GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

TRABAJO FIN DE GRADO

Implementación de visión artificial para la detección y asignación eficiente de espacios de aparcamiento

Autor: Artur Alsina Piró

Director: Federico Muñoz Babiano

Madrid

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

Implementación de visión artificial para la detección y asignación

eficiente de espacios de aparcamiento

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el

curso académico 2022/23 es de mi autoría, original e inédito y

no ha sido presentado con anterioridad a otros efectos.

El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido

tomada de otros documentos está debidamente referenciada.

Fdo.: Artur Alsina Piró Fecha: 21/ 07/2023

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Fdo.: Federico Muñoz Babiano Fecha: 21/ 07/ 2023

Agradecimientos

Deseo expresar mi sincero agradecimiento a todas las personas que han contribuido a la realización de este Trabajo de Fin de Grado.

De manera especial, quisiera agradecer a mi director, Federico Muñoz Babiano, por su invaluable orientación y apoyo constante a lo largo de este proyecto.

También quisiera expresar mi gratitud hacia mi familia por su apoyo incondicional durante toda mi carrera académica.

Finalmente, gracias a todos los profesores y compañeros que, de una manera u otra, han contribuido a mi formación y a este trabajo.

IMPLEMENTACIÓN DE VISIÓN ARTIFICIAL PARA LA DETECCIÓN Y ASIGNACIÓN EFICIENTE DE ESPACIOS DE APARCAMIENTO

Autor: Alsina Piró, Artur.
Director: Muñoz Fabiano, Federico.
Entidad Colaboradora: ICAI – Universidad Pontificia Comillas

RESUMEN DEL PROYECTO

Este proyecto consiste en la implementación de un programa para un aparcamiento inteligente que a través de modelos de visión artificial identifique los vehículos en una entrada de video, detecte las plazas ocupadas por vehículos estacionados y encuentre la ruta óptima para los vehículos en movimiento hacia el nodo con plazas libres más cercano.

Palabras clave: Vision Artificial, Machine Learning, SmartParking

1. Introducción

El creciente número de vehículos en las ciudades presenta, entre otros, un problema al disponer de un espacio limitado dedicado a aparcamientos. Para aumentar su eficiencia ha surgido el concepto de los aparcamientos inteligentes, que tradicionalmente implementan sensores en las plazas para detectar su ocupación y presentar al usuario con esta información. En este trabajo se explorará la posibilidad de utilizar cámaras en lugar de sensores convencionales, que ya suelen estar presentes en la mayoría de los aparcamientos por tema de seguridad, ahorrando costes, facilitando la instalación y el mantenimiento y proporcionando al sistema con más información sobre el estado del aparcamiento para poder guiar al usuario de manera más eficiente.

2. Definición del proyecto

Para guiar usuario por el aparcamiento el sistema requiere traes componentes fundamentales: sensores, procesadores y actuadores. En este caso, los sensores son las cámaras, el procesador el programa que parte de la entrade de video y obtiene el camino óptimo y los actuadores podían ser luces en cada intersección o nodo que indicasen al usuario en qué dirección seguir. El siguiente proyecto es una prueba de concepto de dicho sistema. Por tanto, se centrará en el procesamiento de información, dejando aparte la elección de las cámaras y la interacción con el usuario para una futura implementación.

Para detectar las plazas libres en cada imagen se ha decidido detectar en su lugar a los vehículos que las ocupan por regiones y restar el número de plazas disponibles en cada región el número de vehículos detectados. Estas regiones representarán los nodos del sistema, a través de los que los algoritmos de guiado encontrarán la ruta óptima para un vehículo en movimiento.

Es importante notar que el guiado de los vehículos es constante durante su ruta a la plaza de aparcamiento. Esto es importante ya que no se pueden prever los posibles cambios en el sistema que encontrará el usuario, como que la plaza asignada sea demasiado pequeña, que se libere otra plaza de aparcamiento más cercana o que otro usuario se haya desviado

y haya ocupado la plaza que originalmente se le había asignado. En estos casos, el programa es capaz de recalcular la ruta hacia el nuevo nodo.

3. Descripción del modelo/sistema/herramienta

La ruta que sigue la información a través de las distintas partes del programa hasta obtener el camino óptimo que debe seguir cada coche, empieza en la detección de vehículos a partir de una entrada de video constante de cada cámara. Cada vez que se captura un fotograma se pasa por YOLO y se obtienen las detecciones de la imagen. Estas detecciones se filtran según sus clases para asegurar que solo se detecten vehículos. Posteriormente, se aplica una transformación homográfica a las posiciones de los puntos que delimitan la posición de los vehículos para obtener su posición en coordenadas reales. La lista procesada de detecciones se pasa al algoritmo de rastreo, que las asigna a rastreos ya detectados o a nuevos rastreos tentativos. Con la asignación de cada id de rastreo a los ids de las detecciones de YOLO se accede a un diccionario que contiene los datos de cada rastreo y se actualiza su velocidad y otros valores útiles para futuros cálculos. Con la lista de velocidades de cada vehículo se decide cuáles son los vehículos en movimiento y se separan de los vehículos aparcados. La lista de las posiciones de los vehículos aparcados se pasa por un detector de plazas libres, que compara el número de vehículos en cada región con el número de plazas en esta. Por otro lado, la posición de los vehículos en movimiento se pasa al buscador de ruta, que a través de las posiciones de plazas libres y de la posición actual del coche encuentra la mejor plaza.

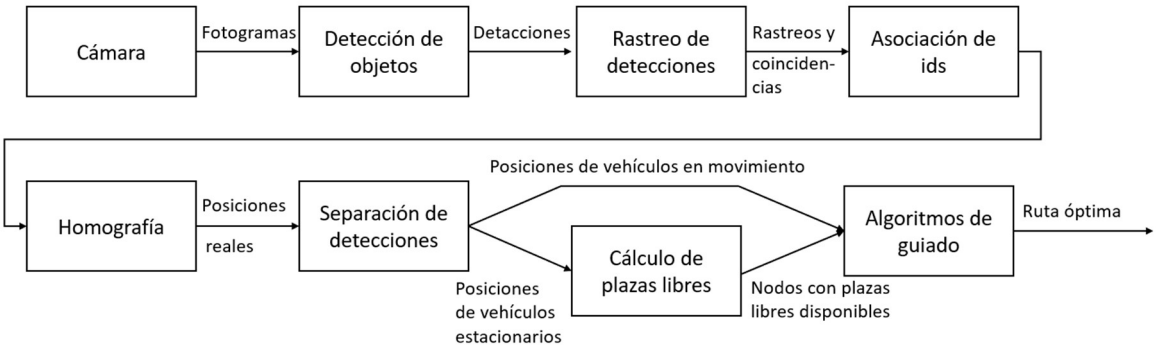

Figura 1: Pipeline del sistema

4. Resultados

La primera parte del proyecto ha consistido en desarrollar un modelo capaz de detectar los vehículos en cada fotograma de una entrada de vídeo, asignarles un id único y rastrearlos a través de los fotogramas. Con cada vehículo identificado, se separa a los vehículos en movimiento de los vehículos estacionarios y en estos se aplica análisis de componentes principales y clustering a los centros de sus detecciones para identificar a que fila pertenecen.

Con los vehículos separados en dinámicos y estacionados, a su vez separados por filas, se cuenta los coches en una fila pertenecientes a una región de la pantalla, que se representa por un nodo en el diagrama de bloques, para determinar si hay plazas libres en ese nodo.

Para visualizar las distintas categorías en las que se han clasificado los distintos vehículos se ha usado los siguientes colores:

-Para identificar los vehículos en las dos filas se han utilizado el verde y el cian, en este caso el verde corresponde a la fila en la que se va a contar las plazas libres

-Para identificar el vehículo en movimiento se ha utilizado el azul

-Par identificar el cuadro delimitador donde se van a contar las plazas se ha utilizado en blanco, al igual que para anotar el centro de las detecciones de la fila que pertenece al nodo analizado que se encuentran dentro del cuadro delimitador. También se anuncia en pantalla en número de plazas libres con ese mismo color.

-Por último, todos los rastreos están acompañadas de su posición en coordenadas reales relativas a un origen arbitrario marcado con un punto negro. Los nuevos ejes tomados son paralelos y perpendiculares a la vía.

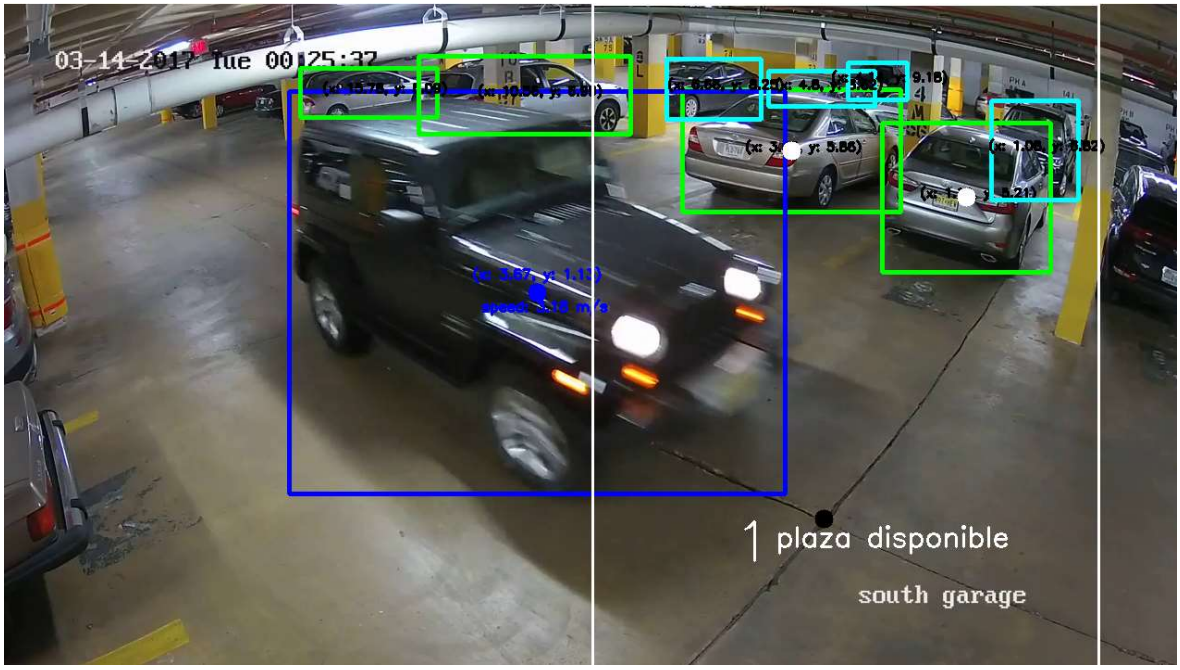

Ilustración 1: Fotograma de la entrada de video anotado

5. Conclusiones

Durante el desarrollo de este proyecto se ha demostrado los modelos de visión artificial, en concreto YOLOv8, puede proporcionar una solución eficiente a un problema tradicionalmente tratado con sensores de posición. El desarrollo de este proyecto ha sido satisfactorio ya que se han logrado los objetivos propuestos.

El siguiente paso a seguir para ampliar este proyecto sería implementarlo en un sistema real con múltiples cámaras y, en caso de que sus campos de visión se superpusieran, adaptar el algoritmo de rastreo para que realizase asociación profunda en las detecciones de múltiples entradas de video para identificar con el mismo id a detecciones en distintas cámaras de un mismo vehículo.

COMPUTER VISION APPLIED TO A SMARTPARKING SYSTEM TO EFFICIENTLY GUIDE USERS TO EMPTY PARKING SPACES.

Autor: Alsina Piró, Artur.
Director: Muñoz Fabiano, Federico.
Entidad Colaboradora: ICAI – Universidad Pontificia Comillas

ABSTRACT

This project involves the implementation of a program for a smart parking system that, through computer vision models, identifies vehicles in a parking lot, detects spaces occupied by parked cars, and finds the optimal route for moving vehicles towards the nearest node with available spaces.

Keywords: Computer Vision, Machine Learning, SmartParking

1. Introduction

The growing number of vehicles in cities presents, among other issues, a problem with the limited space dedicated to parking. To increase efficiency, the concept of smart parking has emerged, which traditionally implements sensors in spaces to detect their occupancy and provide this information to the user. This work will explore the possibility of using cameras instead of conventional sensors, which are usually already present in most parking lots for security purposes, thus saving costs, facilitating installation and maintenance, and providing the system with more information about the parking lot's status to guide the user more efficiently.

2. Project Definition

To guide the user through the parking lot, the system requires three fundamental components: sensors, logic, and actuation. In this case, the sensors are the cameras, the logic is the program that takes video input and obtains the optimal path, and the actuators could be lights at each intersection or node that indicate the direction to follow to the user. The following project is a proof of concept of such a system. Therefore, it will focus on information processing, leaving aside the choice of cameras and user interaction for future implementation. To detect available spaces in each image, it has been decided to detect the vehicles occupying them by regions and subtract the number of detected vehicles from the number of available spaces in each region. These regions will represent the system's nodes, through which the guidance algorithms will find the optimal route for a moving vehicle.

