TRABAJO DE FIN DE GRADO

Detección y geolocalización de objetos mediante visión artificial desde vehículos en movimiento

Grado Universitario en Ingeniería Informática

2020-2021

Autor: Daniel Romero Ureña

Tutor: Daniel Amigo Herrero

ABSTRACT

Computer vision techniques have evolved and gained relevance over the last years within the technological field, being implemented in a great variety of fields, being one of these and the main one in which this work will be mainly focused on, that of autonomous driving.

This project will develop an algorithm capable of detecting and geolocating objects captured from different cameras located in a moving vehicle, making use of images taken from different perspectives, as well as different metadata that provide the necessary information to locate the objects within the route taken by the vehicle.

To achieve these objectives, it will be necessary to analyze and filter a database that provides all the necessary information for the use of the algorithm, as well as the study and application of triangulation techniques through artificial vision to be used by the algorithm. In addition to this, it will be necessary to design and implement the algorithm that will make use of the filtered data and the triangulation techniques previously studied, having the final objective of measuring the effectiveness of its prediction and results with respect to the real position of the object.

As a final result, an algorithm will be obtained which, based on specific information introduced by means of CSV format files, will carry out the process of geolocating the objects contained in the data, obtaining their position within the route taken by the vehicle. In addition to these coordinates, the algorithm will perform a graphical representation with an aerial view perspective of the scenario and the objects that compose it, representing the position of the vehicle at the moment in which it performed each of the detections of the object, as well as the position predicted by the algorithm of the object and its actual position.

As additional output data, the algorithm will generate CSV files with metrics that measure the accuracy of the prediction by calculating errors comparing the actual position of the object and the one predicted by the algorithm.

RESUMEN

Las técnicas de la visión artificial han evolucionado y han ganado relevancia a lo largo de los últimos años dentro del ámbito tecnológico, implementándose en una gran variedad de campos, siendo uno de estos y el principal en el que será principalmente centrado este trabajo el de la conducción autónoma.

En este proyecto se desarrollará un algoritmo capaz de realizar la detección y la geolocalización de objetos captados desde diferentes cámaras situadas en un vehículo en movimiento, haciendo uso de imágenes tomadas desde diferentes perspectivas, así como de diferentes metadatos que proporcionen la información necesaria para ubicar a los objetos dentro del recorrido realizado por el vehículo.

Para llevar a cabo estos objetivos, será necesario el análisis y filtrado de una base de datos que proporcione toda la información necesaria para el uso del algoritmo, así como el estudio y aplicación de técnicas de triangulación mediante visión artificial para ser utilizadas por el algoritmo. Además de esto, se deberá diseñar e implementar el algoritmo que hará uso de los datos filtrados y las técnicas de triangulación previamente estudiadas, teniendo el objetivo final de medir la eficacia de su predicción y resultados con respecto a la posición real del objeto.

Como resultado final, se obtendrá un algoritmo que en base a una información concreta introducida mediante ficheros en formato CSV, realizará el proceso de geolocalizar los objetos contenidos en los datos obteniendo su posición dentro del recorrido realizado por el vehículo. Además de estas coordenadas, el algoritmo realizará una representación gráfica con una perspectiva de vista aérea del escenario y los objetos que lo componen, representando la posición del vehículo en el momento en el que realizo cada una de las detecciones del objeto, así como la posición predicha por el algoritmo del objeto y su posición real.

Como datos de salida adicionales, el algoritmo generará unos ficheros CSV con métricas que midan la precisión de la predicción mediante el cálculo de errores comparando la posición real del objeto y la predicha por el algoritmo.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ACRONIMOS

- RGPD: Reglamento General de Protección de Datos
- LIDAR: Light Detection and Ranging
- FOV: Field Of View (Campo de visión)
- JSON: JavaScript Object Notation (Notación del objeto de JavaScript)
- CNN: Convolutional Neural Network (Redes de Neuronas Convoluciones)
- GPS: Global Positioning System (Sistema de Posicionamiento Global).
- CSV: Comma Separated Values (Valores separados por comas).

DEFINICIONES

- Dataset: conjunto de datos estructurados.
- Bounding box: caja que delimita el alto y ancho de un determinado objeto ubicado dentro de una imagen.
- Metadatos: son datos encargados de describir otros datos.
- Deep learning: es un tipo de aprendizaje automático que tiene como objetivo el entrenar a un ordenador para realizar tareas concretas, como puede ser el reconocimiento de imágenes.
- Machine learning: es una rama de la inteligencia artificial que tiene como objetivo el desarrollar técnicas para que los computadores aprendan a realizar diferentes procesos de manera autónoma.

1. INTRODUCCIÓN

En los años recientes el campo del computer vision ha ganado una gran relevancia tanto en la detección de diferentes objetos mediante imágenes, como la obtención de la localización de estos, teniendo como caso más reciente la detección del uso de mascarillas en lugares públicos mediante la detección en imágenes debido a la situación dada por el COVID-19. Aunque principalmente destaca su uso en vehículos de conducción autónoma, donde se hace uso de diversas técnicas que permitan localizar y ubicar obstáculos en el escenario para que la inteligencia artificial tome las decisiones correspondientes en base a dicha información.

Dentro del campo del computer vision han surgido técnicas derivadas como es el caso de LIDAR, la cual obtiene la distancia con objetos y superficies respecto al punto de emisión, lo cual ha permitido realizar simulaciones de localizaciones concretas, así como sus superficies y objetos en entornos virtuales o con herramientas gráficas.

Con este trabajo se pretende conseguir la capacidad de geolocalizar determinados objetos dentro de un espacio real concreto de manera automática mediante el uso de técnicas de computer vision. Para llevar a cabo esta tarea, se ha realizado el estudio de diferentes técnicas de localización, como es el caso de la triangulación de objetos estáticos o el cálculo de la profundidad mediante imágenes, así como el estudio en profundidad de una base de datos y el filtrado de su información.

De cara al futuro, el algoritmo de geolocalización desarrollado para este trabajo podrá servir como base para automatizar esta localización en tiempo real mediante el uso de inteligencia artificial, como es el caso de las redes neuronales o las redes de neuronas convolucionales. Teniendo este proceso y su automatización utilidades en diversos campos que se alejan de la conducción autónoma, como es el caso de las recreaciones virtuales de entornos o el añadido de información a herramientas GPS como es el caso de Google Maps mediante la identificación y el posicionamiento de objetos dentro de estos mapas.

1.1 Visión general

El trabajo que realizar se basa en la detección y geolocalización de objetos desde la perspectiva de un vehículo en movimiento, mediante el uso de imágenes y metadatos tomados desde diferentes ángulos en diferentes instantes de tiempo a lo largo de un determinado recorrido. Para llevar esto a cabo es esencial el uso y gestión de una buena base de datos inicial, la cual permita la obtención tanto de imágenes como de sensores que permitan obtener datos de posicionamiento como es el caso de LIDAR. Desarrollando con esta información tanto la detección de objetos como su localización real en el espacio en tres dimensiones mediante la realización de un algoritmo.

Tras valorar diferentes conjuntos de datos se ha optado por hacer uso de nuScenes, el cual cuenta con una gran cantidad y variedad de imágenes tomadas desde diferentes ángulos desde diferentes cámaras, así como hacer uso de herramientas como LIDAR o radares, mediante las

cuales se cuenta con la posición georreferenciada de los diferentes elementos presentes en las imágenes, así como el propio vehículo y sus sensores. Estando estos objetos previamente categorizados en diferentes tipos y ubicados dentro de las imágenes mediante bounding boxes.

Una vez filtrados los datos de nuScenes se procede al estudio de posibles algoritmos a implementar para, mediante los datos obtenidos, calcular o deducir la posición real de un determinado objeto captado mediante imágenes. Con este algoritmo ya definido, se procede con su desarrollo, así como la documentación y la validación y testeo final del mismo.

Finalmente, con el problema una vez terminado, se realiza un análisis del resultado alcanzado, así como las posibilidades de mejora o evolución de cara a futuros proyectos.

1.2 Motivación

Con el crecimiento del sector de la conducción autónoma, las técnicas de computer vision han sido esenciales dentro de este campo, siendo fundamentales para la detección y localización de objetos y para la toma de decisiones por parte del vehículo. Por ello, en el ámbito de la seguridad la detección y ubicación de elementos como vehículos cercanos, peatones o determinadas señales es esencial para la prevención de incidentes dentro de este ámbito. Dentro de este ámbito de la conducción autónoma, podemos encontrar tanto la creciente evolución en el ámbito automovilístico realizado por la empresa Tesla para sus automóviles, como el caso de drones tripulados de manera autónoma en base a una inteligencia artificial y técnicas de computer vision.

Otro campo donde puede aplicarse esta identificación y localización de objetos en el espacio es en el campo de las simulaciones y las recreaciones virtuales de localizaciones o entornos, en las cuales mediante la detección y georreferenciación de elementos concretos pueden simularse ubicaciones de manera digital en base a estos datos mediante el uso de herramientas gráficas como es el caso de Unreal Engine 4.

Este análisis del entorno y la ubicación de sus objetos que deriva en simulaciones y recreaciones de escenarios son aplicables a diversos ámbitos y utilidades:

- **Estudio geográfico:** el hecho de poder recrear lugares de forma digital gracias a imágenes tomadas por drones permite el estudio de la geografía de lugares de difícil acceso para el ser humano, lo que también puede derivar en un estudio histórico si se trata de lugares de difícil acceso con una relevancia histórica.
- **Entretenimiento audiovisual:** esta recreación de escenarios y lugares de forma digital gracias a las técnicas de computer vision puede ser llevada al sector del entretenimiento como puede ser el caso de los videojuegos, concretamente en el ámbito del diseño de escenarios, o el cine en el ámbito de efectos especiales.

- **Mejoras en servicios de navegación:** al poder ser capaces de identificar y localizar objetos dentro de un determinado recorrido, es posible obtener mejoras en los sistemas de navegación como Google Maps gracias a la ubicación de objetos dentro de su variante Google Street View y siendo capaces de identificarlos en su posición dentro del GPS. De este modo, se ofrecería detalles al usuario del posicionamiento de dichos objetos dentro del mapa mostrado, pudiendo aportar información más precisa sobre lugares de interés, señalización de tráfico existente, posibles peligros en la calzada como curvas muy cerradas, etc.

1.3 Objetivos

El objetivo final de este trabajo se basa en la detección y geolocalización de un objeto concreto mediante el uso de un algoritmo capaz de identificar dicho objeto dentro de los pixeles de una imagen, así como el uso de otros metadatos obtenidos mediante sensores, para posteriormente poder determinar su posición concreta con respecto al vehículo del que se obtienen todos los datos.

Para alcanzar este objetivo final, primero deben cumplirse una serie de objetivos principales:

- Realizar un estudio sobre diferentes bases de datos relacionadas para obtener una que proporcione una gran variedad tanto de imágenes como de metadatos que satisfagan las necesidades del trabajo.
- Seleccionar y estimar que datos son necesarios para la realización del trabajo, así como que datos pueden ser útiles para evaluar la eficiencia del algoritmo dentro de la base de datos seleccionada.
- Realizar el filtrado y exportación de los datos considerados previamente para su uso en el algoritmo de geolocalización.
- Estudiar diferentes campos u opciones para elaborar el algoritmo de triangulación y detección de objetos en imágenes, así como compararlas entre sí para obtener una técnica a implementar.
- Elaborar y desarrollar el algoritmo para obtener la posición real del elemento seleccionado con los filtrados previamente.
- Experimentar y testear el correcto funcionamiento desarrollado, así como su estudiar su eficiencia y eficacia en diferentes supuestos.
- Valorar futuras implementaciones o mejoras sobre el trabajo realizado de cara a futuros proyectos.

1.4 Marco regulador

Para la realización de este trabajo es necesario el uso de un conjunto de datos compuestos por imágenes tomadas de diferentes calles y lugares públicos. En estas imágenes es inevitable captar a personas o diferentes vehículos junto con sus matrículas.

Este hecho debe tenerse muy en cuenta, ya que abarca derechos como la privacidad del individuo, por ello, pese a que en el ámbito de la visión artificial no se busque obtener la información de personal, debemos tener en cuenta el marco regulador que abarca este aspecto.

En el caso de España se cuenta con las siguientes leyes relacionadas al uso de los datos privados de un determinado individuo:

- Artículo 18 de la constitución española: "Se garantiza el derecho al honor, a la intimidad personal y familiar y a la propia imagen". [1]
- Artículo séptimo de la Ley Orgánica 1/1982: "La utilización del nombre, de la voz o de la imagen de una persona para fines publicitarios, comerciales o de naturaleza análoga". [2]

Si extrapolamos esto a Europa, se debe destacar el Reglamento Europeo de Protección de Datos (RGPD), en el cual la imagen de una persona es considerada como algo de carácter personal y sujeto a protección. De este modo, se estipula que la persona afectada debe dar estrictamente su consentimiento explicito para el uso de los datos previamente comentados. [3]

Por lo tanto, del marco regulador se puede concluir que el aspecto más relevante para tener en cuenta es el uso de imágenes de personas, así como de las matrículas de sus vehículos, son considerados privados, por lo que se debe de obtener el consentimiento explícito del afectado tanto en España como en Europa. En caso contrario, esta información no podrá ser mostrada ni difundida por ningún medio no autorizado, por lo que deberá ser censurado para mantener el anonimato.

1.5 Entorno socioeconómico

La ubicación en el espacio real de objetos mediante imágenes o video es un campo con un gran crecimiento en los últimos años, siendo utilizadas estas aplicaciones de la visión artificial en múltiples campos:

- Recreación y simulación de ubicaciones: recreación digital de un lugar concreto, mediante la captación de dicho lugar mediante imágenes.
- Conducción autónoma: detección y posicionamiento de objetos con respecto al vehículo autónomo.

Este proyecto parte del objetivo de mejorar y facilitar la geolocalización de objetos mediante el uso de imágenes para mejorar el desarrollo de la visión artificial y los campos en la que esta

es utilizada. Esta mejora puede tener un impacto positivo o negativo en función de la manera que sea llevada, derivando un uso poco ético en provocar graves problemas en la privacidad de los individuos, así como posibles accidentes fatales en el ámbito de la conducción autónoma.

Como contraparte a las consecuencias de un mal uso previamente mencionadas, tenemos como beneficios de un uso ético, la potenciación en seguridad de vehículos autónomos, reduciendo al mínimo el número de accidentes y posibles víctimas. Además de esto, se conseguiría una mejora en el desarrollo de simulaciones virtuales de ubicaciones, las cuales varían desde el estudio histórico, hasta casos como la industria del entretenimiento con el uso de estas técnicas para videojuegos, series o películas.

1.6 Estructura del documento

Para tener una visión global de los contenidos que abarca este documento, así como de su distribución y estructura, a continuación, se describe mediante puntos la estructura de contenidos:

- **Introducción:** en este bloque inicial se realiza un primer acercamiento al trabajo de forma general, al desarrollo realizado del mismo y a las fases principales que abarca el mismo, así como una definición de objetivos, motivaciones y su impacto tanto en el marco regulador como en el entorno socioeconómico.

- **Estado del arte:** en este apartado se presenta más en profundidad el problema a resolver durante el desarrollo, además de realizar una explicación de los conceptos y conocimientos esenciales para la comprensión de este. Así como una exposición de los recursos externos que se han utilizado como base para la realización del trabajo.

- **Desarrollo de la solución:** en este apartado se documentará en detalle las fases del desarrollo de la solución, las cuales se basan en el planteamiento inicial de la misma, así como el diseño establecido para el algoritmo.

- **Implementación de la solución:** en este bloque se exponen los pilares fundamentales de la implementación del algoritmo, así como las herramientas utilizadas para llevar a cabo esta implementación. Además, se detalla el funcionamiento del algoritmo y sus diferentes fases.

- **Planificación y presupuesto:** dentro de este bloque se expone la planificación del trabajo a realizar durante su periodo de desarrollo, así como el presupuesto necesario para la elaboración de este.

- **Conclusiones y trabajos futuros:** en este apartado se exponen unas conclusiones para el proyecto, así como mejoras y posibles nuevas rutas a seguir partiendo del algoritmo realizado, todo ello de cara a implementaciones o mejoras de cara al futuro.

2. ESTADO DEL ARTE

A lo largo de este bloque se realiza una explicación completa sobre las técnicas necesarias para la realización del trabajo, así como la materia sobre los campos en los que se encuentran, todo ello con el objetivo de obtener una mejor comprensión del funcionamiento básico de dichas técnicas. En concreto, esta sección se centra dar una visión generalizada del uso de técnicas de computer vision combinadas con el uso de técnicas de triangulación y cálculo de profundidad para la detección y la georreferenciación de objetos.

Además de los aspectos comentados, se introducen ciertos conceptos sobre el uso de la inteligencia artificial en este campo del computer vision, todo ello visto como una perspectiva de cara a aplicaciones en futuros proyectos.

Tras exponer la explicación de estos campos se hará una mención a trabajos previos que comparten similitudes con el trabajo realizado y que han servido de base o inspiración con este.

2.1 Computer vision

El computer vision o visión artificial se basa en la detección automática de elementos u objetos, así como sus propiedades en el mundo tridimensional, a partir de una o varias imágenes tomadas del entorno, llegando a ser estas propiedades dinámicas en diversos casos. Este campo abarca diversos ámbitos físicos como es el caso de la óptica o fotogrametría, además de ámbitos matemáticos como la geometría, la triangulación o la estadística, entre otros.

La visión artificial tiene como objetivo el desarrollo de una metodología automatizada que permita el reconocimiento de patrones complejos dentro de diferentes tipos de imágenes, metodologías de las cuales hacen uso diversos campos como el reconocimiento de objetos o la recreación de escenas. Esta automatización se da mediante el uso de diferentes técnicas de aprendizaje automático, las cuales se encargan de diferenciar los patrones captados mediante el uso de algoritmos matemáticos de manera automática, existiendo dos tipos de aprendizaje:

- **Aprendizaje supervisado:** se basa en realizar un entrenamiento del algoritmo mediante el planteamiento de diferentes incógnitas de las cuales se conoce la respuesta, combinando ambos factores para que el algoritmo realice una determinada predicción.

  Existen dos tipos de aprendizaje supervisado:

  - Regresión: obtiene un resultado numérico, basándose este modelo en minimizar el error que mide la diferencia entre la salida deseada y la obtenida de todos los patrones mediante una aproximación lineal.

  - Clasificación: este tipo de aprendizaje supervisado se basa en realizar una asignación o añadir una etiqueta a diferentes funciones en función de sus características, pudiendo obtener resultados no numéricos o binarios.

- **Aprendizaje no supervisado:** en este tipo de aprendizaje no se cuenta con un conocimiento previo de la solución a diferencia del aprendizaje supervisado, teniendo en este caso el objetivo de agrupar los patrones en diferentes grupos según sus características comunes.

[4]

2.1.1 Detección de objetos

La detección de objetos dentro del ámbito del computer vision se basa en la detección de un determinado objeto o elemento dentro de una imagen en base a su apariencia visual y en la posición que esta ocupa dentro de dicha imagen. Esta detección se da asociando unas características determinadas del objeto para su identificación pudiendo ser dadas previamente o siendo extraídas con modelos matemáticos que definen dichos objetos.

Estas características que definen los objetos en una imagen pueden ser de diversos tipos, como es el caso de histogramas de coloración, la intensidad de la luz o tipos más complejos como es el caso de AdaBoost, el cual se basa en técnicas de aprendizaje automático para la identificación de características.

Para la localización de los diferentes tipos de objetos dentro de la imagen se hace uso de bounding boxes, las cuales delimitan el espacio de la imagen que abarcan los objetos a identificar, teniendo la medida de su alto y ancho en pixeles.

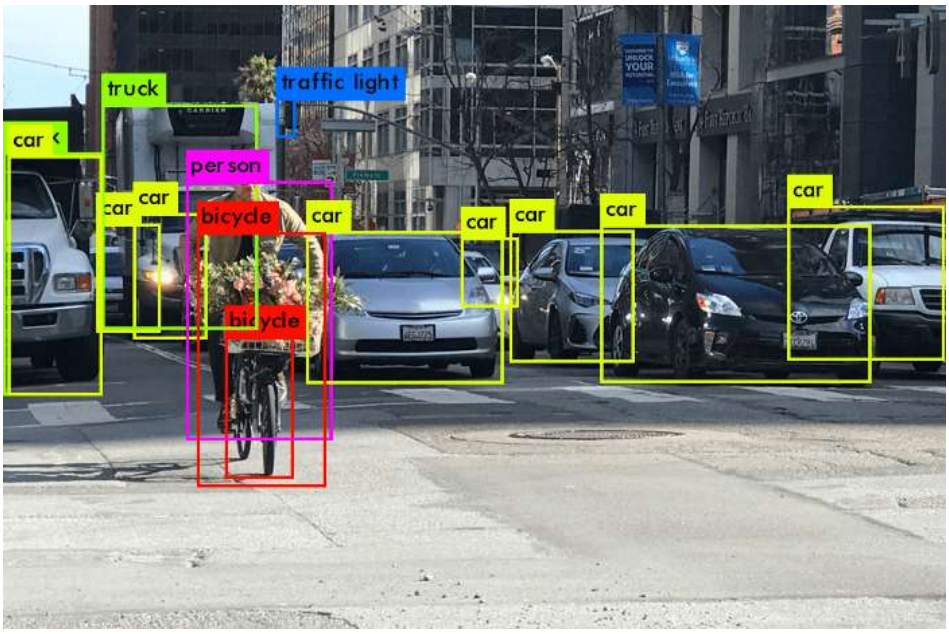

Figura 1: Detección de objetos en una imagen mediante bounding boxes.

[5]

2.1.2 Algoritmos para la detección de objetos en imágenes

Existen múltiples algoritmos para automatizar este proceso de identificación de objetos, los cuales generalmente utilizan técnicas de Deep Learning, mediante las cuales se generan redes entrenadas capaces de rastrea el objeto a identificar en la imagen teniendo como base imágenes de ejemplo del objeto a localizar. La técnica de Deep Learning más utilizada para la detección de objetos es imágenes son las redes de neuronas convolucionales (CNN), las cuales se explicarán con más detalle en los siguientes apartados.

En lo referido al proceso de detección, este se basa en analizar la imagen mediante el uso de ventanas generadas de un determinado tamaño, las cuales van desde un extremo a otro de la imagen, analizando esta mediante estas segmentaciones hasta reconocer los diferentes objetos. [6]

Por lo tanto, el proceso de detección en términos generales se da mediante el siguiente procedimiento:

Figura 2: Esquema del proceso general de detección de objetos en imágenes

[7]

Sliding Windows

Para el escaneo de una imagen presente en el primer punto del proceso de detección de objetos, podemos hacer uso del algoritmo Sliding Windows, el cual consiste en ir deslizando una ventana de tamaño fijo a lo largo de la imagen hasta recorrer a esta por completo.

El tamaño de la ventana de tamaño fijo varía en función del objeto que se esté intentando encontrar en la imagen, por ejemplo, si se trata de un vehículo se deberá hacer uso de una ventana con una forma rectangular, contando con mayor anchura y menos altura, sin embargo, si el objeto a analizar se trata de un árbol, se deberá hacer uso de una ventana rectangular con una menor anchura y una mayor altura.

En lo referido al deslizamiento de la ventana, la imagen se dividirá de columnas y filas de las cuales se deberá determinar el tamaño y el paso de desplazamiento, este último valor será el que defina cada cuantas filas y columnas se analizará la ventana de la imagen. Esto afecta directamente a aspectos como el coste computacional, ya que cuanto menor sea el paso de desplazamiento más análisis se realizarán en un espacio concreto de la imagen, por lo que el coste aumentará, sin embargo, si el valor del paso de desplazamiento es alto, puede realizarse un análisis muy pobre con un bajo coste computacional, por lo que se debe fijar un valor óptimo en función de la imagen. [8]

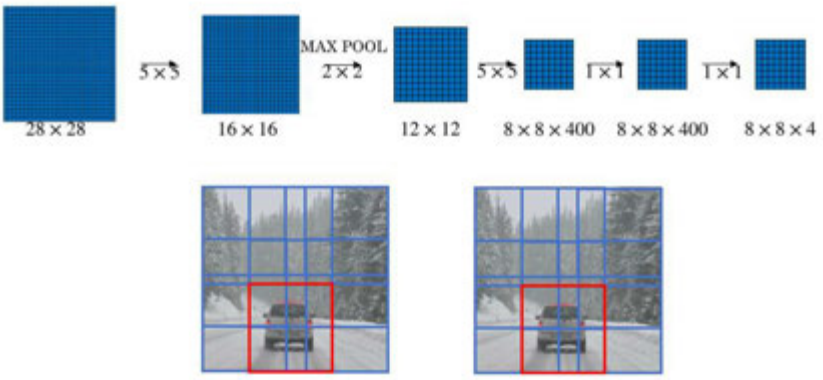

Figura 3: Aplicación del algoritmo Sliding Windows en imágenes

[9]

Pirámide Gaussiana

En el algoritmo anterior se ha mostrado un procedimiento a seguir para realizar un análisis con un tamaño fijo de ventana, pero también se deben tener en cuenta los casos en los que dos objetos de la misma clase cuentan con un tamaño diferente en la misma imagen, como podría ser el caso de dos objetos a diferente distancia de la cámara que toma la imagen, lo que supondría un tamaño de ventana variable y no fijo. Para poder llevar a cabo este objetivo, se aplica un reescalado de la imagen manteniendo el tamaño de la ventana fijo en su tamaño, para no perder información se realiza un preprocesado de la imagen de forma previa al reescalado, donde, para reducir la imagen a la mitad, se aplica un filtro gaussiano a cada uno de los pixeles que componen la imagen para difuminarla, tras esto se tomará una de cada dos columnas de la tabla establecida para la imagen (esta tabla se corresponde con la comentada en el algoritmo Sliding Windows).

