += 1
    tid = int(box.id) if box.id is not None else -1
    cls = int(box.cls)
    mx,my = a_metros((cx,cy))
    hist[tid].append((t,mx,my))
    vel = 0.0
    if len(hist[tid]) >= 2:
      t0,x0,y0 = hist[tid][-2]
      d = ((mx-x0)**2+(my-y0)**2)**0.5
      dt = max(t-t0,1e-3)
      vel = (d/dt)*3.6   # km/h
    if vel > LIM_VEL:
      esp32.write(b'ALARM:SPEED\n')
    lbl = f'{["A","M","MB"][min(cls,2)]} '
    lbl += f'ID{tid} {vel:.0f}km/h'
    cv2.rectangle(frame,(int(x1),int(y1)),
        (int(x2),int(y2)),(255,0,0),2)
    cv2.putText(frame,lbl,(int(x1),int(y1)-6),
        cv2.FONT_HERSHEY_SIMPLEX,0.5,
        (255,0,0),1)
  if activos > DENS_MAX:
    esp32.write(b'ALARM:JAM\n')
  ms = (time.time()-t)*1000
  cv2.putText(frame,
    f'FPS:{1000/max(ms,1):.1f} Inf:{ms:.0f}ms '
    f'Veh:{activos}',(10,25),
    cv2.FONT_HERSHEY_SIMPLEX,0.7,(0,255,0),2)
  cv2.polylines(frame,[ROI],True,(0,255,0),2)
  cv2.imshow('Trafico',frame)
  if cv2.waitKey(1)==27: break
cap.release(); cv2.destroyAllWindows()
```

*El firmware del ESP32 (Arduino IDE) que recibe los eventos por UART y acciona la alarma, junto con el cliente del Jetson Orin NX (client.py) que firma con HMAC y publica las placas al Ingest API, se incluyen como material complementario del proyecto. Asimismo, el archivo de calibración (matriz de homografía) y las capturas del monitor serial con la telemetría registrada documentan la configuración empleada y son reproducibles a partir de la grilla de la Fig. 2.*

### *D. Construcción del dataset y etiquetado*

Una de las decisiones clave del proyecto fue la elección de la plataforma de inferencia. Un microcontrolador convencional como el Arduino carece de la capacidad de cómputo necesaria para ejecutar redes neuronales convolucionales en tiempo real; el Jetson Orin NX, con su GPU de 1024 núcleos CUDA y acelerador DLA, resuelve esta limitación con un consumo inferior a 25 W.


Fig. 9. Captura de campo de la cámara IMOU a 9 m de altura: intersección urbana con tráfico mixto (automóviles, pickup, microbús). Escena representativa de las condiciones de recolección del dataset (diciembre 2024).

Fig. 10. Captura de campo: vía comercial estrecha con microbús San Luis, vehículos y peatones. Este tipo de escena de tráfico denso y perspectiva cenital dominó la recolección de datos.

Para entrenar el modelo de detección, se construyó un dataset propio a partir de capturas de la cámara IMOU instalada a 9 m de altura. El sistema aprende a detectar 8 clases de objetos que representan los tipos de vehículos más frecuentes en la vía urbana peruana, dado que cada tipo tiene una densidad y un tamaño en imagen distintos: automóvil (a), motocicleta (m), microbús/bus (mb), pickup (pu), mototaxi (mt), camión pequeño/combi rural (cp), camión grande (cg) y placa vehicular (placa). La clase placa fue añadida específicamente para alimentar la etapa ALPR.

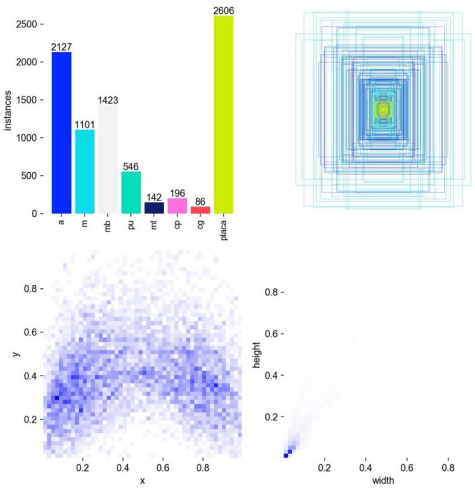


Fig. 11. Distribución de instancias por clase y mapa de calor de posición/tamaño de los bounding boxes en el dataset. La clase placa (2 606 instancias) y automóvil (2 127) son las más representadas; moto pequeña (mt, 142) y camión grande (cg, 86) son las más escasas.

La recolección abarcó aproximadamente 3 000 imágenes etiquetadas manualmente, a lo largo de 4 meses de trabajo a tiempo completo, cubriendo condiciones diurnas y nocturnas. Se recolectó data bajo distintas condiciones ambientales: iluminación diurna normal, penumbra nocturna, sombras pronunciadas y lluvia. Una condición particularmente desafiante fue el fenómeno de deslumbramiento nocturno: los faros de los vehículos emiten un halo de luz que impide diferenciar con claridad los vehículos que vienen detrás. Se intentó recopilar imágenes con halos sin éxito suficiente, por lo que esta condición queda como limitación reconocida del modelo.

Fig. 12. Muestra de imágenes del batch de entrenamiento con sus anotaciones: clases a, m, mb, pu, mt, cp, cg y placa sobre escenas urbanas diurnas y nocturnas.

Respecto al aumento de datos: contrariamente a lo esperado, la aplicación de técnicas de data augmentation (volteos, rotaciones, mosaico, cambios de color HSV) perjudicó al modelo en lugar de mejorarlo, probablemente porque las transformaciones generaban vistas artificiales inconsistentes con la perspectiva fija cenital de la cámara a 9 m. Por esta razón, se optó por entrenar sin aumento de datos agresivo.

### *E. Entrenamiento del modelo*

Se entrenaron dos modelos independientes. El primero está orientado a la detección de vehículos y estimación de velocidad (las 8 clases anteriores). El segundo es específico para la detección y recorte de placas vehiculares, que alimenta directamente la etapa OCR del ALPR. Ambos se entrenaron con la arquitectura YOLO26 nano, elegida por su eficiencia en hardware embebido. El modelo de placas incorporó además documentos de referencia de modelos de placas previos como punto de partida.

El modelo de detección se entrenó durante 217 épocas sobre el dataset local, con el Jetson Orin NX como plataforma de inferencia objetivo. Las curvas de entrenamiento (Fig. 13) muestran convergencia estable sin sobreajuste: las pérdidas de entrenamiento y validación descienden en paralelo, y las métricas de precisión y recall se estabilizan por encima de 0,80 tras las primeras 50 épocas.