It is important to note that vehicle guidance is constant during its route to the parking space. This is crucial since it is impossible to predict the possible changes in the system that the user will encounter, such as the assigned space being too small, another closer parking space becoming available, or another user deviating from their path and occupying the space that was originally assigned to them. In these cases, the program is capable of recalculating the route towards a new node.

3. Model/System/Tool Description

The path that signal follows through the different parts of the program until the optimal path is each car should follow is returned begins with vehicle detection from a constant video input from each camera. Each time a frame is captured, it is passed through YOLO and the image detections are obtained. These detections are filtered by their classes to ensure that only vehicles are detected. Subsequently, a homographic transformation is applied to the points' that define bounding boxes to obtain their position in real coordinates. The processed list of detections is passed to a tracking algorithm, which assigns them to already detected traces or new tentative traces. By assigning each trace id to YOLO's detection ids, a dictionary ca be accessed that contains each trace's data and updates its speed and other useful values for future calculations. With the list of speeds of each vehicle, those in motion are separated from the parked vehicles. The list of the parked vehicles' positions is passed through a free space detector, which compares the number of vehicles in each region with the number of spaces in it. On the other hand, the position of the moving vehicles is passed to the route finder, which through the positions of free spaces and the car's current position finds the best route.

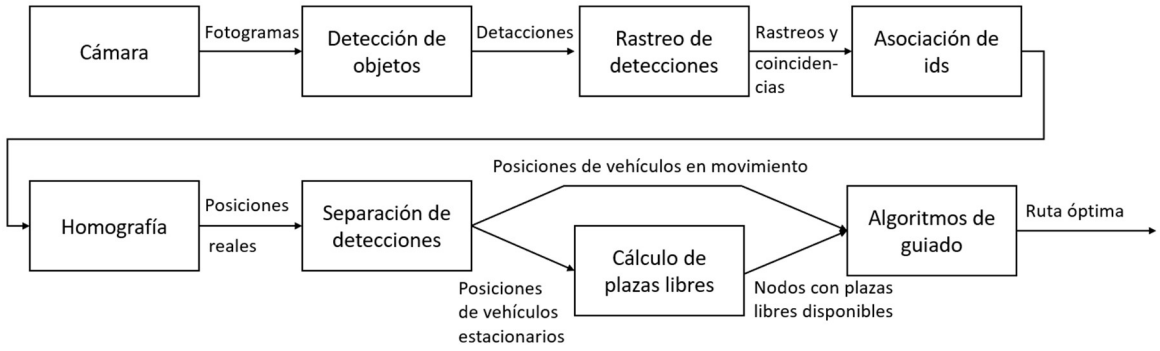

Figura 2: System outline

4. Results

The first part of the project consisted of developing a model capable of detecting the vehicles in each frame of a video input, assigning them a unique id, and tracking them through the frames. Moving vehicles are then separated from stationary vehicles, and principal component analysis and clustering are applied to the centers of their detections to identify which row they belong to. With the vehicles separated into moving and parked, further separated by rows, the cars in a row belonging to a region of the screen, represented by a node in the block diagram, are counted to determine if there are free spaces in that node. To visualize the different categories in which the various vehicles have been classified, the following colors have been used:

- Green and cyan have been used to identify vehicles in the two rows, in this case, green corresponds to the row in which the free spaces will be counted.
- Blue has been used to identify moving vehicles.
- White has been used to identify the bounding box where the spaces will be counted, as well as to note the center of the detections of the row that belongs to the analyzed

node that are within the bounding box. The number of free spaces is also announced on the screen in this same color.

- Finally, all tracks are accompanied by their position in real coordinates relative to an arbitrary origin marked with a black dot. The new axes taken are parallel and perpendicular to the road."

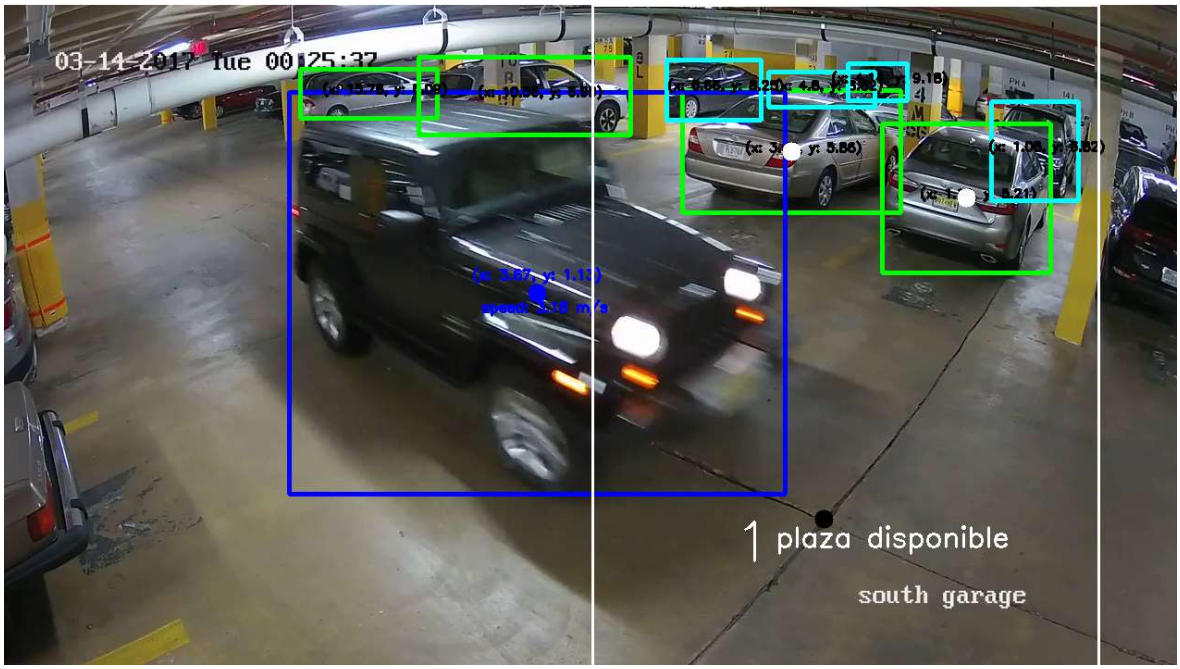

Ilustración 2: Anotated frame

5. Conclusion

Throughout the development of this project, it has been demonstrated that artificial vision models, specifically YOLOv8, can provide an efficient solution to a problem traditionally addressed with position sensors. The progression of this project has been successful as the set objectives have been duly accomplished.

The subsequent step to expand this project would be its implementation of a real-world system with multiple cameras. In instances where their fields of vision overlap, the tracking algorithm would need to be adapted to perform deep association on the detections from multiple video inputs, enabling the identification of different detections of the same vehicle with the same ID.

Índice de la memoria

Índice de figuras

Índice de Tablas

Índice de Ilustraciones

Capítulo 1. INTRODUCCIÓN

El creciente número de vehículos en las ciudades presenta, entre otros, un problema al disponer de un espacio limitado dedicado a aparcamientos. Para aumentar su eficiencia ha surgido el concepto de los aparcamientos inteligentes, que tradicionalmente implementan sensores en las plazas para detectar su ocupación y presentar al usuario con esta información. Además, conocer el nivel de ocupación del aparcamiento en tiempo real abre muchas posibilidades para mejorar la gestión del aparcamiento, como permitir la reserva de plazas a los usuarios o poder establecer los pecios de manera dinámica

En este trabajo se explorará la posibilidad de utilizar cámaras en lugar de sensores convencionales, que ya suelen estar presentes en la mayoría de los aparcamientos por tema de seguridad, ahorrando costes, facilitando la instalación y el mantenimiento y proporcionando al sistema con más información sobre el estado del aparcamiento para poder guiar al usuario de manera más eficiente.

Para guiar usuario por el aparcamiento el sistema requiere traes componentes fundamentales: sensores, procesadores y actuadores. En este caso, los sensores son las cámaras, el procesador el programa que parte de la entrade de video y obtiene el camino óptimo y los actuadores podían ser luces en cada intersección o nodo que indicasen al usuario en qué dirección seguir. El siguiente proyecto es una prueba de concepto de dicho sistema. Por tanto, se centrará en el procesamiento de información, dejando aparte la elección de las cámaras y la interacción con el usuario para una futura implementación.

Para detectar las plazas libres en cada imagen se ha decidido detectar en su lugar a los vehículos que las ocupan por regiones y restar el número de plazas disponibles en cada región el número de vehículos detectados. Estas regiones representarán los nodos del sistema, a través de los que los algoritmos de guiado encontrarán la ruta óptima para un vehículo en movimiento.

Es importante notar que el guiado de los vehículos es constante durante su ruta a la plaza de aparcamiento. Esto es importante ya que no se pueden prever los posibles cambios en el sistema que encontrará el usuario, como que la plaza asignada sea demasiado pequeña, que se libere otra plaza de aparcamiento más cercana o que otro usuario se haya desviado y haya ocupado la plaza que originalmente se le había asignado. En estos casos, el programa es capaz de recalcular la ruta hacia el nuevo nodo.

Capítulo 2. DESCRIPCIÓN DE LAS TECNOLOGÍAS

2.1 YOLOv8

Introducción

En el ámbito de la visión artificial, el algoritmo YOLO proporciona un enfoque innovador para la detección de objetos en imágenes. Al presentar la detección de objetos como un problema de regresión, YOLO supera a sus contrapartes tanto en términos de velocidad como de precisión. Utiliza una única red neuronal de extremo a extremo entrenada para predecir directamente las cajas delimitadoras y las probabilidades de las clases de imágenes completas en una sola pasada.

El aspecto más distintivo del algoritmo YOLO es su perspectiva única sobre la detección de objetos. En lugar de afrontar la detección como un problema de dos partes (primero proponiendo cajas delimitadoras candidatas en una imagen, y luego ejecutando un clasificador en estas cajas propuestas), YOLO trata la detección de objetos como un simple problema de regresión. Lo hace dividiendo la imagen de entrada en una cuadrícula de SxS y luego prediciendo directamente las cajas delimitadoras y las probabilidades de las clases para cada caja.

Cada celda en la cuadrícula es responsable de predecir B cajas delimitadoras. La predicción de la caja delimitadora incluye cinco componentes: (x, y, w, h) y una puntuación de confianza. Las coordenadas (x, y) representan el centro de la caja, en relación con los límites de la celda de la cuadrícula. El ancho y la altura (w, h) se predicen en relación con toda la imagen. La puntuación de confianza representa la estimación de probabilidad de la red de que la caja delimitadora encierre un objeto y la precisión de la caja predicha.

Además, cada celda de la cuadrícula predice C probabilidades condicionales de clase,

$$P(Clase_i|Objeto)$$

para i=1 a C. Esto representa la probabilidad de que el objeto detectado pertenezca a una clase particular. La probabilidad de clase condicional solo se calcula para una caja por celda, ignorando las predicciones adicionales de la caja delimitadora. La puntuación de confianza para cada caja delimitadora se calcula como:

$$P(Objeto) * IOU_{verdad\ pred.}$$

Donde $P(Objeto)$ representa la probabilidad de que la celda contenga un objeto e $IOU_{verdad\ pred.}$ la puntuación de intersección sobre unión entre la caja delimitadora predicha y la verdad del terreno. Si no existe un objeto en esa celda, las puntuaciones de confianza deberían ser cero. De lo contrario, deben ser iguales a la intersección sobre la unión (IOU) entre la caja predicha y la verdad del terreno.

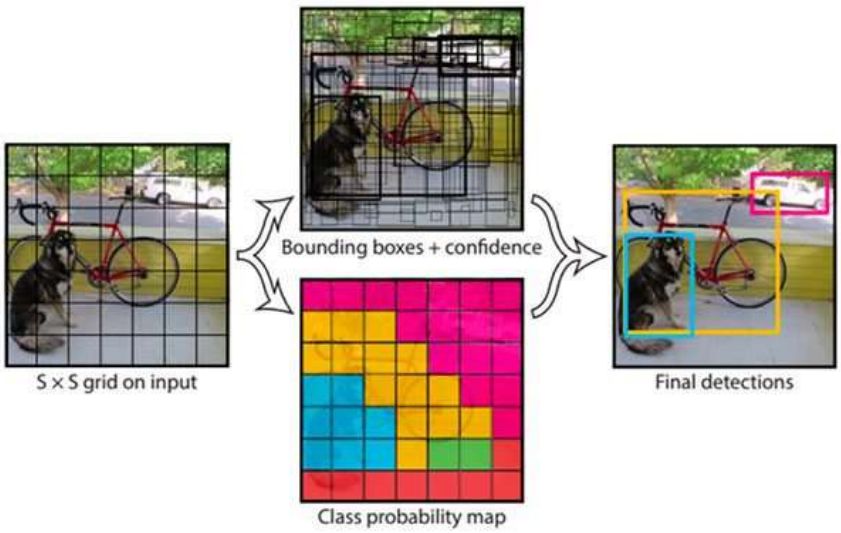

Ilustración 3: Visión simplificada del pipeline de YOLO, J. Redmon, S. Divvala, R. Girshick, and A. Farhadi [11]

Situación actual de YOLO

La versión más reciente de YOLO y la utilizada para este proyecto es YOLOv8.

