Grado en Ingeniería Informática

Seguimiento de vehículos a través de videovigilancia y estimación de su velocidad.

Vehicles tracking through video surveillance and estimation of their speed.

Realizado por
Rubén Garciá Rojas

Tutorizado por
Ezequiel López Rubio
Jorge García González

Departamento
Departamento de Lenguajes y Ciencias de la Computación
UNIVERSIDAD DE MÁLAGA

MÁLAGA, Junio 2021

Agradecimientos

A Ezequiel López Rubio por su aportación matemática en la que se basa este trabajo, y a mi co-tutor Jorge García Gónzález por su ayuda e implicación durante el desarrollo del TFG.

En especial, un agradecimiento a Carmen Vargas, quien fue mi profesora de informática durante cinco años, en los que me ayudó y motivó para que estudiara esta carrera.

Resumen

La aplicación de métodos de aprendizaje profundo relacionados con la visión por computador ha dado lugar a importantes mejoras en muchas aplicaciones industriales. Algunas de las aplicaciones relacionadas pueden ser la mejora en sistemas de videovigilancia, así como la estimación de la concurrencia en lugares. Entre las diferentes líneas de investigación emergentes, este trabajo se centra en la prevención de accidentes a través de las secuencias captadas por los sistemas de videovigilancia, estimando la velocidad de los vehículos que circulan por estas vías.

Los métodos más comunes para la estimación de velocidad se basan en usos de sensores, como por ejemplo los dispositivos lidar, pero estos implican un mayor coste en el presupuesto. Por esa misma razón en este trabajo se propone un método que utilizando cámaras de videovigilancia, las cuáles ya están ampliamente instaladas en una gran cantidad de calles y carreteras, se pueda obtener una estimación fiable.

En este trabajo, se presenta una propuesta para obtener una estimación fiable y suave de la velocidad de un vehículo mediante redes neuronales convolucionales, *CNN*, y regresión por kernel. En primer lugar, se realiza una homografía del plano capturado por la cámara y, para detectar vehículos de forma instantánea, se utiliza un método de detección de objetos basado en CNN. A continuación, se aplica un kernel de regresión para estimar suavemente la velocidad de los vehículos a partir de las coordenadas junto con la marca de tiempo capturada. Para comprobar la fiabilidad del procedimiento presentado, se han realizado varias pruebas con vídeos seleccionados del conjunto de datos *BrnoCompSpeed*.

Keywords: Prevención de accidentes, Estimación de velocidad, Redes neuronales convolucionales, Detección de objetos, Regresión por kernel.

Abstract

The application of deep learning methods related to computer vision has led to significant improvements in many industrial applications. Some of the related applications can be the improvement in video surveillance systems, as well as the estimation of concurrency in places. Among the different emerging lines of research, this work focuses on preventing accidents through sequences captured by video surveillance systems, estimating the speed of vehicles circulating on these roads.

The most common methods for vehicle speed estimation are based on the use of sensors, such as lidar devices, but these involve a higher cost in the budget. For the same reason, in this work we propose a method that using video surveillance cameras, which are already widely installed in a large number of streets and highways, a reliable estimation can be obtained.

In this work, we present a new procedure to obtain a reliable and smooth estimation of the speed of a vehicle through convolutional neural networks, *CNN*, and kernel regression. First, a homography of the plane captured by the camera is performed and, to detect vehicles instantly, a CNN-based object detection method is used. Then a regression kernel is applied to smoothly estimate the speed of the vehicles from the coordinates along with the captured timestamp. In order to check the reliability of the presented procedure, several tests have been performed on selected videos from the *BrnoCompSpeed* dataset.

Keywords: Accident prevention, Speed estimation, Convolutional neural networks, Object detection, Kernel regression.

Índice

1. Introducción **7**

1.1. Motivación 7

1.2. Objetivo 8

1.3. Estructura de la memoria 9

2. Marco teórico **11**

2.1. Redes neuronales artificiales 11

2.2. Redes de aprendizaje profundo 11

2.3. Redes convolucionales 12

2.4. Detección de objetos 12

2.5. Homografía 13

3. Metodología **15**

4. Desarrollo **19**

4.1. Librería: simple-object-detection 19

4.1.1. Características de la librería 20

4.1.2. Modelos implementados 22

4.2. Librería: simple-object-tracking 22

4.2.1. Características de la librería 22

4.3. Librería: vehicle-speed-estimation 23

4.3.1. Características de la librería 24

4.4. Entorno de trabajo 24

4.4.1. Google Colab 26

5. Experimentación **27**

5.1. El dataset BrnoCompSpeed 27

5.2. Detección de vehículos 28

5.3. Seguimiento de vehículos 29

5.4. Métodos de estimación de velocidades de los vehículos 29
5.5. Transformación de la cámara al mundo real 30
5.6. Evaluación 30
5.7. Resultados 32

6. Conclusiones **35**

1 Introducción

1.1. Motivación

La toma de decisiones basada en datos es una parte esencial de la nueva era de la información. Nuestra sociedad cuenta con un número cada vez mayor de dispositivos de recogida de información, como cámaras en la calle, sensores o teléfonos inteligentes. Analizar ese enorme volumen de información no es abordable por un humano, por lo que es necesario utilizar la capacidad de procesamiento de los ordenadores para obtener información útil que aprovechar. Este es un elemento clave en diversos campos como el análisis de mercados [1], las redes sociales [2] o la concesión de créditos [3]. Su punto en común es procesar la información generada por muchos usuarios para predecir o, al menos, reaccionar. Estas características también se dan en el campo del análisis del tráfico o del transporte, donde hay que controlar a muchos conductores.

La aplicación de los métodos de visión por ordenador a la videovigilancia y la seguridad son campos de intensa investigación. Nuestra sociedad cuenta con un número creciente de cámaras de vídeo instaladas en lugares públicos, como las dispuestas en calles, carreteras y autopistas para vigilar el tráfico.

Esta cantidad masiva de imágenes puede emplearse con fines beneficiosos como la estimación de la contaminación generada [4, 5] y la detección o prevención de accidentes de tráfico [6, 7]. La estimación de la velocidad de los vehículos suele ser un elemento clave para todas estas aplicaciones o incluso el objetivo final. Estas mediciones pueden obtenerse mediante el uso de sensores específicos diseñados para ese fin concreto. Aun así, su instalación implica gastos añadidos, por lo que la obtención de la velocidad directamente a partir de secuencias de vídeo facilita la implementación de los sistemas y requiere únicamente un sensor de propósito general: una cámara de videovigilancia.

Las técnicas tradicionales para detectar y rastrear elementos de una imagen implicarían una técnica de segmentación de primer plano [8] basada en distribuciones gaussianas [9], o un modelado estadístico de fondo [10]. Sin embargo, los avances de los últimos años en computación paralela mediante Unidades de Procesamiento Gráfico (GPUs) aplicadas a Redes Neuronales Artificiales han dado lugar a métodos de detección de objetos basados en capas convolucionales que permiten detectar objetos para identificar su posición dentro de una imagen y qué tipo de objeto son [11, 12, 13].

Dentro del campo del transporte y el tráfico, se ha llevado a cabo un uso creciente de los sistemas de videovigilancia para controlar los vehículos con el fin de estimar la densidad del tráfico con el objetivo principal de reducir el número de accidentes. Dentro del campo de la videovigilancia, se puede encontrar un gran número de trabajos relacionados. [14] propone un sistema de detección y recuento de vehículos basado en la visión, extrayendo primero la superficie de la carretera en la imagen para dividirla en una zona remota y otra cercana mediante un nuevo método de segmentación. A continuación, las dos zonas anteriores se introducen en la red YOLOv5 para detectar el tipo de vehículo y su ubicación. Por último, se obtienen las trayectorias de los vehículos mediante el algoritmo ORB. [15] propone la aplicación de la detección de objetos para los sistemas de vigilancia del tráfico con el fin de detectar los vehículos y así poder inferir su clase. El artículo [16] propone un método mejorado para detectar objetos pequeños en secuencias de vídeo de carretera utilizando CNNs y procesos de superresolución, aumentando el número de detecciones. [17] presenta una aplicación de monitorización de tráfico por vídeo en tiempo real basada en la detección y el seguimiento de objetos para controlar parámetros de tráfico como el número de vehículos.