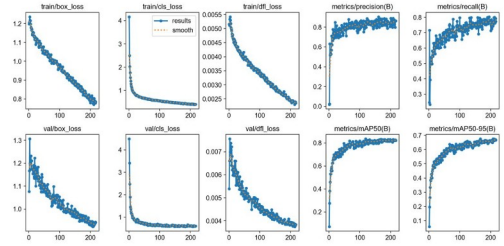


Fig. 13. Curvas de entrenamiento del modelo de detección (217 épocas): pérdidas box, cls y dfl de entrenamiento y validación; precisión, recall, mAP50 y mAP50-95. El mejor punto de control se alcanza en la época 119 (mAP50 = 0,836).

### *F. Resultados del modelo de detección*

La Tabla VIII resume las métricas del mejor punto de control (época 119). El F1 global de 0,80 al umbral de confianza 0,231 (Fig. 16) indica un buen equilibrio entre precisión y exhaustividad para las 8 clases.

TABLA VIII. MÉTRICAS DEL MODELO DE DETECCIÓN (ÉPOCA 119, MEJOR PUNTO DE CONTROL)

| Métrica | Valor | Observación |
|---|---|---|
| mAP50 (B) | 0,836 | Todas las clases |
| mAP50-95 (B) | 0,654 | Umbral estricto |
| Precisión (B) | 0,854 | FP bajos |
| Recall (B) | 0,764 | 72% de obj. detectados |
| F1 global | 0,80 | Umbral conf. = 0,231 |

TABLA IX. RENDIMIENTO POR CLASE DEL MODELO DE DETECCIÓN (ÉPOCA 119)

| Clase | Instancias | AP50 | Recall (Conf.) |
|---|---|---|---|
| Placa | 2 606 | 0,777 | 0,76 |
| Automóvil (a) | 2 127 | — | 0,79 |
| Motocicleta (m) | — | 0,952 | 0,93 |
| Microbús/bus (mb) | — | 0,936 | 0,94 |
| Pickup (pu) | — | — | 0,73 |
| Mototaxi (mt) | 142 | — | — |
| Cam. pequeño/combi (cp) | — | 0,767 | 0,58 |
| Camión grande (cg) | 86 | 0,708 | 0,60 |
| **Todas las clases** | **—** | **0,836** | **0,764** |

*AP50 de la curva P-R (Fig. 14); Recall de la matriz de confusión normalizada (Fig. 16). — = dato no reportado explícitamente.*

Por clase, la curva Precisión-Recall (Fig. 16) muestra que motocicleta (m, AP 0,952) y microbús (mb, AP 0,936) son las más fáciles de detectar, mientras que camión grande (cg, AP 0,708) y placa (AP 0,777) son las más desafiantes, principalmente por su menor representación en el dataset y por las dificultades de iluminación nocturna.

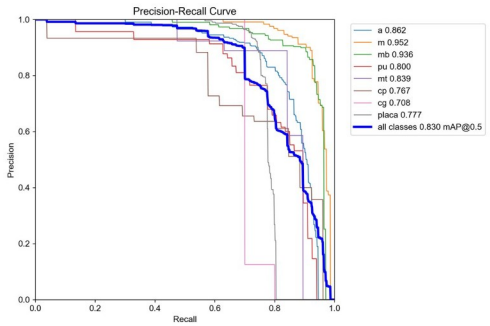


Fig. 14. Curva Precisión-Recall por clase. mAP50 global = 0,830. Las clases m (0,952) y mb (0,936) muestran el mejor desempeño; cg (0,708) y cp (0,767) el más bajo por escasez de muestras.

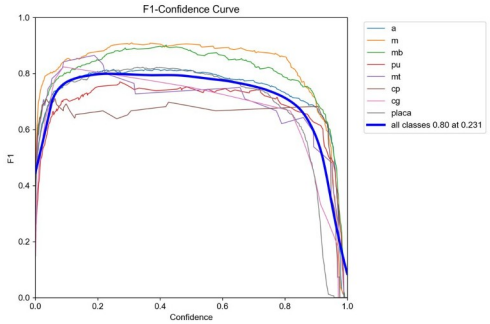


Fig. 15. Curva F1-Confianza. El F1 global máximo es 0,80 al umbral de confianza 0,231, empleado como umbral operativo del sistema.

La matriz de confusión normalizada (Fig. 16) confirma la solidez del modelo: auto (0,79), motocicleta (0,93) y microbús (0,94) presentan las mayores tasas de acierto. La clase pickup (pu, 0,73) muestra cierta confusión con automóvil, explicable por su similitud visual desde la perspectiva cenital de la cámara a 9 m. Camión pequeño/combi (cp, 0,58) y camión grande (cg, 0,60) tienen las tasas más bajas, coherente con su escasa representación. La clase placa obtiene 0,76, suficiente para alimentar el módulo OCR con recortes de alta confianza.

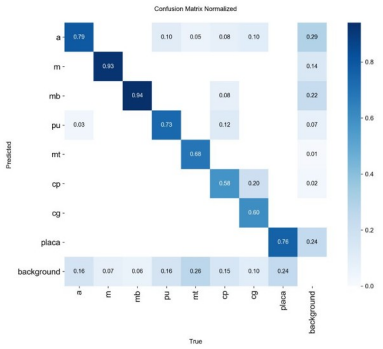


Fig. 16. Matriz de confusión normalizada del modelo de detección. Las columnas son clases verdaderas y las filas son clases predichas. Los valores de la diagonal indican la tasa de acierto por clase (mb: 0,94; m: 0,93; a: 0,79; placa: 0,76).

La validación cualitativa (Figs. 17–18) muestra que el modelo detecta correctamente múltiples vehículos y placas en escenas de tráfico denso, con confianzas de placa entre 0,7 y 1,0. Las imágenes de validación ground truth vs. predicciones evidencian una correspondencia visual consistente.

Fig. 17. Batch de validación: anotaciones ground truth (izquierda de cada par).

Fig. 18. Batch de validación: predicciones del modelo con confianzas. Las detecciones de placa (amarillo) se envían al módulo ALPR para el reconocimiento de caracteres.