La metodología de la Pirámide Gaussiana se basa en realizar una reducción iterativa del número de píxeles presentes en la imagen, proceso que se realizará un número de *n* veces, desde el tamaño original de la imagen hasta la reducción *n*. [10]

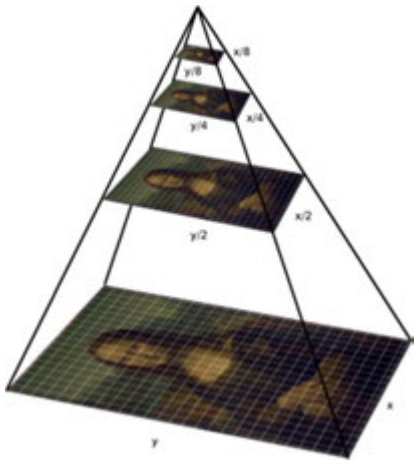

Figura 4: Representación de la Pirámide Gaussiana

[11]

Una vez aplicada la Pirámide Gaussiana, a cada una de estas imágenes reducidas se le aplicará el algoritmo Sliding Windows, obteniendo las diferentes ventanas en versiones reescaladas de la imagen, contando con ventanas de diferente tamaño en la imagen original. Sin embargo, se darán numerosos casos donde estas detecciones estén solapadas entre sí por bounding boxes de diferente tamaño alrededor de un objeto. Debido a este problema, debemos hacer uso de un algoritmo clasificador que filtre estas bounding boxes superpuestas para obtener la predicción más acertada que se haya realizado de la posición del objeto.

Non-maximum Suppression (NMS)

Para dar solución a la problemática anteriormente comentada se debe hacer uso del algoritmo NMS, el cual tiene como objetivo realizar un filtrado entre la totalidad de las bounding boxes superpuestas para un mismo objeto, este filtrado se da en función de cuales de estas se solapan y que grado lo hacen, obteniendo de este modo la bounding box con mayor probabilidad de predicción por cada objeto.

Figura 5: Aplicación del algoritmo NMS en imágenes

Entrando más en profundidad en el funcionamiento del algoritmo, este se basa en recibir un array de bounding boxes, los cuales contienen una determinada probabilidad de pertenecer a la clase de objeto a detectar. De esta entrada se calcula el solapamiento de una ventana A y otra ventana B mediante la siguiente fórmula:

$$solapamiento = \frac{\text{area}(A \cap B)}{\text{area}(A \cup B)}$$

El algoritmo NMS a la hora de fijar cual ventana entre las ventanas A y B debe descartarse, tiene en cuenta la probabilidad de cada ventana a pertenecer a una clase de objeto, de este modo la que tenga una menor probabilidad será descartada y se quedará únicamente con la de mayor probabilidad. Este proceso se realizará para cada una de las ventanas solapadas hasta que quede una única ventana.

[12]

Proceso de detección y clasificación de objetos completo

Con la aplicación de los tres algoritmos previamente comentados, podemos realizar un proceso de detección y clasificación de objetos eficiente, sin embargo, además de estos algoritmos serán necesario estructurar una red de neuronas convolucional y contar con un conjunto de datos para entrenar a dicha red.

Este conjunto de datos contará de imágenes concretas de los objetos, preferentemente imágenes que abarquen diferentes perspectivas de este, y un conjunto de imágenes donde los objetos se encuentren en algún lugar situados para su detección. Estos datos se dividirán en un conjunto de entrenamiento (imágenes donde aparece exclusivamente el objeto a identificar) para entrenar la red y un conjunto de validación para validar el entrenamiento realizado (imágenes donde aparecen los objetos sobre diferentes escenas).

En cuanto al proceso completo, este se compone de las siguientes fases y procedimientos:

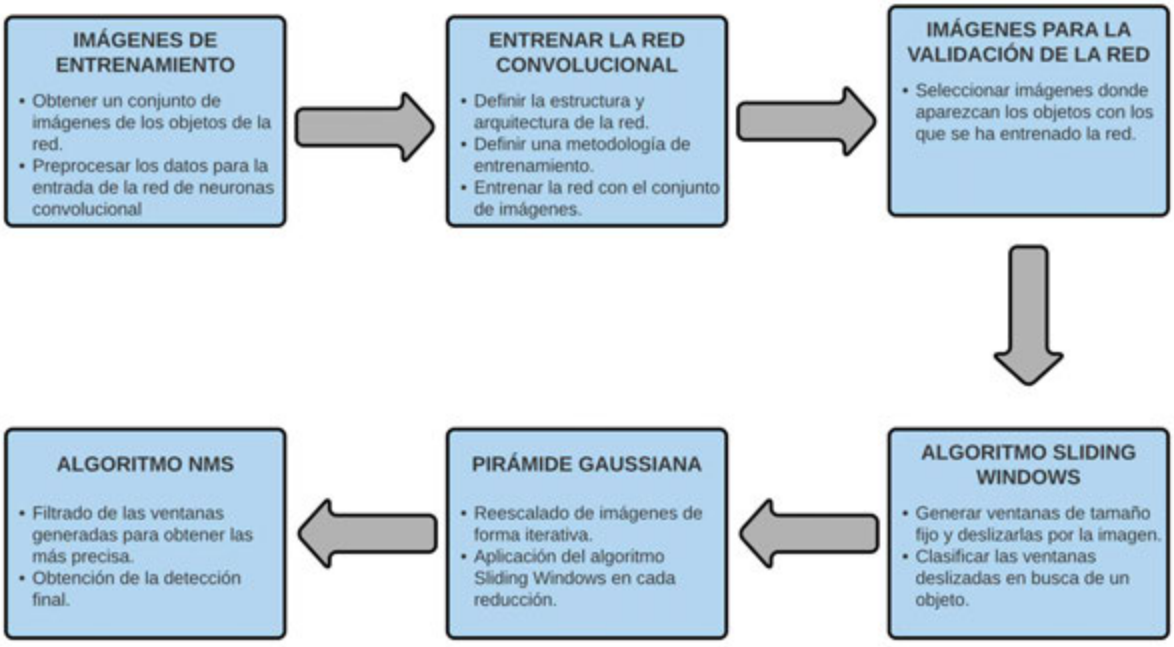

Figura 6: Esquema de la aplicación de algoritmos en la detección de objetos

[7]

2.1.3 Visión en 3D

Además de la localización y detección de objetos dentro de imágenes, también existe la visión artificial dentro de un marco de tres dimensiones, la cual hace uso de diferentes sensores 3D para capturar información en este plano de una determinada escena, dependiendo la obtención y el tipo de información en función del tipo utilizado, siendo los más destacados los siguientes:

- **Nubes de puntos:** su información se basa en las coordenadas de los diferentes puntos presentes en la nube, siendo estas coordenadas expresadas en un plano tridimensional en coordenadas XYZ y teniendo como referencia inicial en el eje de coordenadas el propio sensor.
- **Imagen de profundidad:** la información se da en la propia imagen, donde cada píxel almacena la distancia al centro óptico de la cámara usada, además de contar con la imagen color o a escala de grises.
- **Malla poligonal:** este tipo de información es similar a la nube de puntos, donde el conjunto de puntos está estructurado mediante forma poligonales, generalmente triángulos. Este tipo de formato es generalmente usado en sistemas de triangulación mediante láser.
- **Imagen de rango:** se basa en un conjunto de tres imágenes de profundidad en base a los tres ejes cartesianos XYZ. Este tipo de formato suele darse en sistemas de proyección por luz estructurada.
- **Imagen estereoscópica:** este formato se basa en el uso de dos imágenes a color, correspondiendo cada una a un sensor diferente. Este tipo de formato se da en los sistemas estereoscópicos y suele ser usado para la visualización de escenas en 3D.

[13]

2.1.4 LIDAR

LIDAR es un sensor laser cuya finalidad es la de medir la distancia desde si mismo hacia un punto determinado, esto lo hace mediante los láseres que emite, los cuales le sirven para medir la distancia desde la emisión del punto de luz hasta el lugar donde este impacta, pudiendo obtener recreaciones en tres dimensiones de lugares o superficies concretas en base a los datos proporcionados mediante esta tecnología.

El funcionamiento del cálculo de la profundidad mediante láseres LIDAR se basa en el lanzamiento de rayos de luz a determinadas superficies, obteniendo el rebote de dicho rayo de luz con el sensor, mediante el cual se obtendrá la distancia del objeto en cuestión en función del tiempo que esta tarde en llegar. Además de esto, en función de la inclinación con la que llegue el rebote de luz puede determinarse el ángulo de la superficie del objeto.

Estos rayos de láser generalmente se dan de manera masiva, generando nubes de puntos que determinan con mayor precisión la distancia y la forma de los diferentes objetos presentes, siendo mucho más fiel un mapeo o representación cuantos más puntos se encuentren en la nube de puntos utilizada.

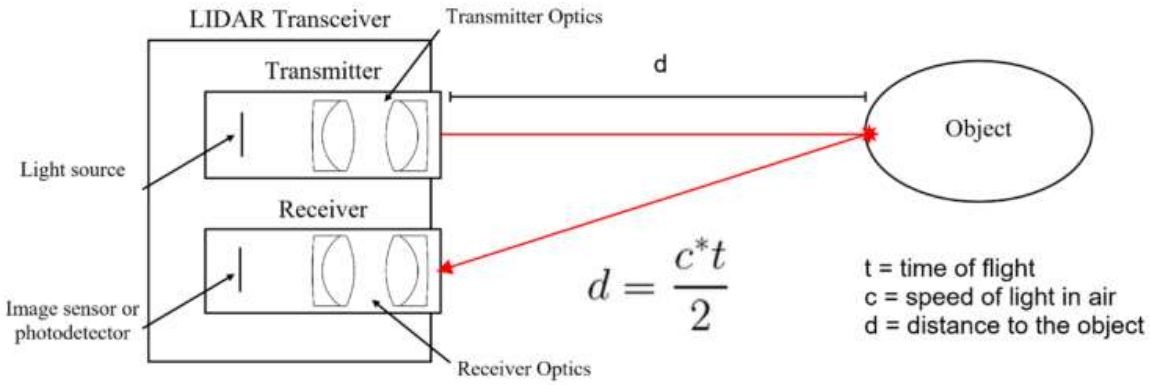

Figura 7: Funcionamiento de un sensor LIDAR en un vehículo autónomo.

[14]

En lo referido a las aplicaciones de este tipo de tecnología, estas son diversas:

- **Mapeo de superficies en 3D:** permite la recreación en tres dimensiones de manera digital de diferentes escenas o superficies, siendo de utilidad para analizar lugares difíciles de alcanzar para el ser humano. Además de esto, esta aplicación permite una recreación de lugares de gran antigüedad para su estudio histórico o arqueológico o el mapeo de zonas forestales para la medición de estas, entre otros.

- **Datos atmosféricos:** mediante este tipo de tecnología se pueden obtener valiosos datos de la atmósfera, como es el hecho de las densidades de los materiales que se encuentran en ella.

- **Modelos topográficos:** la tecnología LIDAR permite la obtención de modelos topográficos de gran detalle, los cuales son utilizados para estudiar el cómo se producen fenómenos naturales como terremotos o erupciones volcánicas, entre otros.

- **Conducción autónoma:** dentro del campo de la conducción autónoma LIDAR es utilizado en la localización de los diferentes objetos que rodean al coche con respecto a este, lo que permite tanto una mejora de la propia movilidad del vehículo como la prevención de accidentes mediante la detección de la distancia con un posible peatón.

[15]

2.1.5 Triangulación de un objeto mediante visión artificial

La triangulación de un objeto mediante el uso de computer vision se basa en la conversión de un determinado punto en una imagen 2D al espacio de las tres dimensiones mediante el uso de dos imágenes tomadas desde diferentes ángulos. Para la obtención de un punto X dentro del espacio en tres dimensiones necesitan conocerse previamente las posiciones de las dos cámaras que han captado las dos imágenes (C1 y C2) y la correspondencia del punto X dentro del espacio en dos dimensiones dentro de las dos imágenes (X1 y X2), de este modo se hace uso de las cámaras y los respectivos puntos de la imagen para obtener los vectores en tres dimensión

que atraviesan las imágenes y cuya intersección resulta en el punto X dentro del plano de las tres dimensiones (L1 y L2).

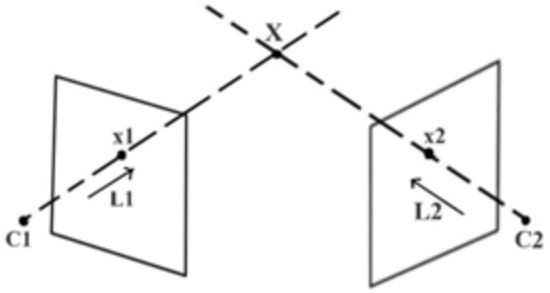

Figura 8: Triangulación de un punto en el espacio 3D mediante el uso de dos imágenes.

[16]

Para llevar a cabo la triangulación de un objeto es necesario conocer el concepto de geometría epipolar, mediante la cual se obtendrán las relaciones geométricas entre los puntos captados en las dos imágenes 2D y la ubicación de los puntos en el espacio real 3D.

La geometría epipolar es la geometría de la visión estéreo, en la cual se obtienen las distintas relaciones geométricas entre los puntos en el espacio 3D de uno o varios objetos y sus proyecciones en las imágenes 2D tomadas por dos cámaras situadas enfocando desde un determinado ángulo.

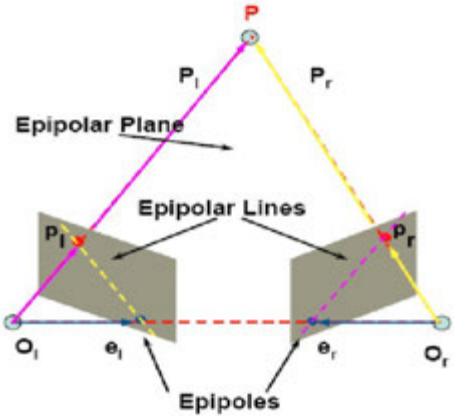

Figura 9: Aplicación de la geometría epipolar para la triangulación de un punto.

[17]

Como se puede observar en la imagen anterior, cada una de las imágenes tomadas abarca un determinado campo de visión, dentro del cual se localizan diferentes puntos (X_L, X_R, e_L, e_R), los cuales dentro de una imagen se encuentran ubicados en una posición distorsionada con respecto a la otra imagen debido a su representación en 2D, siendo X_L y X_R el mismo punto del espacio visto desde dos imágenes distintas. Mediante el conocimiento de esta distorsión, podemos extraer los datos necesarios para generar una recta que atraviese la imagen, derivando en la localización del punto en el espacio real gracias a la intersección de las rectas creadas en ambas imágenes.

2.2 Redes de neuronas artificiales

Dentro del campo del Deep Learning comentado en apartados previos tenemos las redes de neuronas artificiales, las cuales son modelos matemáticos inspirados en la biología, cuyo objetivo es el intentar emular el comportamiento de las redes de neuronas de un cerebro, aprendiendo mediante sus capas de neuronas de la experiencia. Cada red de neuronas cuenta con unidades de procesamiento denominadas neuronas, las cuales se encargan de recibir valores numéricos como parámetros de entrada denominados inputs y genera un valor de salida llamado output.

El funcionamiento de una red de neuronas se basa en procesar unas determinadas entradas o inputs, las cuales pueden ser provenientes de los patrones de entrenamiento o de otra neurona de la red, y obtener una salida u output mediante la aplicación de los diferentes pesos asignados a cada una de las conexiones, sumando las multiplicaciones de pesos e inputs en cada una de estas neuronas hasta llegar a la capa final y obtener la salida de la red.

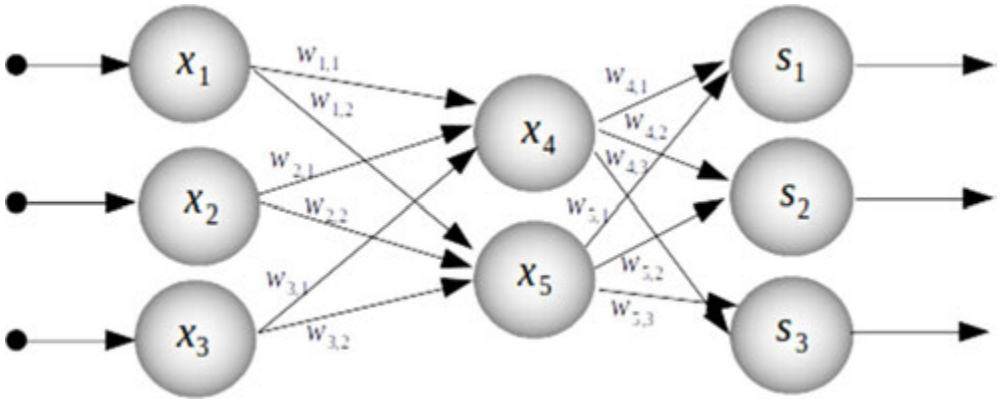

Figura 10: Esquema de una Red de Neuronas Artificial.

[18]

Una vez obtenidos los valores de salida de la red, estos inicialmente son aleatorios y deben ser ajustados al valor esperado del problema, para ello es necesario calcular esta diferencia entre el valor esperado y el obtenido como salida mediante el cálculo del error cuadrático medio o el error absoluto medio. El objetivo de la red será el minimizar el error mediante el ajuste de pesos, entrenando de este modo la red hasta alcanzar una solución óptima.

En lo referido al ajuste de pesos, este se da mediante el algoritmo de retropropagación el cual se basa en calcular el error de la salida de la última capa de neuronas de la red e ir propagando dicho error a cada una de las capas de la red hasta la capa de entrada, ajustando los nuevos pesos en función del error.

[19]

2.2.1 Redes de neuronas convolucionales

Las redes de neuronas convolucionales son un tipo de red de neuronas artificial cuyo input se basa únicamente en imágenes, contando como valores todos y cada uno de los pixeles que conforman la imagen a los cuales la propia red asigna unos pesos para determinar la relevancia de los aspectos de la imagen que permite la diferenciación de objetos. Debido a este uso de imágenes este tipo de redes son aplicadas a campos como la visión artificial y la clasificación de imágenes u objetos.

La estructura de la red consiste en el uso de varias capas convolucionales, las cuales funcionan como un filtro de diferentes dimensiones, es en estas capas donde se extraen las características de la imagen. Una vez extraídas estas características son pasadas como entrada a un perceptrón que se encuentra al final de estas capas convolucionales, mediante las neuronas de este perceptrón se realiza una clasificación de la imagen en base a las características extraídas.

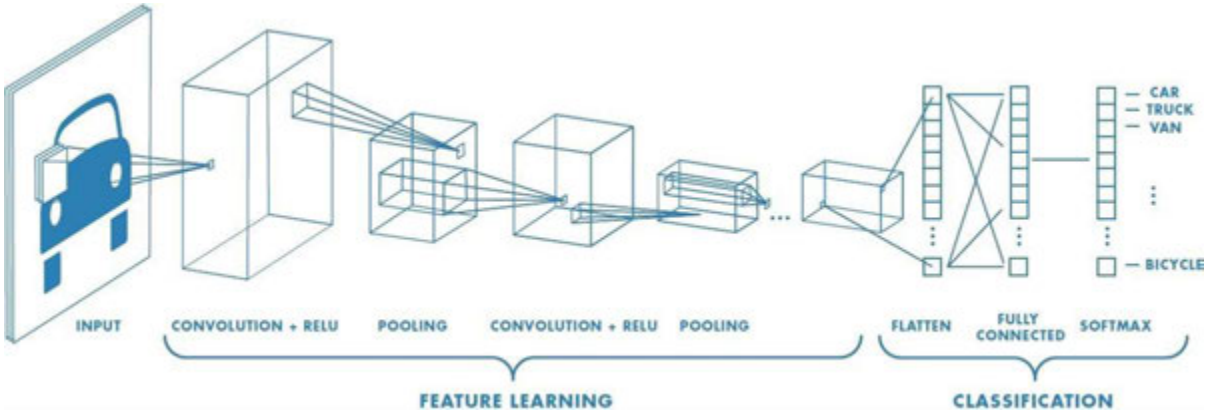

Figura 11: Esquema de una Red de Neuronas Convolucional.

[20]

Cada una de las capas convolucionales que actúan en la red cuentan con dos atributos fundamentales:

- **Número de filtros:** determina el número de filtros a usar dentro de la capa.
- **Número de canales de entrada y canales de salida:** se encargan de determinar el tamaño de filtro, debiendo ser igual el número de canales de entrada y el número de canales de salida de la capa.

En lo referido a la eficacia, las redes convolucionales suponen una alternativa muy eficaz en comparación con otros algoritmos de clasificación de imágenes, esto es debido al uso de los filtros, los cuales son optimizados durante la red durante el aprendizaje, al contrario que en otros clasificadores donde estos son diseñados a mano.

[21]

2.3 Sistemas de referencia y coordenadas

A la hora de trabajar con la posición de un objeto dentro del espacio 3D, se debe tener en cuenta los principales sistemas de referencia puntuales con respecto a este objeto para determinar sus coordenadas relativas, así como establecer unos puntos de referencia globales para ubicar a todos los objetos presentes en la escena bajo las mismas coordenadas globales.

En lo referido a las coordenadas relativas, estas se dan generalmente en el ámbito de la conducción autónoma teniendo como punto de referencia el vehículo que capta las imágenes y datos, siendo el eje 0 en el espacio 3D.

En cuanto a las coordenadas globales dentro de los sistemas de georreferenciación global se dan por una métrica establecida por el propio dataset, como podría ser un circuito generado en el recorrido o un mapa con unas determinadas coordenadas.

Dentro del ámbito de las coordenadas globales puede hacerse uso del sistema de coordenadas WGS84, el cual se basa en un sistema de referencia terrestre que hace uso de unas coordenadas geográficas usadas mundialmente para permitir localizar cualquier punto de la Tierra sin necesidad de un punto de referencia específico, de este modo se elimina la necesidad de contar con una referencia concreta para ubicar los diferentes objetos dentro de un mismo escenario.

[22]

2.4 Trabajos o proyectos similares

Dentro de este apartado, se recogen los proyectos que han servido como inspiración para el trabajo realizado, así como una comparativa entre estos proyectos valorando las diferentes características con las que cuentan cada uno de ellos, para finalmente elegir cuál de estas inspiraciones usar como base para el diseño de la solución.

2.4.1 Mapeo 3D de árboles en Hong Kong utilizando Google Street View

Este proyecto tiene como objetivo el realizar un mapeo con el posicionamiento de los árboles situados por las calles alrededor de la ciudad de Hong Kong, para ello se hace uso de imágenes tomadas mediante Google Street View en las cuales, tras una conversión a una imagen plana de 360º, se hace uso de una red de neuronas convolucional se obtiene el reconocimiento de los árboles en las diferentes imágenes.

Una vez se obtienen estos datos, se realizan técnicas de cálculo de profundidad en los diferentes objetos detectados mediante el FOV con el que cuentan las cámaras, estimando a que distancia se encuentra cada objeto de ellas. Tras obtener estas distancias se puede establecer la posición de cada árbol en las diferentes calles vía GPS.

Michael Li y Wei Yao et al. [23]

2.4.2 Uso de Deep Learning para identificar y localizar postes de luz mediante imágenes de Google Street View

Este trabajo tiene como finalidad el obtener el realizar la detección y geolocalización de los postes de luz situados a lo largo de diferentes carreteras, para ello mediante se hace uso de herramientas como Google Street View y las redes de neuronas convolucionales, mediante la cual se obtiene la detección de los diferentes postes de luz en imágenes.

En lo referido a la predicción de la posición de los postes, esta se da mediante la triangulación de su posición en base a imágenes tomadas desde diferente perspectiva a cada uno de los objetos a localizar, en cada una de estas imágenes se obtendrá mediante el uso del FOV de la cámara una recta que va desde la posición de la cámara hasta la mitad de la bounding box del objeto. Esta línea primero se realiza dentro de las imágenes 2D para posteriormente extrapolarla a una vista de águila para obtener la posición del objeto en el espacio 3D.

Weixing Zhang, Chandi Witharana, Weidong Li, Chuanrong Zhang, Xiaojiang Li y Jason Parent et al. [24]

2.4.3 Clasificación de objetos usando imágenes aéreas y a vista de calle

En este proyecto se tiene como finalidad el detectar y ubicar objetos mediante el uso de imágenes desde una perspectiva aérea y de calle, para ello se hace uso de una Red de Neuronas encargada de realizar la clasificación y de un conjunto de imágenes de diferentes zonas con las dos perspectivas previamente comentadas. Además de estos datos, también se hace uso de un mapa de vista aérea (GPS) de las zonas donde se capten las imágenes para poder posicionar los objetos detectados a lo largo de dicho mapa.

En lo referido a la predicción del posicionamiento de los objetos, se hace uso de metadatos como el posicionamiento en el coche dentro del mapa GPS, además de utilizar imágenes de 360º. Con estos datos se triangula la posición de los objetos mediante la aplicación de funciones trigonométricas en base al ángulo de posicionamiento del vehículo y la geometría de la propia calle con respecto al coche y al objeto.