Esta nueva versión ofrece además modelos de segmentación, que detectan el contorno de los objetos en pantalla, y de clasificación de imagen, que trata a toda la imagen como una detección y la clasifica. Sin embargo, para este proyecto se utilizará los modelos de detección de objetos, que devuelven los objetos en una imagen asociándolos con su id de clase y su posición en la pantalla.

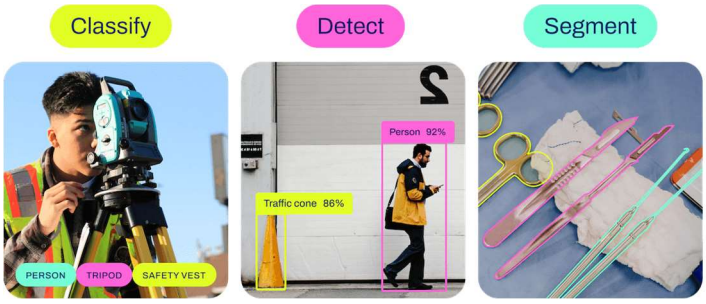

Ilustración 4: Resultados obtenidos con modelos de clasificación, detección y segmentación, Ultralytics [10]

Ultralytics también ofrece versiones de sus modelos preentrenadas con el dataset COCO (Common Objects in Context) [referencia], que distingue 80 categorías de objetos, incluyendo coches y furgonetas, las clases relevantes para este proyecto. Este dataset es uno de los más usados por su calidad en las anotaciones, la variación en los contextos y por su gran tamaño.

2.2 COCO DATASET

En el campo en constante evolución de la visión artificial, los datasets juegan un papel fundamental en la calidad y eficacia de un algoritmo. Un conjunto de datos grande, diverso y adecuadamente anotado puede mejorar drásticamente la capacidad de un algoritmo para identificar y clasificar objetos. Un conjunto de datos clave en este sentido es el conjunto de datos Common Objects in Context (COCO).

Desarrollado y mantenido por Microsoft, el conjunto de datos COCO es un conjunto de datos a gran escala para la detección de objetos y segmentación. Contiene más de 330.000 imágenes, con más de 200.000 de estas imágenes etiquetadas para fines de entrenamiento.

Estas imágenes cubren 80 categorías distintas de objetos, que incluyen humanos, animales, vehículos y objetos cotidianos. Las anotaciones en COCO no se limitan a simples cuadros delimitadores para la detección de objetos, sino que también incluyen segmentaciones poligonales complejas y puntos clave, proporcionando un recurso enriquecido y de alta calidad para diversas tareas de visión artificial.

El conjunto de datos COCO se distingue de otros conjuntos de datos debido a su énfasis en el 'contexto del objeto'. Los objetos anotados se capturan en escenas cotidianas, lo que permite a los modelos entrenados en COCO comprender el contexto del objeto además de la apariencia del objeto.

2.3 DEEPSORT

Deep SORT (Simple Online and Realtime Tracking), se ha convertido en una herramienta crítica dentro del dominio del seguimiento de objetos, avanzando aún más las capacidades del algoritmo SORT original. Basándose en el concepto de seguimiento por detección, la sofisticación de Deep SORT surge al incorporar tanto la información de movimiento como de apariencia para una precisa asociación y seguimiento de objetos, logrando así una mayor precisión y robustez en entornos complejos.

Deep SORT introduce dos componentes centrales: la predicción de movimiento mediante el filtro de Kalman y una métrica de asociación profunda para asociar eficazmente las detecciones. El filtro de Kalman ayuda en la predicción del movimiento, mientras que la métrica de asociación profunda, mediante una red neuronal, extrae las características principales del objeto detectado a través de la sección de la imagen contenida en su cuadro delimitador para proporcionar una solución robusta para el emparejamiento por apariencia.

Concepto de Rastreos y Coincidencias (*tracks and matches)*

En el contexto de Deep SORT, un 'rastreo' se refiere a la trayectoria que sigue un objeto a través de múltiples fotogramas en una secuencia de vídeo. Cada rastreo está asociado a un objeto, y la tarea es asegurar que el mismo rastreo está consistentemente vinculado al mismo

objeto a través de diferentes fotogramas, incluso ante la oclusión o salida temporal del objeto en la entrada de video.

Por otro lado, una 'coincidencia' se refiere a la asociación exitosa de una detección en un nuevo fotograma con un rastreo existente.

Predicción de movimiento utilizando el filtro de Kalman

El filtro de Kalman estima el estado de un objeto, incluyendo su posición, velocidad y otras características, en un fotograma en particular. Sirve como herramienta predictiva, utilizando el estado de los fotogramas anteriores para estimar el estado actual antes de que llegue la medición (detección) real. Proporciona un equilibrio óptimo entre la predicción y la medición.

Métrica de asociación profunda para el emparejamiento

Esta métrica utiliza un modelo de aprendizaje profundo entrenado para generar vectores de características que representan la apariencia de los objetos detectados. Estos vectores de características se utilizan en conjunción con la información de movimiento predicha por el filtro de Kalman para calcular el coste de asignar una detección a un rastreo. El coste tiene en cuenta tanto el error de predicción de movimiento como la incompatibilidad de apariencia.

Asociación de rastreos utilizando el algoritmo húngaro

Los algoritmos para asignar una detección a un rastreo existente se realizan en forma de matriz, calculando la coincidencia de cada detección con cada rastreo existente utilizando la métrica de asociación profunda. El método húngaro es un algoritmo de optimización combinatoria que resuelve el problema de asignación en tiempo polinómico. El algoritmo proporciona la mejor asignación de detecciones a rastreos existentes maximizando la suma

Ilustración 5: Pipeline simplificado de DeepSort

de las puntuaciones de las coincidencias entre detenciones y rastreos.

Gestión de rastreos en Deep SORT

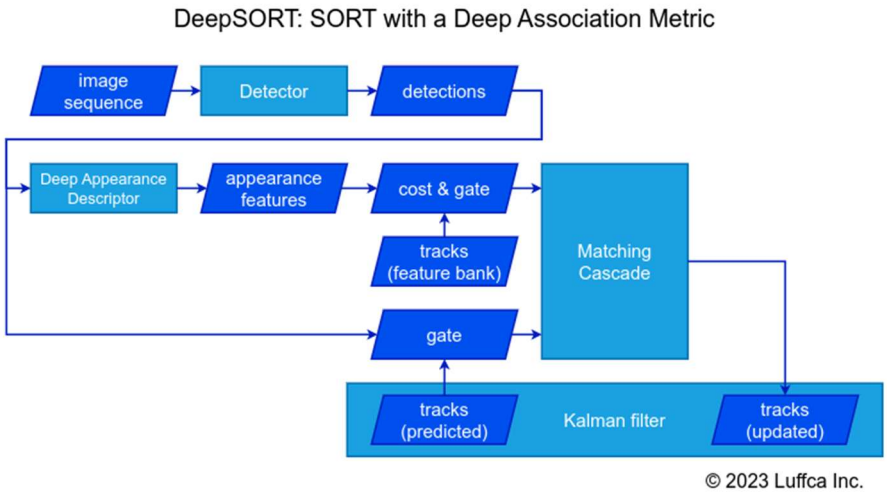

Figura 3: Pipeline de DeepSort, Luffca[1]

La gestión de rastreos es un aspecto crucial de Deep SORT e implica la iniciación, actualización y eliminación de rastreos.

Se inicia un nuevo rastreo cuando una detección no puede ser emparejada con ningún rastreo existente. Esto suele ocurrir cuando un nuevo objeto entra en el fotograma. El estado inicial del nuevo rastreo se basa en la detección, y se calcula una representación de apariencia inicial utilizando la métrica de asociación profunda. Cabe destacar que hay un numero de fotogramas mínimo definido por el usuario entre que se detecta un nuevo objeto y este pasa a formar parte de los rastreos para evitar crear nuevos rastreos con detecciones esporádicas.

Cuando se encuentra una coincidencia entre una detección y un rastreo, se actualiza el estado del rastreo utilizando la detección. El filtro de Kalman juega un papel crucial en esta actualización proporcionando una estimación óptima basada tanto en la predicción como en la medición. Además, se actualiza la representación de la apariencia del rastreo utilizando el vector de características de la detección coincidente.

Un rastreo se elimina cuando, de manera análoga a la inicialización de rastreos, no se empareja con ninguna detección durante un número predefinido de fotogramas consecutivos.

Esto puede suceder cuando un objeto abandona el fotograma de forma permanente o queda ocluido durante un largo periodo de tiempo.

2.4 LIBRERÍAS DESTACADAS

2.4.1 TENSORFLOW CON SOPORTE PARA CUDA

PyTorch

PyTorch, una biblioteca de aprendizaje automático de código abierto desarrollada por el laboratorio de investigación de IA de Facebook proporciona una interfaz de alto nivel para diseñar y entrenar modelos de aprendizaje profundo. Destaca en su capacidad para proporcionar un control detallado y su flexibilidad, lo que permite un rápido prototipado y desarrollo iterativo.

La base de PyTorch se encuentra el tensor, una estructura de datos similar a un array o una matriz. Estos tensores se utilizan para codificar las entradas, salidas y parámetros de un modelo. La fortaleza de PyTorch radica en su capacidad para realizar diferenciación automática y descenso de gradiente, los pilares de cualquier proceso de entrenamiento de redes neuronales.

CUDA

CUDA, o Compute Unified Device Architecture, es una plataforma de computación paralela y un modelo de API creado por NVIDIA. Permite a los desarrolladores utilizar las GPUs de NVIDIA para procesamiento de propósito general, un enfoque denominado GPGPU (Computación de Propósito General en Unidades de Procesamiento Gráfico).

Al explotar las capacidades de procesamiento paralelo de las GPUs, CUDA puede realizar cálculos significativamente más rápidos que un CPU tradicional para ciertos tipos de aplicaciones, especialmente aquellas que pueden descomponerse en muchas tareas pequeñas e independientes. El aprendizaje profundo es una de estas aplicaciones, donde las numerosas operaciones realizadas en matrices y vectores pueden paralelizarse eficazmente.

Operaciones aceleradas por GPU

La esencia de CUDA radica en el uso de kernels CUDA, funciones de C/C++ ejecutadas en la GPU. Cada invocación de una función kernel CUDA se procesa en un elemento de datos diferente, lo que permite el procesamiento simultáneo de grandes cantidades de datos. Para PyTorch, estos kernels CUDA se emplean para diversas operaciones de bajo nivel requeridas en los cálculos de aprendizaje automático, como la multiplicación de matrices o la convolución.

PyTorch está diseñado para integrarse profundamente con CUDA y aprovecha las capacidades de CUDA siempre que es posible. Cuando se ejecuta una aplicación de PyTorch, puede determinar automáticamente si una GPU compatible con CUDA está disponible. Si lo está, PyTorch usará CUDA para acelerar los cálculos de tensores, transfiriendo efectivamente gran parte del trabajo pesado de la CPU a la GPU.

Es importante mencionar que la integración de PyTorch-CUDA va más allá de simplemente realizar cálculos en la GPU. Los tensores de PyTorch pueden moverse a la memoria de la GPU y se pueden realizar operaciones sobre ellos tal como se harían en una CPU. Los cálculos resultantes se ejecutarán automáticamente en la GPU, lo que conlleva una aceleración significativa.

2.5 *ALGORITMO DE DIJKSTRA*

El algoritmo de Dijkstra es un algoritmo de recorrido de grafos que se utiliza para encontrar el camino más corto desde un único nodo origen a todos los demás nodos en el grafo. El grafo debe ser ponderado (es decir, se asocian distancias o costos con cada arista), conectado (es decir, hay un camino entre cada par de vértices), y todos los pesos deben ser no negativos.

El algoritmo de Dijkstra funciona seleccionando iterativamente el nodo "más cercano" no visitado al nodo origen hasta llegar al nodo de destino.