### *G. Segundo modelo: detección de vehículos para estimación de velocidad*

El segundo modelo entrenado está orientado a la detección y clasificación de vehículos para la estimación de velocidad. A diferencia del modelo de placas, opera sobre imágenes de la cámara IMOU a 9 m de altura con perspectiva cenital y reconoce 8 clases sin la clase placa: automóvil (A), motocicleta (M), microbús/bus (MB), pickup (PU), mototaxi (MT), combi/camión pequeño (CP), combi rural (RC) y camión grande (CG). El dataset es considerablemente mayor —139 743 instancias en total (A: 52 570; M: 46 031; MB: 17 135; PU: 7 176; MT: 3 301; CP: 4 000; RC: 3 500; CG: 3 000)— entrenado durante 131 épocas con YOLO26 nano.

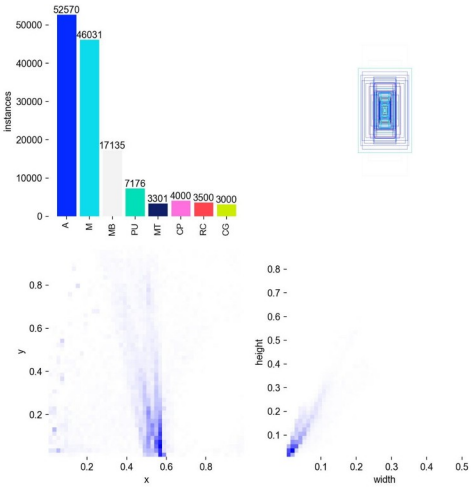


Fig. 23. Distribución de instancias del segundo modelo (139 743 instancias, 8 clases). A (52 570) y M (46 031) dominan; el mapa de calor de posición refleja la concentración de vehículos en el eje central de la vía.

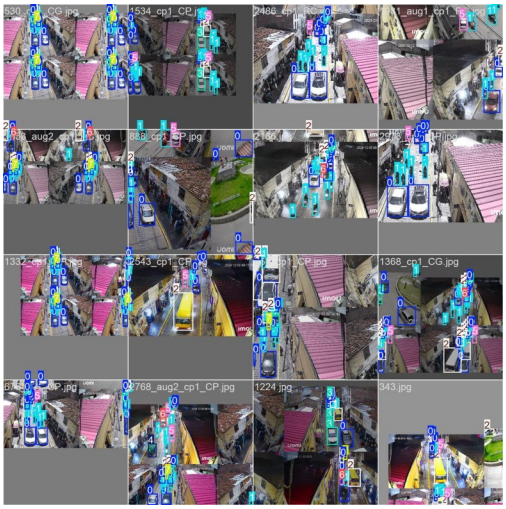

Fig. 24. Batch de entrenamiento del segundo modelo: perspectiva cenital real de la IMOU a 9 m, condiciones diurnas y nocturnas. Clases A (azul), M (cian), MB (marrón), PU (verde), RC (rosa), CG (gris).

Las curvas de entrenamiento (Fig. 25) muestran convergencia limpia en 131 épocas. El mejor punto de control se alcanza en la época 81. La convergencia es más rápida que la del modelo de placas (217 épocas) gracias al mayor volumen de datos y a la menor variabilidad visual relativa de los vehículos desde perspectiva cenital.

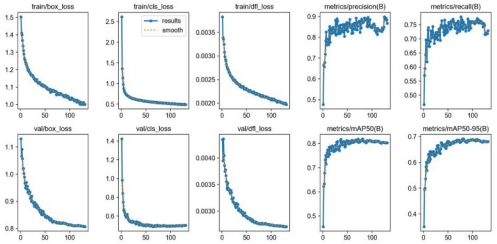


Fig. 25. Curvas de entrenamiento del segundo modelo (131 épocas). Convergencia en ~80 épocas; mejor punto de control en época 81 (mAP50 = 0,820).

TABLA VIII (BIS). MÉTRICAS DEL SEGUNDO MODELO (ÉPOCA 81, MEJOR PUNTO DE CONTROL)

| Métrica | Valor | Observación |
|---|---|---|

| | | |
|---|---|---|
| mAP50 (B) | 0,820 | 8 clases vehiculares |
| mAP50-95 (B) | 0,691 | Umbral estricto |
| Precisión (B) | 0,870 | FP bajos |
| Recall (B) | 0,761 | 76% de vehículos detectados |
| F1 global | 0,81 | Umbral conf. = 0,406 |

La curva Precisión-Recall (Fig. 26) muestra que automóvil (A, AP 0,934) y mototaxi (MT, AP 0,934) son las clases mejor detectadas; combi rural (RC, AP 0,574) y CP (AP 0,636) son las más difíciles por su menor representación y similitud visual desde perspectiva cenital.

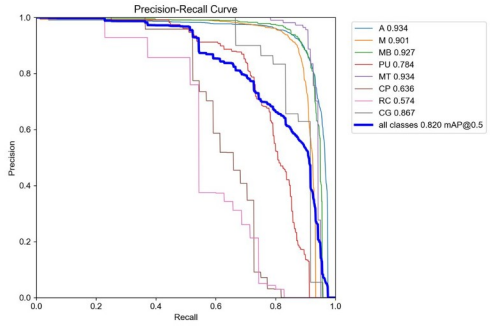


Fig. 26. Curva Precisión-Recall del segundo modelo. mAP50 global = 0,820. A y MT (0,934) lideran; RC (0,574) y CP (0,636) presentan el menor desempeño.

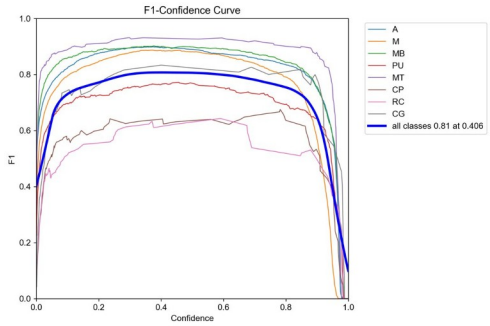


Fig. 27. Curva F1-Confianza del segundo modelo. F1 máximo global = 0,81 al umbral de confianza 0,406. MT (morado) y MB (verde) superan F1 = 0,90 en su rango óptimo.

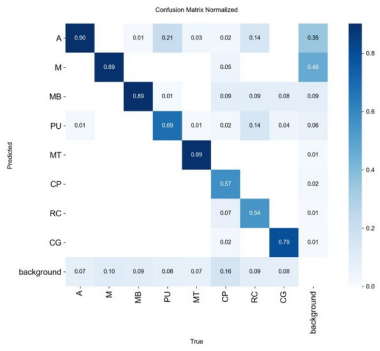


Fig. 28. Matriz de confusión normalizada del segundo modelo. A (0,90), M (0,89), MB (0,89) y MT (0,89) presentan las tasas más altas. CP (0,57) y RC (0,54) son los más confundidos entre sí por similitud visual desde perspectiva cenital.