1.2. Objetivo

Nuestra propuesta utiliza un método de detección de objetos basado en CNN (Convolutional Neural Networks) para obtener información de los vehículos con el fin de detectarlos y rastrearlos. Para obtener una equivalencia con el mundo real, transformamos la perspectiva de la cámara mediante una homografía. Finalmente, aplicamos el estimador de regresión por kernel de Nadaraya-Watson para obtener una estimación suave y fiable de la velocidad.

1.3. Estructura de la memoria

El resto de este trabajo está estructurado como sigue: La sección 3 en la página 15 explica la metodología aplicada, la sección 5 en la página 27 muestra toda la información relacionada con los experimentos realizados para apoyar nuestra propuesta, incluyendo sus resultados, y la sección 6 en la página 35 muestra nuestras conclusiones y posibles trabajos futuros.

2
Marco teórico

2.1. Redes neuronales artificiales

Las redes neuronales artificiales son un sistema de computación basado en las redes neuronales biológicas de los animales.

Pertenecen al campo de aprendizaje computacional, que es el campo de estudio de los algoritmos de aprendizaje automático. Su propósito es facilitar al usuario el aprendizaje a partir de un conjunto de datos, de modo que pueda predecir resultados en datos que no hayan sido vistos previamente.

Las redes neuronales se componen de conjuntos de neuronas cuyas salidas están conectadas a otras. Cada conexión tiene un peso que ponderará el valor de entrada en la neurona, y a su vez, en la salida puede existir una función que se aplique sobre el valor calculado.

El aprendizaje de una red neuronal se realiza utilizando un conjunto de datos el cuál se conoce el valor que se espera que devuelva la red neuronal. Utilizando este conjunto se aplica un algoritmo de aprendizaje que vaya modificando los pesos sinápticos de modo que penalice cuando la red no se ajusta a la salida esperada, y así la red se vaya ajustando a los valores, de modo que cuando se introduzca uno que no haya sido visto anteriormente, la red pueda predecirlo.

Las distintas investigaciones sobre el cerebro propician nuevos modelos de redes neuronales, como son por ejemplo las redes de aprendizaje profundo.

2.2. Redes de aprendizaje profundo

Las redes de aprendizaje profundo son un tipo de redes neuronales artificiales. Se caracterizan principalmente por poseer una gran cantidad de capas ocultas intermedias. Intenta asemejarse más al funcionamiento del cerebro, interconectando una gran cantidad de neuro-

nas de una capa con las de la siguiente, realizando un aprendizaje en la que tras cada etapa modifica los pesos entre todas las neuronas de cada capa.

Sus campos de aplicación son: reconocimiento de voz, procesado de lenguaje natural, traducción, bioinformática, diseño de drogas, análisis de imágenes médicas y visión por computador, entre otros.

En lo que a este trabajo respecta, su principal uso está en la detección de objetos, y su clasificación, en particular, de vehículos.

2.3. Redes convolucionales

Las redes convolucionales son una clase de red de aprendizaje profundo. Consisten en múltiples capas de filtros convolucionales para la extracción de características. En la última capa se encuentran neuronas de perceptrón sencillas para la clasificación de las características extraídas.

En el caso de las imágenes, se utilizan los filtros convolucionales para extraer las características de los objetos, ya sea haciendo la derivada de la imagen y obteniendo los bordes, u otro tipo de filtro. Finalmente se utilizan neuronas perceptrón para clasificar los objetos y asignarles una puntuación.

Su aplicación se extiende por una gran cantidad de campos, como son la detección de objetos en imágenes y vídeos, clasificación de imágenes, segmentación de imágenes, análisis de imágenes médicas, procesado de lenguaje natural, series temporales financieras, etc.

2.4. Detección de objetos

La detección de objetos imágenes es la tecnología que nos permite conocer los distintos objetos que se encuentran en una imagen, y su posición. La forma más usual de obtener estos datos es a través de la segmentación de la imagen en rectángulos y calculando la probabilidad de que haya un objeto ahí, y finalmente clasificar qué tipo de objeto es.

A día de hoy, para la detección de objetos se utilizan redes convolucionales, un tipo de red de aprendizaje profundo, ya que son con las que mejores resultados se obtienen.

2.5. Homografía

La homografía es un tipo de transformación geométrica que nos permite determinar la correspondencia entre dos figuras geométricas planas.

Se puede encontrar un ejemplo de homografía realizado en este trabajo en la imagen 8, la cuál sería una homografía de la imagen6, dado los puntos del rectángulo que forman las líneas marcadas en la imagen.

Su principal utilidad en este proyecto reside en la posibilidad de convertir la perspectiva obtenida desde la cámara de videovigilancia a la perspectiva obtenida a través de un satélite o a una perspectiva ortogonal en la que podamos obtener una correspondencia entre una distancia en la imagen y una distancia en el mundo real de forma constante. Es decir, que la distancia real entre dos puntos del plano se pueda obtener a partir del cálculo de la distancia en píxeles multiplicando por un factor entre la distancia entre cada píxel y la correspondencia en el mundo real.

3
Metodología

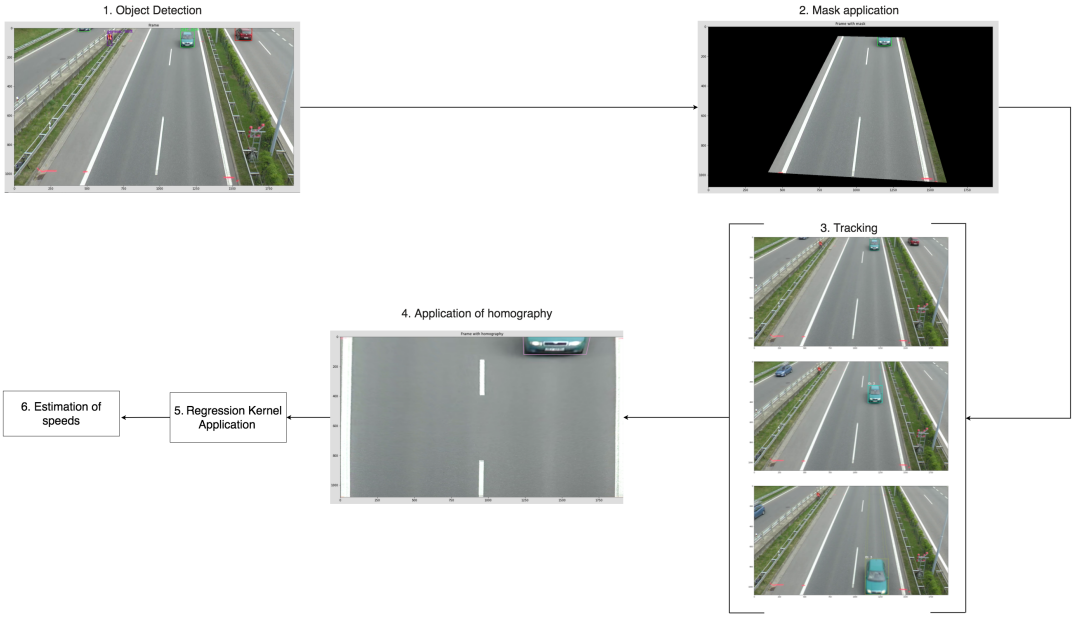


Figura 1: Flujo de trabajo de la técnica propuesta. En primer lugar, se realiza la detección de vehículos. Se aplica una máscara para seleccionar los elementos del carril de la carretera requerido. Posteriormente, se realiza el seguimiento del vehículo. Por último, se aplica la homografía sobre el plano, se realiza la regresión por kernel y se estima la velocidad.

