Universidad del País Vasco Euskal Herriko Unibertsitatea

BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

SISTEMEN INGENIERITZA ETA AUTOMATIKA SAILA
DEPARTAMENTO DE INGENIERÍA DE SISTEMAS Y AUTOMÁTICA

MÁSTER UNIVERSITARIO EN INGENIERÍA

DE CONTROL, AUTOMATIZACIÓN Y ROBÓTICA

TRABAJO FIN DE MÁSTER

DESARROLLO DE ALGORITMOS PARA LA LOCALIZACIÓN DE VEHÍCULOS AUTÓNOMOS MEDIANTE VISUAL ODOMETRY

Estudiante	*Unzueta Martín, Unai*
Director/Directora	*Díez Sánchez, Mikel*
Departamento	*Automática y Control*
Curso académico	*2020-2021*

Bilbao, 24 Junio 2021

Resumen

En este documento se investigan diferentes métodos utilizados en el campo de la visión con el objetivo de mejorar y aportar nueva información en el desarrollo de un sistema de percepción robusto basado en la fusión de distintos sensores mediante el análisis de la información obtenida de las imágenes a través de las cámaras posicionadas en el vehículo autónomo. Para ello, se realizará un estudio de fotogramas consecutivos, que permitirá comprender el desplazamiento relativo del vehículo entre dos imágenes consecutivas. Es decir, se debe obtener suficiente información para que el algoritmo pueda proporcionar el ángulo de giro y el desplazamiento relativo (longitudinal y lateral) del automóvil autónomo entre dos fotogramas consecutivos.

Laburpena

Dokumentu honetan ikusmenaren eremuan erabilitako hainbat metodo ikertzen dira, pertzepzio-sistema sendo baten garapenean informazio berria emateko eta hobetzeko. Sistema hori sentsore desberdinen fusioan oinarritzen da, non ibilgailu autonomoan kokatutako kameren irudietatik lortutako informazioa aztertuko da. Horretarako, ondoz ondoko fotogramen azterketa egingo da, ibilgailua ondoz ondoko bi irudiren artean nola mugitzen den ulertzeko. Hau da, informazio nahikoa lortu behar da algoritmoak auto autonomoaren biraketa-angelua eta desplazamendu erlatiboa (luzetarakoa eta albokoa) eman ahal izateko elkarren segidako bi fotogramen artean.

Abstract

This paper investigates different methods used in the field of vision with the aim of improving and providing new information in the development of a robust perception system based on the fusion of different sensors by analysing the information obtained from the images through the cameras positioned in the autonomous vehicle. For this purpose, a study of consecutive frames will be carried out, which will allow understanding the relative displacement of the vehicle between two consecutive images. In other words, enough information must be obtained for the algorithm to be able to provide the yaw angle and the relative displacement (longitudinal and lateral) of the autonomous car between two consecutive frames.

Palabras Clave

Odometría Visual (Visual Odometry)

CARLA

SURF (Speeded-Up Robust Features)

RANSAC (RANdom SAmple Consensus)

Índice de Tablas

Índice de Ilustraciones

Índice de Gráficas

Índice de Acrónimos

A/D (Analógico/Digital)

API (Application Programming Interfaces)

CCD (Charge-Coupled Device)

CMOS (Complementary Metal-Oxide Semiconductor)

FoV (*Field of Views)*

GPS (Global Positioning System)

IMU (Inertial Measurement Unit)

LIDAR (Light Detection & Ranging)

OF (Optical Flow)

RANSAC (RANdom SAmple Consensus)

RGB-D (Red/Green/Blue – Depth)

SAE (Sociedad de Ingenieros Automotrices)

SFM (Structure From Motion)

SIFT (*Scale-Invariante Feature Transform*)

SKF (Switching Kalman Filter)

SURF (*Speeded-Up Robust Features*)

SVO (Semi-direct Visual Odometry)

VO (Visual Odometry)

Índice de Contenido

MEMORIA

1. INTRODUCCIÓN

Los vehículos autónomos han atraído una gran atención en los últimos años originando que la mayoría de las empresas automovilísticas se encuentren desarrollando sus propios conceptos de vehículos autónomos. Los automóviles sin conductor son vehículos que utilizan tecnología digital para trasladarse sin ninguna intervención humana. En otras palabras, pueden conducir y navegar por la carretera detectando las influencias ambientales.