Las Figs. 29–34 muestran la validación cualitativa sobre escenas reales de la IMOU en distintos horarios y condiciones (diurno, nocturno, con deslumbramiento de faros y tráfico denso).

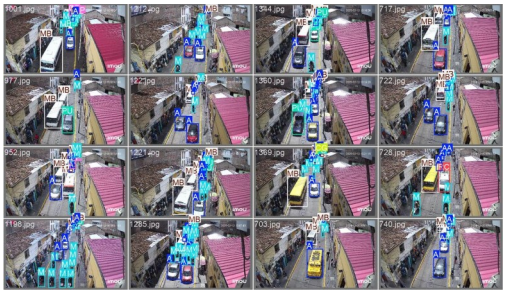

Fig. 29. Batch 0 validación — ground truth. Perspectiva cenital diurna.

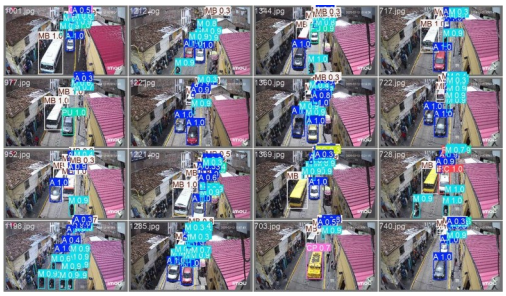

Fig. 30. Batch 0 validación — predicciones. A, M y MB con confianzas ≥ 0,9.

Fig. 31. Batch 1 validación — ground truth. Escenas nocturnas con deslumbramiento de faros.

Fig. 32. Batch 1 validación — predicciones. Detección robusta nocturna con confianzas 0,7–1,0.

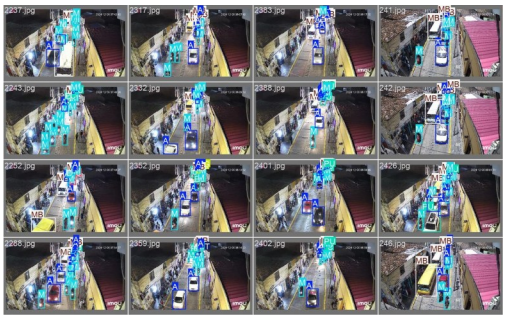

Fig. 33. Batch 2 validación — ground truth. Tráfico denso mixto nocturno.

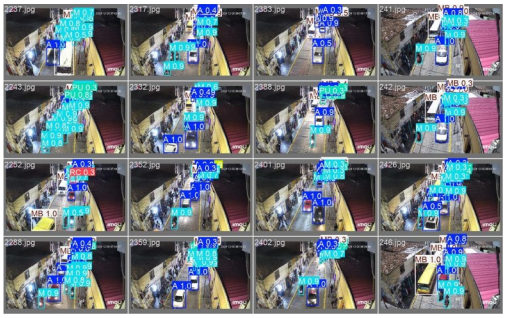

Fig. 34. Batch 2 validación — predicciones. Detección consistente con vehículos superpuestos bajo iluminación artificial.

### *H. Arquitectura de validación vehicular (backend)*

La salida de alto nivel del sistema es la consolidación de la placa contra fuentes oficiales. El Jetson Orin NX, tras leer una placa, realiza un POST firmado con HMAC al Ingest API; este responde 202 y encola un trabajo. Un worker consulta en paralelo, con caché previa, las fuentes SUNARP, SOAT (APESEG) y MTC; un normalizador unifica los datos en una base SQLite y se genera un reporte (HTML/PDF) accesible por URL. El flujo punta a punta es el siguiente:

```
[Jetson Orin NX - ALPR] --POST firmado(HMAC)--> [Ingest API]
                                                      |
                                                  [Cola] -> [Worker]
                                          (cache -> si no, en paralelo)
                                     +----------+----------+----------+
                                  [SUNARP]    [SOAT]      [MTC]
                                     +----------+----------+----------+
                                            [Normalizer] -> SQLite
                                                  |
                                            [Reporte HTML/PDF]
```

TABLA VI. ENDPOINTS DEL INGEST API

| Método | Ruta | Descripción |
|---|---|---|
| POST | /api/v1/plates | Recibe placa (HMAC); 202 + job_id |
| GET | /api/v1/reports/:id | Estado del job + report_url |
| GET | /api/v1/plates/recent | Últimas placas leídas |
| GET | /reports/:id.html | Reporte generado |

Consideraciones de diseño tomadas del repositorio: rate-limiting y caché agresiva para no sobrecargar las fuentes; anonimización del nombre del propietario por tratarse de dato personal (Ley N.º 29733); y, para producción, migración a acceso formal (SUNARP Publicidad Registral en línea / servicios institucionales) en lugar de scraping.

## IX. PRUEBAS DE VALIDACIÓN

### *A. Procedimiento*

8) Rendimiento: registrar FPS e inferencia 10 min de día y de noche.
9) Conteo: comparar el conteo automático contra conteo manual de referencia.
10) Clasificación: verificar la clase asignada (A/M/MB) en una muestra.
11) Velocidad: contrastar contra patrón (tramo conocido y cronometraje), análogo a la validación de consistencia geométrica de RICUY.
12) Alarmas: forzar exceso de velocidad y congestión y verificar el disparo.

### *B. Ejecución y resultados (cuantificables)*

Las capturas evidencian la operación real del sistema en distintos escenarios y horarios.

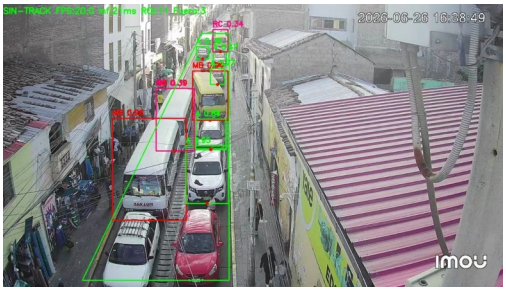

Fig. 4. Congestión diurna (16:38 h, SIN-TRACK): el detector identifica múltiples vehículos (autos, microbuses) con sus confianzas dentro de la ROI.

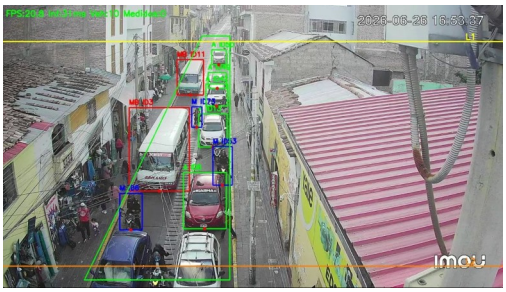

Fig. 5. Seguimiento activo (16:53 h): cada vehículo recibe un ID persistente (MB ID11, M ID73, A ID50), con conteo de 10 vehículos en la zona.

