“Contador de línea virtual”

Visión Artificial | ICED1026-001

Ingeniería Civil Electrónica

Profesor

Oscar Loyola

Estudiante

Bastian Arancibia

Edwin Moya

Jean Grainger

Fecha

05/12/2025

Índice

1. Introducción

El presente proyecto desarrolla un sistema automatizado capaz de realizar el conteo direccional de personas y vehículos mediante técnicas modernas de visión por computador. A partir de un video de tráfico previamente grabado, se implementa un pipeline que combina algoritmos de detección basados en inteligencia artificial con métodos clásicos de análisis de trayectoria. El sistema utiliza el modelo YOLOv8, uno de los detectores de objetos más eficientes y precisos en la actualidad, para identificar peatones y distintos tipos de vehículos presentes en la escena. Luego, un módulo de seguimiento (tracking) permite mantener la identidad de cada objeto detectado a lo largo del tiempo para evitar conteos duplicados.

Posteriormente, el sistema analiza el cruce de dichos objetos a través de una línea virtual ubicada estratégicamente en el campo de visión de la cámara. Este cruce permite determinar si una persona o un vehículo está entrando o saliendo del área monitoreada. La combinación de estas técnicas da forma a una herramienta completamente automatizada, capaz de operar sin supervisión humana y con una precisión significativamente mayor a la del conteo manual.

Este proyecto no solo demuestra el potencial de la inteligencia artificial aplicada a la gestión de movilidad urbana, sino que también evidencia cómo el procesamiento de imágenes puede convertirse en un aliado clave para áreas como seguridad, transporte, gestión de infraestructura y análisis de comportamiento en espacios públicos.

1.1 Definición del Problema

En el contexto contemporáneo de ciudades crecientemente complejas, la **gestión eficiente del flujo de personas y vehículos** se ha convertido en un desafío fundamental. Instituciones públicas, centros comerciales, empresas de transporte y organismos de seguridad requieren datos confiables y en tiempo real para tomar decisiones operativas y estratégicas. Sin embargo, los métodos tradicionales de aforo como el conteo manual o el uso de sensores físicos simples presentan limitaciones sustanciales:

Limitaciones del conteo manual

- Depende completamente del observador, por lo que es propenso a errores humanos.
- Requiere largas jornadas de supervisión, lo que incrementa los costos operativos.
- No permite un monitoreo continuo, especialmente en escenarios de alta afluencia.

Limitaciones de sensores convencionales

- Dispositivos como barreras IR o sensores de presión suelen fallar ante **oclusiones**, es decir, cuando dos o más personas caminan demasiado cerca una de otra.
- Vehículos grandes (buses o camiones) pueden bloquear la visión o generar múltiples activaciones.
- No entregan información contextual, como tipo de objeto, dirección de movimiento o densidad del tránsito.

Escenario tecnológico actual

Con el crecimiento de la infraestructura inteligente, se vuelve indispensable contar con sistemas:

- **Autónomos**, capaces de operar sin intervención humana.
- **Escalables**, aptos para monitorear múltiples puntos simultáneamente.
- **Flexibles**, que puedan adaptarse a distintos entornos urbanos o comerciales.
- **Precisos**, incluso frente a variaciones de iluminación, clima o movimiento complejo.

Frente a este escenario, surge la necesidad de diseñar un sistema capaz de **detectar, clasificar, rastrear y contar objetos en movimiento**, manteniendo su identidad a través de múltiples cuadros del video. La adopción de modelos avanzados como YOLOv8 permite superar las limitaciones de los métodos tradicionales, dotando al sistema de la capacidad de procesar información visual en tiempo real, con alta precisión y sin requerir infraestructura adicional más allá de una cámara estándar.