Los coches sin conductor han sido objeto de investigación desde 1920, cuando se les llamó por primera vez "Coches Fantasmas". Estos vehículos representaron un avance tecnológico impresionante en ese momento debido a que estaban controlados de forma remota, mediante un vehículo ubicado en su parte trasera, a través de un control de radio. En la década de 1950, los pulsos eléctricos eran el método preferido de conducción remota. Estos sensores eran capaces de detectar la posición y la velocidad de otros vehículos y proporcionar información de orientación a los vehículos autónomos. El receptor de este sistema fue incorporado al GM Firebird II en 1956. Aproximadamente, al mismo tiempo, en 1958, el Chrysler Imperial fue el primer automóvil en utilizar un sistema de control de crucero. Mientras que, en 1995, el automóvil autónomo VaMP recorrió 2.000 kilómetros de forma casi autónoma.

A medida que pasaba el tiempo, más y más avances se volvían cada vez más populares y las nuevas empresas se enfrentaban al desafío de crear vehículos comerciales totalmente automatizados. Los más populares hoy en día son Mercedes-Benz, Audi, BMW, Tesla, Hyundai, etc. A pesar del tremendo progreso, los coches autónomos permitidos en la vía pública hoy en día no son completamente autónomos debido a que necesitan un conductor que deberá de permanecer atento por si fuese necesario recuperar el control del vehículo.

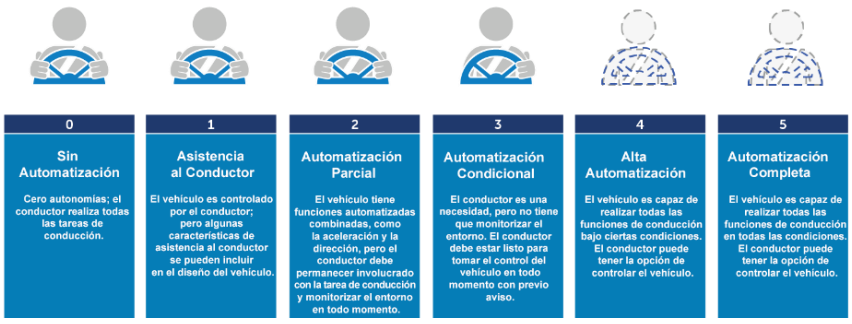

Ilustración 1: Niveles de automatización definidos por SAE

Además, los automóviles sin conductor se dividen en 5 niveles diferentes de automatización definidos por la Sociedad de Ingenieros Automotrices (SAE).

Nivel 0 – Sin automatización: en este caso, hay un 100% de presencia humana. La aceleración, el frenado y la dirección están constantemente controlados por un conductor, aunque admitan sonidos de advertencia o sistemas de seguridad. Este nivel también incluye el frenado de emergencia automatizado.

Nivel 1 – Asistencia al conductor: el ordenador nunca controla la dirección y la aceleración o el frenado simultáneo. Aunque, en determinados modos de conducción, el coche puede tomar el control del volante o de los pedales. Entre los mejores ejemplos del primer nivel se encuentran el control de crucero adaptativo o la asistencia al aparcamiento.

Nivel 2 – Automatización parcial: el conductor puede quitar las manos del volante. En este nivel, hay opciones de configuración en las que el coche puede controlar ambos pedales y el volante al mismo tiempo, pero sólo en determinadas circunstancias. Durante este tiempo, el conductor tiene que prestar atención y, si es necesario, intervenir.

Nivel 3 – Automatización condicional: se acerca a la plena autonomía. En este caso, el coche tiene un modo determinado que puede asumir la responsabilidad total de la conducción en determinadas circunstancias, pero el conductor debe recuperar el control cuando el sistema se lo pida. En este nivel, el coche puede decidir cuándo cambiar de carril y cómo responder a eventos dinámicos en la carretera utilizando al conductor humano como sistema de apoyo.

Nivel 4 – Alta automatización: es similar al nivel anterior, pero es mucho más seguro. El vehículo puede conducirse a sí mismo en las circunstancias adecuadas, y no necesita la intervención humana. Si el coche se encuentra con algo que no puede manejar, pedirá ayuda humana, pero no pondrá en peligro a los pasajeros si no hay respuesta humana.