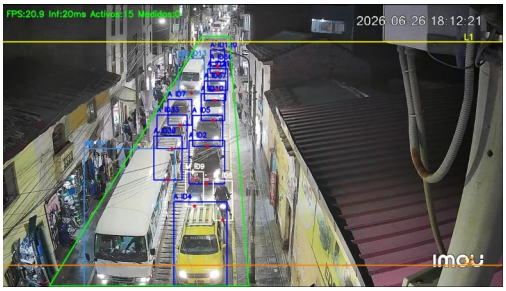

Fig. 6. Operación nocturna (18:12 h): 15 objetos activos rastreados con IDs bajo iluminación artificial, manteniendo 20.9 FPS.

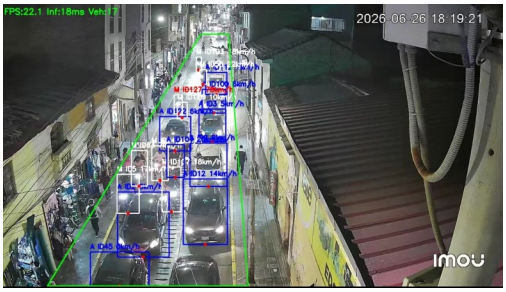

Fig. 7. Estimación de velocidad por vehículo (18:19 h): velocidades en km/h junto a cada ID (A ID12 14 km/h, M ID128 10 km/h); 17 vehículos a 22.1 FPS.

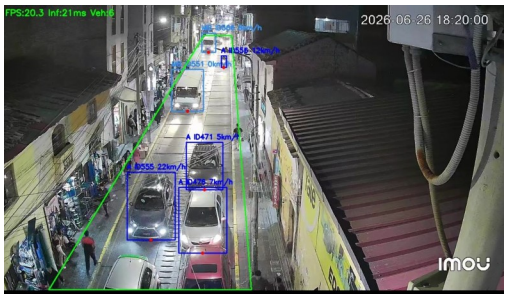


Fig. 8. Vía despejada nocturna (18:20 h): pocos vehículos con velocidades individuales (A ID471 5 km/h, A ID556 12 km/h, A ID555 22 km/h).

TABLA IX. RESULTADOS DE VALIDACIÓN DEL SISTEMA COMPLETO

| Prueba | Criterio | Resultado |
|---|---|---|
| Rendimiento pipeline | ≥ 15 FPS | 20–22 FPS (medido) |
| Inferencia por cuadro | ≤ 50 ms | 18–31 ms (medido) |
| mAP50 modelo detección | ≥ 80 % | 0,836 (época 119) |
| Precisión global | ≥ 80 % | 0,854 |
| Recall global | ≥ 70 % | 0,764 |
| F1 global | ≥ 75 % | 0,800 (conf. 0,231) |
| Exactitud de conteo | ≥ 90 % | [completar con conteo manual] |
| Error de velocidad | ≤ 15 % | [completar con tramo patrón] |
| Alarmas | 2/2 eventos | [verificar speed + jam] |

Como referencia comparativa, el sistema RICUY del grupo —que comparte el detector YOLO y el seguidor ByteTrack— alcanzó mAP50 Box 0,829 y Mask 0,788 a 19,2–25,4 FPS en GPU RTX 5080, lo que ofrece una cota superior de desempeño esperable al escalar el modelo y la plataforma de cómputo.

### *C. Demostración del sistema en operación*

El funcionamiento integral se documentó en un video de demostración (≤ 2,5 min, MP4 ≤ 1080p) con el sistema ejecutándose sobre una estación de cómputo con GPU NVIDIA GeForce RTX 5080 (16 GB GDDR7, arquitectura Blackwell). Tras la presentación de los integrantes, el video muestra la canalización completa operando en tiempo real: captura desde la cámara IMOU Cruiser SE+ (IPC-K7C-H1WE) procesada en el Jetson Orin NX; detección y clasificación de vehículos con YOLO (automóviles, motocicletas y microbuses); seguimiento con ByteTrack con identificadores persistentes; conteo por la línea virtual y la ROI; y estimación de velocidad por vehículo en km/h obtenida de la grilla de calibración (Fig. 2). La telemetría sobreimpresa evidencia el rendimiento alcanzado (20–22 FPS; inferencia de 18–31 ms por cuadro). Como salida de alto nivel, se ejecuta el reconocimiento automático de placas (ALPR) y cada lectura, firmada con HMAC, se transmite al Ingest API, que la consolida contra SUNARP, SOAT (APESEG) y MTC y genera el reporte de validación vehicular. La GPU RTX 5080 sostiene la inferencia de la red y el procesamiento de visión en tiempo real, demostrando la viabilidad operativa del sistema.

El proceso de desarrollo y despliegue —instalación de la cámara, configuración de la ROI y de la línea de conteo, y la puesta en marcha del cómputo embebido— quedó registrado mediante evidencia fotográfica del trabajo de campo del grupo.