1.2 Motivación

La motivación de este proyecto surge de la creciente demanda por soluciones automatizadas que permitan analizar el flujo de personas y vehículos con alta precisión y sin intervención humana constante. En los últimos años, el desarrollo acelerado de la Inteligencia Artificial —particularmente en el área de la visión por computador— ha permitido reemplazar métodos manuales y sistemas tradicionales de sensorización por modelos basados en aprendizaje profundo capaces de interpretar escenas complejas con un grado significativamente mayor de exactitud (Goodfellow et al., 2016).

El Procesamiento de Imágenes, combinado con Redes Neuronales Convolucionales (CNN), ofrece una alternativa eficiente, escalable y adaptable a distintos entornos. Estas tecnologías permiten detectar múltiples clases de objetos en condiciones dinámicas —como variaciones de iluminación, oclusiones, densidad de tráfico o perspectivas difíciles— lo cual sería muy difícil de lograr usando mecanismos clásicos de conteo o sistemas basados únicamente en hardware.

Asimismo, este proyecto busca explorar la aplicabilidad de estas herramientas en escenarios reales, donde la información generada en tiempo real puede apoyar procesos de toma de decisiones en áreas como transporte, seguridad, planificación urbana y gestión de infraestructura crítica (Redmon & Farhadi, 2018). Con esto, se pretende no solo desarrollar un prototipo funcional, sino también demostrar el potencial transformador de la IA en automatización y monitoreo inteligente.

2. Objetivos del Proyecto

2.1 Objetivo General

El objetivo general del proyecto es desarrollar un sistema de Conteo mediante Línea Virtual, basado en visión por computador, capaz de detectar, rastrear y contabilizar objetos en movimiento a partir de secuencias de video. Este sistema debe operar en tiempo real, ser escalable y permitir análisis cuantitativos confiables sobre flujos de movilidad.

2.2 Objetivo Especifico

1. **Detección Robusta:** Implementar un sistema basado en IA capaz de reconocer y diferenciar entre peatones y vehículos (autos, buses, camiones).
2. **Seguimiento (Tracking):** Mantener la identidad de los objetos a través de los cuadros del video para evitar conteos duplicados.

3. **Lógica Direccional:** Establecer una línea virtual que permita discriminar el sentido del movimiento (entradas vs. salidas).
4. **Optimización:** Asegurar que el sistema funcione con fluidez (tiempo real) mediante técnicas de gestión de recursos computacionales.

3. Aspectos Técnicos

3.1 Arquitectura del Sistema

El sistema sigue una arquitectura de "Pipeline" de procesamiento de video secuencial:

1. **Entrada de Video:** Ingesta de flujo de video (.mp4 o cámara en vivo).
2. **Preprocesamiento:** Redimensionamiento de cuadros (720x800 px) para estandarizar la entrada a la red neuronal.
3. **Motor de Inferencia (YOLOv8):** El núcleo del sistema utiliza el modelo **YOLOv8m** (You Only Look Once, versión 8 media). Este modelo se encarga simultáneamente de la detección de objetos y la extracción de características.
4. **Módulo de Tracking (ByteTrack/BoT-SORT):** Integrado en Ultralytics, asigna un ID único a cada objeto detectado, permitiendo reconstruir su trayectoria histórica.
5. **Lógica Heurística:** Un algoritmo personalizado analiza las coordenadas de los centroides respecto a una línea vertical definida por el usuario.

3.2 Datase y Clases

Para la detección de objetos se emplea un modelo preentrenado de la familia **YOLOv8**, específicamente la variante *yolov8m.pt*, la cual ofrece un equilibrio adecuado entre velocidad de inferencia y precisión. Este modelo ha sido entrenado previamente sobre el **dataset COCO (Common Objects in Context)**, uno de los estándares más utilizados en visión por computador para tareas de detección, segmentación y clasificación de objetos.

COCO contiene más de **330.000 imágenes** anotadas manualmente, con **80 clases de objetos** distribuidas en contextos urbanos, interiores, exteriores, situaciones complejas, oclusiones y variaciones de iluminación. Gracias a este volumen y diversidad, los modelos entrenados sobre COCO presentan una alta capacidad de generalización y funcionan correctamente en escenarios reales similares al video analizado.