Nivel 5 – Automatización completa: a este nivel el coche se conduce solo y la presencia humana no es una necesidad, sino una oportunidad. Todas las tareas de conducción las realiza el ordenador en cualquier carretera en cualquier circunstancia, tanto si hay un humano a bordo o no.

2. CONTEXTO

Teniendo en cuenta el amplio campo de investigación del vehículo autónomo, actualmente el desarrollo se sitúa en los niveles 4-5 de automatización definidos por SAE. Por consiguiente, este trabajo se realiza dentro de un proyecto de desarrollo e implementación de sistemas de percepción basados en la fusión de sensores en vehículos autónomos entre los que se encuentra el sensor cámara que será el elemento a desarrollar e investigar en este proyecto.

La implementación de sensores individuales es una forma útil de estudiar su rendimiento y configurarlos. Sin embargo, cada sensor individualmente tiene ventajas e inconvenientes específicos. Combinarlos proporcionará una forma sólida de gestionar la calidad y fiabilidad de los datos obtenidos de los distintos sensores, e incrementar el alcance o reducir la incertidumbre del área de estudio. Existe una gran variedad de sensores que pueden utilizarse para este fin.

LIDAR (Light Detection & Ranging)

El principio fundamental del LIDAR es calcular la profundidad del entorno midiendo la diferencia de tiempo entre la emisión de una señal luminosa y la recepción de la onda reflejada proporcionando una información de 360 grados. Esto conlleva a la utilización de dos componentes, la fuente de luz, y el fotodetector, que determinarán el alcance del sensor.

Una de las ventajas destacable es que no depende de la luz ambiental, lo que facilita la detección de objetos en el mapa creado. Tampoco tiene una limitación de alcance en cuanto a ángulos, excepto desde aquellos ángulos bastante cercanos al coche que no son captados por la perspectiva del LIDAR. El rango de longitud también es bastante bueno, cubriendo hasta 250m y con una precisión de 0,5cm.

Ilustración 2: Representación de los puntos obtenidos por Velodyne LIDAR

Radar

El radar es un sensor de alcance por radiofrecuencia que funciona con un principio similar al del LIDAR, midiendo la diferencia de tiempo entre una señal enviada y otra recibida. Esta señal será de ondas electromagnéticas en el rango de las microondas en lugar de rayos de luz. También mide la frecuencia de la onda recibida para identificar el objeto.

La principal ventaja de utilizar la tecnología radar es que no se ve afectada por las condiciones atmosféricas, como la lluvia, la nieve o el polvo, ya que la longitud de onda no está en el espectro visible. Otro punto beneficioso es el alcance, que puede variar en función de la aplicación. Además, el radar puede medir fácilmente el movimiento de los objetos circundantes. Esta última característica es especialmente útil cuando se trata de la conducción en carretera, donde hay que tener en cuenta a otros vehículos que viajan.

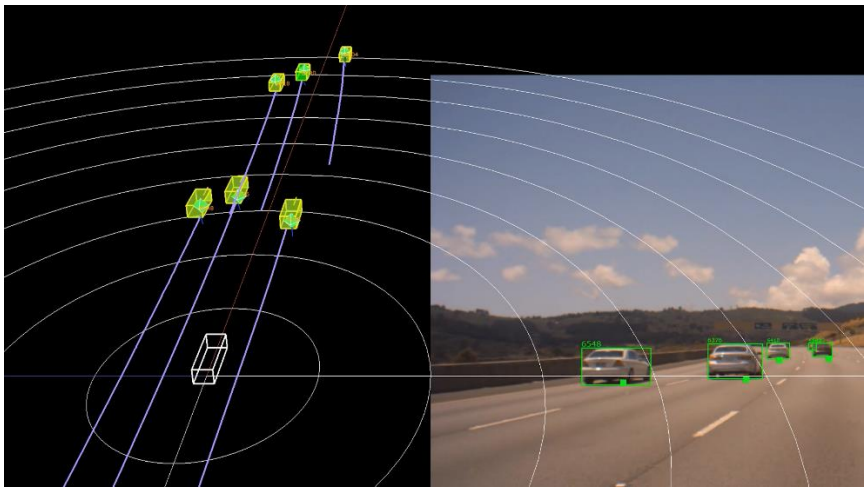

Ilustración 3: Detección de vehículos mediante el sensor Radar

Cámara

El fundamento de una cámara reside en la capacidad de generar una imagen digital del entorno capturado a partir de un sensor de luz pasivo. Estas imágenes pueden capturar muchas propiedades de los objetos alrededor del sensor, como su color y textura, así como capturar objetos estáticos y dinámicos. Además de poder medir la distancia a estos objetos mediante los algoritmos que se han desarrollado, también puede detectar objetos en el entorno, como postes telefónicos, señales de tráfico o líneas en la carretera. Todo el estudio e investigación de visión se detalla en la sección “INVESTIGACIÓN DE PROCESAMIENTO DE IMÁGENES”.