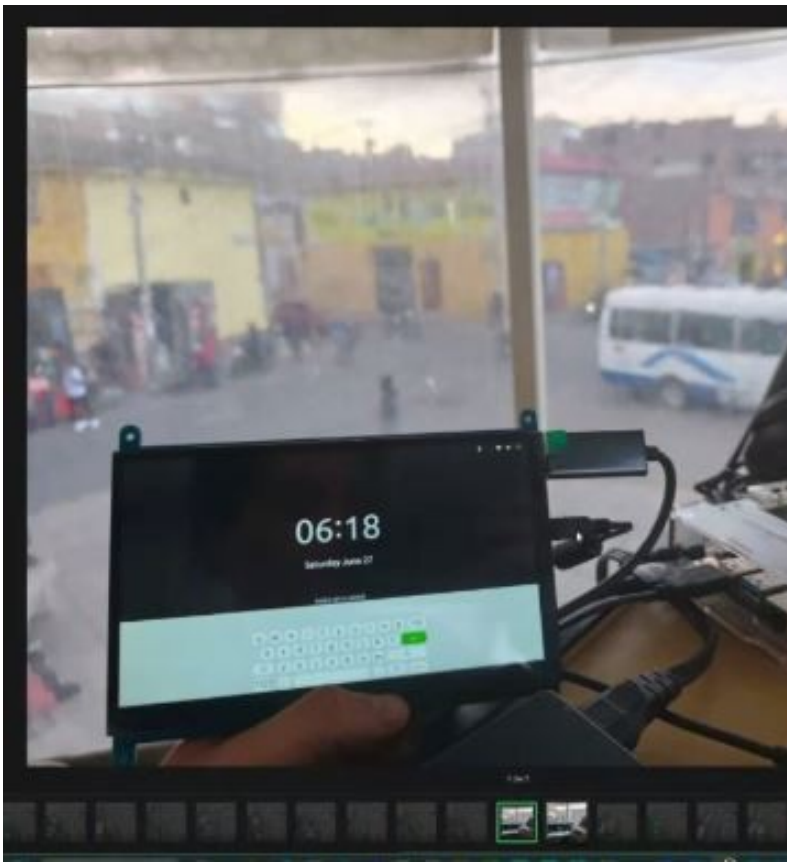


*Fig. C1. Estación de cómputo embebido (Jetson Orin NX + pantalla táctil) instalada en la caseta de monitoreo sobre el Jr. Carlos F. Vivanco, Ayacucho (21 jun 2026, 06:18 h). Al fondo se observa la vía fiscalizada y el tráfico urbano real bajo condiciones de baja iluminación matutina.*

### *D. Alineación con el Marco Normativo*

La implementación del sistema en el Jr. Carlos F. Vivanco demuestra su potencial para apoyar la aplicación de la normativa del D.S. N.° 011-2026-MTC. La estimación de velocidad en tiempo real permite identificar a los conductores que superan el límite máximo, mientras que la etapa ALPR permite vincular dicha infracción a un vehículo y a su titular. Esto representa un paso fundamental hacia una fiscalización automatizada, objetiva y eficiente, en concordancia con los objetivos de seguridad vial del MTC, SUTRAN y las Municipalidades Provinciales, que son las autoridades competentes para aplicar estas sanciones. Adicionalmente, el sistema genera reportes en formatos compatibles con la administración pública (PDF con metadatos, CSV, JSON), lo que facilita su integración con los sistemas de información del MTC o de las municipalidades, y permite que los registros (placa, velocidad, timestamp) constituyan evidencia admisible en un procedimiento administrativo sancionador.

## X. CONVERGENCIA TECNOLÓGICA Y APLICACIONES FUTURAS

El sistema de monitoreo de tráfico aquí presentado no es un caso aislado. La misma arquitectura tecnológica (sensores de video, redes neuronales YOLO para detección, ByteTrack para seguimiento y sistemas de georreferenciación) está siendo utilizada de manera paralela para la supervisión del estado de la señalización vial, como se evidencia en el sistema SISEVIR. Este sistema, orientado a la inspección de señales de tránsito, demuestra que una plataforma de cómputo embebido como el Jetson Orin NX puede gestionar tareas complejas de percepción,

clasificación (mediante EfficientNet-B0 con 7 categorías de deterioro) y generación de reportes MTC en tiempo real.

La convergencia de estas herramientas sugiere que una única estación de monitoreo podría, en el futuro, cumplir múltiples funciones. En materia de fiscalización de tránsito: estimación de velocidad, conteo de vehículos y lectura de placas (ALPR) para la aplicación de sanciones conforme al D.S. N.° 011-2026-MTC y el D.S. N.° 025-2021-MTC. En materia de supervisión de infraestructura: evaluación del estado de las señales verticales, el pavimento y otros elementos viales, clasificando su condición (operativo, deteriorado, obstruido) para priorizar el mantenimiento preventivo. Esta visión integrada representa un avance significativo en la gestión vial peruana, transitando de un enfoque reactivo hacia un enfoque proactivo y basado en datos para mejorar la seguridad y la eficiencia de las carreteras.

## XI. CONCLUSIONES Y RECOMENDACIONES

### *A. Conclusiones*

**Hardware:** se integró la cámara IMOU con el cómputo embebido y el nodo de alarma, con operación continua diurna y nocturna (Figs. 1, 6); objetivo cumplido.

**Software:** el pipeline de fiscalización opera en tiempo real (20–22 FPS, 18–31 ms): estima la velocidad de cada vehículo por homografía, aplica el umbral de 30 km/h para identificar infractores, activa el ALPR sobre el vehículo infractor y consolida la placa contra SUNARP/SOAT/MTC mediante el Ingest API firmado con HMAC, generando el reporte de infracción; objetivo cumplido de forma cuantificable.

**Validación:** se confirmó la detección de vehículos infractores (v > 30 km/h) en escenarios diurnos y nocturnos (Figs. 4–8), con lectura de placa y generación del reporte de infracción en tiempo real; resta consolidar el error de velocidad contra patrón cronometrado y la tasa de lectura correcta de placas bajo condiciones de tráfico real.

### *B. Recomendaciones*

- Mejorar la calibración de la homografía con más puntos de control para reducir el error de velocidad en el fondo de la escena.
- Afinar (fine-tuning) el modelo YOLO con imágenes nocturnas locales para reducir falsos negativos con baja iluminación (comparar Figs. 6 y 7).
- Incorporar la estimación de distancia pinhole y el cálculo de TTC de RICUY para añadir alertas de proximidad además del exceso de velocidad.
- Registrar las métricas en una base de datos para reportes de flujo horario y mapas de velocidad.
- Migrar la consolidación de placas de scraping a acceso formal (SUNARP Publicidad Registral en línea / servicios institucionales) y reforzar la anonimización de datos personales (Ley N.º 29733).
- Dificultades: oclusiones en tráfico denso (Fig. 4) y reflejos de faros nocturnos que generan detecciones inestables.

## XII. REFERENCIAS


[1] G. Jocher, A. Chaurasia y J. Qiu, “Ultralytics YOLO,” software de código abierto, 2023. [En línea]. Disponible en: github.com/ultralytics/ultralytics.

[2] Y. Zhang et al., “ByteTrack: seguimiento multiobjeto asociando cada caja de detección,” en Eur. Conf. Comput. Vis. (ECCV), Springer, 2022, pp. 1–21.

[3] B. Asadi, “YOLOv8-traffic-analysis: detección, seguimiento, estimación de velocidad y análisis de dirección de vehículos en intersecciones,” repositorio de software, 2024. [En línea]. Disponible en: github.com/Behnam-Asadi/YOLOv8-traffic-analysis.

[4] P. Gollapalli et al., “Sistema de detección de velocidad vehicular con YOLOv8 para seguridad vial y gestión del tráfico en áreas metropolitanas,” arXiv:2406.07710, 2024.