A continuación, se especifica en detalle nuestra propuesta. Suponemos que existe un plano por el que se mueven los vehículos, es decir, que el suelo se puede aproximar con un plano. El primer paso de nuestro método consiste en proyectar el plano de la cámara $\mathcal{C}$ sobre el plano de la carretera $\mathcal{R}$ mediante una homografía adecuada. Para cada fotograma de vídeo entrante se obtienen las cajas delimitadoras de los vehículos en la escena mediante una red profunda de detección de objetos correspondiente al paso dos del flujo representado por la figura 1. Posteriormente, como se indica en el punto dos del flujo de trabajo, se aplica una máscara como región delimitadora para seleccionar la carretera deseada. Como se indica en el punto

tres del flujo de trabajo, se realiza el seguimiento de los vehículos. Se aplica la homografía al seguimiento, para obtener las posiciones de los vehículos en el plano, correspondiente al paso cuatro del flujo de trabajo. La posición de cada vehículo se aproxima por el centro de su cuadro delimitador asociado, dado por la media de las esquinas de la caja delimitadora proyectada sobre $\mathcal{R}$. Después se aplica un filtro de Kalman a estas detecciones de vehículos, de modo que se registran las trayectorias de los vehículos. Cada trayectoria registrada S viene dada por un conjunto de posiciones $\mathbf{x} \in \mathbb{R}^2$ en el plano de la carretera $\mathcal{R}$, junto con sus correspondientes marcas de tiempo asociadas $t \in \mathbb{R}$:

$$S = \{(\mathbf{x}_i, t_i) \mid i \in \{1, 2, ..., N\}\} \tag{1}$$

donde N es el número de instantes en los que el vehículo ha sido detectado a través de la secuencia de vídeo.

Debido al ruido y los artefactos de la secuencia de vídeo entrante, la ocultación de los vehículos detrás de otros objetos, y las imperfecciones del procedimiento de de detección de objetos, se espera que cada trayectoria grabada S contenga contenga errores considerables de posición. Este problema debe abordarse para obtener una estimación más fiable de la velocidad del vehículo. Como se indica en el paso cinco del flujo de trabajo, aquí proponemos emplear la regresión kernel para este propósito.

Consideremos el estimador de regresión kernel de Nadaraya-Watson [18, 19, 20]. Para un instante de tiempo t, la posición del vehículo se estimada como una media ponderada de las posiciones registradas cerca del instante t:

$$\hat{\mathbf{x}}(t) = \frac{\sum_{i=1}^{N} K(t - t_i)\, \mathbf{x}_i}{\sum_{i=1}^{N} K(t - t_i)} \tag{2}$$

donde K es el kernel con un ancho de banda h. Por ejemplo, se emplearán kernels Gaussianos, triangulares o cuadráticos:

$$K_{Gauss}(t - t_i) = \exp\left(-\frac{(t - t_i)^2}{h}\right) \tag{3}$$

$$K_{triangular}(t - t_i) = \left(1 - \frac{|t - t_i|}{h}\right) \mathbb{I}(|t - t_i| < h) \tag{4}$$

$$K_{quadratic}\left(t-t_i\right)=\left(1-\left(\frac{t-t_i}{h}\right)^2\right)\mathbb{I}\left(\left|t-t_i\right|<h\right) \tag{5}$$

donde $|\cdot|$ indica el valor absoluto de un número real y $\mathbb{I}$ representa a la función indicatriz.

El vector de velocidad puede ser estimado como la derivada de la posición estimada con respecto al tiempo t:

$$\hat{\mathbf{v}}\left(t\right)=\frac{d\hat{\mathbf{x}}}{dt}\left(t\right)=\frac{\left(\sum_{i=1}^{N}K'\left(t-t_i\right)\mathbf{x}_i\right)\left(\sum_{i=1}^{N}K\left(t-t_i\right)\right)}{\left(\sum_{i=1}^{N}K\left(t-t_i\right)\right)^2}$$

$$-\frac{\left(\sum_{i=1}^{N}K'\left(t-t_i\right)\right)\left(\sum_{i=1}^{N}K\left(t-t_i\right)\mathbf{x}_i\right)}{\left(\sum_{i=1}^{N}K\left(t-t_i\right)\right)^2} \tag{6}$$

donde K' representa la derivada del kernel de la función K:

$$K'_{Gauss}\left(t-t_i\right)=-\frac{2}{h}\left(t-t_i\right)\exp\left(-\frac{\left(t-t_i\right)^2}{h}\right) \tag{7}$$

$$K'_{triangular}\left(t-t_i\right)=-\frac{1}{h}\mathrm{sgn}\left(t-t_i\right)\mathbb{I}\left(\left|t-t_i\right|<h\right) \tag{8}$$

$$K'_{quadratic}\left(t-t_i\right)=-\frac{2}{h^2}\left(t-t_i\right)\mathbb{I}\left(\left|t-t_i\right|<h\right) \tag{9}$$

donde sgn representa la función signo:

$$\mathrm{sgn}\left(z\right)=\begin{cases}-1 & \text{if } z<0\\ 0 & \text{if } z=0\\ 1 & \text{if } z>0\end{cases} \tag{10}$$

Luego, la magnitud ρ y el ángulo θ del vector de velocidad $\mathbf{v}$ se puede estimar, como se indica en el paso seis del flujo de trabajo:

$$\hat{\rho}\left(t\right)=\left\|\hat{\mathbf{v}}\left(t\right)\right\| \tag{11}$$

$$\hat{\theta}\left(t\right)=\arctan\left(\hat{v}_1\left(t\right),\hat{v}_2\left(t\right)\right) \tag{12}$$

donde $\|\cdot\|$ representa la norma Euclídea de un vector.

4
Desarrollo

El trabajo realizado no tiene como objetivo principal el desarrollo de librerías ni herramientas, pero sí que han sido parte importante, ya que toda la fase de experimentación tiene como base el uso de estas librerías.

En el planteamiento inicial se proponía la evaluación de distintos modelos de detección de objetos, así como de seguimiento y estimación de velocidad, por tanto ha hecho falta un entorno que facilitase todo el proceso de pruebas e investigación.

Las tres librerías realizadas se encuentran en sus respectivos repositorios de GitHub con todas las clases y método documentados.

- Simple Object Detection: https://github.com/RubenEu/simple-object-detection/
- Simple Object Tracking: https://github.com/RubenEu/simple-object-tracking/
- Vehicle Speed Estimation: https://github.com/RubenEu/vehicle-speed-estimation/

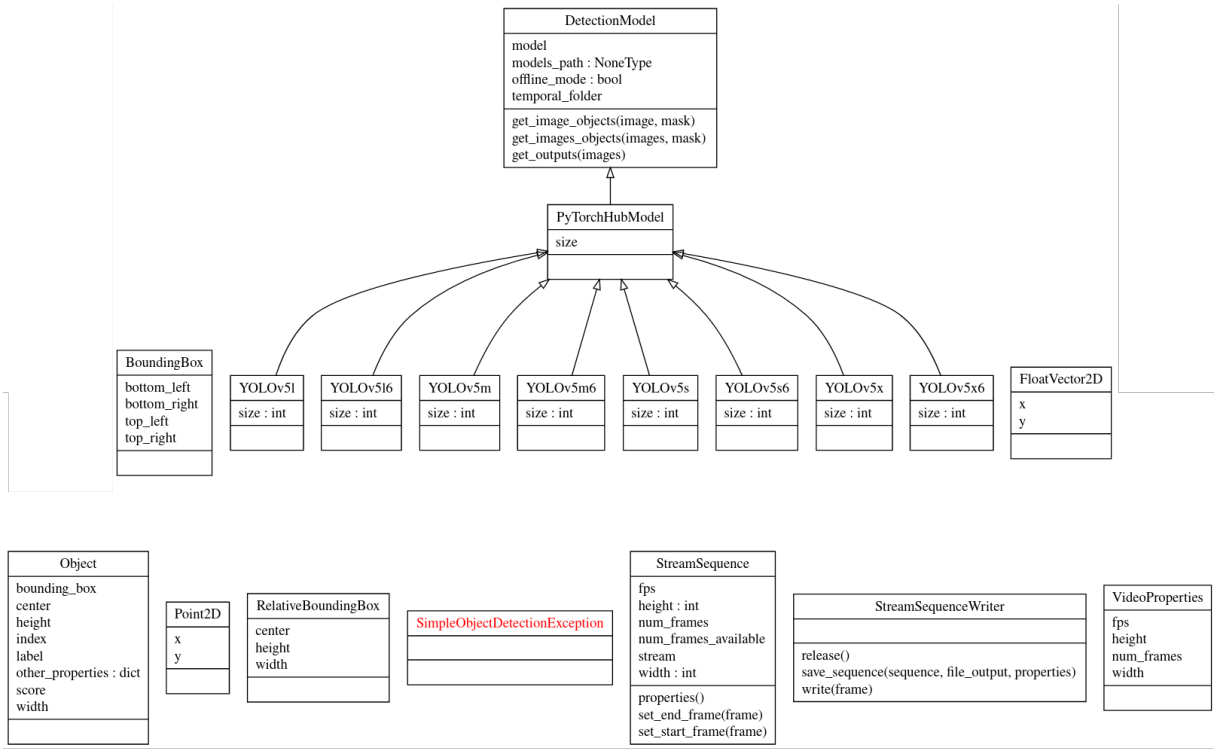
4.1. Librería: simple-object-detection

A día de hoy existen diferentes librerías para trabajar los modelos de detección de objetos, como puedan ser tensorflow o pytorch, e incluyen su propio entorno para facilitar el uso de distintos modelos, como son tensorflow-hub[21] o pytorch-hub[22]. Únicamente se ha quedado implementada la librería pytorch, ya que es la que poseía mayores facilidades para el uso de los modelos YOLO, y tener precargada la librería de tensorflow conllevaba un consumo extra de procesamiento, es por ello que se decidió eliminar de las primeras versiones.