GPS (Global Positioning System)

Los GPS proporcionan información sobre la posición y la velocidad. El sistema GPS se divide en tres segmentos: espacio, control y usuario. El primero corresponde a una constelación de 24 satélites que se mueven en órbitas fijas alrededor de la Tierra, cada uno de los cuales transmite señales que serán descodificadas por el módulo receptor del GPS. El segmento de control trata de las estaciones terrestres que comunican y controlan los satélites. Y, por último, el segmento de usuario, es el receptor GPS que identifica los satélites visibles a través de los cuales se localiza en la Tierra. El tipo de mensaje que se leerá contiene información sobre los satélites, la ubicación, así como la diferencia de tiempo entre el receptor y cada satélite.

Los dispositivos GPS han mejorado notablemente su precisión desde que se inventaron, pudiéndose obtener hoy en día una precisión de centímetros. Lo más importante es que no tienen error acumulativo, ya que miden la posición absoluta.

IMU (Inertial Measurement Unit)

Las IMUs son dispositivos electrónicos compuestos por acelerómetros, giroscopios y magnetómetros que proporcionan información sobre la orientación y movimiento del vehículo. A diferencia del GPS, las IMUs tienen una gran precisión y frecuencia de actualización. Sin embargo, se consideran sensores de cálculo muerto, ya que cada estado depende de mediciones relativas y se calculan sobre estados anteriores provocando un error acumulativo. Pueden ser utilizadas por el vehículo cuando no se recibe la señal del GPS, y como forma de medir la aceleración del coche para el sistema de control.

Odometría

Los sensores de odometría estiman el cambio de posición en el tiempo donde se distinguen los codificadores incrementales y absolutos.

Los codificadores incrementales miden la posición de la rueda acoplada contando los pulsos que genera. Suelen estar construidos con dos o menos líneas de ranuras, las cuales son utilizadas para medir la velocidad o detectar la dirección del movimiento. Sin embargo, los codificadores incrementales generan errores con el tiempo.

Los codificadores absolutos capturan la posición exacta de la rueda cuando se requiere. Se construyen utilizando más canales de ranura que los codificadores incrementales, y detectan la posición de la rueda en una matriz de bits, lo que da lugar a una medición muy precisa. Sin embargo, ninguno de estos sensores mide el deslizamiento de la rueda, lo que puede aumentar los valores de error acumulados.

Un tipo común de codificadores de odometría son los codificadores ópticos basados en la detección de luz. Un fotodiodo emite luz que atraviesa un disco con huecos a la misma distancia entre sí que es detectada por un fotosensor donde la posición de la rueda es medida en función del número de veces que se ha detectado un rayo de luz.

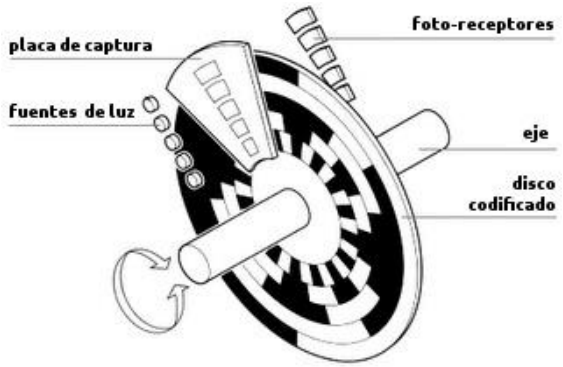

Ilustración 4: Despiece de un codificador óptico

Por consiguiente, mediante la información recibida por los distintos sensores la principal labor del sistema de percepción será la caracterización del entorno y localización en el mismo ofreciendo una robustez que permita detectar errores y estrategias de relocalización del vehículo autónomo.