En este proyecto, no se realiza un entrenamiento desde cero (training) ni una adaptación fina del modelo (fine-tuning), pues las clases necesarias ya se encuentran dentro del dataset COCO. De este modo, se aprovecha directamente la capacidad del modelo para

detectar objetos comunes en escenas de tráfico urbano, lo que permite agilizar el desarrollo del sistema sin requerir datasets adicionales ni tiempos prolongados de entrenamiento.

Clases seleccionadas

De las 80 clases disponibles en COCO, se seleccionaron únicamente aquellas relevantes para el análisis del flujo de personas y vehículos. Estas clases, identificadas mediante su ID numérico dentro del dataset, son:

ID COCO	Clase	Descripción
0	Person	Peatones o individuos caminando por el área analizada.
2	Car	Vehículos livianos, como autos particulares o taxis.
5	Bus	Transporte público o buses privados de gran capacidad.
7	Truck	Camiones de carga y vehículos pesados.

Estas clases representan los grupos más relevantes para estudios de movilidad, análisis de tráfico, estimación de aforo y planificación urbana. Además, al reducir el número de clases evaluadas, se optimiza el rendimiento del sistema al disminuir la carga de procesamiento durante la inferencia.

3.3 Diseño de la solución

El sistema utiliza una línea virtual vertical como punto de referencia, ubicada al 38% del ancho del cuadro (LINE_X_POS = 0.38). Para cada objeto identificado por el tracker, se almacena un historial de posiciones en track_history, donde solo se utilizan las dos posiciones más recientes (la anterior y la actual) para determinar si el objeto cruzó la línea.
El cruce se detecta comparando el centroide anterior (prev_cx) y el actual (curr_cx) respecto a la posición de la línea (line_x):
Cuando prev_cx está a la izquierda y curr_cx pasa a la derecha, se registra una entrada.

- Cuando prev_cx está a la derecha y curr_cx pasa a la izquierda, se registra una salida, pero esta condición se aplica solo a personas.
- En el caso de los vehículos, solo se contabilizan entradas, ya que la calle donde se realizó la grabación es unidireccional, por lo que no existe tránsito de retorno que atraviese la línea en sentido contrario.

Finalmente, los contadores operan de manera independiente: personas entrando, personas saliendo y vehículos entrando, utilizando la lista de IDs [2, 5, 7] para identificar autos, buses y camiones.

Figura N°1. Diseño de la ejecución del código.

4. Implementación

4.1 Desarrollo del Código

El desarrollo del software se realizó en Python, empleando dos componentes principales:

- OpenCV, encargado del procesamiento de imágenes, manipulación gráfica y despliegue visual.
- Ultralytics YOLOv8, responsable de la detección y el seguimiento (tracking) de objetos en tiempo real.

El flujo del programa se estructura en un ciclo principal que procesa cada cuadro del video, aplica detección profunda, actualiza el historial de trayectorias y determina si cada objeto cruza la línea virtual. Antes del procesamiento, el sistema realiza una etapa de preconfiguración donde se cargan los modelos, se inicializan las estructuras de datos para tracking y se establecen los contadores.

Estrategias de Optimización Implementadas

El código incorpora diversas optimizaciones orientadas a mejorar la velocidad sin comprometer la precisión:

a) Frame Skipping

Tal como se observa en la línea FRAME_SKIP = 2, el sistema procesa solo 1 de cada 2 cuadros.