El problema de estas librerías es que únicamente descargan el modelo y permiten el procesado de las entradas dado el formato especificado (generalmente una resolución concreta), pero no están estandarizadas las salidas.

La librería pretende estandarizar las salidas de las distintas implementaciones de modelos, de modo que se pueda trabajar con ellos sin atender a cómo devuelve la información.

Figura 2: Diagrama UML de la librería simple-object-detection.



4.1.1. Características de la librería

- Clase abstracta para la implementación de modelos de redes neuronales convolucionales para la detección de objetos en imágenes.
- Clase para el almacenamiento de la información de los objetos detectados por la red neuronal. Así se obtiene una salida estandarizada en cada modelo implementado.
- Clase abstracta para la implementación de modelos de pytorch-hub.
- Herramientas para dibujar las cajas delimitadoras de los objetos sobre la imagen.
- Herramientas para la carga de secuencias de vídeo.
- Métodos para la generación de los archivos con las detecciones de los objetos.

- Métodos para filtrar los objetos detectados.

La clase abstracta *DetectionModel* permite la implementación siguiendo un estándar dado por la clase *Object*.

Esta clase almacena la siguiente información:

- Índice del objeto. Se asigna en orden secuencial.
- Centro del objeto.
- Caja delimitadora.
- Puntuación asignada por la red neuronal.
- Etiqueta del objeto. Indica a qué clase pertenece.

Además, la caja delimitadora viene dada en los dos formatos más usados: ancho y alto, o dadas sus esquinas. Lo más óptimo sería almacenar únicamente dos esquinas porque las detecciones dadas por las redes suelen estar alineadas con los ejes de la imagen, pero en el caso de aplicarse una homografía esto no tendría por qué cumplirse, por lo que es interesante almacenar la información de lsa cuatro esquinas, para así, si se realiza la homografía de un objeto, este tenga los puntos de las esquinas con la homografía aplicada.

Otro de los grandes problemas cuando se realiza detección en secuencias de vídeo, es que si esta es muy larga, no puede ser cargado directamente en memoria. Como solución se ha creado la clase *StreamSequence* para la lectura de un vídeo a través de un buffer, así como la clase *StreamSequenceWriter* para el guardado de secuencias de vídeo en el caso de que hayan sido editados para añadir información. Esto será útil para utilizar conjuntamente con la librería simple-object-tracking mencionada posteriormente.

Otra de las grandes utilidades de esta librería son los filtros de objetos. Como estos están estandarizados en una clase, pueden usarse métodos para aplicarse como filtros sobre los objetos.

La detección sobre vídeos suele ser muy costosa en tiempo dependiendo de la red, para evitar tener que realizar este proceso cada vez que se quieren obtener los objetos, existen utilidades para guardar las detecciones en un archivo.

4.1.2. Modelos implementados

Durante la fase de desarrollo fueron implementados diferentes modelos ya preentrenados, tales como CenterNet[23] o EfficientDET[24], pero finalmente se eliminaron e implementar únicamente las distintas variantes de la versión de YOLOv5[25].

- YOLOv5s
- YOLOv5m
- YOLOv5l
- YOLOv5x

4.2. Librería: simple-object-tracking

La librería simple-object-tracking propone la implementación de modelos de seguimientos de objetos.

Se provee una estructura de datos para el almacenamiento de los seguimientos de los objetos en una secuencia de vídeo, de tal modo que pueda consultarse de distintas formas acerca de ellos. Los distintos métodos se encuentran documentados en la propia clase.

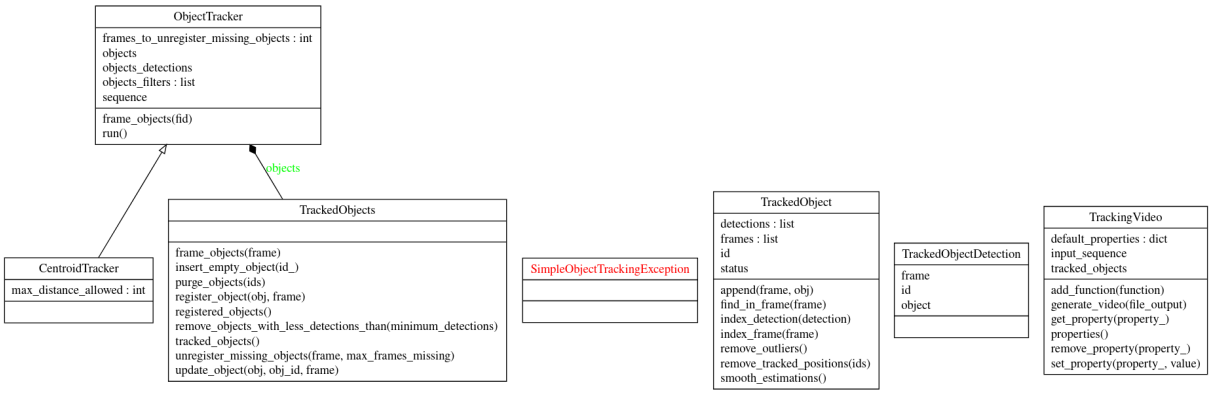
Además, se permite la implementación de modelos de forma que se pueda realizar el seguimiento con distintos modelos y parámetros según las necesidades de la secuencia de vídeo.

Para el seguimiento de los objetos se requiere de la librería simple-object-detection como base de los objetos detectados.

4.2.1. Características de la librería

- Clase abstracta para la implementación de modelos de seguimiento.
- Estructura de datos para el almacenamiento y consulta de los seguimientos de los objetos.
- Modelo de seguimiento basado en un punto dado de los objetos (centro o esquinas).
- Herramientas para la edición de imágenes con los datos de los seguimientos. Se permite la inclusión de los trazados realizados, información del frame, tiempo, etc.

Figura 3: Diagrama UML de la librería simple-object-tracking.



El modelo basado en el centroide implementado posee parámetros que permiten la modificación del comportamiento:

- Distancia máxima permitida para indicar que dos objetos pueden ser emparejados.
- Máscara para indicar sobre qué zona de la secuencia de vídeo realizar los seguimientos.

La distancia máxima permitida nos permite evitar que se realicen emparejamientos incorrectos. Cuando un vehículo desaparece de la secuencia, no pueda ser emparejado con otro vehículo que acaba de aparecer, por ejemplo.