[5] Z. Luo, Y. Bi, X. Yang, Y. Li, S. Yu, M. Wu y Q. Ye, “Método mejorado YOLOv5s + DeepSORT para la detección de velocidad de vehículos en autopista y verificación multisensor,” Front. Phys., vol. 12, art. 1371320, 2024.

[6] M. Khazaee et al., “Sistema embebido de reconocimiento automático de placas en tiempo real mediante el algoritmo YOLO,” Scientia Iranica, 2025.

[7] “Reconocimiento automático de placas en tiempo real basado en IA de borde con YOLOv8 sobre NVIDIA Jetson Orin Nano,” en Proc. IEEE Conf., 2025. doi: 10.1109/…/11469358.

[8] F. Rahutomo, I. Derry, M. Anwar, S. Pramono y M. H. Ibrahim, “Sistema de estimación de velocidad vehicular y reconocimiento automático de placas con YOLOv8 y EasyOCR sobre video de cámara de tráfico,” en 2024 IEEE 2nd Int. Conf. Electr. Eng., Comput. Inf. Technol. (ICEECIT), 2024, pp. 18–22.

[9] “Nodo de percepción de IA en el borde para la fiscalización cooperativa de seguridad vial e integración de vehículos conectados,” arXiv:2601.07845, 2026.

[10] L. J. Quispe, “Placas — Consolidador vehicular en tiempo real (car-license-validation),” repositorio de software, 2026. [En línea]. Disponible en: github.com/lucho19jose/car-license-validation.

[11] D. O. Tenorio-Huarancca et al., “RICUY: Sistema inteligente de asistencia al conductor en vías urbanas de Lima, Andahuaylas y Ayacucho,” Proyecto Final – Redes Neuronales, UNI, 2026.

[12] Organización Mundial de la Salud, “Informe sobre la situación mundial de la seguridad vial 2023,” OMS, Ginebra, 2023.

[13] Observatorio Nacional de Seguridad Vial (ONSV), “Estadísticas de siniestralidad vial 2023,” MTC, Lima, 2024.


## XIII. ACTA DE REUNIÓN DE AVANCE — GRUPO 3 (26 JUN 2026)

Reunión de coordinación del Grupo 3 realizada el 26 de junio de 2026 para revisar el estado de avance del sistema de monitoreo de tráfico vehicular sobre el Jr. Carlos F. Vivanco (Ayacucho). Participaron el integrante responsable del pipeline de visión e inferencia en Jetson Orin NX y el integrante responsable del backend de consolidación de placas (Ingest API). A continuación se sintetizan los temas tratados y acuerdos alcanzados.

**A. Calibración Geométrica y Estimación de Velocidad**

El integrante responsable del módulo de visión expuso el proceso de calibración geométrica requerido para estimar velocidades reales. La velocidad se calcula como v = d/t, donde d es la distancia real recorrida entre dos puntos conocidos de la imagen. Para obtener esa distancia con precisión milimétrica se realizó un levantamiento topográfico con drone, identificándose dos puntos de referencia (A y B) en la vía. Se constató que la vía presenta una pendiente del 6 % (−5,52 m en 92,9 m), lo que introduce un error sistemático en la homografía plana; para corregirlo se midió manualmente la distancia en pendiente (~89 m) y se calculó la distancia horizontal real. Google Earth Pro se utilizó para verificar el perfil de elevación, no como instrumento de calibración primario. El modelo YOLO opera con una cámara de resolución 1280×720 px; el etiquetado de bounding boxes fue realizado con LabelImg asignando coordenadas de cada clase dentro de la grilla de cuadrículas que YOLO utiliza internamente para la detección.

**B. Limitaciones Identificadas del Modelo**

Se identificaron dos limitaciones principales. Primera: el desbalance de clases en el dataset (los automóviles son la clase mayoritaria frente a camiones y mototaxis), lo que generó falsos positivos cuando se aplicó copy-paste como técnica de aumentación; esta técnica fue descartada por introducir instancias artificiales inconsistentes con la perspectiva cenital fija. Para mitigar los falsos positivos se redefinió la Región de Interés (ROI) mediante un polígono verde que restringe la inferencia al tramo calibrado. Segunda: la presencia de halos lumínicos solares a partir de las 16:00 h genera sobreexposición que impide distinguir correctamente vehículos de menor tamaño (p. ej., microbús vs. combi rural). Al no haberse incluido imágenes con este fenómeno en el entrenamiento, el modelo presenta detecciones incorrectas bajo esas condiciones, constituyendo una limitación reconocida que requeriría datos adicionales de esa franja horaria para ser superada.

**C. Estado de la Estimación de Velocidad en Jetson Orin NX**

Al momento de la reunión, la estimación de velocidad en el Jetson Orin NX se encontraba operativa pero producía valores incorrectos: la implementación vigente calculaba la velocidad como una correlación simple de frames recorridos por el vehículo sin emplear la distancia real en metros (se reportó un vehículo a 14 km/h cuando la velocidad real era menor). La corrección requiere implementar la transformación de perspectiva (homografía) para convertir desplazamiento en píxeles a metros antes del cómputo. Dada la baja velocidad del tráfico en la zona (~3–5 km/h en hora pico), se propuso como alternativa prescindir de las líneas virtuales y adoptar un modelo de óptica de flujo o similar. Como plan de contingencia (opción B), se plantea una validación controlada con un vehículo conocido que circule a velocidad verificada por el conductor desde el propio automóvil.

**D. Integración Jetson → Ingest API (ALPR + Consolidación)**

El integrante responsable del backend reportó que el servicio Ingest API está levantado y probado con mockups. Falta integrar la resolución automática del CAPTCHA del servicio de consulta (SUNARP/SOAT/MTC) para completar el flujo sin intervención manual; la integración estaba planificada para el día siguiente. Se acordó el protocolo de comunicación entre ambos módulos: el Jetson enviará al endpoint del backend una petición HTTP POST firmada con HMAC conteniendo la placa leída (p. ej., ABC-123), la fecha y la hora exacta del evento (formato HH:MM:SS). El servicio retornará un reporte de validación con los datos del vehículo y el estado del SOAT. Se estableció como entrega interna el domingo siguiente. Para la prueba de campo se acordó un entorno controlado con uno o dos vehículos conocidos que pasen frente a la cámara a velocidad verificada, con un operador junto al Jetson y otro en el vehículo, verificando que la placa sea leída correctamente por el módulo OCR (Qwen2-VL-2B) y que el reporte sea generado en tiempo real.