Esto tiene dos efectos clave:

- Reduce la carga computacional aproximadamente en un 50%, permitiendo que el modelo YOLOv8 funcione con mayor fluidez en hardware estándar.
- Mantiene suficiente información temporal para un tracking robusto gracias al uso de historiales (deque) y persist=True en YOLO.

b) Redimensionamiento del Frame

frame = cv2.resize(frame, (720, 800))

El redimensionamiento a una resolución controlada reduce la cantidad de píxeles que deben ser analizados por el modelo, aumentando los FPS sin afectar la capacidad del modelo para detectar objetos grandes y medianos.

c) Uso de Tracking Persistente

El método:

results = model.track(frame, persist=True, ...)

permite conservar el ID del mismo objeto entre cuadros, aun cuando no se detecte momentáneamente. Esto evita conteos duplicados y compensa pérdidas temporales por oclusiones o ruido.

4.2 Desafíos Encontrados

Durante la fase de pruebas y validación del sistema, se identificaron tres desafíos principales que afectaban la precisión y la estabilidad del conteo. Estos problemas son comunes en sistemas basados en visión por computadora, especialmente aquellos que dependen únicamente de una cámara RGB monocular.

1. Oclusión de Peatones

Uno de los problemas más recurrentes fue la **oclusión**, fenómeno en el cual dos o más personas caminan muy próximas entre sí, provocando que sus bounding boxes se superpongan. En estas situaciones, el modelo YOLOv8 —a pesar de su alto rendimiento— tiende a fusionar detecciones cercanas en una sola entidad, especialmente cuando la separación espacial entre los individuos es mínima.

Consecuencias observadas:

- Conteo insuficiente de peatones en escenarios densos.
- Pérdida temporal del seguimiento para trayectorias individuales.
- Imposibilidad de determinar con precisión la dirección del cruce en casos puntuales.

Este problema es característico de modelos basados en **detección por cajas** (bounding boxes), donde la segmentación fina no se encuentra disponible.

2. Pérdida de IDs en Vehículos de Gran Tamaño

Otro desafío importante se observó en la detección y seguimiento de vehículos grandes (clases COCO 5 y 7), como buses y camiones:

- Cuando estos objetos ocupaban casi todo el ancho del cuadro, el modelo en ocasiones detectaba únicamente fragmentos de su estructura.
- Al abandonar el campo de visión y reingresar después de uno o dos frames, YOLOv8 asignaba un **nuevo ID**, interpretando el vehículo como una entidad distinta.

Esto derivaba en inconsistencias en el tracking y podía generar conteos duplicados si el vehículo cruzaba la línea virtual en un instante posterior.

3. Latencia por Uso del Modelo YOLOv8-Medium

El uso del modelo **yolov8m.pt**, seleccionado por su precisión intermedia, introdujo un desafío adicional: un nivel de latencia perceptible al ejecutar el sistema en hardware estándar (CPU sin GPU dedicada).

Efectos observados:

- Disminución del número efectivo de FPS.
- Ligero retraso entre el evento real y su visualización en pantalla.
- Incremento del tiempo de procesamiento en cuadros con múltiples objetos.

Si bien el sistema permaneció funcional, esta latencia limitó la fluidez del procesamiento y del HUD en tiempo real.

4.3 Soluciones Aplicadas

Para mitigar los desafíos descritos, se aplicaron diversas soluciones basadas tanto en ajustes internos del código como en configuraciones específicas del modelo YOLOv8. Las mejoras implementadas permitieron aumentar la estabilidad del sistema y mejorar la precisión del conteo.

1. Ajuste de Hiperparámetros del Modelo

Se modificaron los umbrales de detección de YOLOv8 para obtener una separación más clara entre objetos adyacentes:

conf=0.40

iou=0.50

- **Confianza (0.40):** aminora detecciones débiles, reduciendo falsos positivos.
- **IoU (0.50):** ayuda a resolver solapamientos entre cajas, permitiendo distinguir mejor a peatones muy próximos.

Este ajuste contribuyó directamente a mitigar el problema de la oclusión.

2. Activación del Tracking Persistente

El uso del parámetro:

persist=True

obliga al modelo a mantener la identidad de un objeto incluso si temporalmente no es detectado en uno o dos cuadros. Esta solución resultó especialmente efectiva para el problema de:

- pérdida accidental del ID en vehículos grandes,
- oscilaciones del tracking cuando el objeto sale brevemente de la escena,
- reapariciones tardías producto de bajas de FPS.