1. Inicialización: Se selecciona el nodo origen. Y se inicializa la distancia recorrida a 0. Se establece la distancia a todos los nodos no visitados como M, un número muy grande, que indica que aún no conocemos su verdadero valor. También se inicializa la distancia del nodo origen como 0.
2. Relajación de Aristas: Para el nodo actual, se considera todos sus vecinos no visitados y se calcula sus distancias tentativas desde el origen. Por ejemplo, si el nodo actual A está conectado al nodo B con un peso de AB, y la distancia actual desde el origen al nodo A es OA, la distancia a B a través de A será OA+AB (para el nodo inicial OA = 0). Si esta distancia tentativa calculada a un nodo vecino es menor que la distancia actualmente almacenada para ese nodo, que inicialmente es M, se actualiza la distancia como la más corta conocida. Después de considerar todos los vecinos del nodo actual, se marca el nodo actual como visitado (un nodo visitado no será revisado de nuevo).
3. Seleccionar el Siguiente Nodo: Cuando todos los vecinos del nodo actual hayan sido visitados, se selecciona el nodo con la menor distancia conocida desde el nodo origen que aún no ha sido visitado. Se establece como el nuevo "nodo actual" y se vuelve al paso 2. Si todos los nodos han sido visitados, o si la menor distancia tentativa en el conjunto no visitado es M (indicando que no hay conexión con los nodos no visitados restantes), entonces el algoritmo ha terminado. En caso de que tengamos un nodo destino, el algoritmo se podrá terminar cuando se visite dicho nodo.
4. Para encontrar el camino más corto (la secuencia de nodos) en lugar de la distancia más corta, se retrocederá desde cada nodo al nodo origen a lo largo del camino moviéndose

hace el nodo con la distancia más corta, que coincidirá con la distancia actual menos la conexión entre los dos nodos.

2.6 ALGORITMO GENÉTICO

Los algoritmos genéticos son algoritmos de optimización que se inspiran en el proceso de selección natural. Imitan el proceso biológico de la evolución para encontrar soluciones óptimas o casi óptimas a problemas complejos. Los algoritmos genéticos resultan muy útiles para obtener un resultado cercano al optimo en poco tiempo y sin tener que aplicar una solución lógica para resolver el problema complejo.

El esquema básico de su funcionamiento es el siguiente:

1. Inicialización: Se inicializa aleatoriamente a la población con sus genes. Cada miembro de la población tiene una secuencia de genes que dictará, en este caso, la secuencia de movimientos realizados en los nodos para intentar llegar al destino. Para reducir el tiempo de ejecución considerablemente se ha introducido cierta lógica a la asignación de genes, haciendo que solo sea posible que el gen i (nodo al que se viaja en el movimiento i) se asigne si el anterior nodo vinculado al gen i-1 está conectado a ese nodo.
2. Cálculo de resultados: En este paso se calcula el tiempo, o el número de movimientos, que se tarda en llegar al nodo final. Si no se llega se asigna un número mayor al número máximo de movimientos que es el número de genes. Para optimizar la ejecución se suprimen caminos circulares.
3. Cálculo de mejores soluciones: Un enfoque comúnmente usado es comparar las puntuaciones de cada miembro de la población en forma de ranking y quedarse con las mejores soluciones. Este método es intuitivo y rápido, aunque puede llevar a que ciertos genes se propaguen con demasiada rapidez y que se piedra diversidad en la población, que es especial para llegar a una solución óptima. Otro método formar un sistema análogo a un torneo, en el que se seleccionan n candidatos y se comparan de dos en dos en un formato eliminatorio hasta que queda el mejor. Otro enfoque es seleccionar a los

individuos de manera aleatoria, pero asignando probabilidad a cada individuo basado en su puntuación.

4. Cálculo de la población a reemplazar: Para elegir que parte de la población se pueden usar varias técnicas. Una técnica intuitiva es reemplazar la parte de la población con peor puntuación. Sin embargo, este método puede llevar a que máximos relativos en la puntuación se propagan rápidamente en los genes de la población y eviten llegar a una solución óptima. Otro enfoque es tener a los individuos en la población durante un determinado número de iteraciones independiente a su puntuación, análogo a una esperanza de vida constante. Este enfoque es más lento, pero garantiza mayor seguridad de resultados.
5. Cruce: Las soluciones seleccionadas producen descendencia para la siguiente generación. El material genético de los padres se combina utilizando un operador de cruce para producir uno o más descendientes. Este operador puede ser simple, y tratarse de copiar los genes de un progenitor hasta cierto punto y de otro a partir de ese punto, o algo más complejo que mezcle los genes. Este último proceso implementarse con cierta lógica para evitar que dos soluciones completamente distintas, pero igualmente validas lleven a una solución incorrecta.
6. Mutación: La mutación ocurre en la nueva descendencia, y consiste en alterar uno o más valores de genes de manera aleatoria.
7. Iteración: Se vuelve al paso 2 hasta que se llega a un resultado deseable

Capítulo 3. ESTADO DE LA CUESTIÓN

Introducción

La urbanización y el incremento de los vehículos por habitante son algunos de los problemas más importantes en los entornos urbanos actuales. Estos desafíos han generado complicaciones relacionadas con el aparcamiento, el incrementando los tiempos de búsqueda, el consumo de combustible y la innecesaria congestión de tráfico. Para abordar estos problemas, los sistemas de aparcamiento inteligente (SPS) han surgido como una solución en los últimos años. Como parte integral de los sistemas de transporte inteligente (ITS), estos sistemas buscan superar los problemas relacionados con el aparcamiento mediante el uso de tecnología, datos y prácticas innovadoras.

Descripción general de los sistemas de aparcamiento inteligente

Los Sistemas de Aparcamiento Inteligente son una solución sofisticada diseñada para aliviar los complejos desafíos de aparcamiento que enfrentan los entornos urbanos. El objetivo fundamental es gestionar de manera más eficaz los recursos de aparcamiento proporcionando características como información de aparcamiento en tiempo real, la posibilidad de que los conductores reserven plazas de aparcamiento y el guiado hasta el lugar reservado, entre otros. Estos sistemas forman una parte fundamental del panorama más amplio de los Sistemas de Transporte Inteligente (ITS), que incluyen una amplia gama de servicios destinados a mejorar la eficiencia del transporte, mejorar la seguridad y reducir los impactos ambientales.

Tecnologías esenciales en los sistemas de aparcamiento inteligente

Los SPS integran una serie de tecnologías para lograr una funcionalidad óptima. Estos incluyen redes inalámbricas, sistemas de posicionamiento global (GPS) y comunicación vehicular.

Las redes inalámbricas, especialmente aquellas que admiten dispositivos del Internet of Things (IoT), forman la columna vertebral de los SPS. El IoT es crucial para la interconexión y coordinación de múltiples dispositivos y sistemas en SPS, permitiendo la recopilación, análisis y transmisión en tiempo real de los datos que sustentan el funcionamiento del sistema.

Las comunicaciones Vehículo a Infraestructura (V2I) e Infraestructura a Vehículo (I2V) son centrales para muchas funcionalidades de SPS. Estas tecnologías permiten el intercambio de información entre un vehículo y las infraestructuras como los distintos sensores de los aparcamientos. Esta interacción está facilitada por tecnologías como las comunicaciones dedicadas de corto alcance (DSRC), 5G y las redes de área amplia de baja potencia (LPWAN).

Además de estas tecnologías, los SPS también aprovechan algoritmos sofisticados para abordar los complejos desafíos del aparcamiento. Por ejemplo, el programa lineal entero mixto (MILP) resuelve el problema de asignar y reservar espacios de aparcamiento óptimos en función de las necesidades del usuario. El problema de asignación lineal transforma las cuestiones de planificación del aparcamiento, considerando los vehículos como trabajos y los espacios de aparcamiento como agentes, y utiliza algoritmos para ofrecer orientación en tiempo real a los vehículos.

Sensores en los sistemas de aparcamiento inteligente

Sensores Ultrasónicos

Los sensores ultrasónicos desempeñan un papel fundamental en muchos SPS. Estos sensores emiten ondas ultrasónicas y miden el tiempo que estas tardan en rebotar después de golpear un objeto. La distancia entre el sensor y el objeto se calcula en base a este tiempo y se compara con la distancia al suelo, proporcionando información sobe la presencia de un vehículo entre ambos. Su alta precisión, asequibilidad y fiabilidad los convierten en una opción popular para la detección de vehículos en aparcamientos.

Los sensores ultrasónicos suelen estar montados en el techo de la estructura del estacionamiento en cada plaza de estacionamiento, emitiendo y recibiendo continuamente ondas ultrasónicas para detectar la presencia de vehículos. Las ventajas de los sensores ultrasónicos incluyen su funcionamiento sin contacto, su robustez frente a las variaciones de luz y color ambiental, y su idoneidad para detectar una amplia variedad de materiales y superficies. Sin embargo, su precisión de detección puede verse afectada por factores ambientales como la temperatura y la humedad, y también pueden ser propensos a interferencias de señales.

Sensores Infrarrojos

Los sensores infrarrojos funcionan emitiendo luz infrarroja y detectando la luz reflejada en los objetos. Se utilizan ampliamente en los SPS para detectar la presencia o ausencia de vehículos en las plazas de aparcamiento. Estos sensores pueden funcionar en diversas condiciones de iluminación, lo que los hace adecuados tanto para la operación diurna como nocturna.

Los sensores infrarrojos ofrecen una precisión de detección relativamente alta y pueden resistir diversas condiciones climáticas. Sin embargo, pueden verse afectados por obstáculos

como la suciedad o el polvo en la lente, y pueden requerir un mantenimiento regular para garantizar un rendimiento óptimo.

Sensores Magnéticos

Los sensores magnéticos en los SPS detectan cambios en el campo magnético de la Tierra causados por la presencia de un vehículo. Estos sensores suelen utilizarse para la detección de vehículos, el control de su entrada y salida y la monitorización del tráfico. Sus principales ventajas son una fácil instalación, bajo consumo de energía y alta precisión de detección.

Los sensores magnéticos suelen instalarse bajo la superficie de la plaza de aparcamiento, lo que los hace menos susceptibles a ser dañados. No se ven afectados por condiciones ambientales ni por destinos niveles de luz, el calor o la humedad. Sin embargo, su precisión de detección puede verse afectada por la presencia de objetos metálicos grandes en las proximidades.

Sensores de Radar

Los sensores de radar utilizan ondas de radio para detectar objetos. A menudo se utilizan en los SPS para la detección de vehículos, la medición de la velocidad y la monitorización del tráfico. Los sensores de radar ofrecen una alta precisión de detección y alcance, y pueden funcionar en todas las condiciones meteorológicas, lo que los hace adecuados para estacionamientos al aire libre. Sin embargo, su rendimiento puede verse afectado por la presencia de grandes objetos metálicos o estructuras que los obstruyan, especialmente en aparcamientos interiores

Sensores de Bucle Inductivo

Los sensores de bucle inductivo funcionan detectando cambios en un campo electromagnético causados por la presencia de un vehículo. A menudo se utilizan en los SPS para la detección de vehículos en los puntos de entrada y salida, el control del tráfico y la recopilación de datos de tráfico. Estos sensores se diferencian de los sensores magnéticos en su mayor precisión al detectar el vehículo, pudiendo estimar las dimensiones de un vehículo.

Los sensores de bucle inductivo ofrecen una alta precisión de detección y fiabilidad, pero requieren una instalación invasiva, que implica cortar en la superficie de la carretera para instalar el bucle, lo que puede ser una desventaja.

Prácticas comunes en los sistemas de aparcamiento inteligente

El aparcamiento con orientación y reserva es una práctica común en los SPS. Aquí, los conductores pueden reservar una plaza de aparcamiento a través de una aplicación móvil y luego recibir indicaciones para llegar al lugar reservado. Este enfoque reduce el tiempo de búsqueda, reduce la congestión de tráfico y mejora la experiencia general de aparcamiento.

El aparcamiento basado en tarifas dinámicas es otra práctica común en los SPS. Esta práctica ajusta las tarifas de aparcamiento en tiempo real en función de factores como la demanda, la hora del día y la ubicación. Los sistemas de tarifas dinámicas buscan optimizar la utilización del aparcamiento y reducir la congestión mediante la incentivación de patrones de aparcamiento más eficientes.

Conclusión

Con el creciente número de vehículos y la limitada disponibilidad de plazas de aparcamiento, la necesidad de sistemas de gestión de aparcamiento inteligentes, eficientes y fiables se está volviendo cada vez más urgente. Los actuales SPS aprovechan un diverso conjunto de tecnologías, sensores y prácticas innovadoras para abordar estos desafíos. A pesar de los avances significativos, el campo del aparcamiento inteligente sigue presentando oportunidades para la investigación e innovación destinadas a optimizar la experiencia de aparcamiento en los cada vez más congestionados entornos urbanos.