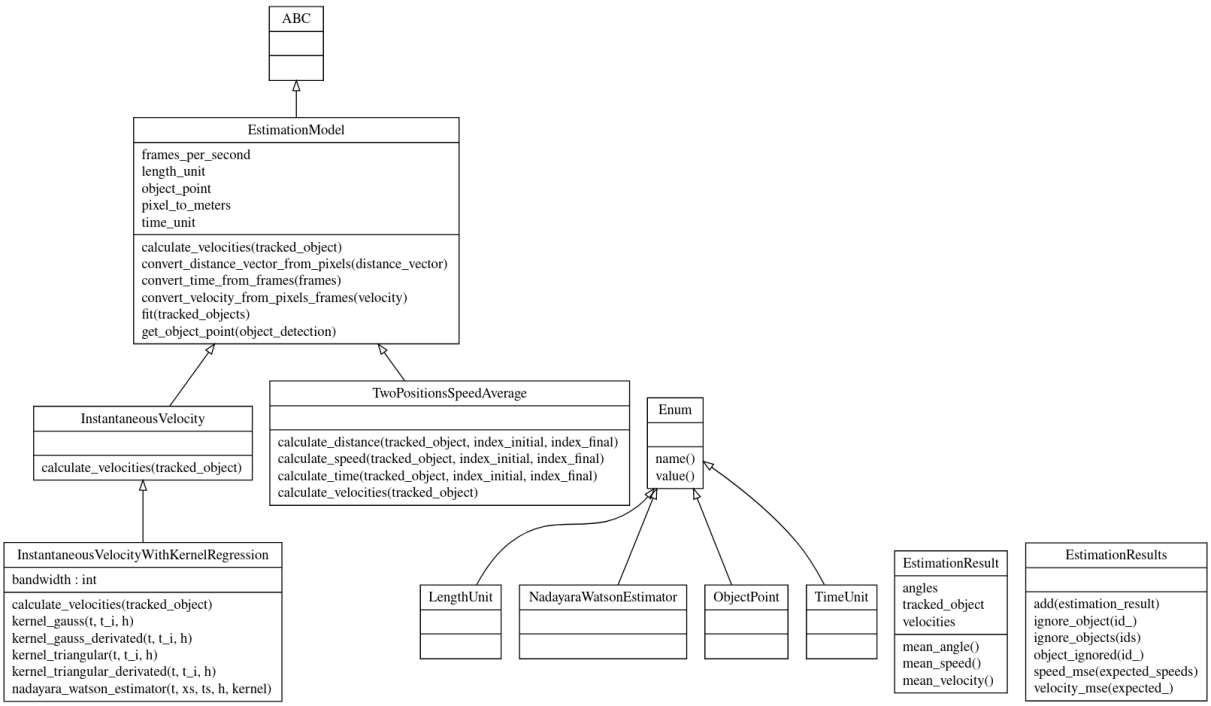
Y la máscara nos permite indicar los carriles de la carretera sobre los que realizar el seguimiento evitando aquellos que no son de interés.

4.3. Librería: vehicle-speed-estimation

Finalmente, para facilitar el proceso de ajuste para las pruebas de experimentación, se ha creado una librería que permite la implementación de distintos modelos de velocidad. Esta hace uso de las clases de las librerías anteriormente mencionadas.

Esta librería permite la estimación de velocidad de vehículos utilizando las técnicas descritas en la sección 5. Para ello hace faltan utilidades para aplicar la homografía tanto a las secuencias de vídeo como a los objetos y sus seguimientos. Todas estas herraimentas se encuentran implementadas aquí.

Figura 4: Diagrama UML de la librería vehicle-speed-estimation.



4.3.1. Características de la librería

- Clase abstracta para la implementación de distintos modelos de estimación de velocidad.
- Herramientas para aplicar la homografía a la secuencia, objetos detectados y seguimientos de los objetos.

Los modelos implementados son:

- Estimación de velocidad media basado en la primera y última posición del objeto.
- Velocidad instantánea basada en los incrementos de posición e instante del objeto.
- Derivada de la posición dada por el estimador de Nadaraya-Watson para el cálculo de la velocidad instantánea (propuesta realizada).

4.4. Entorno de trabajo

La realización de las distintas librerías ha sido realizada a través de PyCharm, utilizando la licencia educativa proveída por la UMA.

El lenguaje de programación utilizado ha sido Python[26] en la versión 3.8. Las librerías realizadas 4.1, 4.2 y 4.3 encuentran disponible para todas las versiones de Python 3.

Se han utilizado distintas durante el desarrollo, además del IDE y lenguaje indicado. Para el control de versiones de las librerías se ha utilizado git, y como alojamiento GitHub.

Librerías utilizadas:

- pytorch. Librería para cargar los modelos de redes neuronales.
- numpy. Librería optimizada para el cálculo numérico y diversas funcionalidades matemáticas.
- tqdm. Utilidad para mostrar los procesos de carga lentos, como puede ser la detección de una secuencia de vídeo o el proceso de generación de los seguimientos de los vehículos.
- opencv. Librería con multitud de funcionalidades enfocada el la visión por computador.
- matplotlib. Librería para la generación de gráficos.
- pillow. Librería para la carga y modificación de imágenes.
- bumpversion. Librería para indicar el número de versión del proyecto. Útil para tener un control de qué versión exacta se está utilizando.

Siendo estas las más relevantes. El resto de librerías utilizadas se pueden encontrar en los archivos requirements.txt de las librerías 4.1, 4.2 y 4.3.

La fase de experimentación se ha realizado utilizando Google Colab, una versión online de Jupyter ofrecida por google. Google Colab permite el uso de computación con GPUs (Graphic Processor Units) lo cuál resulta muy útil para las redes neuronales de detección de objetos.

En cuanto al despliegue de los notebooks, al requerir estos la instalanción de las librerías, ésta ha sido realizada a través de los repositorios etiquetando las versiones de las librerías para posteriormente clonarlos en el entorno del notebook e instalarlos. Esto ha permitido que la ejecución de los notebooks sea independiente de archivos externos y robusta frente a cambios en las librerías.

4.4.1. Google Colab

Colab es una plataforma de computación en la nube de Google usada para la investigación. Está basada en Jupyter, una herramienta utilizada para la creación de notebooks usando el lenguaje de programación Python.

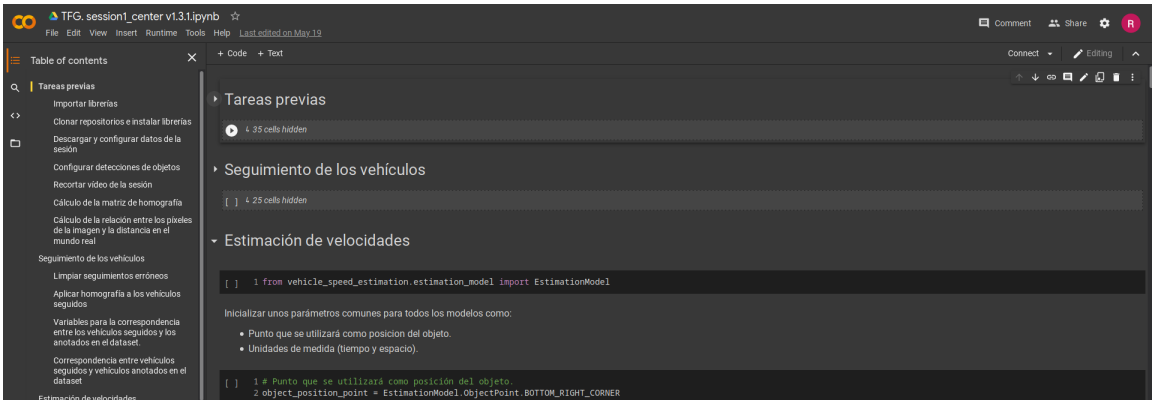


Figura 5: Notebook utilizado para experimentación de la estimación de velocidad de los vehículos con distintos modelos y la propuesta con suavizado por kernel.

Un notebook de Jupyter o Google Colab es un documento que permite la inclusión de celdas de código cuya salida es imprimida a continuación, de modo que permite mezclar el texto de los documentos clásicos con ejecuciones dinámicas de código.

Además, Google ofrece en su plataforma Colaboratory el uso de GPUs para la ejecución de los notebooks, resultando tremendamente útil, ya que el uso de redes neuronales convolucionales requiere un gran coste computacional que delegado en las tarjetas gráficas reduce drásticamente los tiempos de ejecución.

Colab ofrece una gran variedad de opciones. Cuando se ejecuta, se crea una instancia en una máquina virtual que permite incluso la ejecución de comandos Linux, permitiendo así instalar paquetes externos e incluso clonar repositorios e instalarlos manualmente.

En resumen, es una plataforma para crear documentos de texto con la particularidad de poder añadir celdas de código Python que se ejecutan en el propio documento, imprimiendo su salida en este.