Jan D. Wegner, Steve Branson, David Hall, Konrad Schindler y Pietro Perona et al. [25]

2.4.4 Descubrimiento automático y geoetiquetado de objetos a partir de imágenes de Street View

Dentro de este proyecto, se tiene el objetivo de detectar, etiquetar y geolocalizar en el mapa por GPS diferentes objetos de manera automática mediante el uso de técnicas de computer vision, imágenes tomadas por Google Street View, y el uso de una IA que sea capaz de identificar y clasificar los objetos presentes en estas imágenes.

Para llevar a cabo este objetivo, se contemplan tanto técnicas de triangulación mediante el uso de imágenes tomadas desde diferente perspectiva por varias cámaras, como el cálculo de

profundidad de una única imagen mediante el uso de redes de neuronas convoluciones. De este modo, se comparan ambos métodos de detección y geolocalización comparando las ventajas e inconvenientes presentados por ambas en sus resultados.

Vladimir A. Krylov, Eamonn Kenny, Rozenn Dahyot et al. [26]

2.4.5 Comparativa de cualidades entre los proyectos

A continuación, se realiza una comparativa entra las cualidades presentes en cada uno de los proyectos previamente comentados, mediante esta comparativa se realizará la decisión sobre qué proyecto será utilizado como base para el diseño del algoritmo de geolocalización a realizar.

Proyecto	Necesita imágenes en 360º	Necesita imágenes desde distintos ángulos	Triangulación mediante el uso de diferentes imágenes	Cálculo de profundidad en base a una única imagen	Generalizable a otros datasets
2.4.1	✓			✓	✓
2.4.2		✓	✓		✓
2.4.3	✓		✓		
2.4.4		✓	✓	✓	

Tabla 1: Comparativa entre los proyectos de los recursos externos utilizados

Debido a las similitudes con el trabajo que se pretende desarrollar y a sus cualidades, el segundo proyecto será el que se utilice como base para el planteamiento y el desarrollo del algoritmo de triangulación, estableciendo la generación de las rectas mediante el uso del FOV de la cámara, la bounding box detectada del objeto en la imagen y el posicionamiento del vehículo. De estos datos se generarán rectas para cada una de las imágenes tomadas desde diferente perspectiva de un determinado objeto, lo que supondrá obtención de la posición estimada en el punto donde interactúen dichas rectas.

2.4 Datasets tenidos en cuenta para el desarrollo del trabajo

A continuación, se exponen los diferentes datasets que se han valorado para el desarrollo del algoritmo, así como sus pros y contras para establecer una elección de conjunto de datos con el que partir para el desarrollo de este trabajo:

- **NuScenes:** es un dataset de uso público y no comercial, el cual está centrado en la conducción autónoma por el equipo Motional. El dataset cuenta con un gran número de imágenes tomadas desde un total de 6 cámaras situadas a lo largo del vehículo, así como metadatos tomados por un total de 5 radares y un sensor LIDAR.

  De esta forma el dataset cuenta con una gran cantidad de escenas captadas por estos sensores y cámaras, estructurando la información mediante el uso de tokens que relacionan la escena, el fotograma concreto y el sensor con el que fue tomado dicho dato o imagen. Además de esto el dataset cuenta con diferentes versiones de menor o mayor tamaño para su descarga y manejo. [27]

- **Pandaset:** este dataset contiene una cantidad de datos muy completa, contando con diferentes imágenes tomadas desde diferentes cámaras, así como diferentes metadatos obtenidos mediante un sensor LIDAR. Sin embargo, cuenta con una única versión de descarga con un peso de 100 GB, así como una estructuración de los datos poco intuitiva entre sus carpetas y ficheros. [28]

- **Kitti-360:** este dataset cuenta con diferentes imágenes y metadatos gracias al uso de una única cámara y un sensor LIDAR en el vehículo. Además de esto, dentro de las imágenes, se encuentra la aplicación de diferentes filtros a estas, estando orientado este dataset al cálculo de profundidad mediante imágenes.

  La información de este dataset se encuentra estructurada en función del entorno donde se capta la escena, pudiendo ser city, residential, road, campus, etc. Sin embargo, pese a una estructuración que permite la descarga de paquetes de datos de diferente tamaño, el número de imágenes contenido en cada uno de estos es escasa a comparación con el resto de datasets. [29]

- **Waymo open:** en este dataset se encuentra almacenada una gran cantidad de información, la cual tiene un tamaño aproximado de 2 TB, contando con diferentes versiones que dividen esta información en conjuntos de entrenamiento y conjuntos de test. Dentro de esta información se encuentran almacenados tanto imágenes como metadatos recopilados mediante un sensor LIDAR.

  Pese a ser un dataset prometedor cuenta con la desventaja de contar con bloques de información muy pesados, pese a estar divididos en entrenamiento y test, lo que dificulta el manejo y trabajo de la información. Además, a este hecho le debemos sumar la falta de documentación sobre el dataset y la ausencia de un kit de desarrollo. [30]

Dataset	Contiene la información necesaria para el trabajo	Cuenta con diferentes versiones de tamaño reducido	Cuenta con una información dividida y estructurada	Hace uso de diferentes cámaras y sensores	Cuenta con un kit de desarrollo
NuScenes	✓	✓	✓	✓	✓
Pandaset	✓			✓	✓
Kitti-360	✓	✓	✓		✓
Waymo Open	✓	✓		✓	

Tabla 2: Comparativa entre los datasets considerados para el proyecto

Es gracias a este estudio y comparación entre los diferentes datasets tenidos en cuenta para usar en el desarrollo del trabajo, que se puede concluir que nuScenes es el dataset más apto para las necesidades que requiere este trabajo. Por lo tanto, para el desarrollo del algoritmo se hará uso de la información contenida dentro de este dataset, realizando un estudio previo de estos datos y filtrando la información que sea necesaria.

3. DESARROLLO DE LA SOLUCIÓN

3.1 Introducción

Con este trabajo se pretende obtener una solución que permita la detección de un determinado objeto captado mediante imágenes, así como su localización el espacio real en tres dimensiones en base a las diferentes imágenes tomadas desde distintas posiciones de este.

En primer lugar, es necesario contar con un buen dataset que cumpla con las necesidades del proyecto para el correcto desarrollo tanto de la detección de objetos como del algoritmo de triangulación, así como la validación o medición de la eficacia de estos. El dataset utilizado en el proyecto es nuScenes, el cual deberá ser filtrado para obtener la información de entrada deseada para el algoritmo de triangulación, así como trabajar las imágenes contenidas para obtener la localización en imágenes de los objetos a triangular.

Una vez se cuente con el conjunto de datos filtrados, se deberá implementar un algoritmo de triangulación capaz de reconocer dichos datos como parámetros de entrada y manejarlos para obtener una predicción del posicionamiento del objeto, obteniendo como salidas métricas que midan la eficiencia de su predicción, así como una representación de los resultados de forma gráfica desde una perspectiva en vista de águila de la estimación.

3.2 Detección y clasificación de objetos

El conjunto de datos que aporta nuScenes es muy variado, contando con múltiples clases de objetos, en este caso es necesario que el objeto seleccionado para la implementación y validación del algoritmo sea un objeto que permanezca inmóvil durante la transición de la toma de diferentes imágenes por parte del vehículo. Debido a esto, se hará uso de barreras de tráfico, las cuales además de ser objetos inmóviles cuentan con un buen tamaño y forma para la captación de estos en bounding boxes.

Para llevar a cabo la detección y la generación de bounding boxes, nuScenes registra los fotogramas donde aparecen determinados objetos concretos asociados a una determinada clase, por lo que podemos filtrar el conjunto de imágenes para hacer uso de las que contienen barreras de tráfico.

En base a estas imágenes podemos aplicar una red convolucional que ponga en práctica los algoritmos de detección y clasificación desarrollados en el punto 2.1.2 del documento, dibujando en cada una de las imágenes la bounding box que indique el posicionamiento del objeto dentro de la misma.

Sin embargo, la aplicación de esta red convolucional presenta diferentes problemas que complican la detección y el seguimiento de objetos a lo largo de diferentes fotogramas.

Dentro de esta detección de objetos en imágenes, se presenta el caso en el que debemos identificar inequívocamente a cada uno de los diferentes objetos detectados a lo largo de diferentes imágenes, es decir, tras detectar dos barreras en diferentes imágenes se debe identificar si se trata de la misma barrera o de una distinta.

Este hecho sumado al hecho de que por norma general las barreras tienen aspectos idénticos entre sí y generalmente en las escenas se encuentran varias agrupadas en la misma zona complica aún más esta identificación única para cada una de las barreras.

Esta identificación a lo largo de las imágenes es esencial para el desarrollo del algoritmo de triangulación, ya que es necesario conocer que detecciones a lo largo de los diferentes fotogramas se asocian a que objeto concreto. En caso contrario podrían intentar triangularse las posiciones de diferentes barreras generando errores de estimación muy relevantes y perdiendo una gran eficacia en la predicción.

Es por estos motivos por lo que se ha decidido descartar la implementación de una Red Convolucional para la detección e identificación de las barreras en imágenes dentro del algoritmo de triangulación, contando con el posicionamiento de la bounding box de los diferentes objetos dentro de los datos de entrada para el algoritmo de triangulación. En este caso concreto se hará uso de la información almacenada en la base de datos seleccionada que asocie cada una de las detecciones obtenidas a un objeto concreto.

3.3 Diseño de la solución

En lo referido al diseño del algoritmo de triangulación este se basa principalmente en obtener las diferentes rectas generadas por las captaciones del vehículo y sus diferentes cámaras al objeto en concreto que se desea ubicar, extrapolando esta recta generada en cada imagen captada, a una vista área en el espacio real. Una vez obtenidas estas rectas, su intersección de determinará la posición del objeto en el espacio 3D, siendo esta predicción más o menos cercana a la posición real en función del número de captaciones utilizadas en la triangulación, así como la calidad de estas en cuestión de cercanía con el objeto y visibilidad de este en las imágenes.

Para llevar a cabo este proceso es necesario contar con la siguiente información previa:

- **Ubicación y rotación del vehículo:** se debe conocer la posición del vehículo en el plano XYZ, así como su rotación respecto a una determinada referencia, por ejemplo, su posición y rotación dentro del recorrido realizado medido mediante dicho plano.

- **Ubicación de las cámaras en el vehículo y su rotación:** se debe conocer el posicionamiento y la rotación de cada cámara con respecto al vehículo, ya que la posición de estas es el punto donde partirán cada una de las rectas que se generan mediante las diferentes detecciones de objetos, y la rotación determinará cuanto debe desviarse esta recta del centro del campo de visión de la cámara.

- **FOV de cada cámara:** se debe conocer el campo de visión que abarca de ancho cada una de las cámaras, este debe estar medido en grados.

- **Imágenes con su bounding box asociada al objeto:** se debe contar con las imágenes tomadas que contengan al objeto que se desea geolocalizar, el cual debe estar identificado en la imagen mediante una bounding box. Cabe destacar que se deberá conocer la cámara que ha realizado cada imagen para relacionar aspectos como el FOV o la posición de la que parte de la detección.

Para realizar esta conversión el algoritmo genera tres rectas en cada imagen tomada del objeto que atraviesan la bounding box, atravesando su centro y sus extremos las cuales partirán de un determinado píxel del ancho de la imagen.

Para extrapolar estas rectas al mundo real se debe tener en cuenta el campo de visión de la cámara o FOV, el cual indica el cono de visión del entorno que abarca la imagen indicada en un determinado número de grados. Esto se relaciona con el ancho en pixeles de la imagen previamente comentado, donde cada grado del campo de visión en el mundo real puede establecerse como un determinado número de pixeles dentro de la imagen, por lo que obteniendo la proporción píxel-grado se puede obtener el ángulo de giro de las diferentes rectas generadas en la imagen, giro que será aplicado al centro del campo de visión.

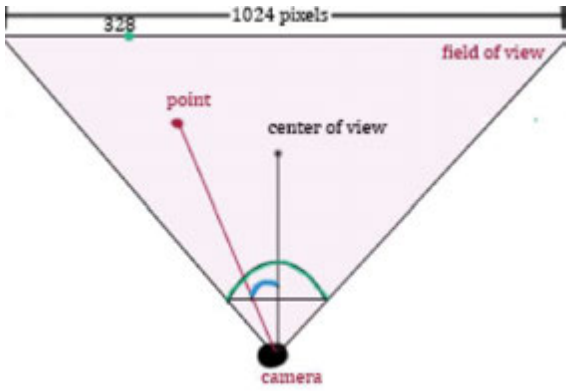

Figura 12: Ángulo de giro de la recta de detección dentro del FOV

Además de este ángulo de giro, se debe tener en cuenta la rotación de la propia cámara, así como la rotación del vehículo, ya que estas pueden afectar a la rotación global de la recta de detección. Tras tener en cuenta estos factores, se aplicará la rotación global a la recta de detección, la cual partirá de la posición global de la cámara hacia el infinito.

Esta transformación previamente explicada de las rectas generadas en las imágenes captadas a una vista área puede verse en la siguiente imagen:

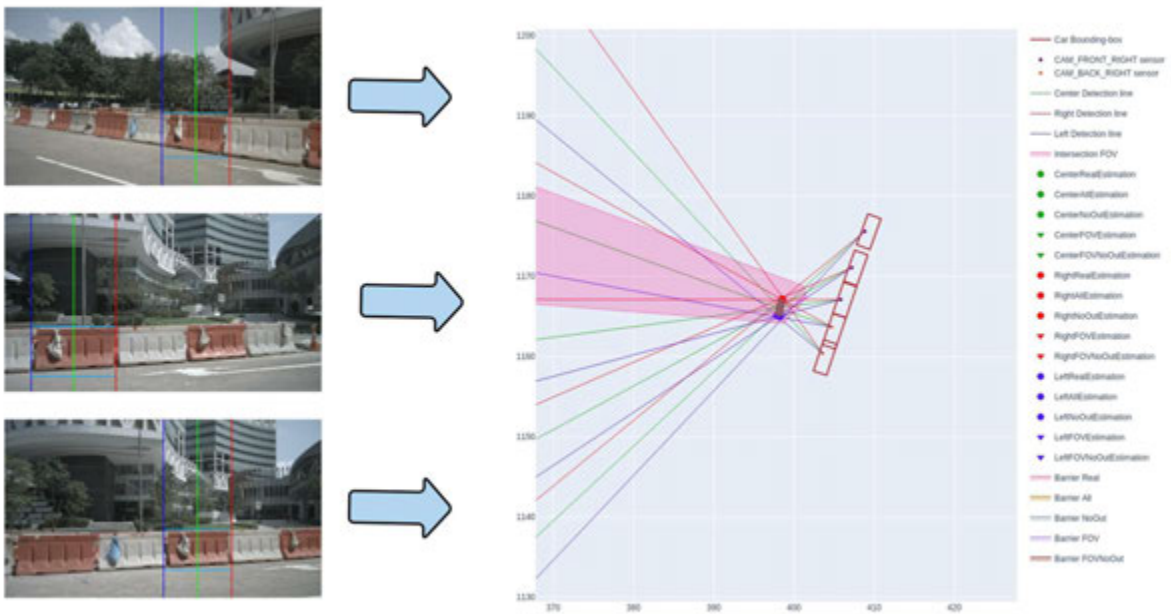

Figura 13: Transformación de las rectas de detección de una bounding box a vista aérea

Una vez generadas las rectas desde una perspectiva de vista aérea, se obtendrá la predicción de la posición del objeto mediante el cálculo de las intersecciones de cada uno de los tres tipos de rectas generados por el algoritmo, obteniendo el centro y los extremos izquierdo y derecho del objeto en el mundo real.

El funcionamiento del algoritmo puede verse de manera visual mediante el siguiente esquema que define su procedimiento:

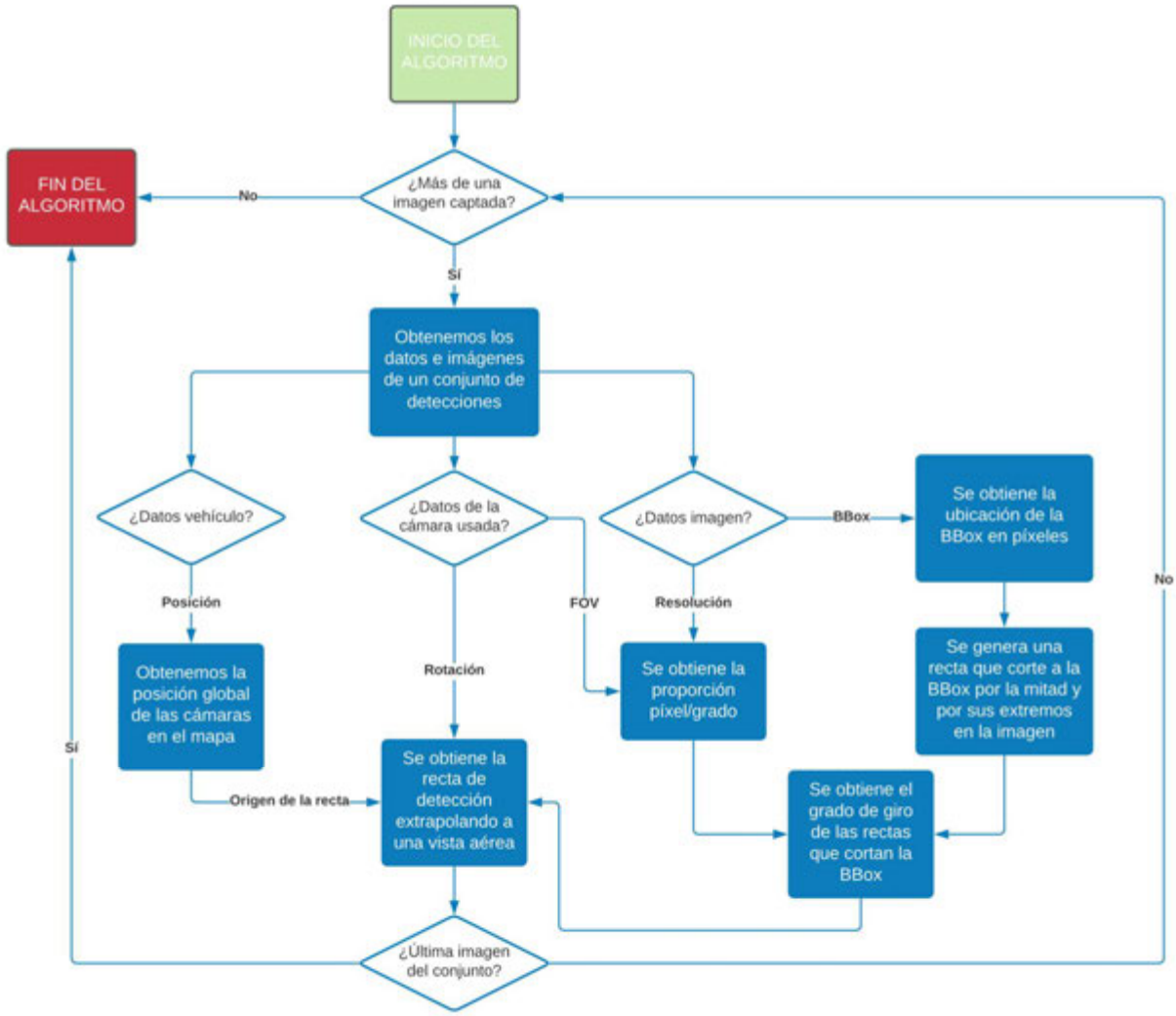

Figura 14: Diseño de la solución, procedimiento del algoritmo

[7]

3.4 Especificación de requisitos

Dentro de este apartado se recoge la especificación de los requisitos necesarios que debe cumplir la solución. El formato de estos requisitos vendrá dado por el siguiente formato, en el cual se detallan los diferentes campos utilizados.

ID			
Nombre			
Prioridad		**Fuente**	
Necesidad			
Claridad		**Verificabilidad**	
Estabilidad			
Descripción			

Tabla 3: Ejemplo de un requisito

Los requisitos se dividirán en funcionales y no funcionales, los cuales contarán con los siguientes atributos:

- **ID:** tiene la finalidad de identificar de manera única a cada uno de los requisitos generados, la nomenclatura de este identificador se da de la siguiente manera:
    - R: indica que se trata de un requisito.
    - C/R: indica si se trata de un requisito de capacidad o restricción.
    - Número: indica la posición del requisito dentro de su subtipo capacidad o restricción.

- **Nombre:** forma resumida de identificar al requisito.

- **Prioridad:** define el nivel de prioridad a la hora de implementar los requisitos en el desarrollo del proyecto. Esta prioridad cuenta con tres niveles:
    - Alta
    - Media
    - Baja

- **Fuente:** origen de donde surge el requisito. Puede tener dos tipos de fuente:
    - Tutor: serán sugerencias o peticiones del tutor que considere necesarias para el desarrollo e implementación del algoritmo.

    - Alumno: se trata de consideraciones del alumno como requisitos necesarios o complementarios a requisitos cuya fuente sea el tutor.

- **Necesidad:** indica la importancia que representa el requisito dentro de la implementación del algoritmo. Se cuenta con los siguientes grados de necesidad:

    - Esencial
    - Deseable
    - Opcional

- **Claridad:** valora el nivel de claridad en la explicación del requisito para comprender su descripción. Se cuenta con los siguientes grados de claridad:

    - Alta
    - Media
    - Baja

- **Verificabilidad:** evalúa la posibilidad de comprobar el correcto funcionamiento del requisito dentro del algoritmo. Se cuenta con los siguientes grados de verificabilidad:

    - Alta
    - Media
    - Baja

- **Estabilidad:** indica la probabilidad de modificar el requisito durante la vida útil del proyecto, suponiendo una mayor estabilidad una menor probabilidad de cambio a futuro. Este campo se mide por los siguientes niveles de estabilidad:

    - Alta
    - Media
    - Baja

- **Descripción:** este campo tiene la finalidad de explicar el requisito de forma precisa.

3.4.1 Requisitos de capacidad

Funcionalidades principales

ID	RC-01		
Nombre	Cálculo de rectas de detección.		
Prioridad	Alta	**Fuente**	Tutor
Necesidad	Esencial		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	Alta		
Descripción	El algoritmo será capaz de calcular las rectas dentro de cada una de las imágenes que atraviesen la mitad y los extremos de la bounding box del objeto.		

Tabla 4: RC-01 Cálculo de las rectas de detección

ID	RC-02		
Nombre	Obtención del ángulo de giro.		
Prioridad	Alta	**Fuente**	Tutor
Necesidad	Esencial		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	Alta		
Descripción	El algoritmo tendrá la capacidad de obtener el ángulo de giro de las rectas respecto a la cámara en base a la recta obtenida en la imagen y el FOV de la cámara.		

Tabla 5: RC-02 Obtención del ángulo de giro

ID	RC-03		
Nombre	Obtención de la rotación global.		
Prioridad	Alta	**Fuente**	Tutor
Necesidad	Esencial		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	Alta		
Descripción	El algoritmo tendrá la capacidad de obtener el giro en el ángulo del giro de la detección en base a la rotación de la cámara, vehículo y el obtenido de la imagen mediante el FOV.		

Tabla 6: RC-03 Obtención de la rotación global.

ID	RC-04		
Nombre	Conversión de las rectas de detección.		
Prioridad	Alta	**Fuente**	Tutor
Necesidad	Esencial		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	Alta		
Descripción	El algoritmo deberá ser capaz de extrapolar las rectas que atraviesan la bounding box al espacio XYZ mediante una visión de águila.		

Tabla 7: RC-04 Conversión de las rectas de detección.

ID	RC-05		
Nombre	Intersección de rectas de detección.		
Prioridad	Alta	**Fuente**	Tutor
Necesidad	Esencial		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	Alta		
Descripción	El algoritmo será capaz de detectar y establecer la posición de los puntos de corte de cada una de las tres rectas de detección con su homónimo en otra imagen.		

Tabla 8: RC-05 Intersección de rectas de detección.

ID	RC-06		
Nombre	Cálculo de error de predicción.		
Prioridad	Alta	**Fuente**	Tutor
Necesidad	Esencial		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	Alta		
Descripción	El algoritmo tendrá la capacidad de calcular el error de predicción con respecto a la posición real del objeto.		

Tabla 9: RC-06 Cálculo de error de predicción.

Representación de los datos de salida

ID	RC-07		
Nombre	Representación gráfica de la vista aérea.		
Prioridad	Alta	**Fuente**	Tutor
Necesidad	Deseable		
Claridad	Media	**Verificabilidad**	Media
Estabilidad	Media		
Descripción	El algoritmo tendrá la capacidad de representar desde una vista aérea la escena con el vehículo, las rectas de detección y la predicción de la posición del objeto mediante las intersecciones de las rectas.		

Tabla 10: RC-07 Representación gráfica de la solución.

ID	RC-08		
Nombre	Dibujado de las rectas de detección en imágenes.		
Prioridad	Alta	**Fuente**	Tutor
Necesidad	Opcional		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	Media		
Descripción	El algoritmo tendrá la capacidad de dibujar las rectas de detección en cada una de las imágenes donde se ha tomado una detección del objeto.		

Tabla 11: RC-08 Dibujado de las rectas de detección en imágenes.

ID	RC-09		
Nombre	Representación gráfica del objeto predicho.		
Prioridad	Alta	**Fuente**	Tutor
Necesidad	Opcional		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	Media		
Descripción	El algoritmo tendrá la capacidad de representar la posición y las dimensiones del objeto en base a la predicción obtenida con las intersecciones de las rectas de detección.		

Tabla 12: RC-09 Dibujado de las rectas de detección en imágenes.