Capítulo 4. DEFINICIÓN DEL TRABAJO

4.1 JUSTIFICACIÓN

Como ya se ha establecido, el creciente número de coches en ciudades en las que el espacio de aparcamiento reducido promueve el uso de aparcamientos inteligentes que mejoren la eficiencia de estos sistemas generando más ingresos para un mismo espacio y mejorando la satisfacción del cliente al reducir el tiempo empleado en aparcar. Además, los aparcamientos con sistemas inteligentes permiten implementar aplicaciones web o móviles que puede incentivar a los clientes a conducir en lugar de usar otros tipos de transporte al dar a esta información en tiempo real sobre el número de plazas disponibles.

Sin embargo, la mayoría de los sistemas actuales dependen de un gran número de sensores para detectar el estado de cada plaza, que se traduce en un gran coste de instalación y de mantenimiento, además de un largo periodo de instalación en el que el aparcamiento queda inutilizado. Para evitar estos inconvenientes, este proyecto propone el uso de cámaras y algoritmos de visión artificial para la detección de los vehículos y de las plazas disponibles en el aparcamiento, que resulta en una instalación más rápida y menos costosa, al reducir significativamente el número de sensores, incluso pudiendo utilizar las cámaras de seguridad previamente instaladas facilitando aún más el proceso de instalación.

Por otra parte, el uso de cámaras en vez de sensores en cada plaza proporciona más información, que en este caso se usará para rastrear a los coches en movimiento por el aparcamiento y guiarles de forma dinámica a la plaza más cercana, mejorando la experiencia del cliente al reducir aún más el tiempo que este pasa buscando el estacionamiento.

4.2 OBJETIVOS

El objetivo principal de este proyecto es desarrollar un paquete completo que demuestre la viabilidad de la propuesta. Este paquete, a través de una entrada de video, debe ser capaz de contar el número de plazas ocupadas, detectar si hay coches moviéndose, y si los hay obtener la mejor ruta a las plazas más cercanas. Al tratarse de una prueba de concepto y no de una implementación completa, se hará una demostración con una única cámara, aunque se simulará una estructura de aparcamiento mayor para poder demostrar la funcionalidad de los algoritmos de guiado.

4.3 METODOLOGÍA

La primera parte que se desarrollara serán los algoritmos de guido, ya que será la única parte del proyecto que se desarrolle en c y por tanto serán independientes del resto del proyecto. Una vez estos estén terminados se pasará a la parte de visión artificial. Primero se implementará una versión simplificada que sea capaz de detectar vehículos en una única imagen, para familiarizarse con los algoritmos, capaz de contar el número de plazas disponibles. El siguiente paso será implementar crear un programa independiente pueda asignar un id único a cada detección en un video y rastrearla en caso de que se mueva. El último paso será juntar las partes anteriores en un programa que para cada fotograma de un video detecte las plazas disponibles, sitúe a un coche en movimiento en el espacio en dimensiones reales y para poder pasar su posición a los algoritmos de guiado para determinar la mejor ruta a la plaza más cercana.

Capítulo 5. SISTEMA/MODELO DESARROLLADO

5.1 ANÁLISIS DEL SISTEMA

El objetivo de este proyecto es partir de la entrada de video de las cámaras en un aparcamiento, procesar las imágenes para detectar los vehículos estacionados y en movimiento, y obtener el camino para guiar a los coches circulando por el aparcamiento hacia la plaza libre más cercana.

Al plantear el uso de cámaras y de algoritmos de visión artificial, debemos decidir el entorno de aplicación de manera específica, ya que este tiene un mayor efecto que en la aplicación de sensores convencionales. Concretamente, es esencial para determinar el proceso de desarrollo de este proyecto la elección de trabajar con aparcamiento al aire libre o cubierto.

Los aparcamientos al aire libre no tienen limitaciones de altura a la hora de colocar las cámaras. Esto permitiría colocar una cámara a una gran altura con un campo de visión que cubriese todo el aparcamiento. Sin embargo, la mayoría de los aparcamientos en las ciudades, donde hay el mayor problema de congestión y por tanto donde se aplicará el proyecto, son aparcamientos cubiertos. Esta decisión conlleva los siguientes obstáculos.

1. Al no poder cubrir todo el aparcamiento con una sola cámara, la solución debe ser capaz de detectar vehículos a través de múltiples imágenes con diferentes perspectivas. Este es el motivo por el que se ha utilizado DeepSort en lugar de un rastreador más simple, que extrae las características principales de la imagen dentro del cuadro delimitador de la detección. A la hora de asociar nuevas detecciones con rastreos anteriores, DeepSort utiliza una métrica que une la confianza de coincidencia de un filtro de Kalman y de la asociación mediante las características principales de la imagen. Si se utilizara más de una cámara, se debería o bien ajustar esa métrica para que las detecciones en los extremos del campo de visión de las

cámaras, donde puede coincidir la detección de dos cámaras, utilizaran únicamente asociación profunda, o modificar la implementación del filtro de Kalman.

2. Otro obstáculo que conlleva la posición de las cámaras en un aparcamiento cubierto es la deformación de las posiciones de las detecciones debido a la mayor diversidad en la distancia a la que las cámaras detectan los vehículos. Este problema causa la necesidad de aplicar una transformación homográfica a las posiciones de las detecciones para conocer su posición real.

   Además, las filas de aparcamiento, que se deben distinguir para saber en qué nodo se encuentra cada vehículo aparcado, también se encuentran relativamente deformadas. Por tanto, se deberá aplicar los algoritmos necesarios para detectar las filas en este espacio deformado y agrupar los coches en las filas que les corresponda.

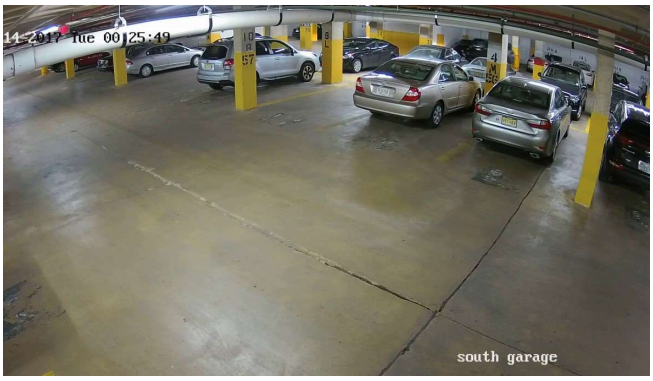

Figura 4: Imágenes de un aparcamiento al aire libre y cubierto, con anotaciones en distintos puntos de tamaño real similar para comparar su deformación. Primera imagen: Car in parking garage - security camera [17]. Segunda imagen: PKLot Dataset [18]

5.1.1 CÁMARAS

La elección de las cámaras se basará en sus capacidades de adaptarse a condiciones de baja luminosidad. Su tasa de refresco deberá ser alta para que el programa no deba esperar mucho a que se capture un fotograma cuando se demande, ya que este proceso ocurre en serie con la detección de objetos que ya tiene un tiempo de inferencia relativamente alto.

5.1.2 DETECCIÓN DE OBJETOS

En esta parte del proceso se utilizará un modelo de visión artificial, en ese caso yolov8x, para detectar los objetos presentes en cada fotograma. A cada detección estará asociada su clase, su confianza y su cuadro delimitador, además de un id único. Sin embargo, estos ids no se mantendrán entre detecciones, es decir, no se podrán utilizar para identificar a un vehículo entre fotogramas. Para ello se recurrirá a algoritmos de rastreo, en este caso DeepSort.

Figura 5: Detección de vehículos con YOLOv8 en aparcamiento cubierto

5.1.3 RASTREO DE DETECCIONES Y ASIGNACIÓN DE IDS

DeepSort se utilizará para obtener una lista de rastreos a partir de una lista de detecciones para cada fotograma. Al llamar a las funciones de DeepSort también se podrá establecer una relación entre los ids de las detecciones para un determinado fotograma y los ids de los rastreos, que serán constantes y únicos durante todo el intervalo de detección. Esta relación se podrá usar para añadir funcionalidad a el rastreador, como un intervalo de confianza dinámico para el que las detecciones se tienen en cuenta, que servirá para evitar las detecciones intermitentes.

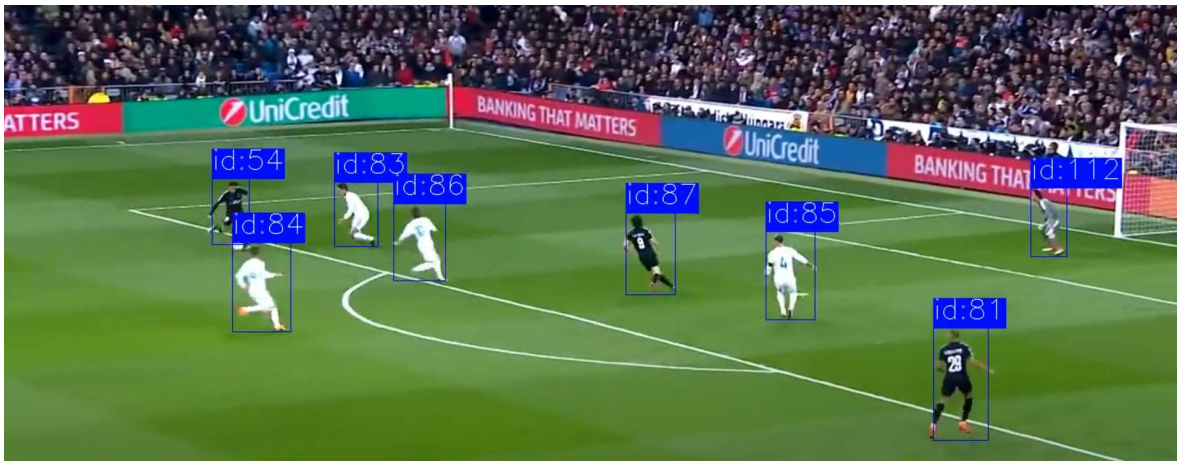

Figura 6: Ejemplo de detecciones asociadas a un id único con DeepSort, Sanyam [2]

5.1.4 TRANSFORMACIÓN HOMOGRÁFICA Y SEPARACIÓN ENTRE VEHÍCULOS EN REPOSO Y MOVIMIENTO

Para diferenciar entre los vehículos en movimiento y estacionados se calculará su velocidad y las detecciones que pasen de un umbral se considerarán en movimiento. Sin embargo, las posiciones de las detecciones no se encuentran en el espacio real, sino que se tiene su posición en la imagen. Para hallar su posición en coordenadas reales se utilizará una transformación homográfica a través de puntos reales conocidos en la imagen. La detección se asociará con un id de rastreo a través de DeepSort, y a ese id se asociará sus coordenadas reales para cada fotograma y su velocidad en los últimos n fotogramas.

5.1.5 CÁLCULO DE NODOS CON PLAZAS DISPONIBLES

Como ya se ha mencionado, la perspectiva de la cámara deforma el espacio en el que se encuentran los vehículos. Además, las filas de plazas pueden no encontrarse en el plano frontal a la cámara, quedando en diagonal en la imagen capturada. También es notable que es común encontrar dos filas de aparcamiento contiguas pero separadas por varios nodos, ya que no se puede atravesar de una a otra (aparente en la primera imagen de la figura 4). Todo esto hace que distinguir que detecciones pertenecen a cada nodo no sea trivial.

Para ello se ha decidido utilizar análisis de componentes principales de dos dimensiones a los centros de las detecciones ya que, al tratar con filas, debe haber una dirección en la que los coches aparcados estén alineados, y en el caso de que haya dos filas contiguas que se deban separar se podrá hacer clustering de dos grupos para separarlas.