Ejemplo de notebook creado en Google Colab: https://colab.research.google.com/drive/1_CEV72mZPaV4PXqMA1NC2GjzToPdv7wI?usp=sharing

5
Experimentación

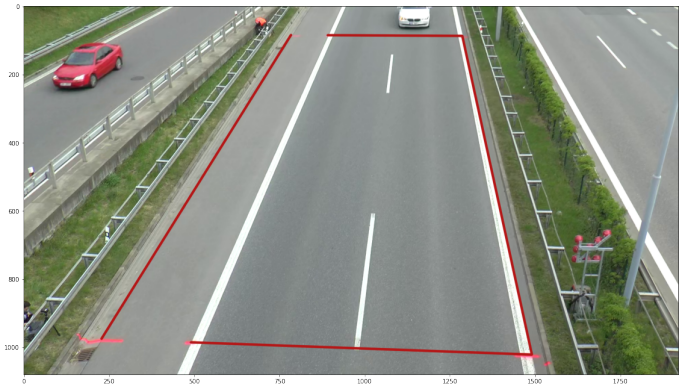
A continuación se presentan los resultados experimentales que se han llevado a cabo. La subsección 5.1 presenta el conjunto de datos de referencia considerado. Las subsecciones 5.2 y 5.3 tratan de los procedimientos de detección y seguimiento de vehículos, respectivamente. A continuación, se detallan los métodos de estimación de la velocidad del vehículo comparados (Subsección 5.4), y se especifica el procedimiento para transformar el sistema de coordenadas de la cámara al sistema de coordenadas del mundo real (Subsección 5.5). Las medidas de evaluación cuantitativa se describen en la subsección 5.6. Por último, los resultados obtenidos se muestran en la subsección 5.7. La figura 1 muestra el flujo de trabajo completo propuesto.

5.1. El dataset BrnoCompSpeed

Se ha utilizado el dataset *BrnoCompSpeed* [27] para probar nuestra propuesta. El conjunto de datos seleccionado consta de seis sesiones de una hora. Cada sesión se graba en una ubicación diferente y consta de tres vídeos desde tres ángulos diferentes capturados por los sistemas de videovigilancia de las autopistas: izquierda, centro y derecha con una vista desde arriba de la carretera. Estas sesiones están anotadas con las mediciones en la carretera sobre las líneas dibujadas como se ve en la Figura 6, también con las velocidades medidas de los vehículos obtenidas de *LIDAR* (Light Detection and Ranging o Laser Imaging Detection and Ranging). Este dispositivo obtiene una serie de puntos del vehículo tomados con un escáner láser. Los dispositivos LIDAR se utilizan a menudo en zonas de mucho tráfico, ya que el rayo láser puede enfocar fácilmente los vehículos individuales, lo que permite una precisión milimétrica [28]. De este modo, es posible medir con precisión la velocidad de un vehículo, incluso en condiciones de tráfico intenso.

Cabe señalar que estas sesiones presentan varios problemas. En primer lugar, los vehículos no se registran en el conjunto de datos en el orden en que aparecen en la escena. Cuando

Figura 6: Carretera de la sesión una con las marcas de las líneas



se realiza el seguimiento, éste se establece según el orden de aparición de los vehículos. Otro problema corresponde a las anotaciones. Dado que la estimación se realiza mediante dispositivos LIDAR, es posible que dos vehículos circulen en paralelo por ese tramo de carretera. Esto provoca que algunos vehículos no sean anotados en el conjunto de datos, mientras que con el método propuesto se registra su seguimiento. En estos casos, es imposible comparar nuestra estimación con la del LIDAR, ya que ésta no está disponible. Además, hay anotaciones erróneas, asignando erróneamente clases a ciertos elementos debido a errores del método de detección de objetos.

Dada la problemática expuesta, es difícil realizar una comparación automática de todos los vehículos que circulan por la carretera de forma sencilla y eficaz. Por lo tanto, es necesario asociar manualmente los vehículos obtenidos por el seguimiento y los del conjunto de datos. Por esta razón, las pruebas se han realizado con la primera sesión de vídeo para el punto de vista central, obteniendo de ella una secuencia de 10 minutos de duración. A pesar de utilizar sólo una pequeña parte de la sesión, las pruebas se realizan para estimar la velocidad de 140 vehículos.

5.2. Detección de vehículos

Para la detección de vehículos, se ha elegido Yolov5 [13, 25] como la red neuronal profunda de detección de objetos. Yolov5 es un método lo suficientemente rápido como para poder procesar imágenes en tiempo real. Además, no suele generar detecciones dobles. No obstante, cualquier método de detección de objetos con un rendimiento razonable podría utilizarse para

esta tarea.

5.3. Seguimiento de vehículos

Se ha utilizado un algoritmo basado en el seguimiento de un punto del vehículo, eligiendo entre su centro o una de las cuatro esquinas de la caja delimitadora dependiendo del ángulo de la cámara. La figura 7 muestra la trayectoria de los vehículos y sus cuatro puntos de las esquinas de la caja delimitadora.

Para cada vehículo detectado a lo largo del vídeo, se guarda su posición (píxel) y la hora (fotograma).

5.4. Métodos de estimación de velocidades de los vehículos

Para probar y comparar nuestra propuesta, se utilizan tres métodos diferentes para estimar la velocidad de los vehículos:

- Método 1: obtener la velocidad a partir de la primera y la última posición e instante del vehículo detectado para obtener la velocidad media.
- Método 2: calcular el incremento de posición y tiempo de cada dos detecciones consecutivas de vehículos para obtener el valor medio de todos los vectores de velocidad.
- Nuestro método propuesto en la Sección 3 usando la estimación de Nadaraya-Watson por kernel.

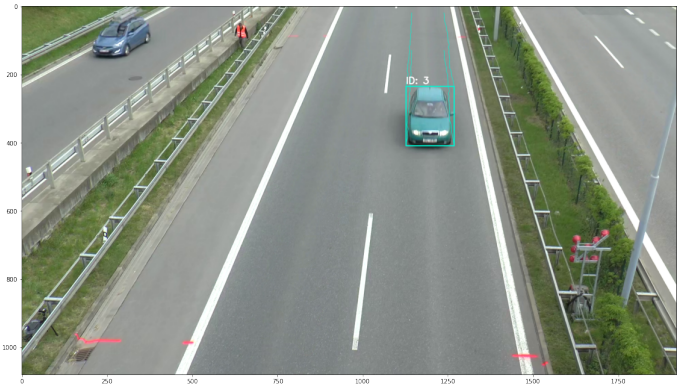


Figura 7: Seguimiento del vehículo

5.5. Transformación de la cámara al mundo real

Lo primero que hay que hacer es calcular la homografía. Se necesitan al menos cuatro puntos, obtenerlos a partir de una imagen satélite (conociendo también la distancia real entre ellos) y cotejar manualmente la imagen de satélite y la imagen de la sesión grabada podría ser una opción. En este caso, el conjunto de datos proporciona varios puntos marcados en la carretera, que podemos aproximar a un rectángulo y obtener directamente la homografía conociendo la distancia real entre ellos. Las figuras 6 y 8 muestran la equivalencia de la homografía.

El siguiente paso es aplicar la homografía a las posiciones de los vehículos y a las medidas tomadas en la carretera. De este modo, podemos conocer la distancia entre los píxeles de los ejes X e Y y su correspondencia con la distancia en el mundo real.