3.4.2 Requisitos de restricción

Datos de entrada

ID	RR-01		
Nombre	Formato de los datos de entrada.		
Prioridad	Alta	**Fuente**	Alumno
Necesidad	Esencial		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	Media		
Descripción	Los datos de entrada necesarios para el funcionamiento del algoritmo serán detectados mediante un fichero CSV.		

Tabla 13: RR-01 Formato de los datos de entrada.

ID	RR-02		
Nombre	Información de entrada.		
Prioridad	Alta	**Fuente**	Alumno
Necesidad	Esencial		
Claridad	Alta	**Verificabilidad**	Alta
Estabilidad	Media		
Descripción	El algoritmo deberá de reconocer y trabajar con la siguiente información de entrada: • Posición global del vehículo. • Rotación global del vehículo. • Posición de la cámara en el vehículo. • Rotación de la cámara en el vehículo. • FOV de la cámara. • Imágenes captadas. • Resolución de las imágenes. • Ubicación de la bounding box en la imagen.		

Tabla 14: RR-02 Información de entrada.

Datos de salida

ID	RR-03		
Nombre	Formato de los datos de salida.		
Prioridad	Alta	**Fuente**	Alumno
Necesidad	Esencial		
Claridad	Alta	**Verificabilidad**	Media
Estabilidad	Media		
Descripción	El algoritmo generará un fichero CSV donde proporcione datos como la posición predicha del objeto, así como diferentes mediciones que indiquen el error de predicción.		

Tabla 15: RR-03 Formato de los datos de salida.

4. IMPLEMENTACIÓN DE LA SOLUCIÓN

4.1 Introducción

Dentro de este bloque se documentan los diferentes estudios y procedimientos realizados para obtener la solución definida en el apartado anterior, abarcando el estudio del dataset a utilizar en el proyecto, el manejo y filtrado de los datos necesarios para alcanzar la solución, y la implementación del algoritmo de triangulación, además del estudio de sus resultados y la eficacia de estos.

4.2 Base de datos nuScenes

NuScenes es un dataset de uso público y no comercial, el cual está centrado en la conducción autónoma por el equipo Motional. Los datos que componen este conjunto de datos se basan en la aplicación de técnicas de visión artificial, constando de un gran número de imágenes y metadatos sobre diferentes recorridos realizados en diferentes localizaciones mediante el uso de un vehículo dotado de diferentes herramientas para la obtención de datos.

Actualmente está compuesto de unas 1000 escenas captadas durante diferentes recorridos dados mediante un vehículo que cuenta con diferentes cámaras, radares y un sensor LIDAR. Estas escenas son realizadas recorriendo las ciudades de Boston y Singapur, principalmente debido a su gran afluencia de tráfico.

Como resultado de esto, la base de datos alberga un conjunto total de datos compuesto por 1.4 millones de imágenes tomadas mediante las cámaras, 390 mil detecciones mediante LIDAR y 1.4 millones barridos de radar.

[27]

Datos contenidos en el dataset

Entrando más en profundidad, dentro del dataset cada uno de los fotogramas realizados desde diferentes cámaras situadas en un vehículo desde diferentes ángulos, siendo estas un total de 6 cámaras distribuidas en diferentes puntos del vehículo.

Además de las imágenes tomadas por estas cámaras, el dataset incluye metadatos de posicionamiento tomados mediante 5 radares cuyo sensor cuenta con un alcance de 200 a 300 metros, así como del sensor LIDAR situado en la parte superior del vehículo. Estos datos se basan en la detección y clasificación de los diferentes objetos que rodean al vehículo en sus diferentes trayectos, situando a estos elementos dentro de diferentes bounding boxes, pudiendo estar clasificados en un total de 23 tipos.

Esta forma de reconocer y clasificar los diferentes objetos mediante bounding boxes se amplió en julio de 2020, donde nuScenes lanzó su versión nuScenes-lidarseg, la cual complementa esta forma de clasificar objetos mediante cajas por una detección y clasificación mediante puntos de LIDAR, encontrándose un total de 1.4 billones de estos puntos en el conjunto del dataset.

Mediante lidarseg se añaden a los 23 tipos de objetos existentes en la clasificación, añadiendo 9 clases de objetos adicionales, las cuales abarcan un conjunto de objetos más generalizables.

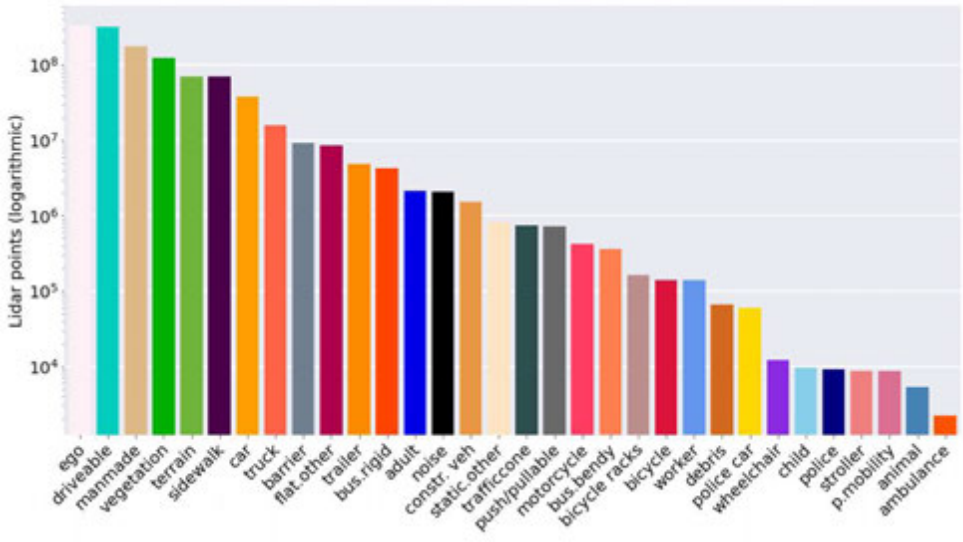

Figura 15: Número de puntos LIDAR presentes para cada clase de objeto.

[27]

En la siguiente imagen puede verse la distribución de las herramientas previamente comentadas para la obtención de los diferentes tipos de datos en el vehículo que desarrolla los diferentes recorridos:

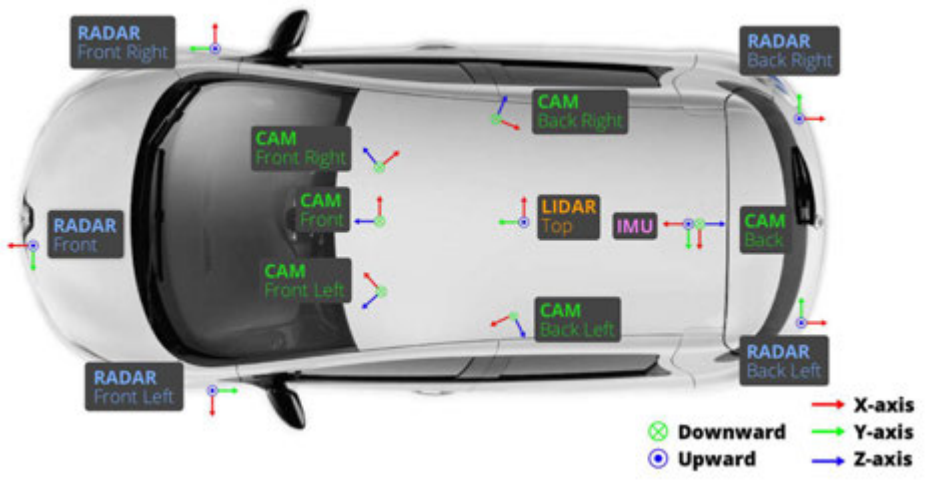

Figura 16: Distribución y posicionamiento de sensores en el vehículo ego.

[27]

Estructuración de los metadatos

El dataset está organizado de una forma similar a una base de datos, contando con diferentes objetos los cuales cuentan con diferentes atributos, siendo algunos de ellos una referenciación a otro objeto lo que permite el traspaso de información.

Esta organización brinda la posibilidad de trabajar con los diferentes tipos de datos obtenidos del dataset de forma separada o en un conjunto de dataset completo donde se incluyen los datos obtenidos por la totalidad de las herramientas disponibles (cámaras, radares, LIDAR…), contando con diferentes sets de datos con diferente tamaño para el manejo de una porción del conjunto de datos.

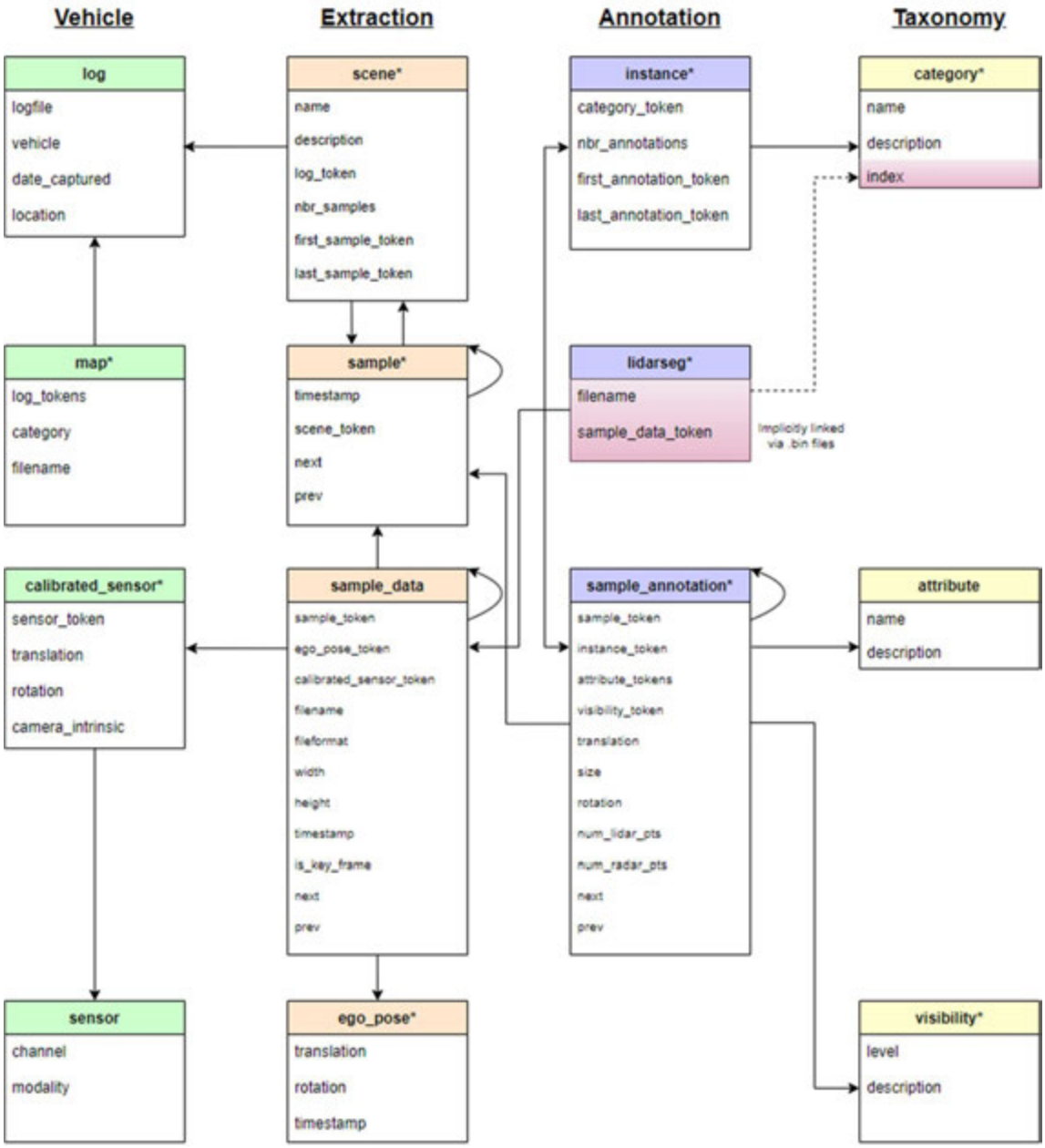

Figura 17: Estructuración de los datos presentes en nuScenes.

[27]

Un aspecto muy importante a tener en cuenta es la referencia que se utiliza para la métrica en los valores de posicionamiento del vehículo, los sensores utilizados y objetos almacenados dentro de nuScenes:

- **Posicionamiento del vehículo ego:** la posición registrada para el vehículo ego, viene dada por los mapas generados de cada una de las escenas. Cada escena genera un mapa a modo de circuito en el cual se generan unas coordenadas globales en base al trazado que genera el coche en su recorrido, de este modo el vehículo ego siempre tendrá el valor 0 en sus coordenadas del eje Z.

- **Posicionamiento de los sensores:** las coordenadas de los diferentes sensores en el eje XYZ vienen dadas respecto a la posición del vehículo ego previamente comentada, siendo dicha posición el punto 0 para cada uno de estos sensores.

- **Posicionamiento de los objetos detectados:** las coordenadas de cada uno de los objetos detectados en los ejes X e Y viene dado por las coordenadas globales del mapa, de forma similar a como se da en la posición del vehículo ego. En lo referido al eje Z, viene dado por la distancia a la que está el objeto del vehículo ego en un determinado momento.

Cada uno de estos objetos es representado mediante un fichero JSON, los cuales almacenan diferentes atributos y se asocia a otros objetos mediante atributos concretos que hacen uso de tokens.

A continuación, se detalla una breve explicación de cada uno de los objetos:

- **log:** almacena la información relativa al registro de información recolectada en las diferentes escenas.

- **map:** se encarga de almacenar los metadatos referentes a los diferentes mapas originados a partir del recorrido realizado en cada escena.

- **calibrated_sensor:** obtiene los datos de calibración de los diferentes sensores a lo largo de los diferentes fotogramas captados.

- **sensor:** guarda los datos básicos de los sensores utilizados en el vehículo ego, relacionándose con el objeto calibrated_sensor mediante un token identificativo.

- **scene:** almacena los datos principales de la escena como su token identificativo, su primer y último fotograma, relacionándose de este modo con el objeto sample.

- **sample:** incluye la información relativa a cada uno de los fotogramas que componen las diferentes escenas, indicando su escena padre, así como el fotograma anterior y posterior al actual.

- **sample_data:** contiene la información relativa a un instante captado por un determinado sensor en un fotograma concreto, teniendo datos como la ruta de la imagen tomada y relaciones con las clases sample_token, ego_pose o calibrated_sensor_token.

- **ego_pose:** almacena las coordenadas globales de posicionamiento del vehículo ego, así como la rotación de este.

- **instance:** es la identificación de un objeto concreto, el cual aparece a lo largo de diferentes fotogramas durante la escena, relacionándose con los datos de los diferentes instantes en los que este objeto aparece, además de la categoría en la que este contenido.

- **lidarseg:** almacena el mapeo de los diferentes puntos LIDAR vinculados a una determinada bounding box en un fotograma. De este modo, se asocian los diferentes ficheros que contienen las nubes de puntos con la clase sample_data.

- **sample_annotation:** contiene la información relativa a los datos de una determinada bounding box

- **category:** se encarga de almacenar todas las categorías almacenadas en nuScenes previamente comentadas, asociando cada una de ellas a un token para su referenciación en el resto de las clases.

- **attribute:** almacena información complementaria para las diferentes categorías, las cuales dan un calificativo a algunas de estas, como puede ser el caso de un vehículo parado, en movimiento o aparcado.

- **visibility:** determina diferentes niveles de visibilidad para una determinada bounding box, cada uno de estos niveles con su determinado token, siendo usado este token en la clase sample_annotation para determinar la visibilidad en un determinado fotograma.

[27]

Otros datasets tenidos en cuenta

Como se puede apreciar nuScenes es un dataset muy prometedor en cuanto a información aplicable al computer visión y al proyecto a desarrollar. El dataset cuenta con diversas ventajas que lo hacen un dataset preferente frente a otros tenidos en cuenta para el desarrollo de este trabajo:

- Cuenta con numerosa documentación en su página web, la cual abarca tanto información de los propios datos, como estos han sido obtenidos, así como tutoriales que indican el manejo y tratamiento de estos.

- Hace uso de diferentes herramientas para la captación de datos (radares, LIDAR...), lo que sumado a las numerosas cámaras para la toma de imágenes por fotogramas dota de una gran cantidad y variedad en los datos. Además de esto, gracias al uso de LIDAR el

dataset tiene la posición real de los objetos en cada instante, lo que es verdaderamente relevante a la hora de analizar el error de precisión del algoritmo de triangulación.

- Información estructurada, separando diferentes conjuntos de datos en función de su tipo y su herramienta en la toma de datos, lo que permite un rápido y ágil manejo de los diferentes tipos de datos que componen el dataset.
- Cuenta con diferentes versiones para su descarga, contando con el dataset completo o variantes de datasets separados por el tipo de dato. Además, cada uno de estos conjuntos cuenta con una versión mini con peso reducido, lo que facilita el manejo y uso de información.
- Cuenta con kit de desarrollo en lenguaje python en github, así como una gran cantidad de documentación sobre la estructuración del dataset y el cómo manejar los diferentes datos proporcionados.

4.3 Filtrado de datos

Una vez realizado el estudio de la base de datos, se debe recopilar del conjunto total de datos la información útil para el desarrollo del proyecto, la cual será procesada y recopilada mediante un CSV que el algoritmo de triangulación deberá recibir como entrada.

Para llevar a cabo este objetivo se ha desarrollado un programa que aplique este filtrado de datos de manera automática. El funcionamiento del programa consiste en obtener de una determinada escena de la base de datos, la cual será introducida como parámetro de entrada, los diferentes datos deseados para todas las barreras de tráfico que aparezcan en dicha escena.

El seguimiento de estos objetos dentro de la clase se realizará analizando los diferentes objetos que aparecen en cada fotograma de la escena, accediendo a su instancia y clase de objeto mediante la relación de tokens presente en nuScenes. De este modo, si se detecta el tipo de objeto como “movable_object.barrier”, se realizará un seguimiento de los fotogramas en los que esta instancia aparece recopilando sus datos y almacenándolos en fichero CSV, generando un fichero CSV por cada instancia donde su número de líneas será el número de fotogramas donde aparezca la instancia.

Los datos almacenados en los ficheros, así como su ordenación dentro de los CSV resultantes de la aplicación del filtrado de datos es el siguiente:

- **Posición de las esquinas de la bounding box:** se basa en una versión convertida al 2D de la posición de la bounding box en imágenes presente en 3D, la cual se encuentra almacenada en el archivo JSON “sample_annotation”. La base de datos nuScenes proporciona en su kit de desarrollo una función capaz de realizar esta conversión para cualquier bounding box almacenado, por lo que antes del almacenamiento de la posición en el filtrado de datos se hará uso de esta función.

Como resultado obtenemos la posición de la esquina superior izquierda y la esquina inferior derecha. Estas coordenadas son dadas en píxeles y vienen representadas mediante un array de cuatro elementos, donde los dos primeros valores hacen referencia a la coordenada de la esquina superior izquierda y los dos últimos las coordenadas de la esquina inferior derecha.

- **Posición global del vehículo:** se registra la posición global del vehículo dentro del mapa del recorrido generado por nuScenes en cada una de las escenas. Esta posición se obtiene accediendo al fichero JSON "ego_pose" accediendo a la posición del vehículo en los diferentes instantes de los fotogramas donde aparece un objeto tipo barrera de tráfico.

  La posición del vehículo viene dada por un array de tres elementos que representa las coordenadas en los ejes XYZ, siendo el eje Z siempre 0 debido a que en los mapas generados en nuScenes, la profundidad representada mediante este eje se basa en la distancia con el coche en su recorrido.

- **Rotación global del vehículo:** almacena la rotación del vehículo en cada instante donde aparezca la instancia de la que se está realizando su seguimiento, esta rotación, al igual que la posición del vehículo, se da con respecto al mapa generado por nuScenes en el recorrido.

  Esta rotación es representada mediante el uso de un cuaternión que se representa mediante los ejes WXYZ.

- **Posición de la cámara respecto al vehículo:** registra la posición de la cámara mediante coordenadas en los ejes XYZ con respecto a la posición del vehículo, teniendo como sistema de referencia en el centro del vehículo. Este valor se obtiene a partir de la relación del instante donde aparece un objeto de tipo barrera de tráfico con la calibración de la cámara que la capta en ese momento.

  Este dato es almacenado como un array de tres valores que indican su posición en los ejes XYZ.

- **Rotación de la cámara respecto al vehículo:** al igual que con el valor de la posición de la cámara, la rotación es en base a la orientación del vehículo, por lo que para extrapolarse a una rotación global se deberá tener a esta en cuenta. El modo de obtener este dato es similar al de la posición de la cámara, mediante los datos de calibración de la cámara.

  El valor almacenado, al igual que la posición del vehículo, se basa en un cuartenión de los ejes WXYZ representado mediante un array de cuatro valores.

- **Tamaño de la imagen:** debido a que todas las imágenes almacenadas en nuScenes cuentan con una resolución de 1600x900 píxeles, se almacenará un array de dos valores que almacene este ancho y alto en píxeles de la imagen.

- **Ruta de la imagen:** para cada uno de los fotogramas asociados, se almacenará la ruta del directorio donde se encuentra almacenada la imagen tomada en dicho instante. Esta ruta se almacenará en forma de string y se obtendrá mediante el token del fotograma donde se realiza la detección de un objeto de tipo barrera de tráfico, donde se almacena el dato de la ruta de la imagen tomada.

- **FOV de la cámara:** para obtener el FOV de la cámara utilizada en cada detección debemos tener en cuenta que en el caso de nuScenes no todas las cámaras cuentan con el mismo campo de visión. Para almacenar este valor se deberá comprobar que cámara del vehículo ha sido la que ha captado el fotograma mediante su nombre, siendo 70º en caso de ser cualquier cámara salvo la cámara trasera ("CAM_BACK"), teniendo esta última un campo de visión de 110º.

  Este valor se almacena como un número de tipo entero dentro del fichero CSV.

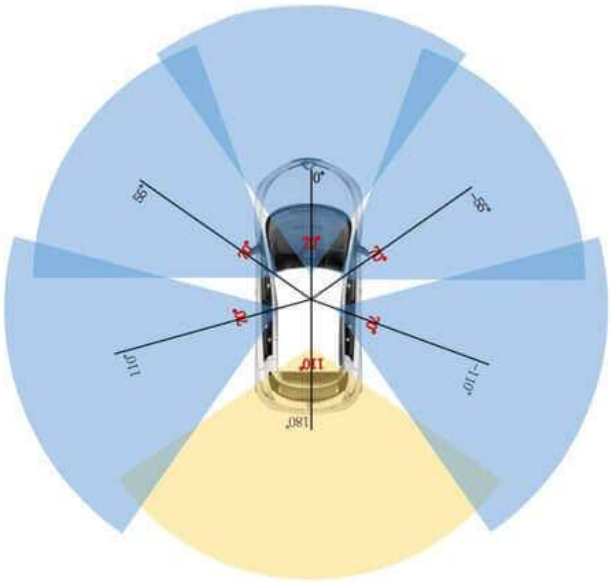

Figura 18: Campo de visión de las cámaras utilizadas en nuScenes.

[27]

- **Coordenadas globales del objeto:** de forma opcional, se incluirán las coordenadas reales del objeto almacenadas en nuScenes en el plano XYZ para realizar representaciones y cálculos del error del algoritmo. Estas coordenadas se obtienen mediante el token de los objetos detectados, donde se obtiene su posición global registrada en el mapa.

- **Nombre de la cámara:** otro parámetro opcional es la captación de los nombres de las cámaras utilizadas, las cuales simplemente se usarán para métodos de representación gráfica de la solución, teniendo los siguientes nombres de cámaras:

  - CAM_FRONT
  - CAM_BACK
  - CAM_BACK_LEFT

- CAM_FRONT_LEFT
- CAM_FRONT_RIGHT
- CAM_BACK_RIGHT

4.4 Elaboración y funcionamiento del algoritmo realizado

Una vez definido el diseño de la solución, así como los datos de entrada necesarios para el funcionamiento de esta, se debe proceder con la implementación del algoritmo en forma de código. Esta implementación ha sido desarrollada en lenguaje Python y se ha hecho uso del entorno de Google Colab, al igual que la implementación del script de filtrado de datos.

La elección de este lenguaje y entorno es debido a contar con kits de desarrollo para el manejo de los datos proporcionados por nuScenes, así como la capacidad de manejar información muy pesada como es el caso de las imágenes, metadatos y ficheros CSV utilizados con almacenamiento en la nube de Google Drive.

Dentro de este bloque se tratará tanto el funcionamiento del algoritmo implementado desde una visión global que relacione todas las funcionalidades entre sí, así como una visión más en detalle de diferentes secciones del algoritmo. Finalmente se analizarán los resultados de salida obtenidos, lo que permitirá valorar la eficiencia del algoritmo y en que situaciones demuestra obtener unos mejores resultados.

4.4.1 Funcionamiento general del algoritmo

Como se especifica en el diseño de la solución, el funcionamiento principal del algoritmo se basa en extrapolar la posición de un objeto dentro de una imagen a un entorno en tres dimensiones representado mediante una vista aérea. Para llevar a cabo este objetivo se generarán unas rectas de detección dentro de la imagen que mediante datos conocidos como el FOV de la cámara o su posición se extrapolarán a la vista área, siendo las intersecciones de dichas rectas la predicción de la posición predicha por el algoritmo.