Por lo tanto, el proyecto de investigación a realizar del sensor de visión tendrá como resultado final la mejora del sistema de percepción. Para ello, se realizará una investigación profunda con el objetivo de extraer la mayor cantidad de información de las imágenes capturadas por las distintas cámaras. Así como el estudio de proyectos similares, simuladores de vehículos autónomos para la captura de imágenes y su posterior procesamiento y análisis de las imágenes obtenidas que será detallado en profundidad en los próximos apartados.

3. OBJETIVOS Y ALCANCE DEL TRABAJO

Como se ha ido comentando en el apartado anterior el principal objetivo del proyecto es mejorar y aportar nueva información al trabajo de desarrollo de sistemas de percepción robustos basados en la fusión de distintos sensores mediante el análisis de la información de las imágenes obtenidas a través de la cámara. Para ello, se realizará un estudio de los fotogramas consecutivos que permitirá conocer el desplazamiento relativo del vehículo entre dos imágenes consecutivas. Es decir, se deberá de obtener la información suficiente como para que, mediante el algoritmo, el sensor sea capaz de darnos el ángulo de giro (ángulo Yaw) y el desplazamiento (longitudinal y lateral) relativo realizado por el vehículo autónomo entre los dos fotogramas consecutivos.

Por ello, los siguientes puntos describirán el alcance del trabajo:

- Estudio de los distintos proyectos realizados hasta la actualidad en base a la utilización del sensor cámara en vehículos autónomos.
- Búsqueda e implementación del simulador apropiado de vehículos autónomos para la obtención de la base de datos e imágenes deseadas para su procesamiento.
- Búsqueda e implementación de algoritmos de procesamiento de imágenes con el objetivo de obtener la información necesaria para el conocimiento del desplazamiento del vehículo.
- Búsqueda e implementación de algoritmos de procesamiento para la eliminación de datos erróneos.
- Diseño del algoritmo final que permita conocer el desplazamiento relativo realizado por el vehículo autónomo entre dos fotogramas consecutivos.
- Aplicación del algoritmo en distintos escenarios para el análisis de datos en las diferentes situaciones.

4. BENEFICIOS QUE APORTA EL TRABAJO

En la actualidad todo lo que envuelve a la conducción autónoma sugiere claramente que existen ventajas que obtener al invertir en vehículos sin conductor. Uno de los principales futuros atractivos trataría de un mundo en el que las calles y carreteras estén llenas de vehículos sin conductor que se muevan en perfecta sincronía entre sí. Al mismo tiempo, los accidentes de tráfico causados por errores humanos serían cosa del pasado y el viaje diario al trabajo sería relajado y seguro.

Entre los beneficios que se obtendrían de la utilización de vehículos autónomos estarían:

- **Mejora de la seguridad:** diferentes estudios indican que hasta un 90% de los accidentes de tráfico son causados por los conductores.
- **Mayor eficiencia:** el tráfico será más fluido y la congestión se verá reducida mediante la comunicación entre vehículos autónomos.
- **Menor impacto ambiental:** se verá reducido el número de vehículos junto con una mayor eficiencia del combustible.
- **Mejor confort:** el piloto pasará a ser un pasajero nuevo.

Por otro lado, a través de la utilización del sensor de visión en el sistema de percepción se aumentará la robustez del sistema debido a la redundancia de sensores. Esta redundancia permitirá solventar la información errónea recibida por alguno de los sensores en alguna de las distintas situaciones reales como, por ejemplo, la pérdida de la señal GPS en túneles. También, gracias a la redundancia de sensores se mejorará la precisión del desplazamiento relativo del vehículo. Y, al mismo tiempo, mediante el sensor de visión se podrá identificar los distintos elementos del entorno como señales, otros vehículos o peatones que ayudarán al sistema de percepción a obtener nueva información.

Por lo tanto, mediante la realización de este trabajo de investigación se conseguirá obtener un nuevo paso en la consecución de vehículos completamente autónomos que mejoren la vida cotidiana.

5. ANÁLISIS DEL ESTADO DEL ARTE

Definidos los objetivos del trabajo, antes de abordar el desarrollo de los algoritmos, se ha realizado un estudio de los antecedentes o proyectos realizados hasta la actualidad donde se centrarán los recursos en 3 puntos principales:

- Estudio del desarrollo de vehículos autónomos en la actualidad.
- Búsqueda de la base de datos adecuada para la obtención de los parámetros y elementos necesarios para los distintos sensores utilizados en la fusión.
- Búsqueda de proyectos de procesamiento de imágenes: Odometría visual.

