

# Memoria Práctica 2: Conteo de Vehículos en Vías de Tráfico

Fabio Nesta Arteaga Cabrera y Agustín Darío Casebonne

## 1. Introducción

La presente memoria documenta la implementación de un sistema de conteo de vehículos que transitan en una vía de tráfico utilizando la librería **OpenCV** en Python. Se ha logrado detectar y contar vehículos en **dos de los tres carriles definidos**, demostrando la capacidad del sistema para manejar vehículos con **distintas velocidades**. Se han implementado técnicas de procesamiento de imagen y visión por computador mediante la librería antes mencionada, para detectar y contabilizar los vehículos.

## 1. Análisis de OpenCV

La librería **OpenCV** ha sido fundamental para el desarrollo de esta práctica. Se utilizaron las siguientes técnicas:

1. **Background Subtraction**: Para identificar objetos en movimiento.
2. **Detección de Contornos**: Mediante **cv2.findContours** para obtener la posición de los vehículos.
3. Se aplican operaciones de **Cierre** y **Apertura** para eliminar ruido en la máscara de fondo:
  - **Cierre (MORPH\_CLOSE)**: Rellena pequeños huecos dentro de las áreas detectadas.
  - **Apertura (MORPH\_OPEN)**: Elimina pequeños objetos no deseados o ruido en la imagen.

## 2. Descripción del Problema

El objetivo principal es desarrollar un sistema que:

1. Detecte y cuente vehículos en un vídeo de tráfico.
2. Permite la monitorización de distintos carriles.
3. Maneje vehículos a diferentes velocidades y con distintas dimensiones.
4. Detecte vehículos que pasan muy cerca uno de otros.
5. Saber manejar el problema de las sombras durante el conteo de los coches.

## 3. Descripción del Código

Las bases principales de nuestro código son las que a continuación mencionaremos, estas técnicas fueron aprendidas y explicadas en clase por el profesorado.

### 3.1. Lectura y Configuración del Vídeo

Se utiliza la clase **cv2.VideoCapture** para cargar el vídeo y definir sus dimensiones.

### 3.2. Extracción del Fondo

Se implementa el método **Background Subtraction** usando **cv2.createBackgroundSubtractorMOG2** para identificar vehículos.

Este método permite diferenciar los objetos en movimiento, en nuestro caso vehículos, del fondo estático.

### 3.3. Configuración de Carriles

Cada carril incluye:

- **Posiciones de líneas:** Para determinar el cruce de los vehículos.
- **Color:** Facilita la visualización en pantalla.
- **Contador:** Almacena el número de vehículos detectados.

### 3.4. Detección y Conteo de Vehículos

El código procesa cada frame del vídeo:

- Aplica **Background Subtraction** y operaciones morfológicas para mejorar la detección.
- Encuentra **contornos** de los objetos en movimiento.
- Verifica si un vehículo cruza una línea predefinida del carril

## 4. Resultados Obtenidos

El sistema implementado logró los siguientes resultados:

1. **Conteo de Vehículos:**
  - Se contabilizaron correctamente los vehículos que cruzaron las líneas de los **carriles 1 y 2**.
  - El conteo funciona para vehículos de **distintas velocidades**.
2. **Visualización:**
  - Se muestran los contornos de los vehículos y las líneas de detección.
  - Cada carril cuenta con un contador independiente y visualizado con su color correspondiente.
3. **Eficacia:**
  - El sistema puede distinguir entre vehículos en movimiento y el fondo estático con buena precisión.

## 5. Limitaciones

Si bien se logró cumplir con el objetivo del enunciado, existen algunas limitaciones:

- El conteo en el carril 3 no funciona del todo bien, ya que hay algunos vehículos con los que tiene problemas.
- No se distinguen diferentes tipos de vehículos.

- El sistema podría mejorarse para manejar vehículos que circulan en **direcciones opuestas**.

## 6. Conclusión

El sistema desarrollado cumple con los objetivos principales al contar vehículos en dos carriles, demostrando eficacia frente a vehículos de distintas velocidades. El uso de **OpenCV** permitió la implementación de técnicas eficientes de detección y conteo, sentando las bases para futuras mejoras, como la detección de tipos de vehículos y el conteo en escenarios más complejos.