El algoritmo se rige mediante proceso de ejecución:

1. **Tratamiento de la información de entrada y generación de rectas:** en este primer proceso, el algoritmo se encarga de acceder a la ruta donde se encuentran almacenados los ficheros CSV con los datos de entrada, recorriendo mediante un bucle todos los ficheros almacenados línea por línea.

   Cada uno de estos ficheros almacena todas las detecciones realizadas a una barrera concreta en diferentes instantes y ángulos, por lo que cada una de las líneas de un fichero CSV corresponde a los datos de una detección. Estas líneas podrán ser descartadas si se considera una mala detección del objeto, como podría ser el caso de una detección muy lejana del objeto o con mala visibilidad de este.

Dentro de un bucle iterativo para cada una de las detecciones, de forma previa al cálculo de las rectas de detección, se identifican la posición y rotación del vehículo, así como la posición y rotación de la cámara que realiza la detección en el mapa. Además de calcular y obtener el triángulo formado por el FOV de la cámara con dicha rotación.

A continuación, se realizará el proceso de generación de rectas, calculando su posición inicialmente en la imagen mediante la posición de la bounding box dentro de la imagen. De este modo se generarán tres rectas, una que pase por el centro de la bounding box del objeto y dos que pase por sus extremos izquierdo y derecho.

Una vez se generan estas tres rectas dentro de la imagen, se obtendrá el ángulo de giro de esta mediante el dato del campo de visión de la cámara utilizada y la resolución de la imagen captada, estableciendo una proporción que determine cuantos grados de giro supone cada píxel de la imagen. Este ángulo de giro puede darse hacia la derecha o la izquierda dependiendo de la posición del objeto, alcanzando una inclinación máxima igual a la mitad de los grados que abarca el FOV de la cámara que capta la imagen.

Finalmente, se obtendrá la representación de esta recta fuera de la imagen mediante la combinación de los ángulos de rotación del coche, rotación de la cámara y rotación de la recta de detección dentro del FOV de la imagen, la cual en base a la ubicación de la cámara que es fijada como punto de origen, generará la recta desde una perspectiva de vista aérea.

2. **Proceso de triangulación para predecir el posicionamiento del objeto:** tras obtener la posición y representación de las rectas en vista de águila de cada una de las detecciones de un fichero CSV concreto, se debe proceder con la detección y cálculo de intersecciones entre las diferentes rectas de detección.

   Para ello, mediante el uso de un bucle que recorra todas las rectas de detección generadas en el punto 1 para cada uno de los tres tipos de detección, almacenando los puntos donde se producen intersecciones en los ejes XY dentro de la vista de águila.

   Durante este proceso se obtendrá la posición real del objeto y se establecerá su bounding box desde una perspectiva de vista área para la medida de métricas de error y precisión en el punto 3.

3. **Obtención y representación de los datos de salida:** en los anteriores puntos se ha realizado la generación de las rectas de detección y el cálculo de posicionamiento del vehículo, el sensor, las intersecciones que determinan la predicción del objeto y la propia posición real del objeto. En base a la información obtenida, se procede a medir la eficacia del algoritmo mediante los resultados en base a métricas de error y precisión, así como representar la vista aérea para cada una de las detecciones de los diferentes ficheros CSV, representando una escena por fichero.

Cada uno de estos tres procesos, así como el formato y el tipo de dato con los que estos trabajan serán explicados más en detalle a lo largo de los siguientes apartados.

En base a las tres funcionalidades del algoritmo previamente explicadas, podemos representar el funcionamiento general del algoritmo de forma gráfica mediante el siguiente diagrama:

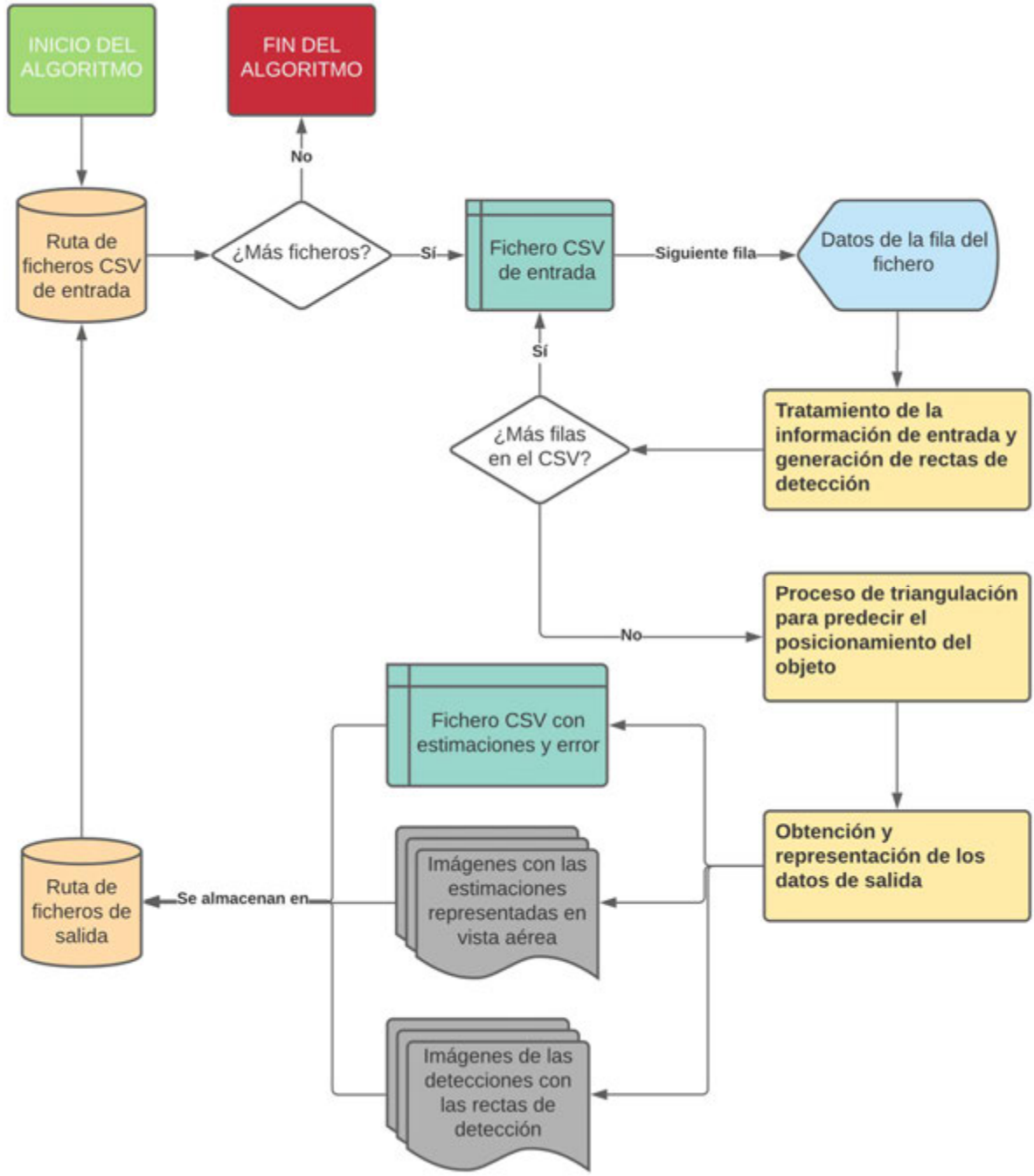

Figura 19: Funcionamiento general del algoritmo de triangulación implementado.

[7]

4.4.2 Tratamiento de la información de entrada y generación de rectas

El tratamiento de la información de entrada es el primer realizado por el algoritmo, donde este debe leer los diferentes ficheros CSV recibidos y tratar su información para el desarrollo del algoritmo, este proceso se da mediante un doble bucle que recorre los diferentes ficheros y para cada uno de ellos se recorren todas las filas almacenadas.

Cada fichero CSV equivale a todas las detecciones de un objeto concreto a lo largo de una escena, siendo cada línea del fichero una detección. De este modo, la generación de rectas se dará dentro de este segundo bucle de filas del fichero CSV, donde se generarán las rectas de detección para cada una de estas en función de los datos reconocidos por entrada. De esta forma, se deberá de recorrer la totalidad de los ficheros almacenados y todas sus filas haciendo uso de un bucle anidado sobre otro.

Para cada una de estas filas se debe valorar si se trata de una detección viable, para ello se calculará el área o tamaño de la bounding box del objeto en la imagen de la detección, descartando de esta manera a las detecciones donde el objeto se encuentre a una gran distancia de la cámara. Estos casos son descartados debido a desembocar en intersecciones de rectas de detección poco precisas y sin valor real de predicción de la posición del objeto.

Figura 20: Ejemplo de detección descartada por ser demasiado lejana.

Una vez descartados estos casos, se procede a la generación de las tres rectas de detección para cada una de la bounding boxes de las diferentes detecciones, generando una recta central (verde) que pasa por el centro y dos rectas que pasan por los laterales de esta bounding box (azul y roja).

Esta generación de rectas se da en base a la posición de la esquina superior izquierda y la inferior derecha de la bounding box, las cuales se encuentran almacenadas en la fila del fichero CSV que corresponde a la detección. En base a estos valores, se calcula el centro de la caja en las posiciones XY en pixeles dentro de la imagen, así como las posiciones de las esquinas laterales, lo que concluye en la obtención de las tres rectas pasando a través de la imagen por estos puntos.

Una vez generadas las rectas y conociendo su posición en el ancho de los píxeles dentro de la imagen, se obtiene la proporción de cuantos pixeles equivalen a un grado de visión dentro del FOV de la cámara, obteniendo este último dato del fichero CSV.

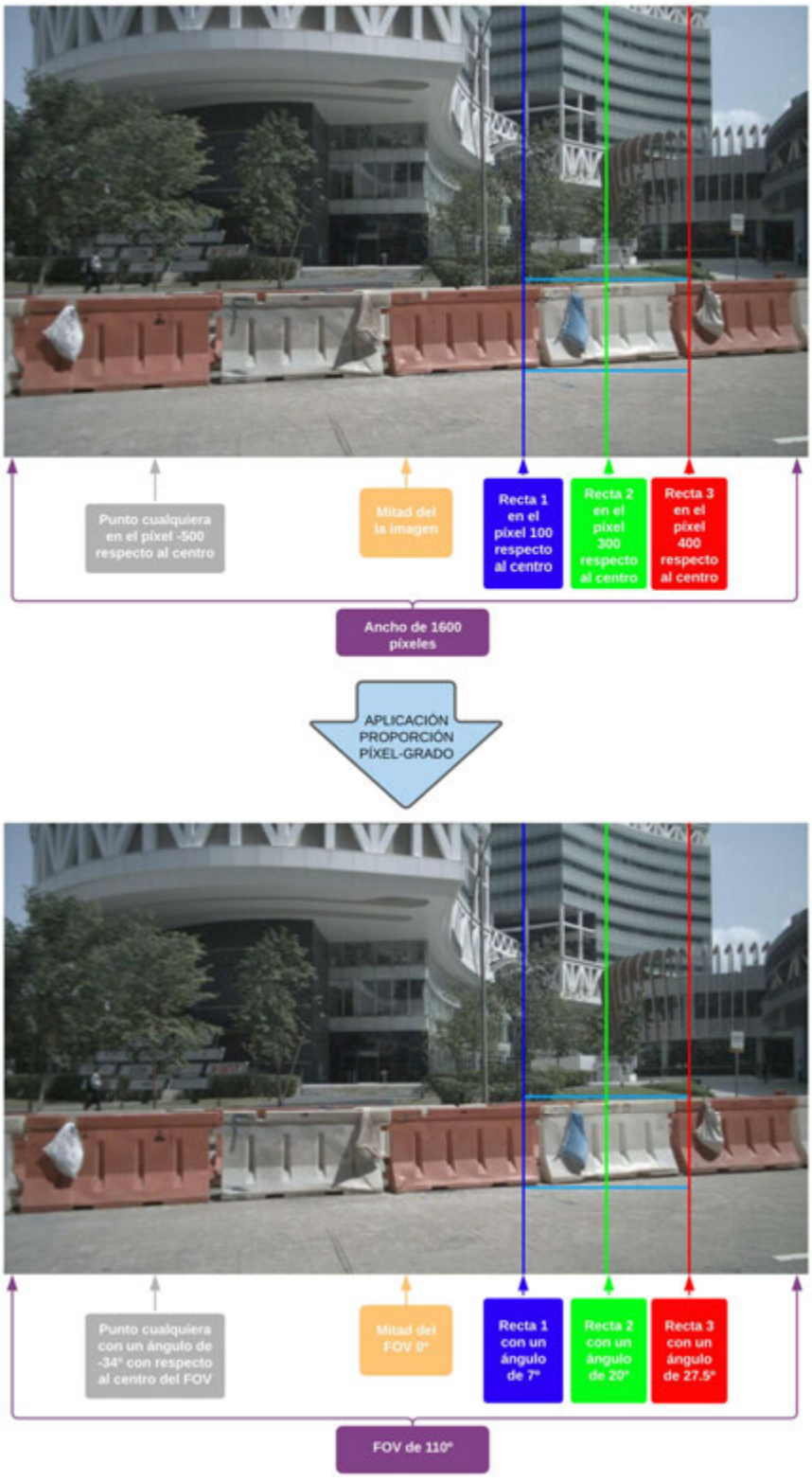

Figura 21: Transformación de píxeles a grados en las rectas de detección en imágenes.

[7]

Tras haber obtenido la rotación de estas imágenes dentro del FOV de la cámara, debemos extrapolar esta información a una vista área con el conjunto de elementos que participan en la escena. Como se puede observar en la siguiente figura, el campo de visión que abarca la imagen depende directamente de la rotación de la cámara y del vehículo, ya que la inclinación de cualquiera de estos factores afecta al cono de visión y por tanto al ángulo global de las rectas de detección contenidas en dicho cono.

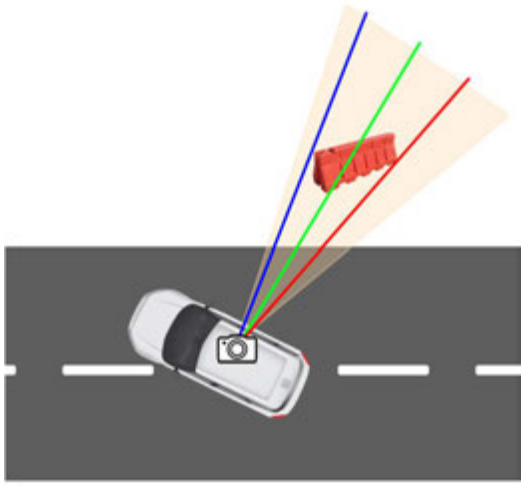

Figura 22: Esquema de la vista aérea de las rectas de detección

[7]

Para aplicar a las rectas de detección una rotación general, se calcula la combinación de las rotaciones almacenadas en el fichero CSV como es la rotación del vehículo y la cámara, con el giro calculado mediante la transformación de las rectas de detección en imágenes. De este modo se obtiene una rotación de la recta global para las rectas de detección que pueden ser representadas desde una vista aérea del mapa del recorrido.

4.4.3 Proceso de triangulación para predecir el posicionamiento del objeto

Una vez obtenidas las rectas de detección con su rotación final para cada una de las detecciones de un determinado fichero CSV, se procede con la detección y localización de las diferentes intersecciones generadas por las rectas de detección obtenidas, siendo estos puntos los que determinarán la predicción de la posición del objeto.

Durante la generación de cada una de las rectas de detección, estas han sido almacenadas en una lista de líneas que se ha ido rellenando a lo largo del recorrido de las diferentes detecciones del fichero CSV, estando esta lista compuesta por tres columnas, donde se almacenan cada uno de los tres tipos de rectas de detección que segmentan la bounding box.

En base a la lista de rectas de detección se determinarán las diferentes intersecciones entre las rectas del mismo tipo (izquierda, centro o derecha), recorriendo cada una de las tres columnas para detectar únicamente intersecciones entre rectas que pertenezcan a la misma categoría, evitando de esta forma falsas detecciones.

Esta detección se realiza mediante la aplicación de dos matrices, mediante las cuales se irá comprobando si alguna de las rectas presentes llega a colisionar con otra mediante el uso de geometría dentro de este plano de vista aérea, obteniendo las coordenadas del punto donde sucede dicha colisión.

En una primera etapa del algoritmo, se contaban todos los puntos de intersección entre las diferentes rectas. Sin embargo, esto generaba en ciertos casos puntos muy dispares entre sí, lo que dificultaba la predicción de la posición del objeto en el escenario.

Para alcanzar una mayor precisión en la predicción de la posición del objeto se descartarán las intersecciones fuera del punto medio de colisiones dadas entre rectas. Este punto medio es calculado mediante la media de las coordenadas de todas las intersecciones obtenidas, quedando esta zona acotada en el lugar donde más colisiones se producen, la cual se traduce en la zona con mayor probabilidad de encontrarse el objeto. De este modo, se descargan los puntos que quedan fuera de este rango de valor medio calculado, siendo estos denominados como outliners.

Para pulir aún más la precisión en la predicción dada por los puntos de intersección entre rectas, se establece una limitación en la consideración de puntos de intersección para el cálculo del valor medio en base a los puntos que se encuentran en la intersección del área de visión del FOV de las diferentes detecciones realizadas. Descartando de este modo los puntos que quedan fuera de esta zona delimitada, contando con una versión mejorada de este nuevo filtro que se encarga de descartar los outliners que se encuentran fuera del rango del valor medio calculado.

Por lo tanto, tras realizar diferentes mejoras iterativas para la precisión de las intersecciones entre rectas, el algoritmo cuenta con cuatro modos para la identificación de intersecciones, siendo entre ellos el modo “FOV” el más preciso de todos debido a su descarte de outliners y su descarte a puntos fuera de la intersección de los diferentes campos de visión de las detecciones realizadas.

Los diferentes modos con los que cuenta el algoritmo son los siguientes:

- **All:** calcula el punto medio entre todas las intersecciones realizadas por las rectas de detección, incluyendo las que se encuentran fuera de la intersección de los diferentes FOV de las detecciones. Sin embargo, este modo descarta los outliers para la obtención de la posición del objeto, siendo estos outliers puntos de intersección que se quedan fuera del rango de la media de intersecciones calculada.

- **NoOut:** es similar al modo “All”, aunque en este caso no se descarta a los outliers.

- **FOV:** tiene en cuenta únicamente las intersecciones producidas dentro de la intersección de los diferentes FOV para el cálculo de la media de intersecciones. Es el modo que obtiene mayor precisión en sus predicciones, realizando al igual que el modo “All” el descarte de los outliers.

- **FOVNoOut:** este modo es similar al modo “FOV”, diferenciándose en el hecho de tomar en cuenta los outliers que se alejan de la media de intersección calculada.

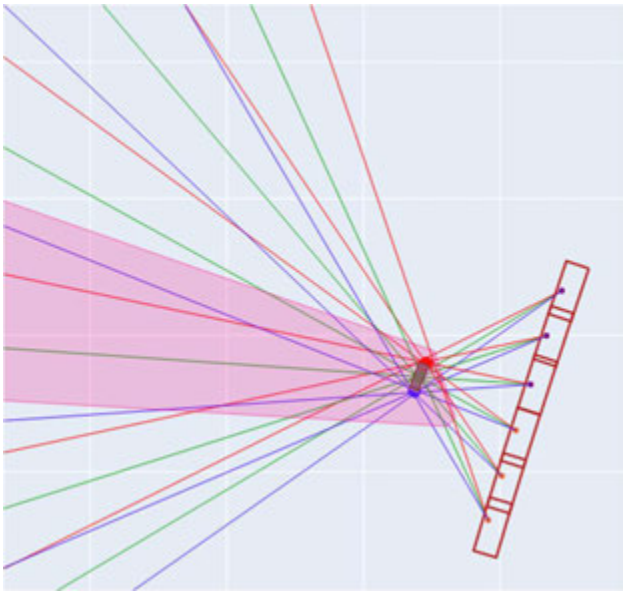

Figura 23: Filtrado de intersecciones dentro del área de intersección entre los FOV

Como se puede apreciar en la imagen, el modo empleado para la detección de intersecciones es ”FOV”, ya que las intersecciones obtenidas fuera del área donde colisionan los diferentes FOV (zona sombreada en rojo) son completamente ignorados por el algoritmo, así como las intersecciones fuera del área del punto medio de las intersecciones, pese a encontrarse algunas intersecciones dentro del área de colisión entre los diferentes FOV.

Una vez obtenidas las diferentes intersecciones y habiendo estimado cuales se encuentran más cerca del punto medio de intersecciones, se estima la rotación y posicionamiento de las cuatro esquinas de la barrera y de su centro. Estos datos serán usados para estimar el error de posicionamiento y rotación respecto a la barrera real, así como para poder representar la barrera junto con el resto de la escena en la recreación gráfica de vista aérea.

4.4.4 Obtención y representación de los datos de salida

Una vez generadas las rectas de detección en vista aérea, así como las colisiones filtradas para evitar el ruido en los datos, se deben evaluar las predicciones obtenidas con respecto a la posición real del objeto en caso de encontrarse este dato almacenado en el CSV.

Para valorar la precisión de la predicción obtenida, así como medir diferentes parámetros, se calculan diferentes métricas que serán almacenadas en un fichero CSV, siendo cada una de las filas una detección de una determinada barrera, obteniendo de este modo un fichero CSV por barrera presente en la escena.

Para cada una de las filas de cada fichero CSV se cuenta con los siguientes parámetros:

- **idBarrier:** número del objeto barrera, siendo identificadas en función de su aparición en la escena, siendo la barrera 0 la primera en aparecer.
- **Frame:** numera cada uno de los fotogramas donde aparece una determinada barrera.

- **numDetections:** indica el número de detecciones captadas para una determinada barrera.

- **YawError:** indica el error de rotación del objeto predicho respecto a la rotación del objeto real. Este error será calculado para los cuatro modos de detección de intersecciones de las rectas de detección, siendo el nombre de estos errores "YawErrorAll", "YawErrorNoOut", "YawErrorFOV" y "YawErrorFOVNoOut".

- **area2DBB:** tamaño de la barrera en la detección 2D captada en la imagen.

- **distBarrierCar:** distancia del centro de la barrera al vehículo que la detecta.

- **ErrMeanCenter:** este error se encarga de medir la cercanía o diferencia de las coordenadas del centro predicho para el objeto con respecto a la ubicación del centro de la posición real del objeto. Este error será calculado para los cuatro modos de detección de intersecciones de las rectas de detección, siendo el nombre de estos errores "ErrMeanCenterAll", "ErrMeanCenterNoOut", "ErrMeanCenterFOV" y "ErrMeanCenterFOVNoOut".

- **ErrMeanLeft:** este error se encarga de medir la cercanía o diferencia de las coordenadas del lateral izquierdo predicho para el objeto con respecto a la ubicación de dicho lateral en la posición real del objeto. Este error será calculado para los cuatro modos de detección de intersecciones de las rectas de detección, siendo el nombre de estos errores "ErrMeanLeftAll", "ErrMeanLeftNoOut", "ErrMeanLeftFOV" y "ErrMeanLeftFOVNoOut".

- **ErrMeanRight:** este error se encarga de medir la cercanía o diferencia de las coordenadas del lateral derecho predicho para el objeto con respecto a la ubicación de dicho lateral en la posición real del objeto. Este error será calculado para los cuatro modos de detección de intersecciones de las rectas de detección, siendo el nombre de estos errores "ErrMeanRightAll", "ErrMeanRightNoOut", "ErrMeanRightFOV" y "ErrMeanRightFOVNoOut".

- **ErrFrameCenter:** representa el error con respecto al punto medio de intersecciones para las rectas de detección del centro de la bounding box (recta verde). Este error será calculado para los cuatro modos de detección de intersecciones de las rectas de detección, siendo el nombre de estos errores "ErrFrameCenterAll", "ErrFrameCenterNoOut", "ErrFrameCenterFOV" y "ErrFrameCenterFOVNoOut".

- **ErrFrameLeft:** representa el error con respecto al punto medio de intersecciones para las rectas de detección de la pared izquierda de la bounding box (recta azul). Este error será calculado para los cuatro modos de detección de intersecciones de las rectas de detección, siendo el nombre de estos errores "ErrFrameLeftAll", "ErrFrameLeftNoOut", "ErrFrameLeftFOV" y "ErrFrameLeftFOVNoOut".

- **ErrFrameRight:** representa el error con respecto al punto medio de intersecciones para las rectas de detección de la pared derecha de la bounding box (recta roja). Este error será calculado para los cuatro modos de detección de intersecciones de las rectas de detección, siendo el nombre de estos errores "ErrFrameRightAll", "ErrFrameRightNoOut", "ErrFrameRightFOV" y "ErrFrameRightFOVNoOut".

- **NumIntCenter:** indica el número de intersecciones que se quedan dentro del rango del valor medio calculado en el caso de la recta que atraviesa el centro de la bounding box (recta verde), descartando de este modo los outliners. Este parámetro es calculado para los modos de detección de intersecciones de las rectas de detección "FOV" y "All", siendo el nombre de estos parámetros "NumIntCenterNoOut" y "NumIntCenterFOVNoOut".

- **NumIntLeft:** indica el número de intersecciones que se quedan dentro del rango del valor medio calculado en el caso de la recta que atraviesa el lateral izquierdo de la bounding box (recta azul), descartando de este modo los outliners. Este parámetro es calculado para los modos de detección de intersecciones de las rectas de detección "FOV" y "All", siendo el nombre de estos parámetros "NumIntLeftNoOut" y "NumIntLeftFOVNoOut".