Gracias a esta función, el seguimiento se volvió más estable y coherente a lo largo del tiempo.

3. Feedback Visual mediante Cambio de Color

Se incorporó un mecanismo de retroalimentación visual que modifica dinámicamente el color de la línea virtual cuando ocurre un cruce:

- **Rojo:** cruce de izquierda a derecha.
- **Azul:** cruce de derecha a izquierda.

Esta solución no mejora directamente la precisión del modelo, pero facilita la depuración del sistema y permite validar, a simple vista, que la detección del cruce se ha realizado correctamente. En entornos académicos y demostrativos, esta función es clave para interpretar de forma clara los eventos detectados.

4. Filtrado de Detecciones por Tamaño (ruido visual)

Se agregó un filtro geométrico mínimo:

```
if w < 30 or h < 30:
    continue
```

Con esto se descartaron detecciones asociadas a:

- objetos muy lejanos,
- ruido digital,
- reflejos,
- artefactos del detector.

El filtrado contribuyó tanto a reducir falsos positivos como a mejorar la velocidad general del sistema al disminuir la cantidad de objetos a procesar.

5. Demo y Resultados

5.1 Funcionamiento del código

El sistema implementado realiza un procesamiento cuadro a cuadro del video de entrada, sobre el cual despliega una interfaz visual (HUD, *Heads-Up Display*) que permite interpretar fácilmente el funcionamiento interno del algoritmo. Esta interfaz combina elementos gráficos provenientes de OpenCV con los resultados de detección y tracking generados por el modelo YOLOv8. Su propósito es tanto informativo como diagnóstico, permitiendo validar visualmente el comportamiento del sistema durante su ejecución.

Elementos Visuales del HUD

1. Bounding Boxes

Cada objeto detectado —persona o vehículo— es encerrado dentro de un **rectángulo delimitador** (bounding box).
Además, se incluye:

- El color representativo según el tipo de clase.
- Una etiqueta que indica el ID del objeto y su clase (persona, auto, bus, camión).
- El puntaje de confianza otorgado por YOLOv8.

Estos rectángulos permiten observar si el modelo está clasificando correctamente y si mantiene la identidad de cada objeto entre distintos frames.

2. Línea de Referencia

Se dibuja una **línea vertical** fija en la posición definida por LINE_X_POS (38% del ancho del cuadro por defecto).
Esta línea constituye el elemento central del sistema, ya que actúa como umbral para determinar:

- entradas (izquierda → derecha),
- salidas (derecha → izquierda).

El color de la línea cambia dinámicamente (verde → rojo o azul) para reflejar un cruce detectado, proporcionando retroalimentación visual inmediata al usuario y facilitando la validación del conteo.

3. Contadores en Tiempo Real

En la esquina superior izquierda se despliegan tres indicadores principales:

- **Autos In:** contabiliza vehículos (IDs COCO 2, 5, 7) que cruzan la línea hacia la derecha.
- **Personas In:** contabiliza peatones que se desplazan de izquierda a derecha.
- **Personas Out:** registra peatones que cruzan en sentido contrario, es decir, de derecha a izquierda.

Estos contadores se actualizan en tiempo real y reflejan la lógica del sistema basada en el historial de centroides y en la detección del cruce con respecto a la línea virtual.

Flujo General del Funcionamiento

1. Se lee un frame del video.
2. Se aplica *frame skipping* para controlar la carga computacional.
3. YOLOv8 detecta y realiza tracking sobre personas y vehículos.

4. Se calcula el centro (centroid) de cada detección.
5. Se actualiza el historial de trayectorias por ID.
6. Se evalúa si hubo cruce respecto de la línea.
7. Se actualizan los contadores.
8. Se renderiza el HUD sobre el frame.
9. Se muestra el resultado en pantalla.