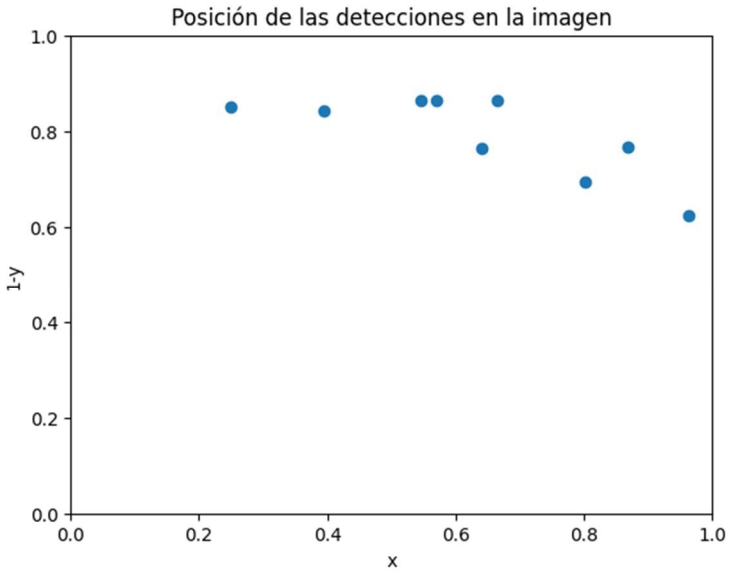

Figura 7: posición de las detecciones en la anterior figura, nótese que el eje y está invertido

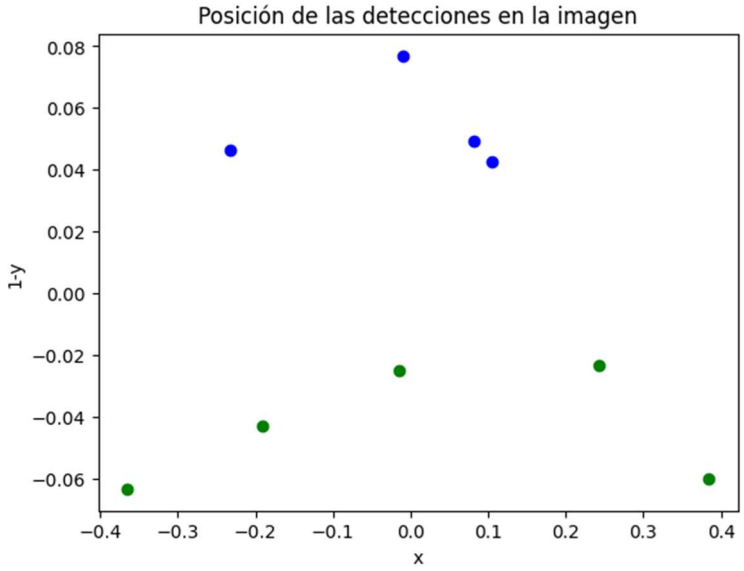

Figura 8: Posición de las detecciones tras aplicar ACP de dos dimensiones y clustering de dos grupos.

5.1.6 ALGORITMOS DE GUIADO

Los algoritmos de guiado deben ser capaces de recibir los nodos con plazas disponibles y la posición de un vehículo en movimiento y calcular la óptima que debe seguir ese vehículo hasta la plaza más cercana. Dado que los algoritmos de reconocimiento de objetos tienen un tiempo de inferencia significativo y que para que el sistema funcione en tiempo real este debe ser tan rápido como sea posible, se ha decidido implementar los algoritmos en el lenguaje de programación c, que es considerablemente más rápido que Python.

	Energy
(c) C	1.00
(c) Rust	1.03
(c) C++	1.34
(c) Ada	1.70
(v) Java	1.98
(c) Pascal	2.14
(c) Chapel	2.18
(v) Lisp	2.27
(c) Ocaml	2.40
(c) Fortran	2.52
(c) Swift	2.79
(c) Haskell	3.10
(v) C#	3.14
(c) Go	3.23
(i) Dart	3.83
(v) F#	4.13
(i) JavaScript	4.45
(v) Racket	7.91
(i) TypeScript	21.50
(i) Hack	24.02
(i) PHP	29.30
(v) Erlang	42.23
(i) Lua	45.98
(i) Jruby	46.54
(i) Ruby	69.91
(i) Python	75.88
(i) Perl	79.58

	Time
(c) C	1.00
(c) Rust	1.04
(c) C++	1.56
(c) Ada	1.85
(v) Java	1.89
(c) Chapel	2.14
(c) Go	2.83
(c) Pascal	3.02
(c) Ocaml	3.09
(v) C#	3.14
(v) Lisp	3.40
(c) Haskell	3.55
(c) Swift	4.20
(c) Fortran	4.20
(v) F#	6.30
(i) JavaScript	6.52
(i) Dart	6.67
(v) Racket	11.27
(i) Hack	26.99
(i) PHP	27.64
(v) Erlang	36.71
(i) Jruby	43.44
(i) TypeScript	46.20
(i) Ruby	59.34
(i) Perl	65.79
(i) Python	71.90
(i) Lua	82.91

	Mb
(c) Pascal	1.00
(c) Go	1.05
(c) C	1.17
(c) Fortran	1.24
(c) C++	1.34
(c) Ada	1.47
(c) Rust	1.54
(v) Lisp	1.92
(c) Haskell	2.45
(i) PHP	2.57
(c) Swift	2.71
(i) Python	2.80
(c) Ocaml	2.82
(v) C#	2.85
(i) Hack	3.34
(v) Racket	3.52
(i) Ruby	3.97
(c) Chapel	4.00
(v) F#	4.25
(i) JavaScript	4.59
(i) TypeScript	4.69
(v) Java	6.01
(i) Perl	6.62
(i) Lua	6.72
(v) Erlang	7.20
(i) Dart	8.64
(i) Jruby	19.84

Tabla 1:Eficiencia de Python respecto a otros lenguajes en cuanto a energía, tiempo y memoria empleados, R. Pereira [14]

Cabe destacar que la mayoría de las librerías de Python, como PyTorch, están escritas de igual manera en c para aprovecharse de este aumento de velocidad, y que por tanto la mayoría del código de este proyecto escrito en Python, que se basa en estas librerías, ya está optimizada.

5.2 DISEÑO

La ruta que sigue la información a través de los distintos programas hasta obtener el resultado final, que es la ruta óptima a la plaza más cercana, empieza en la detección de vehículos a partir de una entrada de video constante de cada cámara. Para la prueba de concepto que se demostrará en este proyecto solo se utilizará una cámara, pero aumentar ese número no representaría ningún cambio fundamental en el sistema. Cada vez que se captura un fotograma y se pasa por YOLO se obtienen las detecciones en la imagen. Estas detecciones se filtran según sus clases para asegurar que solo se detecten vehículos. La lista procesada de detecciones se pasa por DeepSort, que las asigna a rastreos ya detectados o a nuevos rastreos tentativos. Posteriormente, se aplica una transformación homográfica a las posiciones de los puntos que delimitan la posición de los vehículos para obtener su posición en coordenadas reales. A través de la asignación de cada id de rastreo a los ids de las detecciones de YOLO se accede a un diccionario que contiene los datos de cada rastreo y se actualiza su velocidad y otros valores útiles para futuros cálculos. Con la lista de velocidades de cada vehículo se decide cuáles son los vehículos en movimiento y se separan de los vehículos aparcados. La lista de vehículos aparcados se pasa por un detector de plazas libres, que compara el número de vehículos en cada región con el número de plazas en esta. Por otro lado, la posición de los vehículos en movimiento se pasa al buscador de ruta, que a través de las posiciones de plazas libres y de la posición actual del coche encuentra la mejor plaza.

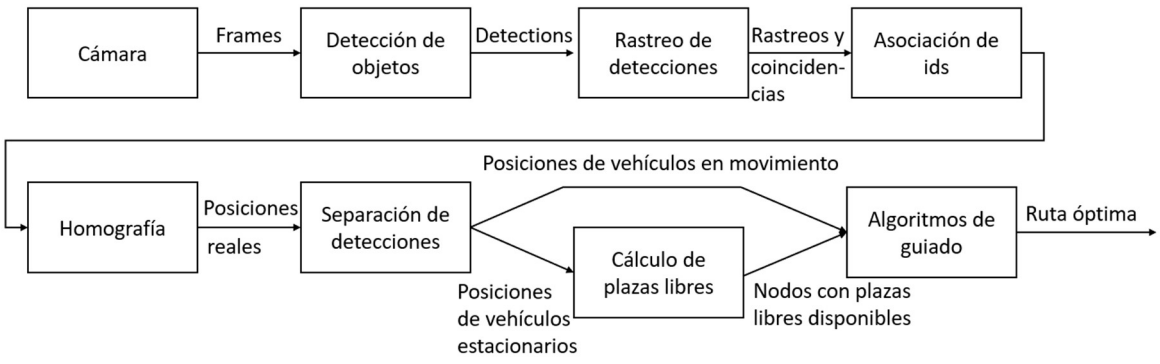

Ilustración 6: Pipeline del proyecto

5.3 IMPLEMENTACIÓN

5.3.1 ALGORITMO DE DIJKSTRA

La implementación consistió en aplicar el algoritmo, que está bien documentado y definido, con dos modificaciones principales respecto a su implementación más común. La primera es que en vez de tener un único nodo de destino se tiene múltiples, ya que hay múltiples nodos con plazas libres y se quiere guiar al coche al nodo más cercano. La segunda es que el algoritmo implementado calcula el camino a seguir para llegar de manera óptima de un nodo a otro en vez de únicamente calcular la distancia más contra entre ellos. Para ello, una vez calculada la distancia de todos los nodos que conectados al nodo final y el inicial, y partiendo del nodo final, se buscaba un nodo cuya distancia al nodo inicial fuese igual a la distancia del anterior nodo al inicial menos la distancia que los conecta. Este acercamiento resulta en un único camino óptimo y descarta otros caminos paralelos de la misma distancia, aunque para la aplicación de este proyecto esto no resulta un problema.

5.3.2 ALGORITMO GENÉTICO

La implementación del algoritmo genético consiste en tener una matriz bidimensional que representa a la población, siendo cada fila un individuo, y siendo cada una de las columnas de cada fila los genes de ese individuo. En esta implementación, los generes representan la secuencia de movimientos que el individuo realiza para intentar llegar al nodo final. Estas secuencias se inicializan de manera aleatoria con dos condiciones. La primera es que el primer nodo de todas las secuencias sea el nodo inicial. La segunda es que cada gen en la secuencia de genes de cada individuo sea inicializado con un valor aleatorio dentro de una lista de los nodos conectados al nodo anterior en vez de ser un nodo aleatorio de todo el conjunto de nodos.

Estas dos condiciones se podrían haber evitado, ya que si un individuo no empieza en el nodo inicial o viaja a través de conexiones de nodos inexisten puntuación se vería afectada

negativamente. Sin embargo, implementar esta pequeña parte lógica en el algoritmo genético permite disminuir el tiempo para llegar a un resultado dentro del rango admisible considerablemente (un algoritmo genético no garantiza llegar a la solución óptima pero estadísticamente si garantiza llegar a una solución cercana en un tiempo de ejecución relativamente corto [Referencia]).

El esquema básico del funcionamiento del algoritmo genético utilizado es el siguiente:

1. Inicialización: Como ya se ha mencionado se inicializa semi-aleatoriamente a la población con sus genes. Cada miembro de la población tiene una secuencia de genes que dictará, en este caso, la secuencia de movimientos realizados en los nodos para intentar llegar al destino.
2. Cálculo de puntuaciones: Se calcula el tiempo, o el número de movimientos, que se tarda en llegar al nodo final. Si no se llega se asigna un número mayor al número máximo de movimientos que es el número de genes. Para optimizar la ejecución se suprimen caminos circulares.
3. Cálculo de mejores soluciones: En este caso se ha optado por ordenar a todos los individuos de mayor a menor puntuación y escoger los n coches en la parte superior de la lista.
4. Cálculo de la población a reemplazar: Se ha elegido asignar una edad a cada individuo de la población y reemplazarlos al cabo de un determinado número de generaciones independientemente de su puntuación. Para ello, en cada iteración se eliminan los últimos n individuos, se mueve a toda la población n posiciones hacia abajo (envejeciéndoles 1 iteración) y se añade a los nuevos individuos en la parte superior de la lista.
5. Cruce: Con la parte de la población seleccionada, se coge a individuos de dos en dos (análogo a los padres) y se asigna a dos descendentes la primera parte del código genético de un padre y la otra parte del código genético del otro padre. Hasta qué punto el código genético transferido es de un padre o de otro se calcula de manera aleatoria para cada caso, escogiendo un punto en el que el reemplazo no afecte a la continuidad de los nodos al viajar entre dos nodos no conectados

6. Mutación: Una vez se tiene a los nuevos individuos, estos se someten a mutación de manera aleatoria con una probabilidad predefinida. Si un individuo muta, se selecciona uno de sus genes o nodos y se reemplaza por otro que esté conectado tanto al gen anterior como al posterior.
7. Iteración finalizada: Se vuelve al paso 2 hasta que se llega a un resultado dentro de rango aceptable o hasta que el tiempo de ejecución supera el deseado.

5.3.2.1 COMPARATIVA

Como ya se ha mencionado, el algoritmo de Dijkstra llega uno de los posibles múltiples resultados óptimos. Por otro lado, el algoritmo genético nos permite ejecutar el programa durante un tiempo definido y llegar a una solución cada vez más cercana a la óptima. Para el caso de un sistema con un numero de nodos reducido, el algoritmo de Dijkstra es más útil, ya que en sistemas modernos el tiempo de ejecución es ínfimo y no resulta un problema. Sin embargo, a medida que escalan los nodos N el tiempo de ejecución escala de manera $O(|N^2|)$, o $O(|E| + |N| * \log|N|)$, E siendo el número de conexiones entre nodos, en caso de utilizar un montículo de Fibonacci, algo que resulta útil cuando $|E| \ll |V^2|$, es decir, en gráficos con pocas interconexiones. En el apartado de resultados se comparará de manera práctica los resultados para diferentes números de nodos y se explicará la elección del algoritmo a utilizar.