- **NumIntRight:** indica el número de intersecciones que se quedan dentro del rango del valor medio calculado en el caso de la recta que atraviesa el lateral derecho de la bounding box (recta roja), descartando de este modo los outliners. Este parámetro es calculado para los modos de detección de intersecciones de las rectas de detección "FOV" y "All", siendo el nombre de estos parámetros "NumIntRightNoOut" y "NumIntRightFOVNoOut".

- **PercentIntCenter:** almacena el porcentaje de las intersecciones obtenidas dentro del rango del valor medio calculado para las rectas de detección que atraviesan el punto medio de la bounding box (recta verde), descartando de este modo los outliners. Este parámetro es calculado para los modos de detección de intersecciones de las rectas de detección "FOV" y "All", siendo el nombre de estos parámetros "PercentIntCenterNoOut" y "PercentIntCenterFOVNoOut".

- **PercentIntLeft:** almacena el porcentaje de las intersecciones obtenidas dentro del rango del valor medio calculado para las rectas de detección que atraviesan el lateral izquierdo de la bounding box (recta azul), descartando de este modo los outliners. Este parámetro es calculado para los modos de detección de intersecciones de las rectas de detección "FOV" y "All", siendo el nombre de estos parámetros "PercentIntLeftNoOut" y "PercentIntLeftFOVNoOut".

- **PercentIntRight:** almacena el porcentaje de las intersecciones obtenidas dentro del rango del valor medio calculado para las rectas de detección que atraviesan el lateral derecho de la bounding box (recta roja), descartando de este modo los outliners. Este parámetro es calculado para los modos de detección de intersecciones de las rectas de detección "FOV" y "All", siendo el nombre de estos parámetros "PercentIntRightNoOut" y "PercentIntRightFOVNoOut".

- **Overlap:** este parámetro indica el porcentaje de solapamiento entre la figura del objeto predicha en el mapa, en este caso un rectángulo que representa la barrera, y la figura del objeto en su posición real. Suponiendo un mayor porcentaje de solapamiento una predicción más acertada.

  Este parámetro se mide para los cuatros modos de detección de intersecciones presentes en el algoritmo, siendo el nombre de estos parámetros "OverlapAll", "OverlapFOV", "OverlapNoOut" y "OverlapFOVNoOut".

- **widthBarrier:** almacena el ancho del objeto en el espacio real.

- **lengthBarrier:** indica la altura del objeto en el espacio real.

- **height2DBB:** alto de la barrera en la detección 2D dentro de la imagen medido en píxeles.

- **width2DBB:** indica el ancho de la barrera en la detección 2D de la imagen captada, estando esta anchura medida en píxeles.

Mediante este conjunto de métricas podemos valorar tanto la precisión de las diferentes predicciones realizadas, como contrastar la mejora de dicha precisión en base a las mejoras realizadas con los diferentes modos de detección de intersecciones entre rectas de detección realizados.

Además de estas métricas almacenadas en los diferentes ficheros CSV, para cada una de las barreras detectadas en la escena se genera una recreación desde una perspectiva de vista aérea de la escena mediante el uso de Plotly. En esta representación se muestran las diferentes detecciones captadas del objeto junto con sus líneas de detección, la posición y rotación del vehículo en los diferentes instantes, los puntos de intersección de las rectas, la estimación del posicionamiento y rotación de la barrera en base a los puntos de intersección obtenidos, y la posición y rotación exactas de la barrera real.

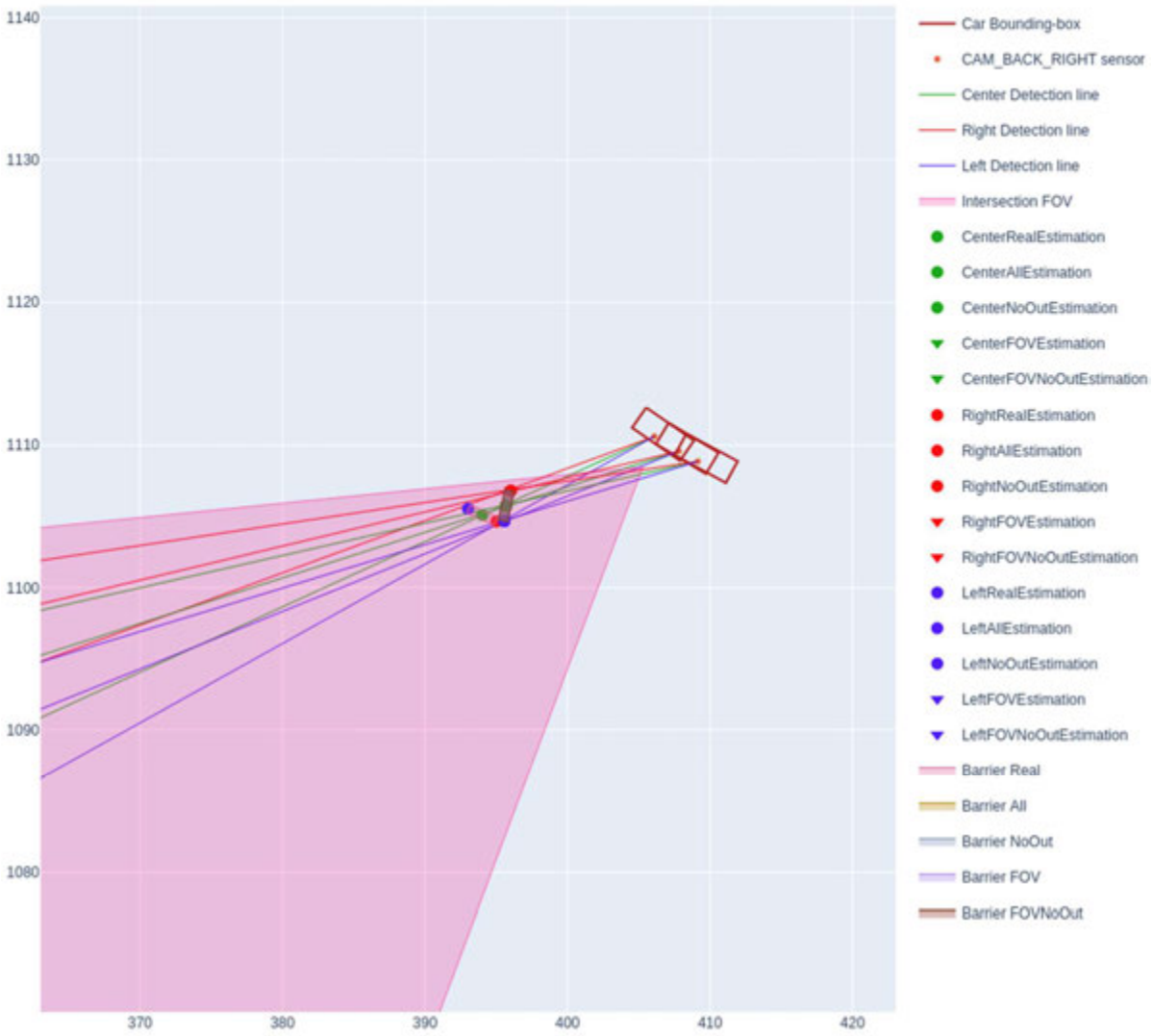

Figura 24: Recreación desde vista aérea de una determinada barrera

En base a estas representaciones, podemos observar de manera visual la precisión de la predicción con respecto a la posición real de la barrera, así como los diferentes valores almacenados en el fichero CSV.

A toda esta información de salida se le suman los fotogramas individuales de cada una de las captaciones realizadas a cada objeto, donde se encuentran dibujadas tanto las tres líneas de detección, como la bounding box que abarca al objeto. El formato de este tipo de imágenes puede verse en las figuras 20 y 21.

4.4.5 Herramientas utilizadas durante la implementación

Durante el desarrollo del proyecto se ha hecho uso de diferentes herramientas para el desarrollo del algoritmo, las cuales han permitido trabajar con grandes conjuntos de datos y representar correctamente los resultados de salida.
Las principales herramientas utilizadas han sido las siguientes:

- **Google Collab:** entorno de programación en la nube, en este caso se ha hecho uso del lenguaje Python para el desarrollo, siendo útil este entorno en el manejo de grandes cantidades de información debido a su funcionamiento en la nube.

- **Google Drive:** plataforma de almacenamiento en la nube, útil para el manejo de grandes cantidades de información como es el caso de los datos manejados por el algoritmo.
  A esto le debemos sumar la capacidad de trabajar en Google Collab con los datos almacenados en Drive, lo que permite una mayor comodidad en el manejo de la información.

- **Plotly:** librería de código abierto para Python centrada en la generación de diferentes tipos de gráficos. Para este trabajo se ha hecho uso de dicha herramienta para el dibujado de la vista aérea y los diferentes elementos que participan en la escena, como es el caso de las diferentes posiciones del vehículo, las rectas de detección generadas, los puntos de intersección, la posición estimada del objeto, la posición real, etc.
  Además de esto, se ha hecho uso de Plotly para el dibujado de las bounding boxes en 2D, así como la representación de las rectas de detección dentro de cada uno de los fotogramas captados de los diferentes objetos.

- **Dev-kit nuScenes:** kit de desarrollo proporcionado por nuScenes dentro del entorno de Google Collab, donde se manejan los diferentes datos almacenados en el dataset, así como el renderizado de imágenes con sus bounding boxes o puntos LIDAR.
  Este dev-kit ha permitido la creación de un script para la obtención de los datos necesarios para el desarrollo del algoritmo de triangulación y almacenar dichos datos en ficheros CSV.

4.5 Análisis y evaluación de la solución

Dentro de este bloque se analizará la eficacia de predicción del algoritmo en base a las métricas obtenidas mediante la información de salida generada. Además de esta medida de eficiencia se analizarán los diferentes casos donde el algoritmo presenta unos mejores o peores resultados y a que podrían deberse estos hechos.

4.5.1 Análisis de las mejoras obtenidas con el algoritmo

Dentro de este apartado se realiza un análisis de los resultados presentados por el algoritmo en cuanto a eficacia de predicción se refiere. Para llevar este objetivo a cabo se realiza un estudio en base a los resultados obtenidos para las predicciones de diferentes objetos con un determinado número de detecciones, sacando de este modo el valor medio de los errores y parámetros de salida obtenidos.

Para la realización del estudio se toman los datos obtenidos para diferentes objetos con un total de 6 y 15 detecciones a lo largo de un determinado recorrido, realizando la media de los parámetros obtenidos para su posterior análisis y comparación.

Número de intersecciones detectadas

En base al número de intersecciones medias detectadas para el caso de las 6 detecciones se puede apreciar que en los diferentes modos de detección de intersecciones entre las rectas de detección no se aprecia una gran diferencia. Esto es debido a que al tratarse de un bajo número de detecciones las líneas de detección no experimentan muchas intersecciones fuera de rango, por lo que la aplicación de los diferentes modos de mejora de detección no hace un descarte relevante en este aspecto.

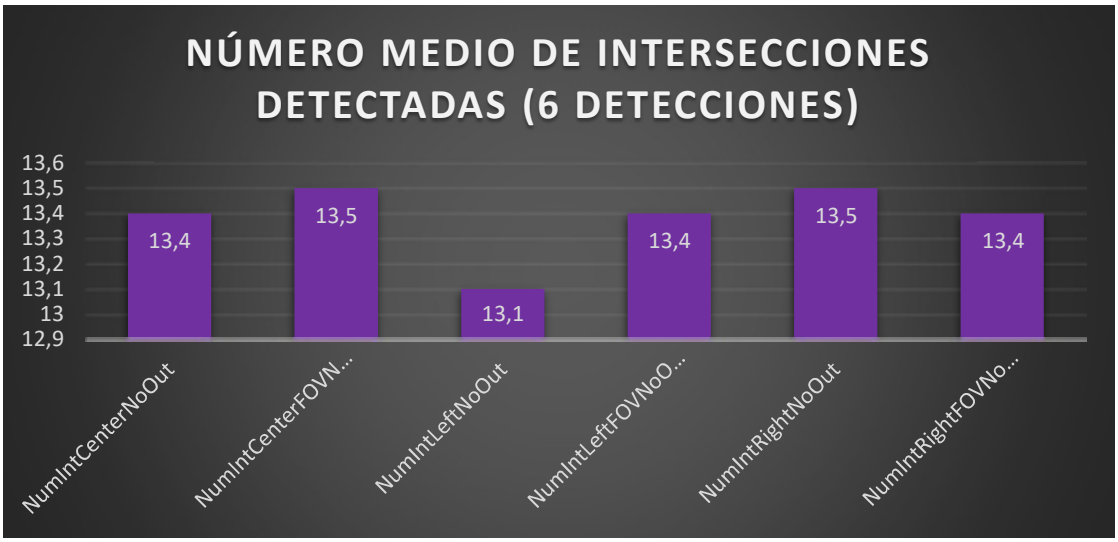

Figura 25: Gráfica del número medio de intersecciones detectadas para 6 detecciones

Por otro lado, tenemos el caso de los objetos que han tenido un total de 15 detecciones, en los cuales si se aprecia una diferencia significativa de descartes en el número de intersecciones por parte del modo "FOVNoOut". Este modo realiza un descarte de los puntos de intersección que se encuentren fuera de la intersección de los campos de visión de las diferentes detecciones realizadas, además de descartar los outliners.

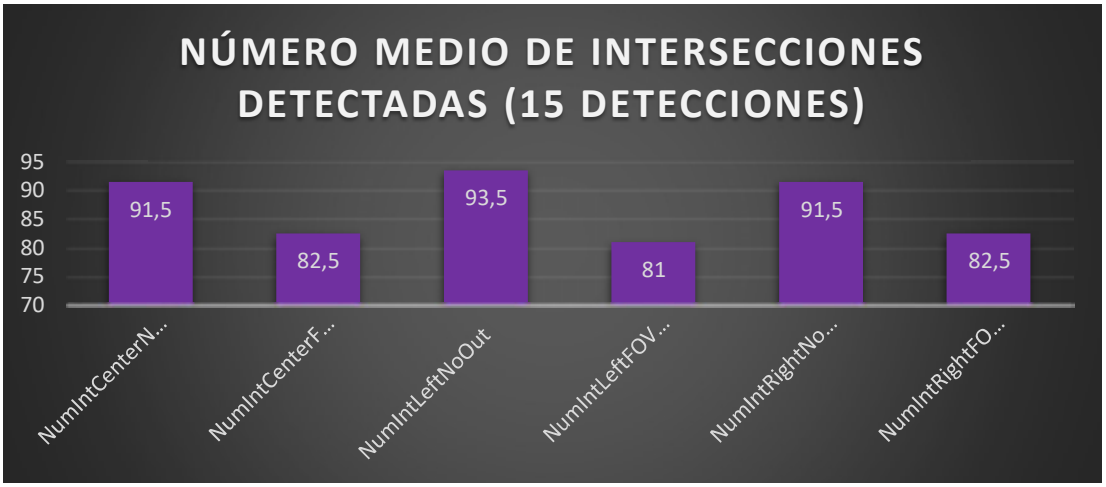

Figura 26: Gráfica del número medio de intersecciones detectadas para 15 detecciones

Como se ha podido comprobar, el descarte de intersecciones depende directamente del número de detecciones realizadas, siendo mucho más probable obtener puntos de intersección sin utilidad en objetos con un gran número de detecciones y por lo tanto un gran número de rectas.

Solapamiento entre figura predicha y figura real del objeto

En lo referido al porcentaje de solapamiento, en el caso de hacer uso de 6 detecciones presenta unos buenos resultados, siendo el modo "FOVNoOut" el modo que presenta una mayor precisión en sus resultados.

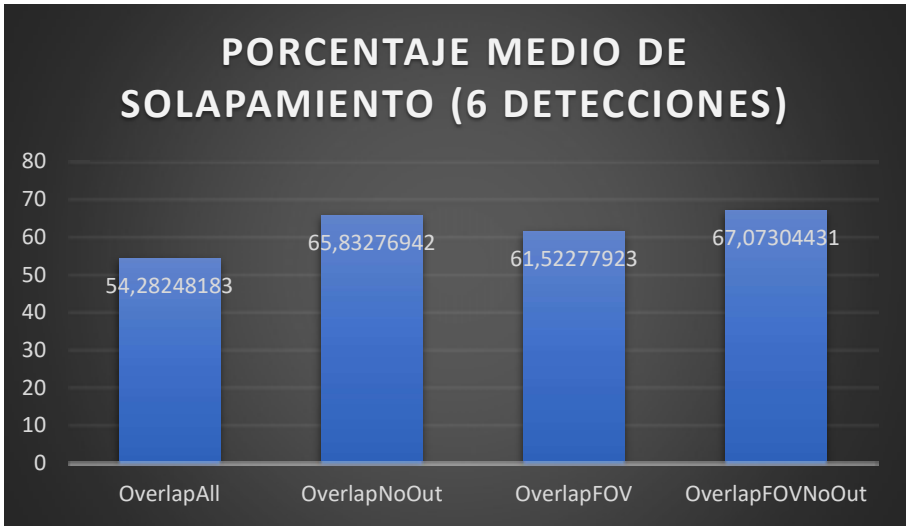

Figura 27: Gráfica del porcentaje medio de solapamiento para 6 detecciones

En cuanto al uso de 15 detecciones obtenemos unos resultados considerablemente peores, llegando a obtener un 0% de solapamiento en el modo FOV, lo que supone una predicción de la posición del objeto que no llega a colisionar o solaparse con la posición real.

Esto se debe a que al hacer uso de un gran número de intersecciones se cuenta con un gran número de intersecciones e información que carece de relevancia real para predecir la predicción, la cual es complejo reconocer y descartar en su totalidad. De este modo, en base a dicha información sin relevancia, se generan predicciones incorrectas alejadas de la realidad.

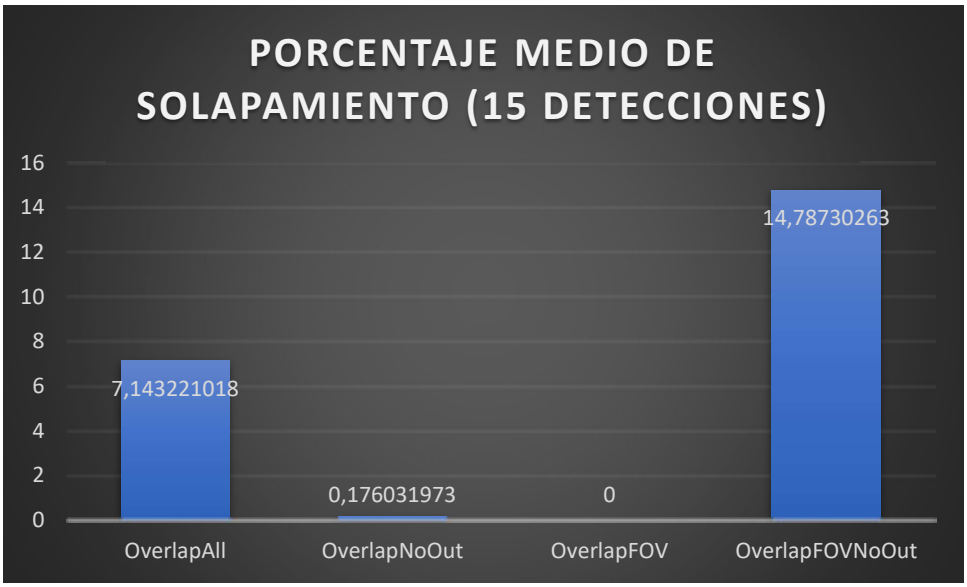

Figura 28: Gráfica del porcentaje medio de solapamiento para 15 detecciones

Como se ha podido verificar, el algoritmo presenta resultados considerablemente mejores en el caso de hacer uso de un número bajo o moderado de detecciones, perdiendo el algoritmo eficacia de predicción en casos con un alto número de detecciones.

Error de rotación con respecto al objeto real

Dentro de esta sección se miden los resultados referidos al error de rotación presentado por la predicción objeto. La rotación es un factor relevante de la predicción ya que no solo se debe solo estimar la posición del objeto en el entorno, sino que se debe conocer su orientación en el mismo.

Los resultados obtenidos por los objetos con 6 detecciones tomadas presentan buenos resultados de error, especialmente en los casos donde las intersecciones fuera de rango (outliners) son descartadas. Gracias a este descarte de información innecesaria el error es reducido a menos de la mitad, obteniendo el mejor resultado el presentado por el modo "NoOut", seguido de cerca por el modo "FOVNoOut".

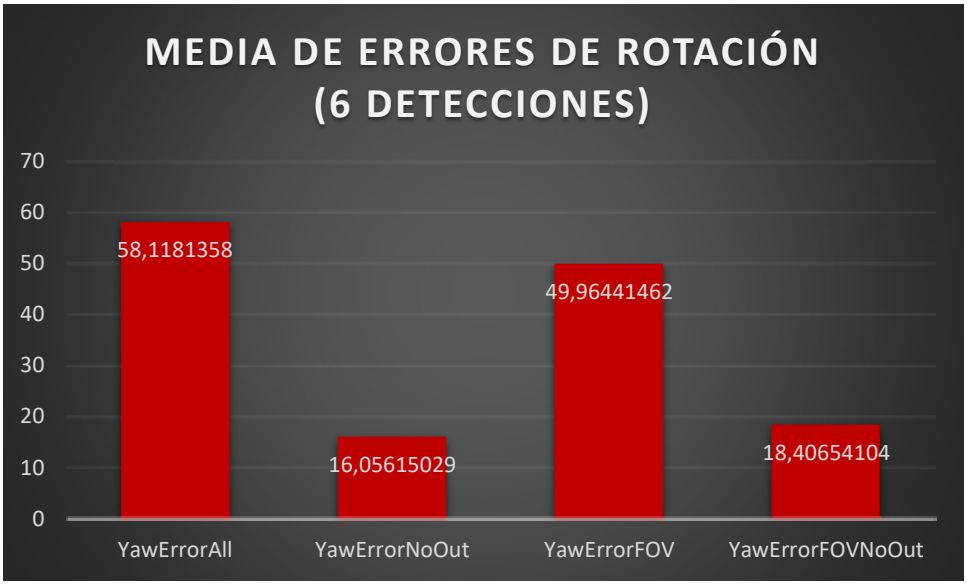

Figura 29: Gráfica de la media de errores de rotación para 6 detecciones

En lo referido al caso de los objetos que han tenido 15 detecciones, en ellos se presenta un error considerablemente elevado, derivado de los inconvenientes vistos en el análisis del porcentaje de solapamiento.

Como se puede apreciar, al hacer uso de un gran número de rectas de detección el mejor resultado en la predicción de la rotación se da con el modo "All", el cual no tiene en cuenta ningún tipo de filtro para descartar puntos de intersección y tomando cualquier intersección como válida.

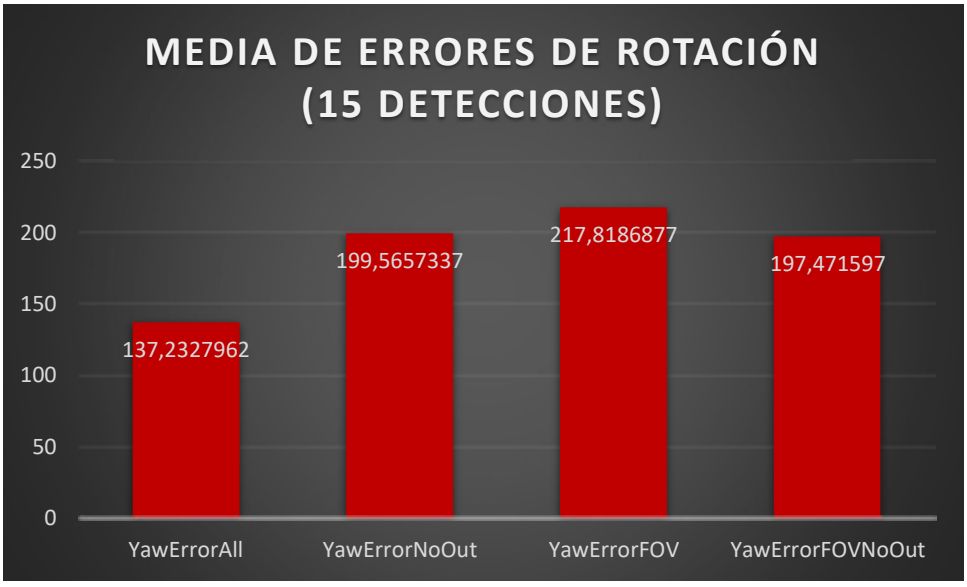

Figura 30: Gráfica de la media de errores de rotación para 15 detecciones

Como se ha podido comprobar en este estudio el hecho de hacer uso de un gran número de detecciones empeora en gran medida la predicción, pasando las mejoras en el cálculo de intersecciones de esencial en el caso de 6 detecciones a contraproducente en el caso de 15.

4.5.2 Casos especiales

El algoritmo pese a obtener una predicción cercana a la posición real del objeto en gran parte de las escenas, presenta algunos errores de imprecisión en algunos casos puntuales.

Un caso especial de predicción son los escenarios donde las detecciones realizadas al objeto son desde una posición alejada de este, lo que supone que sus rectas de detección colisionen en repetidas ocasiones generando falsos puntos de predicción debido a las intersecciones entre estas rectas.

Como ya se ha visto en el apartado anterior, al existir un gran número de intersecciones en una zona el algoritmo valida el punto de la intersección como una predicción valida, por lo que este tipo de casos pueden derivar en errores de predicción considerablemente alejados de la posición real del objeto.