ESTUDIO DEL DESARROLLO DE VEHÍCULOS AUTÓNOMOS

En primer lugar, para comenzar con el análisis de los antecedentes se presentan algunos de los avances o prototipos más exitosos hasta la actualidad. Hay muchas categorías diferentes de tecnología de conducción autónoma que se utilizan hoy en día en la industria del automóvil. Una aplicación muy popular, disponible como accesorio opcional en muchos coches nuevos, es el sistema de asistencia al aparcamiento. Este sistema detecta el entorno inmediato y aparca el vehículo en una plaza de aparcamiento. Entre las opciones más populares se encuentra el "Asistente de aparcamiento de Bosch" donde el sistema toma el control total de la dirección, liberando al conductor que únicamente controla la operación de aparcamiento mediante la aceleración y el frenado. El conductor puede elegir entre aparcar en paralelo o aparcar en batería.

El siguiente sistema de aparcamiento avanzado trata del "Aparcamiento autónomo de Volvo". Al igual que el servicio de aparcacoches tradicional, este sistema permite aparcar un vehículo una vez que el conductor ha salido del mismo. El conductor se comunica con el sistema a través de un dispositivo móvil para dirigirlo a la plaza de aparcamiento preferida y reclamar al vehículo aparcado para que lo recoja cuando lo necesite. Actualmente el sistema se encuentra en fase de desarrollo.

Ilustración 5: Sistema de aparcamiento: Bosch / Volvo

El siguiente paso dado en los vehículos autónomos trata de una serie de vehículos de gama alta, como el Mercedes Clase S Intelligent Drive32, el BMW i333 o el Hyundai Genesis34, que están equipados con una aplicación específicamente desarrollada para ayudar a los conductores en las autopistas. Esta función se activa en caso de tráfico intenso, y mantiene de forma autónoma el vehículo en su carril, frenándolo y acelerándolo hasta los 60 km/h. Esta velocidad máxima probablemente aumente con el tiempo ya que, técnicamente, esta aplicación puede soportar velocidades superiores a los 100 km/h. Es probable que la aplicación piloto en carretera evolucione para permitir cambiar de carril y adelantar.

Todos estos pequeños avances realizados tienen como objetivo la conducción autónoma donde, actualmente, se encuentran distintas empresas desarrollando cada una de ellas su propio vehículo autónomo. Entre ellas [1] se encuentran empresas como Tesla donde Elon Musk, consejero delegado de Tesla, afirma que todos los coches de Tesla serán completamente autónomos. El modelo "S" de Tesla es un coche semiautónomo en el cual los diferentes coches aprenden unos de otros mientras trabajan conjuntamente. Por ejemplo, las señales procesadas por los sensores se envían a otros vehículos para que puedan desarrollarse mutuamente. Esta información enseña a los coches a cambiar de carril y a detectar obstáculos, y mejoran continuamente.

Por otro lado, el equipo de Google lleva años trabajando en coches sin conductor donde destaca el prototipo “Waymo”. Su propio vehículo autónomo utiliza sensores Bosch como otros equipos fabricados por las empresas LG y Continental. Además, Google también apoya a otros fabricantes de automóviles con tecnologías de autoconducción.

Ilustración 6: Interior del vehículo autónomo de Tesla / Prototipo Waymo

BÚSQUEDA DE SIMULADOR PARA LA OBTENCIÓN DE LA BASE DE DATOS

Dada la dificultad de crear una propia base de datos que permita desarrollar el proyecto en todo su rango de sensores se han estudiado dos posibles fuentes de obtención:

- Base de datos obtenida mediante el escaneo de un escenario real a través de un vehículo con los distintos sensores integrados.
- Base de datos obtenida mediante simulador de vehículos autónomos.

Base de datos obtenida mediante escenario real

Se ha ido observando cómo ha ido aumentando la cantidad de alternativas de obtención de base de datos mediante dicho sistema debido al gran número de opciones reales que existen en la actualidad. Es por ello, que dentro de todas las posibilidades se destacan las siguientes.