Este flujo garantiza un sistema estable, interpretativo y acorde a los objetivos del proyecto.

5.2 Métricas de desempeño observadas

El desempeño del sistema fue evaluado mediante un conjunto de pruebas realizadas sobre el video de muestras *AutoPersona.mp4*, donde se analizó la precisión del conteo, la fluidez visual y la consistencia del tracking.

1. Distinción Clara de Dirección de Movimiento

El sistema logró identificar de manera correcta el sentido de desplazamiento de peatones y vehículos, gracias a:

- el seguimiento persistente por ID,
- el análisis del cambio de posición del centro del objeto,
- la comparación de la trayectoria con respecto a la línea virtual.

Esto permitió que el conteo fuera estable y congruente en ambos sentidos.

2. Consistencia en el Conteo Vehicular

Los vehículos —autos, buses y camiones— fueron detectados de forma robusta debido al tamaño relativamente grande de sus bounding boxes.
En consecuencia, el sistema mantuvo:

- buena separación entre clases,
- mínima pérdida de IDs,
- alta fiabilidad en el cálculo de entradas.

Esta consistencia permitió validar la lógica direccional para flujos vehiculares.

3. Latencia Aceptable gracias al Frame Skipping

A pesar de utilizar un modelo de tamaño medio (*yolov8m.pt*), el rendimiento se mantuvo dentro de niveles aceptables para visualización en tiempo real. Esto se logró fundamentalmente por:

- la reducción de carga computacional (procesar solo 1 de cada 2 frames),

- el escalado uniforme del video,
- el tracking persistente que evita reprocesos innecesarios.

El sistema mostró una fluidez suficiente para uso demostrativo y académico en hardware de especificaciones estándar.

6. Conclusiones

6.1 Logros

El proyecto culminó con la implementación exitosa de un prototipo funcional capaz de detectar, rastrear y contabilizar personas y vehículos en tiempo real mediante un sistema de visión por computador completamente autónomo. Se logró cumplir con todos los objetivos planteados, destacando especialmente la creación de un contador direccional basado en una línea virtual, lo cual eliminó la necesidad de sensores físicos instalados en terreno (como barreras infrarrojas, lazos inductivos o sensores de presión). Esta característica representa una ventaja significativa al reducir los costos de infraestructura y facilitar la instalación en entornos donde la intervención física es difícil o no deseada.

Asimismo, el sistema demostró flexibilidad en su configuración: la ubicación de la línea de conteo (LINE_X_POS) puede adaptarse a cualquier posición del encuadre, permitiendo ajustar la lógica del conteo según la perspectiva de la cámara o el comportamiento específico del flujo en estudio. El prototipo también consiguió mantener un equilibrio adecuado entre precisión y velocidad, incluso ejecutándose en hardware de recursos limitados, gracias a las estrategias de optimización aplicadas (frame skipping, filtrado de ruido y tracking persistente).

En conjunto, estos logros posicionan al sistema como una herramienta práctica y escalable para aplicaciones reales de monitoreo de movilidad, aforo vehicular y análisis urbano.

6.2 Limitaciones

A pesar del rendimiento satisfactorio del sistema, persisten varias limitaciones propias de los métodos de visión por computadora monocular, es decir, aquellos basados en una sola cámara RGB sin sensores adicionales:

1. Sensibilidad a Condiciones Lumínicas Extremas

El rendimiento del modelo disminuye en entornos con:

- **iluminación muy baja** (noche sin fuentes de luz artificial),
- **contraluces fuertes** (cámara apuntando hacia una fuente de luz intensa),
- **reflejos o sombras pronunciadas**.

En estas situaciones, la red neuronal puede presentar fallos en los bordes de los objetos, pérdida del tracking o falsos positivos debido a variaciones bruscas en la luminancia.