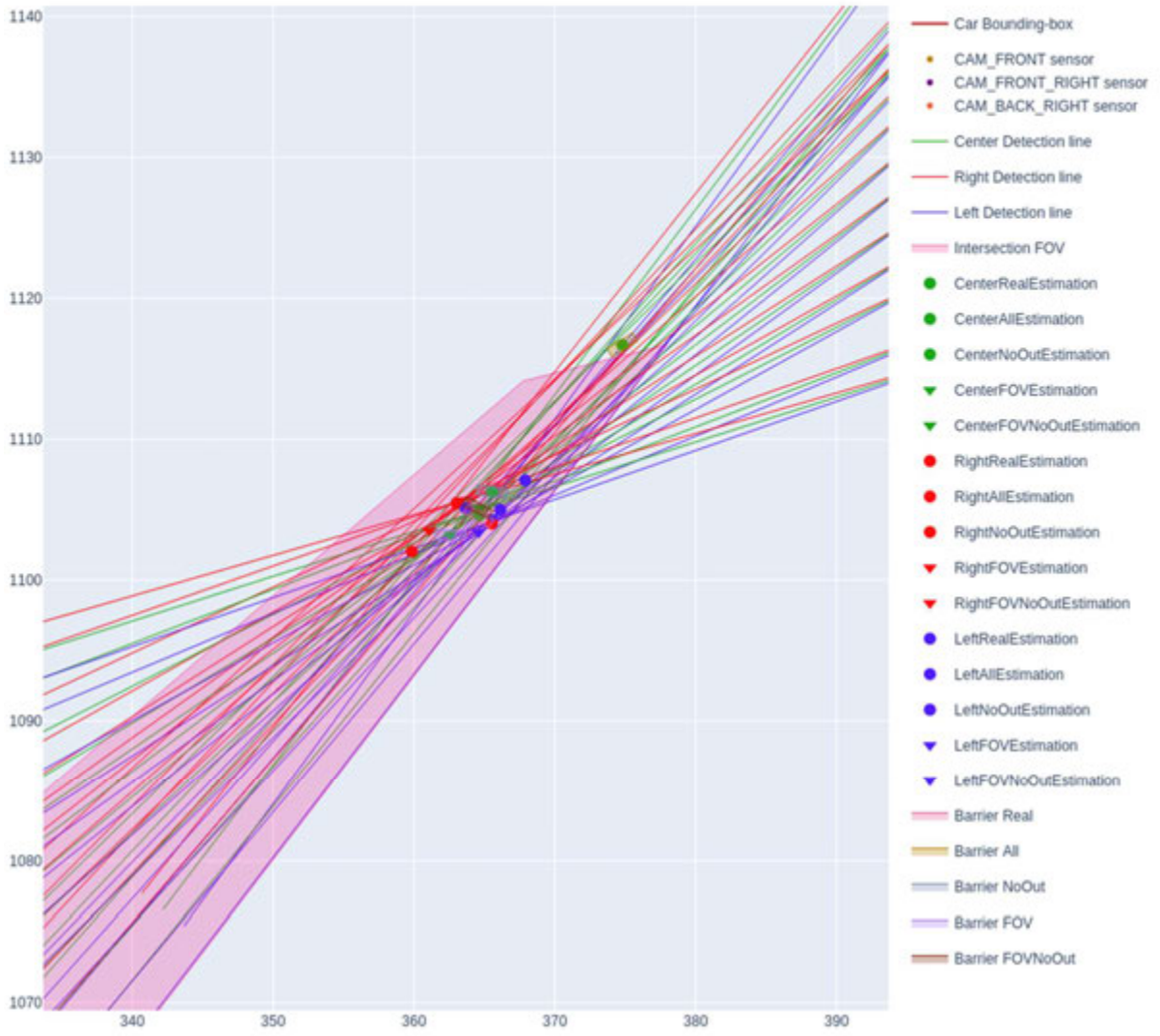

Figura 31: Caso especial de detecciones lejanas

Otro tipo de caso especial es el de los escenarios donde el objeto es captado únicamente por la camara delantera o trasera desde el vehículo, contando con un angulo de las rectas de detección entre cada detección tomada muy bajo. Esto genera el mismo efecto que las detecciones lejanas, generandose numerosas intersecciones que derivan en falsos puntos de predicción del objeto y un error en la estimación de la posición del objeto con respecto a su posición real.

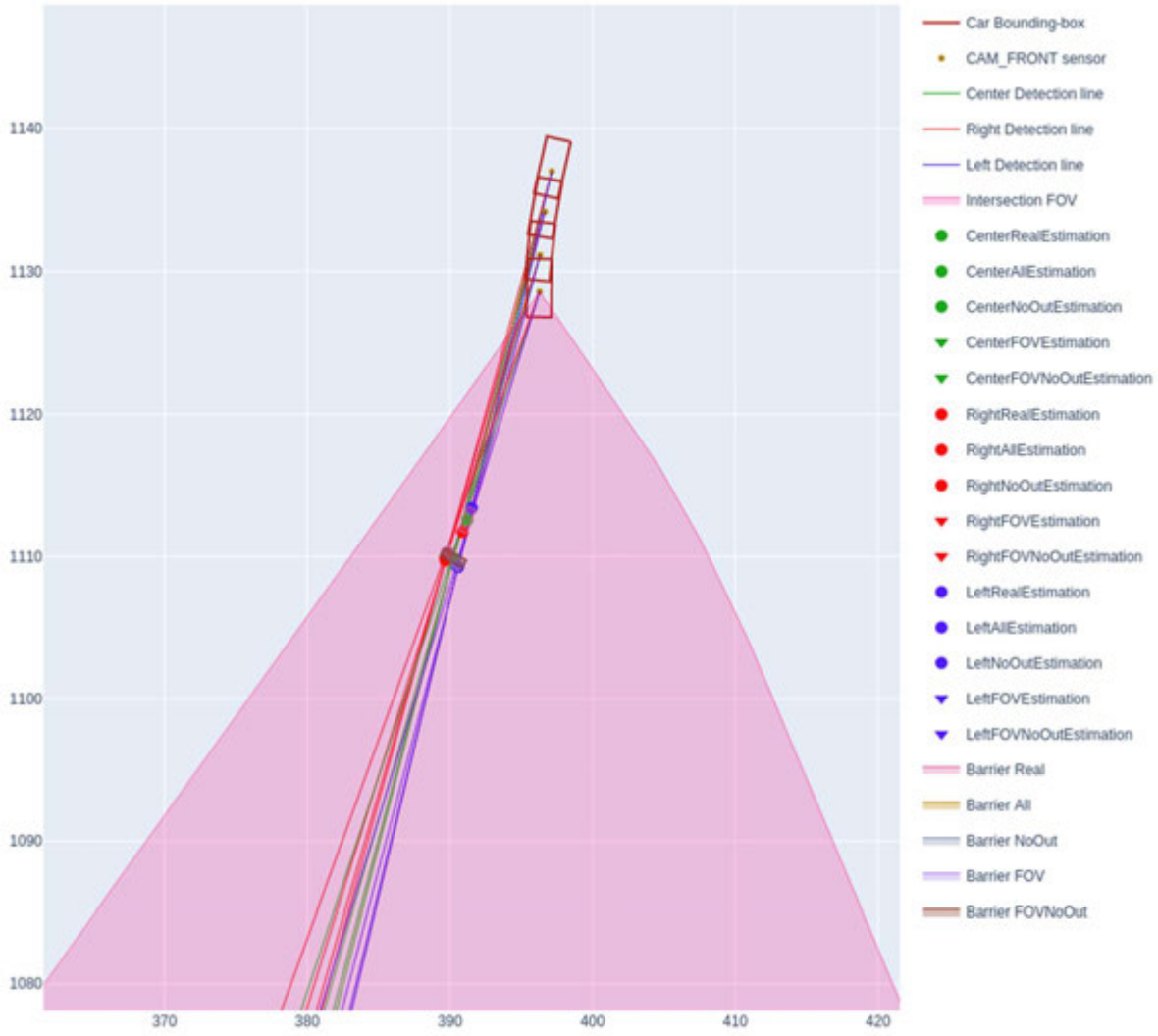

Figura 32: Caso especial de detecciones únicamente frontales

Como se ha podido comprobar este tipo de casos se trata de situaciones puntuales que no afectan en gran medida a la eficacia del algoritmo, ya que fuera de estos casos, los resultados obtenidos como se ha podido comprobar tienen un buen nivel de precisión. Sin embargo, este tipo de casos especiales deberán ser tomados en cuenta de cara a posibles mejoras del algoritmo o trabajos futuros.

5. PLANIFICACIÓN Y PRESUPUESTO

5.1 Metodología de trabajo

En este bloque se presenta en la evolución del desarrollo de las diferentes tareas planteadas para el trabajo, así como la duración de estas y los recursos necesarios para llevar estas a cabo:

- **Tarea 1:** Especificación de objetivos y búsqueda de recursos relacionados.
  - Descripción: establecer un punto inicial y un tema principal para el trabajo e investigar sobre él para definir el desarrollo de este.
  - Número de horas: 8 horas
  - Duración: 7 febrero – 14 febrero
  - Recursos humanos utilizados: estudiante de ingeniería informática.
  - Recursos tecnológicos utilizados: portátil, Microsoft Word y Zotero.

- **Tarea 2:** Búsqueda y elección de una base de datos.
  - Descripción: búsqueda de diferentes bases de datos y comparación entre sí para seleccionar una base de datos final cuya información cumpla con las necesidades de la realización del proyecto.
  - Número de horas: 8 horas
  - Duración: 14 febrero – 21 febrero
  - Recursos humanos utilizados: estudiante de ingeniería informática.
  - Recursos tecnológicos utilizados: portátil.

- **Tarea 3:** Estudio y filtrado de datos de la base de datos seleccionada.
  - Descripción: estudio de la información contenida tanto dentro de la base de datos como la obtenida mediante la detección de objetos, así como la realización de una selección de la información útil para el desarrollo del trabajo, realizando un filtrado y extracción de dichos datos para su posterior uso durante el proyecto.
  - Número de horas: 30 horas
  - Duración: 21 febrero – 24 marzo
  - Recursos humanos utilizados: estudiante de ingeniería informática.
  - Recursos tecnológicos utilizados: portátil, Google Drive, Google Collab y Microsoft Office.

- **Tarea 4:** Redacción de la memoria.
  - Descripción: redacción de los tres primeros capítulos de la memoria para documentar el trabajo realizado hasta la fecha.
  - Número de horas: 10 horas
  - Duración: 24 marzo – 14 abril
  - Recursos humanos utilizados: estudiante de ingeniería informática.
  - Recursos tecnológicos utilizados: portátil, Microsoft Office y Zotero.

- **Tarea 5:** Estudio y comparación de algoritmos de detección y geolocalización de objetos.
    - Descripción: realizar un estudio sobre los posibles algoritmos que permitan abarcar el problema, así como cuál de estos resulta ser la mejor opción.
    - Número de horas: 10 horas
    - Duración: 14 abril – 21 abril
    - Recursos humanos utilizados: estudiante de ingeniería informática.
    - Recursos tecnológicos utilizados: portátil y Microsoft Office.

- **Tarea 6:** Elaboración y programación del algoritmo de geolocalización de objetos seleccionado.
    - Descripción: desarrollar en código Python el algoritmo seleccionado para la obtención de la ubicación de objetos en el plano 3D.
    - Número de horas: 35 horas
    - Duración: 21 abril – 19 mayo
    - Recursos humanos utilizados: estudiante de ingeniería informática.
    - Recursos tecnológicos utilizados: portátil, Google Drive, Google Collab y Microsoft Office.

- **Tarea 7:** Experimentación y validación del algoritmo desarrollado.
    - Descripción: validar el correcto funcionamiento del algoritmo, así como medir su eficacia en la estimación de la posición del objeto con respecto a su posición real.
    - Número de horas: 25 horas
    - Duración: 2 junio – 23 junio
    - Recursos humanos utilizados: estudiante de ingeniería informática.
    - Recursos tecnológicos utilizados: portátil, Google Collab y Microsoft Office.

- **Tarea 8:** Documentar los contenidos del diseño e implementación del proyecto en la memoria.
    - Descripción: documentar las fases de diseño e implementación que abarcan desde la búsqueda de una base de datos y su estudio, hasta el diseño, desarrollo y análisis de resultados del algoritmo desarrollado.
    - Número de horas: 40 horas
    - Duración: 23 junio – 11 agosto
    - Recursos humanos utilizados: estudiante de ingeniería informática.
    - Recursos tecnológicos utilizados: portátil y Microsoft Office.

- **Tarea 9:** Finalizar y revisar los diferentes apartados de la memoria.
    - Descripción: documentar los apartados restantes del documento, así como revisar todo el contenido realizado-
    - Número de horas: 15 horas
    - Duración: 11 agosto – 25 agosto

- Recursos humanos utilizados: estudiante de ingeniería informática.
- Recursos tecnológicos utilizados: portátil y Microsoft Office.

5.2 Diagrama de Gantt

En este apartado se muestra de manera gráfica la planificación previamente explicada en el apartado anterior mediante el uso de un diagrama Gantt:

Figura 33: Diagrama de Gantt del desarrollo del TFG

[31]

5.3 Presupuesto

A continuación, se presenta el presupuesto de gasto necesario para la realización del trabajo, el cual abarca desde los equipos informáticos utilizados, hasta las herramientas software, teniendo también en cuenta costes indirectos como la luz y el internet.

Equipos informáticos

El trabajo ha tenido una duración de 7 meses, habiéndose utilizado equipos concretos para el desarrollo de este. El precio mostrado en la tabla se basa en una amortización a unos 6 años, que es el tiempo de vida útil aproximado de este tipo de hardware.

Descripción	Precio/unidad (21% de IVA Incluido)	Años de vida	Amortización	Total (7 meses)
Portátil: Lenovo Yoga 510 / Procesador Intel Core i5 de 6a generación / 8GB de RAM / 256GB SSD	550,55 €	6	7,65 €/mes	53,55 €
Sobremesa: AMD Ryzen 7 3800XT / 32GB de RAM / 500GB SSD / 1TB HDD	1200,25 €	6	16,67 €/mes	116,69 €
Total:			24,32 €/mes	170,24 €

Tabla 16: Presupuesto de los equipos informáticos utilizados

Herramientas de software

En lo referido al coste de herramientas de software, se ha hecho uso de diversos programas para el desarrollo del proyecto, siendo algunas de estas gratuitas. Para las herramientas de pago, se tendrá en cuenta su coste mensual, así como su precio total tras 7 meses de uso.

Descripción	Cantidad	Precio (21% de IVA incluido)	Total (7 meses)
Visual Studio Code	1	0 €/mes	0 €
Microsoft 365 Empresa Estándar	1	10,5 €/mes	73,5 €
Google Drive 2TB al mes	1	9,99 €/mes	69,93 €
Google Collab	1	0 €/mes	0 €
Total:		20,49 €/mes	143,43 €

Tabla 17: Presupuesto de las herramientas de software utilizadas

Costes de personal

Durante el desarrollo de este trabajo se ha contado con un total de dos trabajadores, siendo estos un ingeniero informático y un jefe de proyecto software. En base a su sueldo y a las horas trabajadas a lo largo de los meses se obtiene el siguiente presupuesto de costes:

Descripción	Coste	Horas de trabajo	Total
Jefe de proyecto	25 €/hora	50	1250 €
Ingeniero informático	16 €/hora	131	2096 €
Total:	-	181	3346 €

Tabla 18: Presupuesto de los costes de personal

Costes indirectos

Debido a haberse realizado el trabajo de forma telemática, se deben tener en cuenta costes indirectos como el servicio mensual de internet, así como el gasto de luz.

Descripción	Coste (21% de IVA incluido)	Total (7 meses)
Internet Fibra simétrica 300Mb	29,90 €/mes	209,3 €
Factura promedio de luz para una persona.	35,48 €/mes	248,36 €
Total:	65,38 €/mes	457,66 €

Tabla 19: Presupuesto de los costes indirectos

Costes totales

A continuación, se muestra el presupuesto global en base a todos los costes presentados anteriormente, este coste final se mostrará tanto con su valor sin IVA como con este valor incluido.

Descripción	Coste
Coste de equipos informáticos	170,24 €
Herramientas de software	143,43 €
Costes de personal	3346 €
Costes indirectos	457,66 €
Total:	4117,33 €
Total sin IVA:	3252,69 €

Tabla 20: Presupuesto de los costes totales

Por lo tanto, el presupuesto de costes total asciende a TRES MIL DOSCIENTOS CINCUENTA Y DOS EUROS con SESEINTA Y NUEVE CÉNTIMOS.

6. CONCLUSIONES Y TRABAJOS FUTUROS

6.1 Conclusiones

Como conclusión principal de este trabajo podemos afirmar que se ha cumplido el objetivo principal del mismo, obteniendo un algoritmo de triangulación de objetos que presenta unos buenos resultados en términos generales, sin llegar a ser perfecto en su predicción para algunos casos concretos.

En lo referido al planteamiento inicial, debido a diferentes problemáticas en la detección e identificación de cada objeto detectado, se ha descartado el hecho de incluir una Red Convolucional capaz de detectar y clasificar los objetos en imágenes dentro del algoritmo, contando con estos datos de manera inicial a partir del dataset utilizado en el trabajo.

Este trabajo se ha basado en gran medida en la investigación y el estudio de diferentes técnicas relacionadas con la visión artificial, así como la búsqueda y el análisis de datos a utilizar, así como su filtrado para su uso dentro del algoritmo. Además de esto, una tarea fundamental en el desarrollo del trabajo ha sido el diseño del propio algoritmo, así como la aplicación de las diferentes técnicas estudiadas y la implementación y manejo del conjunto de datos obtenido dentro de un programa implementado en código Python.

Dentro de la solución inicial obtenida para el algoritmo, se han ido realizando diferentes mejoras de forma iterativa para mejorar la precisión de los resultados. Estas mejoras se han dado principalmente en los siguientes procesos:

- **Filtrado de detecciones al objeto:** dentro de esta funcionalidad en base a los resultados obtenidos se han ido descartando diversas detecciones que producían ruido en los datos. Este tipo de detecciones se basaban principalmente en detecciones tomadas desde una detección muy lejana del objeto y casos donde la bounding box del objeto aparece cortada en un extremo de la imagen, faltando por lo tanto datos para generar las tres rectas de detección.

- **Filtrado en la obtención de intersecciones de las rectas de detección:** en esta funcionalidad del algoritmo inicialmente se tenían en cuenta todas las intersecciones producidas por las diferentes rectas obtenidas. En base a estos puntos se obtiene un valor medio para determinar un rango de posiciones donde era más probable que se encontrara la posición real del objeto.

  Sin embargo, para ganar precisión y eliminar puntos que no aportaban información relevante se iteraron diferentes métodos para filtrar que intersecciones tener en cuenta para el cálculo de las coordenadas finales de la predicción del objeto. Este filtro se basa principalmente en tener en cuenta únicamente los puntos de intersección que se encuentren en el área donde intersecan los diferentes campos de visión de las detecciones realizadas, complementándose este filtro con el hecho de descartar las intersecciones que se encuentren fuera del rango del valor medio calculado (outliners).

Para concluir, a título personal, considero que el tema del computer vision o visión artificial es un tema realmente interesante y con un gran potencial de cara al futuro, especialmente en el ámbito de la conducción autónoma. Debido a ello, este campo ha sido el ámbito donde más se ha centrado el trabajo desarrollándose principalmente el algoritmo para detecciones tomadas desde vehículos a objetos presentes durante el trayecto en carretera a lo largo de un determinado recorrido.

6.2 Trabajos futuros

A lo largo del proyecto se han tomado diferentes decisiones que han afectado al resultado final y sus resultados, pudiendo mejorar e implementar ciertos aspectos de cara a futuros trabajos relacionados.

En primer lugar, de cara a futuros proyectos se podría implementar una red convolucional capaz de identificar un determinado objeto mediante imágenes para el algoritmo de triangulación, sin tener la necesidad de contar con el posicionamiento de las bounding boxes en imágenes de forma previa. Además, esta red de neuronas convolucional deberá de contar con un método para identificar inequívocamente cada objeto detectado a lo largo de las imágenes, para evitar de esta forma la triangulación en base a diferentes detecciones de dos objetos diferentes como si se tratase de uno mismo.

Esta red de neuronas convolucional supondría que el algoritmo tuviera una mayor complejidad y fuese más generalizable de cara a diferentes conjuntos de datos, evitando el hecho de tener que contar con objetos detectados previamente en las imágenes e identificados a lo largo de los fotogramas.

Otra mejora de cara a futuro es el hecho de utilizar unas líneas de detección horizontales en las imágenes donde se detectan objetos, permitiendo de esta forma obtener la localización de objetos en suspensión. Relacionado a esto se podría evolucionar el algoritmo de detección y triangulación a una vista 3D, descartando de esta manera la vista aérea desarrollada, donde gracias a estas líneas horizontales se podría determinar la altura de los objetos, así como si se trata de objetos suspendidos en el aire como es el caso de los semáforos o algunas señales.

De este modo se obtendría unas predicciones más completas que abarcarían una mayor cantidad y variedad de objetos a geolocalizar, obteniendo datos como su altura o elevación con respecto al suelo.

Una posible mejora de cara al futuro para el algoritmo sería el hecho de incorporar la posibilidad de hacer uso de herramientas GPS como es el caso de Google Maps, donde los objetos detectados pudiesen ser dibujados dentro de los mapas en su posición predicha. Esto supondría proporcionar de más información a estos navegadores con información útil para el usuario, ya sea por señalizaciones o lugares de interés indicados dentro del mapa en su respectiva posición.

De cara al ámbito de la recreación de entornos de manera virtual, de cara a futuro se podría implementar la capacidad de hacer uso de herramientas gráficas como Unreal Engine 4, en la cual se pudiesen extrapolar los datos de posicionamiento obtenidos de los diferentes entornos a esta herramienta con el fin de recrear un determinado escenario con sus objetos y la distribución de estos dentro del mismo.

Finalmente, como pequeños cambios a mejorar del algoritmo de cara a trabajos futuros, sería el hecho de modificar el algoritmo para obtener una mayor precisión en los casos especiales previamente comentados en el apartado 4.5.2, donde debido a la lejanía con el objeto o ángulo de giro del vehículo en las diferentes detecciones se generan un gran número de colisiones de las rectas de detección debido a su cercanía entre sí, lo que provocan una predicción menos precisa.

7. BIBLIOGRAFÍA

[1] «BOE.es - BOE-A-1978-31229 Constitución Española.» https://www.boe.es/buscar/act.php?id=BOE-A-1978-31229 (accedido jun. 10, 2021).

[2] «BOE.es - BOE-A-1982-11196 Ley Orgánica 1/1982, de 5 de mayo, de protección civil del derecho al honor, a la intimidad personal y familiar y a la propia imagen.» https://www.boe.es/buscar/act.php?id=BOE-A-1982-11196 (accedido jun. 10, 2021).

[3] «REGLAMENTO (UE) 2016/ 679 DEL PARLAMENTO EUROPEO Y DEL CONSEJO - de 27 de abril de 2016 - relativo a la protección de las personas físicas en lo que respecta al tratamiento de datos personales y a la libre circulación de estos datos y por el que se deroga la Directiva 95/ 46/ CE (Reglamento general de protección de datos)», p. 88.

[4] J. Peters, *Foundations of Computer Vision*, vol. 124. 2017. doi: 10.1007/978-3-319-52483-2.

[5] 330ohms, «¿Cómo detectar objetos con YOLO?», *330ohms*, nov. 17, 2020. https://blog.330ohms.com/2020/11/17/deteccion-de-objetos-con-yolo/ (accedido jun. 05, 2021).

[6] «Modelos de Detección de Objetos | Aprende Machine Learning». https://www.aprendemachinelearning.com/modelos-de-deteccion-de-objetos/ (accedido jun. 21, 2021).

[7] «Software de diagramación en línea y solución visual», *Lucidchart*. https://www.lucidchart.com/pages/es (accedido ago. 16, 2021).

[8] «Sliding Windows for Object Detection with Python and OpenCV», *PyImageSearch*, mar. 23, 2015. https://www.pyimagesearch.com/2015/03/23/sliding-windows-for-object-detection-with-python-and-opencv/ (accedido jun. 21, 2021).

[9] rohan.arora, «Convolutional implementation of the sliding window algorithm», *Medium*, mar. 06, 2020. https://medium.com/ai-quest/convolutional-implementation-of-the-sliding-window-algorithm-db93a49f99a0 (accedido jun. 21, 2021).

[10] E. Adelson, C. Anderson, J. Bergen, P. Burt, y J. Ogden, «Pyramid Methods in Image Processing», *RCA Eng*, vol. 29, nov. 1983.

[11] B. Chandra, «A Beginners Guide to Computer Vision (Part 4)- Pyramid», *Medium*, abr. 27, 2020. https://medium.com/analytics-vidhya/a-beginners-guide-to-computer-vision-part-4-pyramid-3640edeffb00 (accedido jun. 21, 2021).

[12] S. K, «Non-maximum Suppression (NMS)», *Medium*, abr. 30, 2021. https://towardsdatascience.com/non-maximum-suppression-nms-93ce178e177c (accedido jun. 20, 2021).

[13] E. Gutierrez Alegre, M. Pajares, y A. de la Escalera Hueso, *Conceptos y métodos en visión por computador*. España: s.l., 2016.