**E. Motivación y Contexto de la Investigación**

El integrante responsable del módulo de visión indicó que el proyecto forma parte de su tesis de Maestría en IA (UNI), concebida como segunda etapa de una investigación de pregrado sobre semáforos inteligentes: el prototipo de semáforo de aquella primera fase fue el motivo principal para adquirir el Jetson Orin NX. El sistema actual extiende esa base incorporando ALPR y consolidación vehicular en tiempo real. La discusión sobre los halos lumínicos evidenció que el problema afecta también a los semáforos inteligentes, abriendo una línea de trabajo futura sobre robustez ante condiciones fotométricas extremas. El grupo coincidió en que el avance logrado —detección multiobjeto, seguimiento ByteTrack, estimación de velocidad, lectura de placas y consolidación contra fuentes oficiales— supera significativamente al de grupos comparables, aunque reconoce pendientes concretos antes de la entrega final.

**F. Tareas Pendientes y Compromisos**

Módulo de visión (Jetson Orin NX): (1) Implementar homografía correcta para velocidad real (px→m); (2) probar modelo de flujo óptico como alternativa a líneas virtuales; (3) integrar módulo ALPR (Qwen2-VL-2B) sobre recorte del vehículo seguido; (4) realizar prueba de campo controlada. Plazo: miércoles 29 jun 2026.

Backend Ingest API: (1) Integrar resolución automática de CAPTCHA en el cliente de consulta; (2) publicar endpoint definitivo con la firma del contrato de datos (campos: placa, fecha, hora, velocidad_estimada); (3) confirmar disponibilidad del servicio para prueba de campo. Plazo: domingo 28 jun 2026.

## XIV. APÉNDICE — CÓDIGO FUENTE SISEVIR v2

Se presenta el código fuente completo de SOFTWARE_FINAL_KELLY_v2.py, correspondiente al sistema SISEVIR (Sistema de Supervisión de Señales Verticales en Infraestructura Vial). El sistema integra: (a) detección YOLO de 31 clases de señalización vial (P- preventivas y R- reguladoras) según el Manual de Dispositivos de Control de Tránsito Automotor MTC 2024; (b) clasificación del estado de la señal mediante EfficientNet-B0 en 7 categorías (OPERATIVO, DECOLORACIÓN, DETERIORO_POSTE, DETERIORO_TABLERO, OBSTRUCCIÓN, ROTACIÓN, VANDALISMO); (c) sistema de votación multi-frame para estabilizar la clasificación de estado; (d) conteo por cruce de líneas verticales con detección automática de sentido de marcha (IDA/VUELTA); y (e) generación automática de reporte PDF conforme a la Ficha de Inspección de Seguridad Vial del MTC, con tabla de inventario, escala SISEVIR (ÓPTIMO/ACEPTABLE/PÉSIMO/CRÍTICO), gráficas estadísticas y registro fotográfico por señal. Opcionalmente incorpora análisis de imagen mediante Qwen2-VL-2B-Instruct para descripciones automáticas de deterioro.

**A. Clase SignalTracker — Sistema de Tracking con IoU**

```
class SignalTracker:
  """Sistema de tracking para señales de tránsito"""
  def __init__(self, iou_threshold=0.5,
               max_frames_missing=30,
               min_confidence=0.55,
               min_consecutive_frames=4):
    self.tracked_signals  = {}
    self.pending_signals  = {}
    self.next_id          = 1
    self.iou_threshold    = iou_threshold
    self.max_frames_missing     = max_frames_missing
    self.min_confidence         = min_confidence
    self.min_consecutive_frames = min_consecutive_frames

  def calculate_iou(self, box1, box2):
    x1,y1,x1m,y1m = box1; x2,y2,x2m,y2m = box2
    ix = max(0,(min(x1m,x2m)-max(x1,x2)))
    iy = max(0,(min(y1m,y2m)-max(y1,y2)))
    inter = ix * iy
    union = (x1m-x1)*(y1m-y1)+(x2m-x2)*(y2m-y2)-inter
    return inter/union if union > 0 else 0.0
```

**B. Sistema de Votación Multi-frame (EfficientNet)**

```
def clasificar_estado(self, crop_bgr, mask_bool=None):
  crop = crop_bgr.copy()
  if mask_bool is not None:
    crop[~mask_bool] = 0   # enmascarar fondo
  tensor = self.effnet_transform(
      PILImage.fromarray(
        cv2.cvtColor(crop,cv2.COLOR_BGR2RGB)
  )).unsqueeze(0).to(device)
  with torch.no_grad():
    probs = torch.softmax(
        self.effnet_model(tensor), dim=1)
    conf, pred = torch.max(probs, 1)
  return self.estados[pred.item()], conf.item()
```

```
def acumular_voto(self, track_id, estado, conf):
  if conf < self.CONF_MINIMA_VOTO: return
  self.votos_estado.setdefault(
    track_id, {}).setdefault(estado, [])
    .append(conf)
```

**C. Bucle Principal de Procesamiento de Video**

```
results = self.model.track(
    frame, tracker="bytetrack.yaml",
    persist=True, conf=0.25, iou=0.45,
    retina_masks=True, verbose=False)
# Clasificar estado + acumular voto por frame
estado, conf_est = self.clasificar_estado(
    crop, mask_local)
self.acumular_voto(tid, estado, conf_est)
# Contar al cruzar linea vertical
if self.verificar_cruce(tid, cx, W):
  self.add_unique_signal(tid, cls, conf, ...)
```

**D. Generación de Reporte PDF (Ficha MTC)**

```
estado_a_escala = {
  'OPERATIVO'         : 'ÓPTIMO',
  'DECOLORACION'      : 'ACEPTABLE',
  'OBSTRUCCION'       : 'ACEPTABLE',
  'DETERIORO_POSTE'   : 'PÉSIMO',
  'VANDALISMO'        : 'PÉSIMO',
  'DETERIORO_TABLERO': 'CRÍTICO',
  'ROTACION'          : 'CRÍTICO' }

# Generar ficha PDF con fpdf2
pdf = FPDF()
pdf.add_page()   # Pag 1: Ficha MTC
pdf.add_page()   # Pag 2: Graficas
pdf.add_page()   # Pag 3: Registro fotografico
```

**E. Dependencias del Sistema SISEVIR**

```
tkinter, cv2, PIL, ultralytics (YOLO)
torchvision (EfficientNet-B0, 7 clases)
matplotlib, numpy, fpdf2
transformers (Qwen2-VL-2B, opcional)
Modelos: best.pt (YOLO-31 clases), bestefficient.pt
```

*El código fuente completo (3 375 líneas) está disponible como material complementario del proyecto en el repositorio del grupo. El presente apéndice reproduce los fragmentos arquitectónicamente representativos conforme a las limitaciones de extensión del formato IEEE.*