2. Problemas con Objetos Parcialmente Visibles

Si un objeto aparece solo parcialmente en el primer cuadro del video, el modelo puede:

- asignar un ID tardío,
- no contabilizar el cruce si el objeto entra ya posicionado en la zona posterior a la línea virtual,
- fragmentar la detección hasta que el objeto entre completamente en escena.

Esto es una limitación inherente a cualquier sistema cuya lógica depende del seguimiento de trayectoria desde un punto inicial visible.

3. Dependencia de la Perspectiva de la Cámara

La precisión del conteo puede verse afectada por ángulos de cámara muy inclinados, cámaras muy cercanas al flujo o videos con distorsión óptica. En estos casos, las trayectorias laterales pueden volverse difíciles de interpretar correctamente.

6.3 Trabajo Futuro

Para elevar este prototipo a estándares industriales y mejorar su desempeño en escenarios complejos, se propone el siguiente conjunto de mejoras y extensiones:

1. Uso de Segmentación Semántica (YOLOv8-seg)

Reemplazar los bounding boxes rectangulares por **máscaras de segmentación por píxel**. Esto permitiría:

- resolver de mejor forma las oclusiones,
- distinguir personas agrupadas,
- obtener contornos precisos para análisis avanzados.

Las máscaras también habilitan métricas adicionales, como tamaño, área o densidad de flujo.

2. Integración con una Base de Datos

Persistir los conteos y estados del sistema mediante:

- **SQL** (PostgreSQL, MySQL), o
- **Firebase/Firestore** para aplicaciones en la nube,

permitiría generar:

- estadísticas históricas,

- reportes periódicos,
- análisis comparativos por horario, día o temporada.

Esto transforma el sistema en una herramienta de monitoreo continuo a nivel institucional o municipal.

3. Dashboard Web y API REST

Desarrollar un panel web que reciba datos en tiempo real mediante una API REST permitiría:

- monitoreo remoto del flujo en vivo,
- gráficos de tendencia,
- combinación con mapas interactivos,
- integración con aplicaciones municipales o de transporte.

Esta funcionalidad es esencial para aplicar el sistema a contextos de **Smart Cities**.

4. Fine-Tuning con Dataset Local

Entrenar el modelo con imágenes capturadas en Chile (patentes nacionales, tipos específicos de buses, vehículos de transporte público locales, condiciones climáticas típicas) permitiría:

- mejorar la precisión en contextos específicos,
- reducir errores de clasificación,
- adaptar el modelo a cámaras con diferentes resoluciones o ángulos,
- incrementar la estabilidad del tracking en entornos reales.

El fine-tuning es una de las mejoras con mayor impacto potencial en el sistema a mediano plazo.

Bibliografía

Beck, A. (2020). *Computer Vision: Models, Learning, and Inference*. Cambridge University Press.

Bewley, A., Ge, Z., Ott, L., Ramos, F., & Upcroft, B. (2016). Simple Online and Realtime Tracking. *2016 IEEE International Conference on Image Processing (ICIP)*, 3464–3468. https://doi.org/10.1109/ICIP.2016.7533003

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.

He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask R-CNN. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2961–2969. https://doi.org/10.1109/ICCV.2017.322

Jocher, G., Chaurasia, A., & Qiu, J. (2023). *YOLOv8: Ultralytics YOLO for State-of-the-Art Real-Time Object Detection*. Ultralytics.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., & Zitnick, C. L. (2014). Microsoft COCO: Common Objects in Context. *European Conference on Computer Vision (ECCV)*. https://doi.org/10.1007/978-3-319-10602-1_48

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 779–788. https://doi.org/10.1109/CVPR.2016.91

Redmon, J., & Farhadi, A. (2018). YOLOv3: An Incremental Improvement. *arXiv*. https://arxiv.org/abs/1804.02767

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2015). Going Deeper with Convolutions. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1–9. https://doi.org/10.1109/CVPR.2015.7298594

Ultralytics. (2023). *Ultralytics YOLO Documentation*. https://docs.ultralytics.com