5.3.3 TRANSFORMACIÓN HOMOGRÁFICA

Para calcular la matriz necesaria para realizar una transformación homográfica se debe conocer puntos de la imagen, que se encuentra en el espacio transformado, asociados a puntos en el espacio real. Para ello, se ha diseñado una función que permita al usuario cargar una imagen y hacer clic en los puntos cuya distancia real a un origen relativo conoce, devolviéndole su posición en el espacio transformado. El resultado de esta función tiene el siguiente formato:

```
Point coordinates are: (887, 555)
Point coordinates are: (408, 324)
Point coordinates are: (235, 230)
Point coordinates are: (160, 187)
Point coordinates are: (1058, 336)
Point coordinates are: (415, 167)
```

Estos puntos junto con sus coordenadas reales se deberán introducir en la función principal, que se usarán para calcular las coordenadas de los rastreos en cada iteración

```
pts_src = np.array([[887, 555], [408, 324], [235, 230], [160, 187], [1058, 336],
[415, 167]])  # Points in source image

pts_dst = np.array([[0 ,0]    , [5 ,0]    , [10, 0]   , [15, 0]   , [0,
3]    ,  [10, 3]])  # Same points in real coordinates
```

5.3.4 YOLOv8

Para este proyecto se ha decidió utilizar una versión preentrenada de YOLOv8 ofrecida por Ultralytics, la empresa desarrolladora del modelo. Los modelos que ofrece han sido entrenados con la base de datos COCO, un dataset de referencia en cuanto a detección y segmentación de objetos. Entrenar el modelo utilizando un dataset especializado de vehículos en aparcamientos sería más apropiado para el proyecto, ya que la riqueza de contextos que ofrece COCO es solo útil cuando se requiere detectar objetos en contextos diversos, y el gran número de clases para las que está entrenada el modelo (80) le añade una complejidad innecesaria, ya que la mayoría no se usarán. Sin embargo, el modelo preentrenado que se ha decidido usar, Yolov8x, consta de 68.2 millones de parámetros [Referencia], y entrenarlo en la GPU disponible hasta llegar al punto en el que se encuentra el modelo preentrenado habría requerido semanas de computación. Además, entrenar estos modelos requiere un entendimiento profundo de su funcionamiento para ajustar sus hyperpárametros de manera óptima, algo que habría requerido emplear una gran cantidad de tiempo para obtener unos resultados ligeramente superiores.

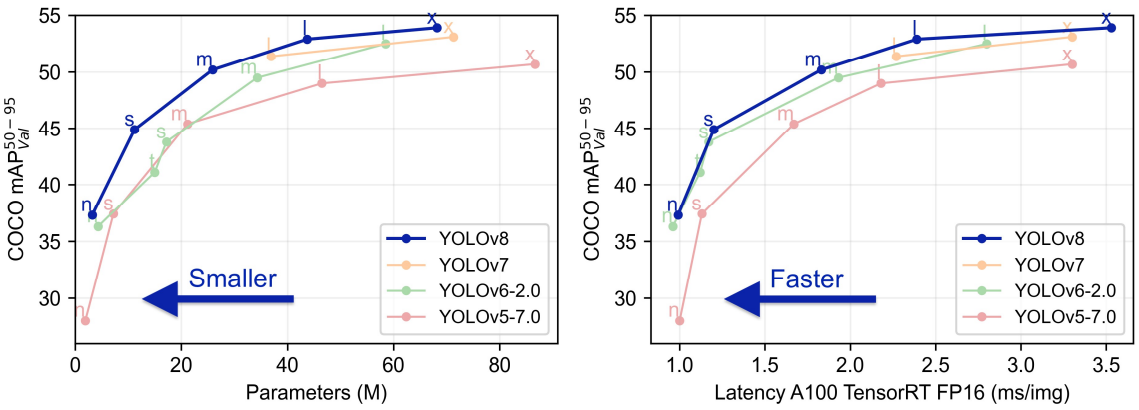

Figura 9: Comparativa de modelos preentrenados de YOLOv8, Ultralytics [10]

5.3.5 DEEPSORT

Para la implementación de DeepSort, es necesario clonar el entorno de trabajo directamente desde su repositorio en GitHub, debido a que no existe una opción para instalar la biblioteca a través de un gestor de paquetes convencional, como pip. Sin embargo, independientemente de su disponibilidad en pip, la adquisición directa del código fuente es un requisito para este proyecto, ya que este requiere modificar parte del código para poder extraer datos internos del modelo y crean nuevas funciones que se llamen desde dentro de su ejecución.

Capítulo 6. ANÁLISIS DE RESULTADOS

6.1 DETECCIÓN Y RASTREO

El objetivo de esta sección era desarrollar un modelo capaz de detectar los vehículos en cada fotograma de una entrada de vídeo, asignarles un id único y rastrearlos en los siguientes fotogramas. Una vez se hubiera identificado a cada vehículo, se pretendía separar a los vehículos en movimiento de los vehículos en estacionados y aplicar análisis de componentes principales y clustering a los centros de las detecciones los vehíuclos aparcados para identificar a que fila pertenecían. Con los coches ya separados entre coches en movimiento y coches estacionados a su vez separados por filas, el objetivo era poder contar los coches en una fila pertenecientes a una región de la pantalla que representaría un nodo en el diagrama de bloques, para determinar si había plazas libres en ese nodo. Todos estos objetivos se han cumplido en el resultado final presentado.

Para visualizar las distintas categorías en las que se han clasificado los distintos vehículos se ha usado los siguientes colores:

-Para identificar los vehículos en las dos filas se han utilizado el verde y el cian, en este caso el verde corresponde a la fila en la que se va a contar las plazas libres

-Para identificar el vehículo en movimiento se ha utilizado el azul

-Par identificar el cuadro delimitador donde se van a contar las plazas se ha utilizado en blanco, al igual que para anotar el centro de las detecciones de la fila que pertenece al nodo analizado que se encuentran dentro del cuadro delimitador. También se anuncia en pantalla en número de plazas libres con ese mismo color.

-Por último, todos los rastreos están acompañadas de su posición en coordenadas reales relativas a un origen arbitrario marcado con un punto negro. Los nuevos ejes tomados son paralelos y perpendiculares a la vía de circulación.

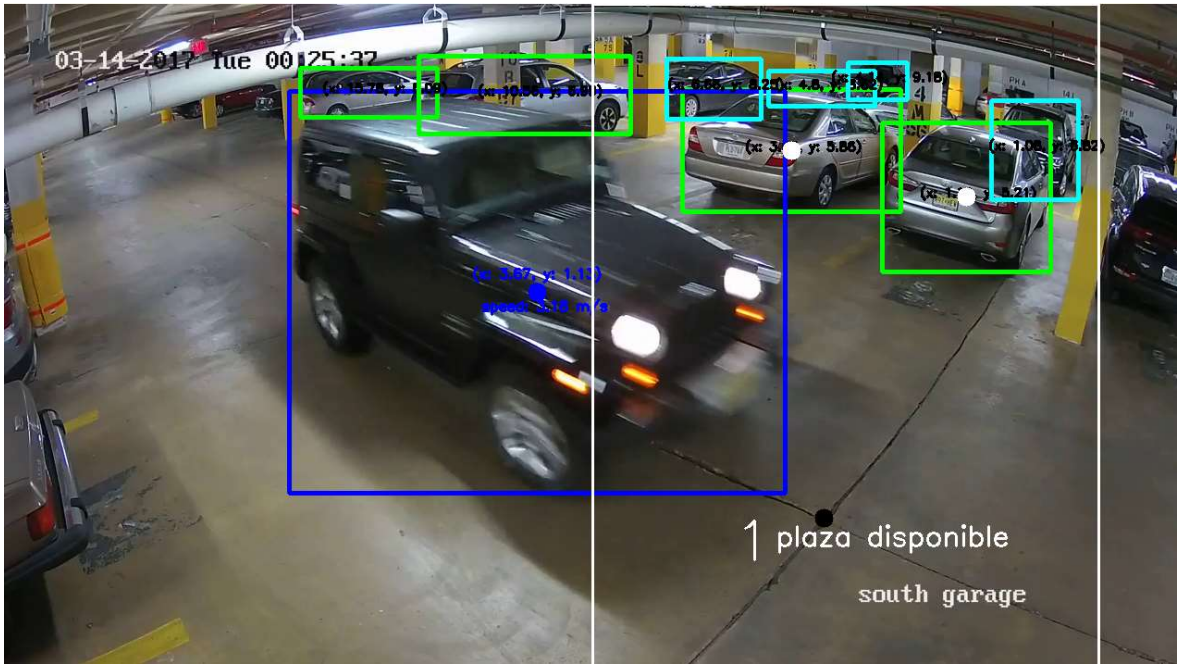

Ilustración 7: Fotograma del video anotado

Esta visualización tiene únicamente el fin de presentar los resultados, y en la ejecución real del programa no se emplearía tiempo de ejecución en ella.

Cabe destacar que cuando un coche en movimiento pasa por delante de un coche aparcado, los cuadros delimitadores de ambos tienden a deformarse, dando lugar a resutlados erróneos. Sin embargo, en esos casos se puede detener la actualización del número de plazas disponibles con pocos cambios al codigo, que remedia los problemas que esto pudiera causar.

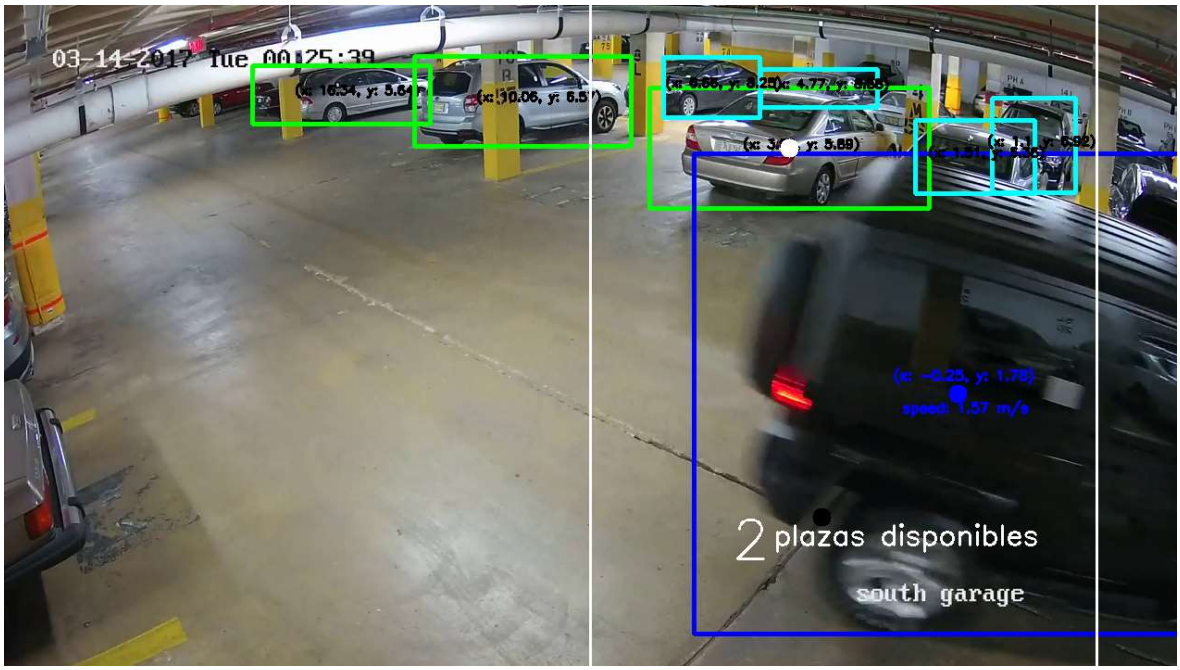

Ilustración 8: Medida incorrecta de la posición de uno de los vehículos en la región

6.2 ALGORITMOS DE GUIADO

Como ya se ha mencionado, el algoritmo genético se puede ejecutar indefinidamente, y por tanto se deberá decidir las generaciones de ejecución de antemano. Para ello, se ejecutará primero el algoritmo de Dijkstra para saber cuál es el tiempo de ejecución a superar con el algoritmo genético, ya que solo se implementará este algoritmo en caso de que represente una mejora respecto al anterior. Una vez se tenga ese tiempo, se ajustará el número de generaciones para que el tiempo de ejecución sea inferior y se compararán los resultados para ver que el resultado obtenido por el algoritmo genético, que no es determinista, es el óptimo. Si el algoritmo genético cumple estas condiciones, se considerará su uso para la situación estudiada.