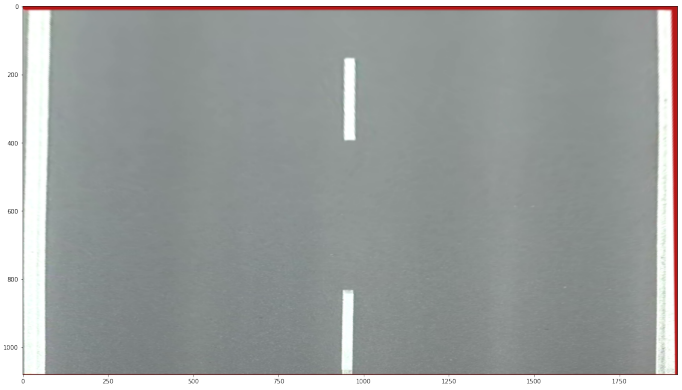
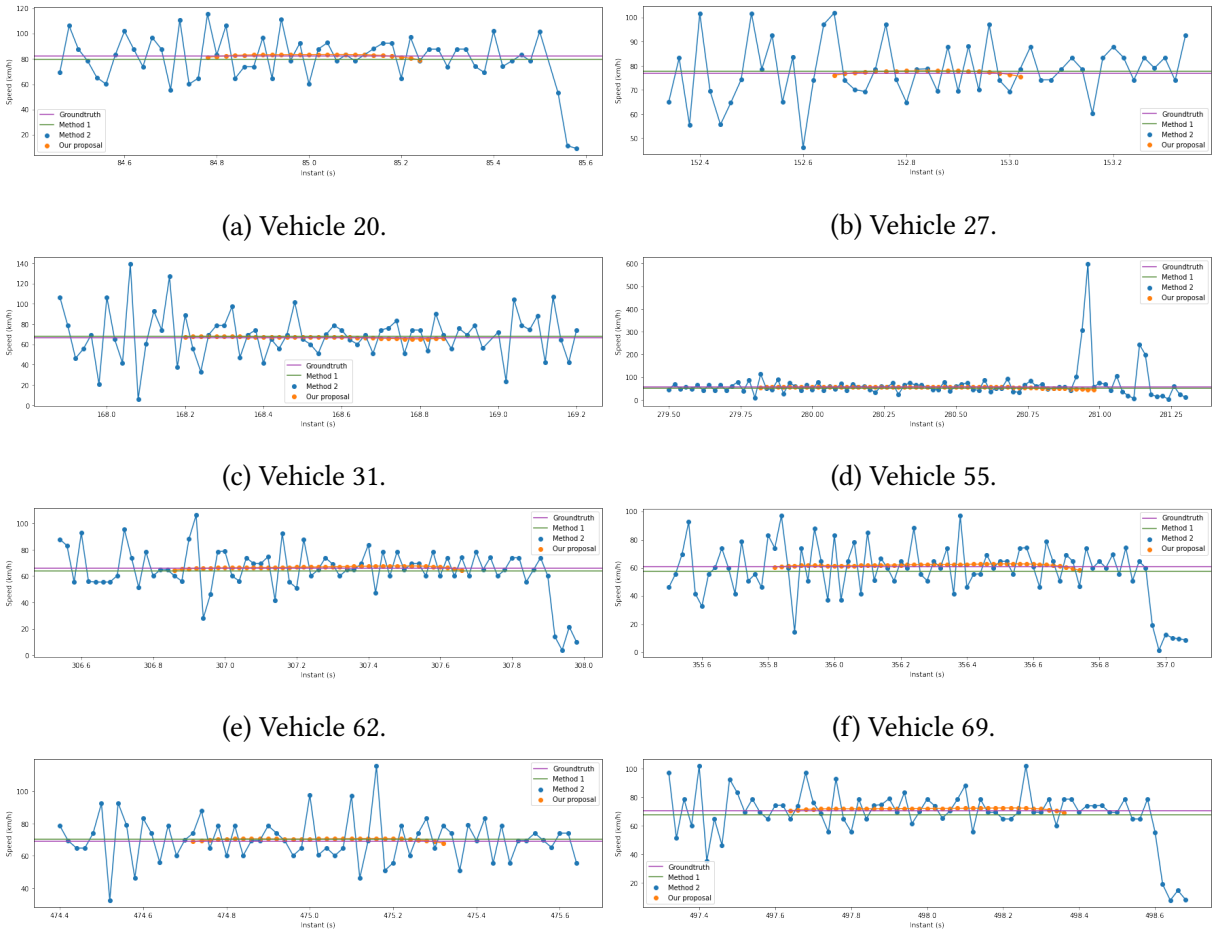


Figura 8: Road from session one with line marks and homography applied

Finalmente, para estimar la velocidad, se aplica la ecuación 3, y se obtienen los instantes de velocidad en píxeles/frame. Se realiza una conversión para obtener la velocidad en kilómetros por hora. Con el factor de conversión obtenido entre píxeles y metros, se realiza la conversión de distancia, y conociendo los fotogramas por segundo del vídeo, podemos convertir los fotogramas a horas.

5.6. Evaluación

Como estimador del error se ha considerado el *MSE*. El conjunto de datos proporciona la velocidad media del vehículo entre la línea inicial y la final, por lo que para calcular el error cuadrático medio se realiza el módulo de la media de todos los vectores de velocidad estimados. Y con estos datos se calcula el MSE.



(a) Vehicle 20.

(b) Vehicle 27.

(c) Vehicle 31.

(d) Vehicle 55.

(e) Vehicle 62.

(f) Vehicle 69.

(g) Vehicle 100.

(h) Vehicle 105.

Figura 9: Cada figura muestra la velocidad real junto a las estimaciones realizadas por el método 1, el método 2 y nuestra propuesta para ocho vehículos aleatorios.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(\widehat{v_i} - v_i)^2 \tag{13}$$

donde $\widehat{v_i}$ es la velocidad medida en el dataset y v_i la velocidad estimada.

Principares factores que afecta a la estimación de velocidad:

- Precisión de los puntos seleccionados para el cálculo de la matriz de homografía.
- Deformación de la caja delimitadora del vehículo al acercarse a la cámara.

5.7. Resultados

Se utilizó la primera sesión para llevar a cabo la fase de experimentación descrita en la subsección 5.1. La figura 9 de la página 31 muestra las estimaciones de velocidad obtenidas por todos los métodos. Para ello, se han seleccionado ocho vehículos aleatorios capturados por el modelo de detección de objetos. Además, para cada uno de ellos se ha representado la velocidad real de los vehículos capturados por el sistema emphLIDAR.

Como se muestra en este gráfico, obtenemos que la estimación de la velocidad obtenida por el método 1 se aproxima a la velocidad medida pero esta es constante, incapaz de adaptarse a los cambios de velocidad. Sin embargo, nuestra propuesta es mucho más precisa, determinando así velocidades más exactas de los vehículos que circulan por esa carretera. El método 2 no es aplicable dada la varianza entre las velocidades estimadas en diferentes instantes. Podemos determinar que este comportamiento se repite en los distintos vehículos detectados por el modelo. Al realizar la estimación de la velocidad con el kernel, nuestra propuesta elimina las estimaciones $2 * sqrt(h)$ tanto del principio como del final, para evitar la introducción de errores al realizar la derivada de la regresión por el kernel.

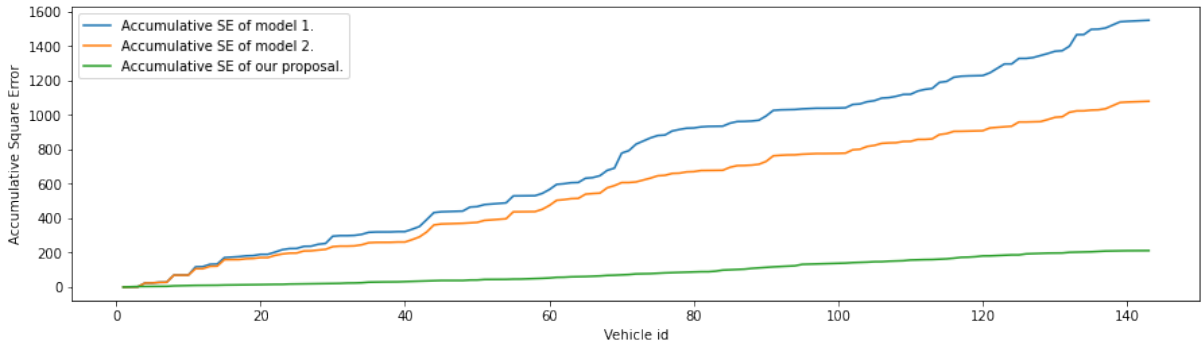


Figura 10: Error cuadrático acumulado a medida que se estima la velocidad de los vehículos.

Cuadro 1: Error Cuadrático Medio (cuanto más bajo, mejor) para la estimación de la velocidad de los vehículos detectados en la primera sesión. El mejor resultado está marcado en **negrita**.

Técnica	ECM (Error Cuadrático Medio)
Método 1	11.912
Método 2	8.296
Nuestra propuesta	**1.633**

Para la primera sesión, se calculó el error cuadrático medio de los 140 vehículos detectados para la perspectiva central. Estos valores se representan en la tabla 1. Como puede verse en la tabla, los métodos 1 y 2 no son adecuados debido al mayor error cuadrático medio. Nuestra propuesta es 7,295 veces más precisa que el modelo 1 y 5,08 veces más precisa que el modelo 2. La figura 10 de la página 32 muestra el error cuadrático acumulado de los métodos 1, 2 y nuestra propuesta. Como muestra la figura, nuestra propuesta es más precisa en cuanto a la estimación de la velocidad, obteniendo un error acumulado inferior a 200, en contraste con los métodos 1 y 2, que superan este umbral.