[14] V. Varshney, «LiDAR: The Eyes of an Autonomous Vehicle», *Medium*, sep. 19, 2019. https://medium.com/swlh/lidar-the-eyes-of-an-autonomous-vehicle-82c6252d1101 (accedido jun. 05, 2021).

[15] P. Chazette, J. Totems, L. Hespel, y J.-S. Bailly, «Principle and Physics of the LiDAR Measurement», en *Optical Remote Sensing of Land Surface: Techniques and Methods*, 2016, pp. 201-247. doi: 10.1016/B978-1-78548-102-4.50005-3.

[16] «Introduction to Epipolar Geometry and Stereo Vision | Learn OpenCV», dic. 28, 2020. https://learnopencv.com/introduction-to-epipolar-geometry-and-stereo-vision/ (accedido jun. 05, 2021).

[17] F. Cordova y I. Brilakis, «On Site 3D Vision Tracking of Construction Personnel», ene. 2008, pp. 809-820.

[18] Andrea Navarro, «Redes neuronales artificiales ¿Qué son?», *Junco TIC*, jul. 29, 2016. https://juncotic.com/redes-neuronales-artificiales-que-son/ (accedido jun. 05, 2021).

[19] K. Suzuki, *ARTIFICIAL NEURAL NETWORKS – ARCHITECTURES AND APPLICATIONS*. 2013.

[20] S. Saha, «A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way», *Medium*, dic. 17, 2018. https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53 (accedido jun. 05, 2021).

[21] A. Ghosh, A. Sufian, F. Sultana, A. Chakrabarti, y D. De, «Fundamental Concepts of Convolutional Neural Network», 2020, pp. 519-567. doi: 10.1007/978-3-030-32644-9_36.

[22] «Data collection of WGS 84 information — or is it?», *GPS World*, nov. 02, 2016. https://www.gpsworld.com/data-collection-of-wgs-84-information-or-is-it/ (accedido ago. 16, 2021).

[23] M. Li y W. Yao, «3D MAP SYSTEM FOR TREE MONITORING IN HONG KONG USING GOOGLE STREET VIEW IMAGERY AND DEEP LEARNING», *ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci.*, vol. V-3-2020, pp. 765-772, ago. 2020, doi: 10.5194/isprs-annals-V-3-2020-765-2020.

[24] W. Zhang, C. Witharana, W. Li, C. Zhang, X. Li, y J. Parent, «Using Deep Learning to Identify Utility Poles with Crossarms and Estimate Their Locations from Google Street View Images», *Sensors*, vol. 18, p. 2484, ago. 2018, doi: 10.3390/s18082484.

[25] J. D. Wegner, S. Branson, D. Hall, K. Schindler, y P. Perona, «Cataloging Public Objects Using Aerial and Street-Level Images — Urban Trees», en *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, jun. 2016, pp. 6014-6023. doi: 10.1109/CVPR.2016.647.

[26] V. A. Krylov, E. Kenny, y R. Dahyot, «Automatic Discovery and Geotagging of Objects from Street View Imagery», *Remote Sens.*, vol. 10, ago. 2017, doi: 10.3390/rs10050661.

[27] «nuScenes dataset - Nuscenes». https://www.nuscenes.org/nuscenes (accedido jun. 05, 2021).

[28] «PandaSet - Open-source Dataset for Self-driving Cars». www.pandaset.org (accedido jun. 05, 2021).

[29] «KITTI-360». http://www.cvlibs.net/datasets/kitti-360/ (accedido jun. 05, 2021).

[30] «Open Dataset», *Waymo*. https://waymo.com/open/ (accedido jun. 05, 2021).

[31] «Creador de Infografías Gratuito», *Venngage*. https://venngage.com/ (accedido ago. 16, 2021).

A. ANEXO I: EXTENDED ABSTRACT

A.1 Introduction

The field of computer vision has gained relevance over the years, especially in the field of object detection through images, being used in different fields such as autonomous driving, health, surface scanning, etc.

The objective of this work is to obtain an algorithm capable of detecting and geolocating certain objects in real space through the use of different triangulation techniques and a certain dataset that provides the necessary data. All this will be given through the perspective of a moving vehicle, being the field of autonomous driving the main focus of this work.

The development of the project will be given by different main objectives that will cover different phases of work:

- Conduct a study on different datasets containing the desired information for the development of the algorithm.
- Study and filter the information stored within the selected dataset.
- Study the possibilities of triangulation techniques to be implemented by the algorithm.
- Design, implement and evaluate the triangulation algorithm.

Regarding the main motivation of the work, this is given by the promising future that appears to have the field of autonomous driving, requiring a large number of security systems derived from computer vision techniques as in the case of this work. In addition to this field, this work could have future applications in other fields such as the geographical or historical study in virtual recreations, where this algorithm can be useful for the detection and location of objects within a given terrain.

For the development of the work, aspects related to the regulatory framework should be taken into account, as is the case of privacy in the use of images where other vehicles or different pedestrians appear. For this reason, article 18 of the Spanish Constitution [1] and article 7 of the Organic Law 1/1982 [2] should be taken into account at the level of legislation in Spain, and the European Data Protection Regulation [3] should be taken into account on the European side.

A.2 State of art

This section presents the theoretical information necessary to understand the main concepts of the main techniques used during the development of the work in the field of computer vision, as well as the use of AI techniques for the automatic recognition of objects in images.

A.2.1 Computer vision

Computer vision is based on the automatic detection of elements or objects, as well as their properties in the three-dimensional world, from one or several images taken from the environment, these properties becoming dynamic in several cases. This field covers various physical fields such as optics or photogrammetry, as well as mathematical fields such as geometry, triangulation, statistics, etc.

Therefore, through the use of computer vision techniques, a process of object recognition within images can be automated, this automation occurring through the use of AI techniques that are capable of learning. This can be supervised or unsupervised, the appearance of the different objects to subsequently identify them after having learned them.

Regarding the detection of objects within the image itself, the space occupied by the object is identified through the use of bounding boxes, which have the functionality of defining by a height and width the space occupied by the object within the image, this area of occupied space being represented by coordinates measured in pixels of the image.

Algorithms for object detection in images

To automate the process of detecting objects in images, Deep Learning techniques are used to generate trained networks capable of identifying objects autonomously after prior learning. Among these techniques, the most widely used in this field are Convolutional Neural Networks (CNN).

This object detection process is carried out in three phases: a scan of the image at different positions and scales, the execution of the classifier on the generated windows, and finally the elimination of overlapping windows within the same object.

The following algorithms and methods can be used to carry out the three phases mentioned above:

- **Sliding Windows:** this algorithm takes care of the image scanning process to detect the object, making a fixed size window along the image. The size of this window will change depending on the object to be detected in its shape. [8]
- **Gaussian Pyramid:** to take into account the cases in which there are objects of different sizes in the same image and making use of a fixed size window in the Sliding Windows algorithm, it is necessary to make use of a Gaussian Pyramid.

This pyramid is responsible for rescaling the image size so that when the Sliding Windows algorithm is applied, the window can detect these different sized objects. [10]

- **Non-maximum Suppression (NMS):** this algorithm is in charge of filtering the possible overlapping detections generated by the Sliding Windows algorithm, leaving only the detection with the highest probability of being the most accurate. [12]

3D vision

In addition to the detection of objects in images, computer vision is also applicable within a 3D space, thanks to the use of different sensors that are able to generate and represent the information.

The information within the 3D plane is presented in various forms: point cloud, depth image, polygon mesh, range image, stereoscopic image, etc.

LIDAR

LIDAR is a laser sensor whose objective is to measure the distance from its position to the nearest object or surface at a given point in space where the laser light hits. In this way the sensor generates a large number of rays in different directions obtaining point clouds that determine the composition of the environment in 3D, thus allowing to generate virtual recreations of these scenarios.

This technology has several applications: 3D surface mapping, atmospheric data acquisition, generation of topographic models, autonomous driving, etc. [14]

Triangulation of an object by computer vision

The triangulation of an object in 3D space using computer vision is based on the use of two or more images taken from different angles. To achieve this goal, certain lines are generated that extrapolate a point of the image to the real space through the image. [16]

In this way a point P within two different images can be extrapolated to the 3D plane by generating the previously mentioned lines and obtaining their intersection, all these concepts being based on epipolar geometry.

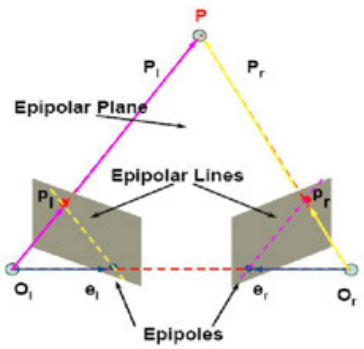

Figura 34: EA-01 Application of epipolar geometry for the triangulation of a point.

[17]

A.2.2 Artificial neural networks

Within the field of deep learning seen in previous sections are the Artificial Neuron Networks, which are responsible for simulating the behavior of neurons in a brain by learning along its layers in order to learn and automate a certain process.

Within the field of object detection in images by means of computer vision, the most popular technique within deep learning is the convolutional neural networks. This type of artificial neural network bases its operation on the use of different convolutional layers that are dedicated to extract the characteristics of the objects to be detected by means of a set of input images.

After this feature extraction, a multilayer perceptron is used to classify the objects present in the different scenes represented in images, thus obtaining the detection and classification of objects within the images.

A.2.3 Related works

3D mapping of trees in Hong Kong using Google Street View

This project aims to map the position of the different trees located along the streets of the city of Hong Kong. To achieve this goal, Google Street View is used, converting its 360º view into 2D images and obtaining the position of the trees by means of depth calculation techniques.

Michael Li y Wei Yao et al. [23]

Using Deep Learning to identify and locate light poles using Google Street View images

The purpose of this work is to perform the detection and localization of different light poles located along a route, using Google Street View and convolutional neural networks as tools for image detection.

The positioning of the poles is obtained by applying triangulation techniques through the use of various images, extrapolating 2D detection lines to 3D space.

Weixing Zhang, Chandi Witharana, Weidong Li, Chuanrong Zhang, Xiaojiang Li y Jason Parent et al. [24]

Object classification using aerial and street view imagery

This project aims to detect and locate certain objects through the use of neural networks and images taken from an aerial perspective and at street level, focusing the network on performing the identification in images.

The location of the objects on the map is obtained by means of triangulation techniques and metadata obtained from the GPS of the vehicle capturing the images.

Jan D. Wegner, Steve Branson, David Hall, Konrad Schindler y Pietro Perona et al. [25]

Automatic Discovery and Geotagging of Objects from Street View Imagery

Within this project, the objective is to automatically detect, tag and geo-locate different objects on the map by GPS through the use of computer vision techniques, images taken by Google Street View, and the use of an AI capable of identifying and classifying the objects present in these images.

To achieve this objective, we compare and contrast both triangulation using images taken from different perspectives by several cameras, and the depth calculation of a single image using convolutional neural networks.

Vladimir A. Krylov, Eamonn Kenny, Rozenn Dahyot et al. [26]

A.3 Solution development

The aim of this work is to obtain a solution that allows the detection and location of a certain object through the use of images, extrapolating its position to 3D space.

To do this, it will be necessary to find a good dataset containing the desired information, studying and filtering the information, following with the development of the triangulation algorithm that performs the detection and geolocation of the object.

A.3.1 Object detection and classification

Within the data contained in nuScenes (the dataset used in the project), the different objects detected are organized by different categories. In the case of this project, it is necessary to have an object that remains motionless throughout the frames to be able to perform the triangulation, so the object to be detected will be the traffic barriers.

As the dataset has a record of frames where each object appears, we can establish a set of images on which to apply a convolutional neural network and the techniques and algorithms seen in section 7.2.1 for image detection.

However, a major problem arises when automating this detection process, since each of the barriers detected throughout the images taken must be uniquely identified. Due to the fact that this is a very complex process to carry out, the algorithm will use this detection as input data provided by nuScenes, thus discarding the automatic identification process.

A.3.2 Solution design

The triangulation algorithm is mainly based on generating detection lines that go through the bounding box that locates the object in the 2D image and extrapolating these lines to the 3D space, in order to obtain the estimated position of the object by means of the intersection between the detection lines.

In order to carry out this process, the following prior information is available:

- **Vehicle location and rotation:** both the position of the image capturing vehicle and its rotation in 3D space must be known.
- **Location of the cameras in the vehicle and their rotation:** it is necessary to know the positioning of the cameras inside the vehicle, as well as their rotation with respect to it.
- **FOV of each camera:** the field of view that each camera covers in width must be known, this must be measured in degrees.
- **Images with their bounding box associated to the object:** it is necessary to have the set of images containing the objects located by means of a bounding box.

A.4 Solution implementation

This section documents the different studies and procedures carried out to obtain the final solution previously defined, covering the different phases of implementation of the triangulation algorithm.

A.4.1 NuScenes database

For the initial phase of the study and filtering of the data needed for the algorithm, the nuScenes dataset has been selected, which is for public and non-commercial use. This dataset has different types of information, ranging from images taken by a total of 6 cameras, to positioning metadata through the use of different radars and a LIDAR sensor. *[27]*

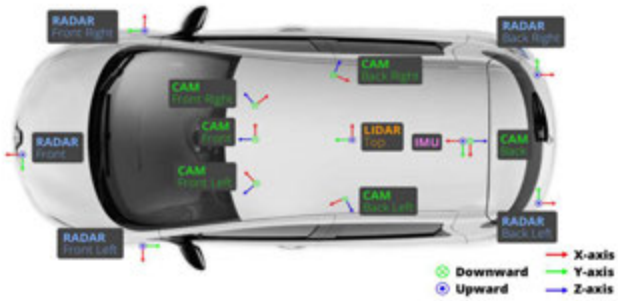

Figura 35: EA-02 Distribution and positioning of sensors in the ego vehicle.

[27]

Regarding the structuring of the information, nuScenes works in a similar way to a conventional database, dividing its data into different classes which have certain attributes, some of which are shared between different classes to relate them to each other. This division of classes is given through different JSON files that contain the attributes and relate the classes through tokens stored in the shared attributes of the different classes.

In addition to nuScenes, other datasets were taken into account throughout the development process but were finally discarded, such as Pandaset [28], Kitti-360 [29] and Waymo open [30].

A.4.2 Data filtering

Once the information contained in nuScenes has been studied, it is necessary to filter the data that are useful for this project, in particular those related to the traffic barriers present in the dataset.

To achieve this goal, a script has been developed in Google Collab to perform data extraction automatically. This extraction is based on obtaining only all the attributes related to certain objects belonging to the traffic barrier class that are found throughout the different scenes stored in the dataset.

As a result, a CSV file containing the following attributes is obtained as output:

- **Position of the bounding box corners:** the position of the upper left corner and the lower right corner of the bounding box represented in 2D within the image is obtained, these coordinates being expressed in pixels.
- **Overall vehicle position:** the exact position of the vehicle within the map generated by nuScenes is obtained in the XYZ axes, being Z always 0, because this axis is used to represent the distance of the objects with respect to the vehicle.
- **Overall vehicle rotation:** stores the rotation of the vehicle within the map generated by nuScenes, being this representation represented by a quartenion with the WXYZ axes.
- **Camera position relative to the vehicle:** the position of each of the cameras with respect to the center of the vehicle is obtained, representing it by means of the XYZ axes.
- **Camera rotation relative to the vehicle:** the rotation of the cameras used is obtained, taking as a reference this rotation to the vehicle where the cameras are placed.
- **Image size:** the size in pixels of the image is stored, since nuScenes always uses a fixed size of 1600x900 pixels, an array with this image height and width will be stored.
- **Image path:** the path where a particular frame is stored is stored.
- **Camera FOV:** if the back camera is used, 110° FOV will be stored, otherwise 70° FOV will be stored.

A.4.3 Algorithm development and operation

The algorithm has been developed in the Google Collab environment, using the Python language, having implemented the data filtering script also in this environment.

As for the algorithm operation, it is based on receiving an input CSV with the parameters specified in the section which will be processed obtaining a prediction of the object position, as well as a graphical representation of the scene with the position of the vehicle in each detection, the predicted and the real position of the object, and the detection lines with their intersections.

This process is divided into three phases:

1. **Processing of input data and generation of straight lines:** this process initially reads the CSV file containing the input data, using a loop to read all its lines, each of these lines containing the information related to a specific detection of the same object.

   As each CSV file is read, detections will be discarded if they are too far away from the object or have poor visibility of the object.

   Once the data of a file row has been read, we proceed with the generation of the detection lines, which are based on three lines that cross the bounding box present in the image of each detection, passing through its sides and center.

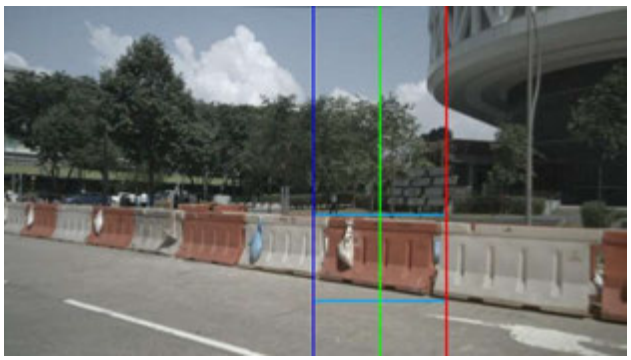

Figura 36: Example of detection lines in an image

   Finally, after having generated the detection lines and making use of data such as camera FOV, camera rotation and vehicle rotation, the rotation of these detection lines within the real space from an aerial view perspective will be obtained.

   It should be noted that these steps will be repeated iteratively for each of the detections (rows) of an input CSV file.

2. **Triangulation process to predict object positioning:** after having obtained all the detection lines of the different detections taken of the same object and having extrapolated them to an aerial view plane, we proceed with the calculation of

intersections between these lines, which will determine possible predictions of the object's position.

To filter out possible noise in the intersections, the algorithm will discard the intersections that occur outside the intersection of the different FOVs of each of the detections performed on the object. In this way, only the intersections close to the points where a greater number of intersections occur within this FOV intersection will be taken into account.

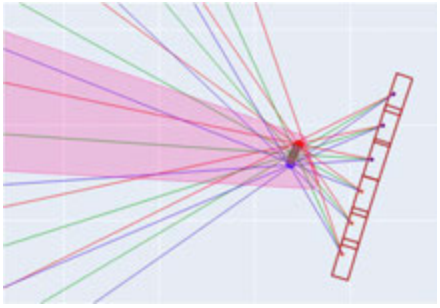

Figura 37: Intersection filtering within the intersection area between FOV

Based on the intersections between these three types of detection line, an estimation of the position of the object will be obtained by means of its center and sides, and this object can be represented in real space by means of graphic tools such as Ploty.

3. **Obtaining and representation of output data:** Once the object position prediction has been made, the algorithm generates output data that presents the prediction obtained, as well as the positioning of the objects and the vehicle graphically using the Ploty tool. In addition to this, the algorithm generates a CSV file that stores different accuracy and error metrics of the algorithm in contrast to the actual position of the object.

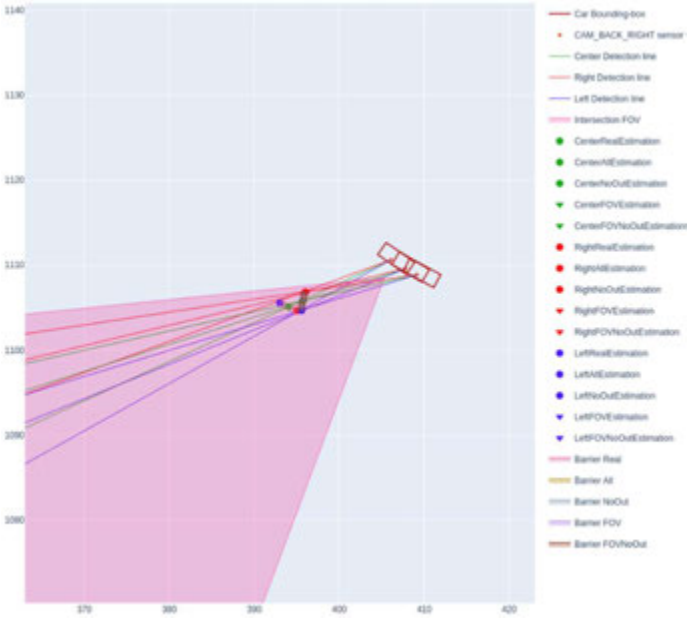

Figura 38: Aerial view recreation of a given barrier

Solution analysis and evaluation

Within this block, the prediction efficiency of the algorithm will be analyzed based on the metrics obtained from the generated output information. In addition to this measure of efficiency, we will analyze the different cases where the algorithm presents better or worse results and to what these facts could be due.

To perform this analysis, the data obtained for different objects with a total of 6 and 15 detections along a given route will be taken, averaging the parameters obtained for subsequent analysis and comparison.

For this analysis, the following errors and parameters were studied:

- **Number of intersections detected:** based on the number of average intersections detected for the case of the 6 detections, it can be seen that in the different modes of intersection detection between the detection lines there is not a great difference. This is due to not having a large number of detection lines.

  On the other hand, we have the case of objects that have had a total of 15 detections, in which there is a significant difference in the number of discards in the number of intersections by the "FOVNoOut" mode.

  Therefore, it can be concluded that the discarding of intersections depends directly on the number of detections made, being much more likely to obtain intersection points without utility in objects with a large number of detections and therefore a large number of straight lines.

- **Overlap between predicted figure and real figure of the object:** in terms of the percentage of overlapping, in the case of using 6 detections, it presents good results. The "FOVNoOut" mode is the mode with the highest accuracy in its results.

  For objects using 15 detections, we obtain considerably worse results, reaching 0% overlap in FOV mode, which means a prediction of the object's position that does not collide or overlap with the real position.

  Due to these results, it can be concluded that the algorithm presents considerably better results in the case of making use of a low or moderate number of detections, losing predictive efficiency in cases with a high number of detections.

- **Rotation error with respect to the real object:** rotation is a relevant factor in the prediction since not only the position of the object in the environment must be estimated, but also its orientation in the environment must be known.

  The results obtained for objects with 6 detections taken show good error results, especially in cases where outliners are discarded.

  Regarding the case of the objects that have had 15 detections, in them there is a considerably high error. Having the best result in the prediction of rotation with the

"All" mode, which does not take into account any type of filter to discard intersection points and taking any intersection as valid.

Due to these results we can verify that making use of a large number of detections greatly worsens the prediction, the improvement in the intersection calculation going from essential in the case of 6 detections to counterproductive in the case of 15.

A.5 Planning and budgeting

For the realization of this project, a specific plan has been followed, consisting of 9 tasks to be carried out during the months of development:

- **Task 1:** specify the initial objectives of the work and conduct research on related topics to define it (February 7 - February 14).
- **Task 2:** search for a dataset to use that meets the needs of the job. (February 14 - February 21).
- **Task 3:** to carry out a study of the selected dataset and filter its information based on the desired data. (February 21 - March 24).
- **Task 4:** draft the first three blocks of the report to document the activity carried out to date. (March 24 - April 14).
- **Task 5:** study of different object detection and geolocation algorithms using images, as well as a comparison between them. (April 14 - April 21).
- **Task 6:** design and implementation of the geolocation algorithm based on the algorithm studied. (April 21 - May 19).
- **Task 7:** experimentation and validation of the developed algorithm based on the accuracy of its results. (June 2 - June 23).
- **Task 8:** document the design and implementation of the algorithm in the report. (June 23 - August 11).
- **Task 9:** finalize the remaining sections of the report and carry out a general review of the written content. (August 11 - August 25).

Regarding the budget, taking into account the costs of computer equipment and software tools used during development, as well as indirect costs such as electricity or internet, we obtain the following total cost for 7 months of development:

Description	Cost
Cost of computer equipment	170,24 €
Software tools	143,43 €
Personnel costs	3346 €
Indirect costs	457,66 €
Total:	4117,33 €
Total excluding VAT:	3252,69 €

Tabla 21: EA–01 Budget of total costs

A.6 Conclusions and future work

Despite having variations with respect to the initial objective of the project, it can be concluded that the main objective of the development has been achieved. The result is an algorithm capable of predicting with a good result the position of an object in space by using images and various metadata.

This objective has been achieved through a long process of research and study of different techniques and methodologies used in computer vision, as well as the search, study and filtering of a dataset with the necessary information for the project. All this in order to finally design and implement a triangulation algorithm that puts into practice everything studied previously.

Throughout the development of the solution, improvements have been implemented to achieve better results. These improvements have been mainly in the filtering of detections provided by the input data and in the calculation of intersections of the detection lines.

As far as future work is concerned, the algorithm could receive various improvements or take other development routes to improve its results or cover other avenues:

- **Use of a convolutional neural network:** for future projects, a network capable of uniquely identifying each object across frames could be implemented within the algorithm itself. This would make the algorithm more generalizable to different datasets.

- **Application of horizontal detection lines:** in order to develop another approach, the use of vertical detection lines that are able to determine whether an object is suspended or how high it is could be implemented in the algorithm in the future. This could open the possibility of making use of 3D recreations instead of an area view.

- **Improvement of special cases obtained:** as an improvement for the future, the algorithm could be modified to improve the less accurate results obtained from the special cases detailed in section 7.4.3.

- **Implement the use of GPS tools:** a possible future improvement for the algorithm would be to incorporate the possibility of making use of GPS tools such as Google Maps, where the detected objects could be drawn on the maps at their predicted position.

- **Ability to make use of graphic tools:** the capacity to make use of graphic tools such as Unreal Engine 4 could be implemented, in which the positioning data obtained from the different environments could be extrapolated to this tool in order to recreate a certain scenario with its objects and their distribution within it.