Para generar la matriz que representa el gráfico de nodos se ha creado la siguiente función, que genera una matriz en la que todos los nodos están conectados a partir de la que los algoritmos de guiado generan la ruta óptima.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define M 100000
#define MAX_VALUE 20

void generate_matrix(int n) {
    srand(time(NULL));
    int i, j, chance;
    int mat[n][n];

    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            mat[i][j] = (i == j) ? M : M;
        }
    }

    for (i = 0; i < n - 1; i++) {
        mat[i][i + 1] = rand() % MAX_VALUE;
        mat[i + 1][i] = mat[i][i + 1];
    }


    for (i = 0; i < n; i++) {
        for (j = i + 2; j < n; j++) {
```

```
            chance = rand() % 4;
            if (chance == 1) {
                mat[i][j] = rand() % MAX_VALUE;
                mat[j][i] = mat[i][j];
            }
        }
    }

    printf("int mat[%d][%d] = {\n", n, n);
    for (i = 0; i < n; i++) {
        printf("    {");
        for (j = 0; j < n; j++) {
            if (j != 0)
                printf(", ");
            if (mat[i][j] == M) {
                printf("M");
            } else {
                printf("%d", mat[i][j]);
            }
        }
        printf("}%s\n", (i == n - 1) ? "" : ",");
    }
    printf("};\n");
}
```

Resultado

La primera prueba se realizará para el numero de nodos V = 15

```
int mat_generated[15][15] = {
    {M, 9, M, M, M, M, M, 7, M, M, M, 7, 11, M, M},
    {9, M, 16, M, M, M, M, 17, M, 5, 11, M, M, M, 14},
    {M, 16, M, 12, 18, M, M, 17, M, M, M, M, M, M, 9},
    {M, M, 12, M, 6, M, M, M, M, 18, 4, M, 16, M, M},
    {M, M, 18, 6, M, 7, 19, 8, M, M, M, M, 3, M, 1},
    {M, M, M, M, 7, M, 19, M, 17, M, M, 15, M, 18, M},
    {M, M, M, M, 19, 19, M, 9, 7, M, 8, M, M, 19, M},
    {7, 17, 17, M, 8, M, 9, M, 6, 4, 12, M, M, M, M},
    {M, M, M, M, M, 17, 7, 6, M, 1, M, M, M, M, 3},
    {M, 5, M, 18, M, M, M, 4, 1, M, 16, M, M, M, M},
    {M, 11, M, 4, M, M, 8, 12, M, 16, M, 17, 8, 13, M},
    {7, M, M, M, M, 15, M, M, M, M, 17, M, 3, M, M},
    {11, M, M, 16, 3, M, M, M, M, M, 8, 3, M, 11, 14},
    {M, M, M, M, M, 18, 19, M, M, M, 13, M, 11, M, 1},
    {M, 14, 9, M, 1, M, M, M, 3, M, M, M, 14, 1, M}
};
Dijkstra Algorithm:
Distance: 14
Path: 0, 11, 12, 4, 14.
Time spent in dijkstra: 0.001000 seconds
Genetic algorithm:
```

```
Puntuacion: 15
Path: 0, 12, 4, 14.
Time spent in genetic algorithm: 0.057000 seconds
Press any key to continue.
```

En estos resultados no se ha ajustado el algoritmo genético para ser igual de rápido que el algoritmo de Dijkstra, ya que este es prácticamente instantáneo y para un numero de nodos tan pequeño no es justificable utilizar un algoritmo genético para intentar mejorar el tiempo.

La siguiente table muestra los tiempos de ejecución para diferentes números de nodos.

	Tiempo de ejecución (segundos)	
Nodos	Algoritmo de Dijkstra	Algoritmo Genético
10	0,001	0,009
20	0,001	0,014
40	0,001	0,027
70	0,001	0,048
100	0,002	0,069

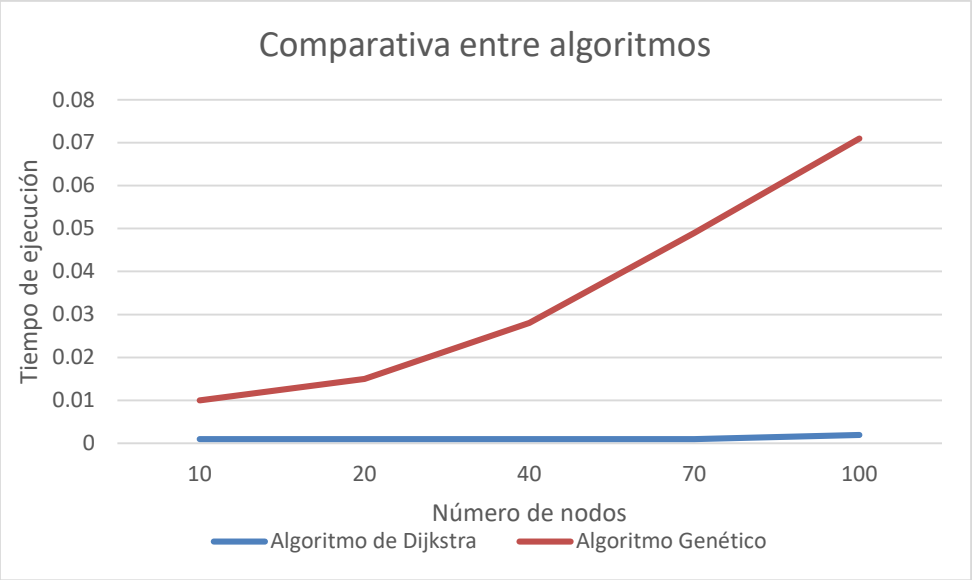

Figura 10: Gráfica comparativa entre los dos algoritmos

Como se puede ver, el algoritmo de Dijkstra es más rápido que el algoritmo genético para cualquier número de nodos dentro del rango razonable para un aparcamiento. Por tanto, será el algoritmo a utilizar para el resultado final.

Capítulo 7. CONCLUSIONES Y TRABAJOS FUTUROS

Este proyecto tenía como objetivo desarrollar un modelo que a partir de una entrada de video de una cámara en un aparcamiento permitiera obtener las indicaciones para guiar a los vehículos en movimiento detectados a las plazas más cercanas. El resultado final ha cumplido estos objetivos, con un buen tiempo de inferencia y una gran fiabilidad gracias a la elección de DeepSort en lugar de un tracker convencional.

Como ya se he mencionado en el documento, una aplicación real del proyecto requeriría usar más de una cámara para cubrir todo el aparcamiento. Este cambio requeriría modificar ligeramente el algoritmo de DeepSort para poder asignar un mismo id a dos detecciones simultaneas de un mismo vehículo en diferentes cámaras. Además, la aplicación real requeriría desarrollar el entorno de comunicación entre las cámaras, las luces indicadoras para informar al usuario del camino a tomar y el ordenador central.

Otra posible mejora sería, en caso de que se tratase de un aparcamiento muy transitado con muchos vehículos en movimiento, sería utilizar un algoritmo de asignación que tuviera acceso a la matriz de la distancia de cada coche a todos los nodos con plazas libres disponibles para elegir la plaza a la que se guía a cada coche teniendo el en cuenta el coste total de todas las rutas, ya que no necesariamente guiar a cada coche a la plaza más cercana es la manera más rápida de guía a todo el conjunto de coches. Sin embargo, esta ruta podría ser contraintuitiva para los usuarios que podrían desviarse de la ruta establecida para llegar a una plaza más cercana y causar problemas para los demás usuarios.

Capítulo 8. BIBLIOGRAFÍA

[1] "DeepSORT: Sort with a Deep Association metric," Luffca, https://www.luffca.com/2023/05/multiple-object-tracking-deepsort/ (accessed Jul. 20, 2023).

[2] Sanyam, "Understanding multiple object tracking using DeepSORT," LearnOpenCV, https://learnopencv.com/understanding-multiple-object-tracking-using-deepsort/ (accessed Jul. 20, 2023).

[3] C. A. Pascual, "El Número de Coches por habitante aumenta en España," Newtral, https://www.newtral.es/numero-coches-espana-europa/20220822/ (accessed Jul. 20, 2023).

[4] "Homography transformations," OpenCV, https://docs.opencv.org/4.x/d9/dab/tutorial_homography.html (accessed Jul. 20, 2023).

[5] M. G. Diaz Ogás, R. Fabregat, and S. Aciar, "Survey of smart parking systems," MDPI, https://www.mdpi.com/2076-3417/10/11/3872 (accessed Jul. 20, 2023).

[6] Smart Parking Systems: Comprehensive Review based on ... - Cell Press, https://www.cell.com/heliyon/pdf/S2405-8440(21)01153-1.pdf (accessed Jul. 20, 2023).

[7] IEEE Xplore Full-text PDF:, https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7169508 (accessed Jul. 20, 2023).

[8] Urban traffic detectors - IEEE xplore, https://ieeexplore.ieee.org/document/9515014/ (accessed Jul. 20, 2023).

[9] Roboflow, "Home," Supervision, https://roboflow.github.io/supervision/ (accessed Jul. 20, 2023).

[10] "Ultralytics," GitHub, https://github.com/ultralytics (accessed Jul. 20, 2023).

[11] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," arXiv.org, https://arxiv.org/abs/1506.02640 (accessed Jul. 20, 2023).

[12] R. Kanjee, "DeepSORT - deep learning applied to object tracking," Medium, https://medium.com/augmented-startups/deepsort-deep-learning-applied-to-object-tracking-924f59f99104 (accessed Jul. 20, 2023).

[13] R. Pereira, G. Carvalho, L. Garrote, and U. J. Nunes, "Sort and deep-sort based multi-object tracking for mobile robotics: Evaluation with New Data Association Metrics," MDPI, https://www.mdpi.com/2076-3417/12/3/1319 (accessed Jul. 20, 2023).

[14] R. Pereira, et al., "Energy efficiency across programming languages. How do energy, time, and memory relate?", in Proc. 10th ACM SIGPLAN Int. Conf. Software Language Engineering, Vancouver, Canada, Oct. 23–24, 2017, pp. 256–267.

[15] V. Lyashenko, "Pytorch Cuda - the definitive guide," cnvrg, https://cnvrg.io/pytorch-cuda/ (accessed Jul. 20, 2023).

[16] "Understanding nvidia cuda: The basics of GPU parallel computing," Hire the World's Most Deeply Vetted Remote Developers, https://www.turing.com/kb/understanding-nvidia-cuda (accessed Jul. 20, 2023).

[17] "Car in parking garage - security camera," YouTube, https://www.youtube.com/watch?v=zBBVnq20HFU (accessed Jul. 20, 2023).

[18] [1] A. A. Ali, "PKLOT dataset," Kaggle, https://www.kaggle.com/datasets/ammarnassanalhajali/pklot-dataset (accessed Jul. 20, 2023).

ANEXO I: ALINEACIÓN DEL PROYECTO CON LOS ODS

Este proyecto se alinea con varios de los Objetivos de Desarrollo Sostenible (ODS) de las Naciones Unidas, que tienen como objetivo abordar los desafíos globales y promover la prosperidad mientras se protege el medio ambiente. Algunos de los ODS que este proyecto apoya incluyen:

ODS 11- Ciudades y Comunidades Sostenibles: Al optimizar la gestión del estacionamiento y hacer un mejor uso de los espacios de estacionamiento disponibles, el sistema de estacionamiento inteligente contribuye a crear entornos urbanos más sostenibles y eficientes. La mejora de los sistemas de estacionamiento puede reducir la congestión del tráfico, mejorar la movilidad urbana y proporcionar un mejor acceso al transporte público, aspectos esenciales de las ciudades y comunidades sostenibles.

ODS 9- Industria, Innovación e Infraestructura: El desarrollo e implementación del sistema de estacionamiento inteligente puede impulsar la innovación en los campos de la visión artificial, la inteligencia artificial y la planificación urbana. Al utilizar tecnologías avanzadas y promover su integración en la infraestructura urbana, este proyecto fomenta la innovación y respalda la industrialización sostenible.

ODS 13- Acción por el Clima: Un sistema de estacionamiento inteligente eficiente puede ayudar a mitigar el cambio climático al reducir el consumo de combustible y las emisiones de vehículos. Al guiar a los conductores rápidamente a los espacios de estacionamiento disponibles, el sistema puede minimizar el tiempo empleado en buscar estacionamiento, lo que lleva a una menor emisión de gases de efecto invernadero y contribuye a los esfuerzos de mitigación del cambio climático.

ODS 12- Consumo y Producción Responsables: El sistema de aparcamiento inteligente tiene como objetivo utilizar los recursos de manera más eficiente al reducir la cantidad de sensores necesarios para monitorear y gestionar los espacios de estacionamiento. Al emplear cámaras

y técnicas de procesamiento de imágenes, el sistema puede minimizar el consumo de recursos y los residuos, alineándose con el objetivo de promover patrones de consumo y producción responsables.

En resumen, el proyecto del sistema de aparcamiento inteligente se alinea con varios ODS de las Naciones Unidas al promover el desarrollo urbano sostenible, fomentar la innovación y abordar el cambio climático. Al trabajar hacia estos objetivos, el proyecto contribuye a un futuro más sostenible y próspero.