6

Conclusiones

Este trabajo propone una metodología que emplea redes neuronales convolucionales y un estimador de regresión por kernel para aproximar de forma suave y fiable la velocidad a la que circulan los vehículos por una carretera. En primer lugar, se realiza una transformación proyectiva que determina una correspondencia biyectiva entre los puntos del plano captado por la cámara y la propia carretera. A continuación, se utiliza un método de detección de objetos basado en redes neuronales convolucionales para identificar los vehículos y su posición, con el fin de capturar su trayectoria y almacenar la posición así como una marca de tiempo. Con esta información, se procede a aplicar el estimador de regresión kernel de Nadaraya-Watson utilizando kernels gaussianos, triangulares o cuadráticos, respectivamente, para suavizar los errores de medición.

Para probar la propuesta, se han realizado experimentos con la red de detección de objetos Yolov5 ya que permite el procesamiento de imágenes en tiempo real. Se han seleccionado una serie de vídeos del conjunto de datos BrnoCompSpeed, que contiene información sobre la velocidad de los vehículos que aparecen en la escena. Se ha utilizado la medida de rendimiento MSE (Mean Squared Error) como método de evaluación y se han incluido como referencia otros dos métodos para calcular la velocidad.

Nuestra propuesta se ve respaldada por los resultados mostrados en la subsección 5.7 de la página 32 ya que la velocidad estimada se acerca más a la proporcionada por el conjunto de datos que otros métodos.

Es importante destacar que esta propuesta es aplicable a cualquier método de detección de objetos que permita la detección en tiempo real, por lo que consideramos que es una estrategia recomendable a la hora de abordar problemas en tiempo real utilizando secuencias capturadas por cámaras de videovigilancia. También es necesario destacar que, aunque en nuestros experimentos se ha creado la matriz de homografía utilizando puntos de referencia del conjunto de

datos, los puntos clave para realizar la equivalencia podrían extraerse de imágenes de satélite siempre que se disponga de una cámara instalada en un lugar conocido.

Como trabajo futuro, nuestra propuesta podría aplicarse a zonas urbanas en las que interactúan vehículos y personas, los cambios de velocidad son habituales y los accidentes son más probables.

Referencias

[1] Polamuri Subba Rao, K. Srinivas, and A. Krishna Mohan. A survey on stock market prediction using machine learning techniques. In Amit Kumar, Marcin Paprzycki, and Vinit Kumar Gunjan, editors, *ICDSMLA 2019*, pages 923–931, Singapore, 2020. Springer Singapore.

[2] Soufien Jaffali, Salma Jamoussi, Nesrine Khelifi, and Abdelmajid Ben Hamadou. Survey on social networks data analysis. In Siddharth Swarup Rautaray, Gerald Eichler, Christian Erfurth, and Günter Fahrnberger, editors, *Innovations for Community Services*, pages 100–119, Cham, 2020. Springer International Publishing.

[3] Rimpal R. Popat and Jayesh Chaudhary. A survey on credit card fraud detection using machine learning. In *2018 2nd International Conference on Trends in Electronics and Informatics (ICOEI)*, pages 1120–1125, 2018.

[4] Miguel A Molina-Cabello, Rafael Marcos Luque-Baena, Ezequiel López-Rubio, Lipika Deka, and Karl Thurnhofer-Hemsi. Road pollution estimation using static cameras and neural networks. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7. IEEE, 2018.

[5] Miguel A Molina-Cabello, Benjamin N Passow, Enrique Dominguez, David Elizondo, and Jolanta Obszynska. Infering air quality from traffic data using transferable neural network models. In *International Work-Conference on Artificial Neural Networks*, pages 832–843. Springer, 2019.

[6] V. C. Maha Vishnu, M. Rajalakshmi, and R. Nedunchezhian. Intelligent traffic video surveillance and accident detection system with dynamic traffic signal control. *Cluster Computing*, 21:135–147, 2018.

[7] F Chen, C. Wang, Y. Dai, W. Zhou, and Y. Geng. A vision-based video crash detection framework for mixed traffic flow environment considering low-visibility condition. *Journal of Advanced Transportation*, 2020, 2020.

[8] Thierry Bouwmans. Traditional and recent approaches in background modeling for foreground detection: An overview. *Computer Science Review*, 11-12:31 – 66, 2014.

[9] Z. Zivkovic and F. Van Der Heijden. Efficient adaptive density estimation per image pixel for the task of background subtraction. *Pattern Recognition Letters*, 27(7):773–780, 2006.

[10] R.M. Luque, E. Domínguez, E.J. Palomo, and J. Muñoz. An art-type network approach for video object detection. pages 423–428, 2010.

[11] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks, 2016.

[12] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. Ssd: Single shot multibox detector. *Lecture Notes in Computer Science*, page 21–37, 2016.

[13] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection, 2016.

[14] Huansheng Song, Haoxiang Liang, Huaiyu Li, Zhe Dai, and Xu Yun. Vision-based vehicle detection and counting system using deep learning in highway scenes. *European Transport Research Review*, 11, 12 2019.

[15] Daniel Lorenčík and Iveta Zolotová. Object recognition in traffic monitoring systems. In *2018 World Symposium on Digital Intelligence for Systems and Machines (DISA)*, pages 277–282, 2018.

[16] Iván García, Rafael Marcos Luque, and Ezequiel López. Improved detection of small objects in road network sequences, 2021.

[17] Kantip Kiratiratanapruk and Supakorn Siddhichai. Vehicle detection and tracking for traffic monitoring system. In *TENCON 2006 - 2006 IEEE Region 10 Conference*, pages 1–4, 2006.

[18] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning*. Springer, New York, 2nd edition, 2013.

[19] F. Ferraty and P. Vieu. Nonparametric models for functional data, with application in regression, time-series prediction and curve discrimination. *Journal of Nonparametric Statistics*, 16(1-2):111–125, 2004.

[20] V. Katkovnik, A. Foi, K. Egiazarian, and J. Astola. From local kernel to nonlocal multiple-model image denoising. *International Journal of Computer Vision*, 86(1):1–32, 2010.

[21] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[22] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[23] Kaiwen Duan, Song Bai, Lingxi Xie, Honggang Qi, Qingming Huang, and Qi Tian. Centernet: Keypoint triplets for object detection, 2019.

[24] Mingxing Tan, Ruoming Pang, and Quoc V. Le. Efficientdet: Scalable and efficient object detection. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10778–10787, 2020.

[25] Glenn Jocher, Alex Stoken, Jirka Borovec, NanoCode012, Ayush Chaurasia, TaoXie, Liu Changyu, Abhiram V, Laughing, Tkianai, YxNONG, Adam Hogan, Lorenzomammana,

AlexWang1900, Jan Hajek, Laurentiu Diaconu, , Marc, Yonghye Kwon, , Oleg, Wanghaoyang0106, Yann Defretin, Aditya Lohia, Ml5ah, Ben Milanko, Benjamin Fineran, Daniel Khromov, Ding Yiwei, , Doug, Durgesh, and Francisco Ingham. ultralytics/yolov5: v5.0 - yolov5-p6 1280 models, aws, supervise.ly and youtube integrations, 2021.

[26] G. van Rossum. Python tutorial. Technical Report CS-R9526, Centrum voor Wiskunde en Informatica (CWI), Amsterdam, May 1995.

[27] Jakub Sochor, Roman Juránek, Jakub Špaňhel, Lukáš Maršík, Adam Široký, Adam Herout, and Pavel Zemčík. Comprehensive data set for automatic single camera visual speed measurement. *IEEE Transactions on Intelligent Transportation Systems*, 20(5):1633–1643, 2019.

[28] Jiaxing Zhang, Wen Xiao, Benjamin Coifman, and Jon Mills. Vehicle tracking and speed estimation from roadside lidar. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 09 2020.

UNIVERSIDAD DE MÁLAGA | uma.es

E.T.S. DE INGENIERÍA

E.T.S de Ingeniería Informática
Bulevar Louis Pasteur, 35
Campus de Teatinos
29071 Málaga