La base de datos nuScenes [2] es un conjunto de datos de conducción autónoma a gran escala con anotaciones de objetos en 3D. Entre las principales características se encuentran el conjunto completo de sensores compuesto por 1x LIDAR, 5x Radar, 6x Cámara, IMU y GPS; 1000 escenarios diferentes de 20 segundos cada uno; 1.400.000 imágenes e información detallada del mapa recorrido por el vehículo. El artículo [3] presenta un análisis detallado de la base de datos nuScenes donde analiza las ventajas y desventajas de cada uno de los sensores utilizados para la base datos dando información de cómo podrían mejorarse.

Ilustración 7: Vehículo utilizado en nuScenes y disposición de los sensores

La base de datos RobotCar [4] recoge más de 1.000 km de conducción grabada con casi 20 millones de imágenes recogidas de 6 cámaras montadas en el vehículo, junto con el correspondiente LIDAR, GPS y Ground Truth. Todas ellas realizadas en el mismo escenario durante distintos días y situaciones. En los artículos [5] y [6] se presentan el análisis de las bases de datos obtenidas a partir de RobotCar donde el segundo de ellos además aporta la odometría del radar optimizada según el Ground Truth para dar un impulso adicional a la investigación.

Ilustración 8: Disposición de los sensores en RobotCar

La base de datos KITTI [7] ha sido recogida mediante la modificación de los actuadores del pedal para el control de la aceleración y el frenado a través de 1x Laser, 2x Cámara en escala de grises, 2x Cámara en color y 1x Cámara de lentes variables que recogen 10 fotogramas por segundo. Adicionalmente, se dispone de GPS, LIDAR, IMU y Ground Truth con una base de datos muy amplia en distintos escenarios. En el artículo [8] igual que en las anteriores bases de datos se ofrece un amplio análisis de los datos extraídos por KITTI.

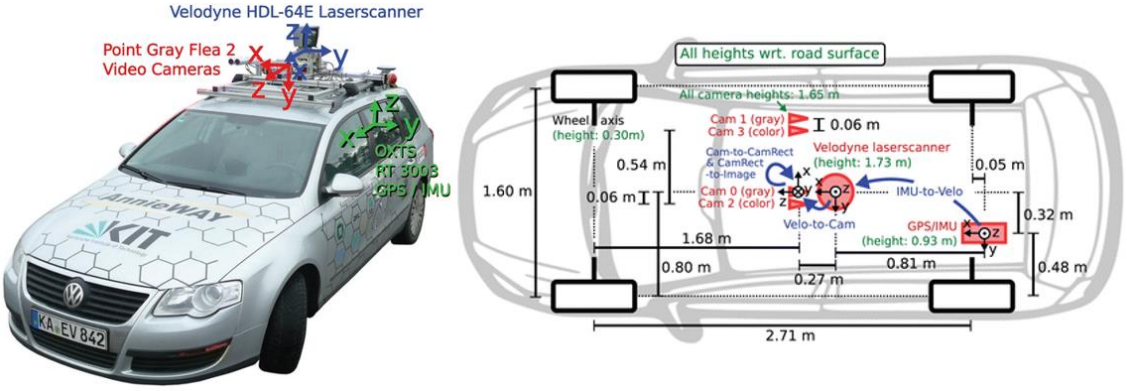

Ilustración 9: Vehículo utilizado en KITTI y disposición de los sensores

Base de datos obtenida mediante simulador de vehículos autónomos

En el lado opuesto se encuentran los simuladores de vehículos autónomos en los cuales se pueden diseñar todos los elementos necesarios como el entorno a recorrer, el vehículo o la posición y número de sensores a utilizar. Al mismo tiempo, existe la posibilidad de integrar el algoritmo y observar mediante el simulador los errores para su posterior mejora u obtener los datos del recorrido realizado por el vehículo mediante los sensores conociendo exactamente los datos reales.

El SVL Simulator [9] ofrece la posibilidad de probar el sistema de principio a fin o por módulos en un entorno virtual. También, permite realizar pruebas de integración desde el primer día sin necesidad de crear una infraestructura de pruebas desde cero mediante una plataforma de simulación en tiempo real y gran rendimiento en la cual pueden contralarse todos los elementos del entorno, así como la velocidad del vehículo. Al mismo tiempo, existe la posibilidad de trabajar en la nube para compartir el contenido con un equipo de trabajo. Asimismo, ofrece poder trasladar un entorno real a la plataforma de simulación. Por consiguiente, en el artículo [10] se analiza e integra dicho simulador para las distintas plataformas de diseño del entorno, adquisición de datos y simulación de entorno real.

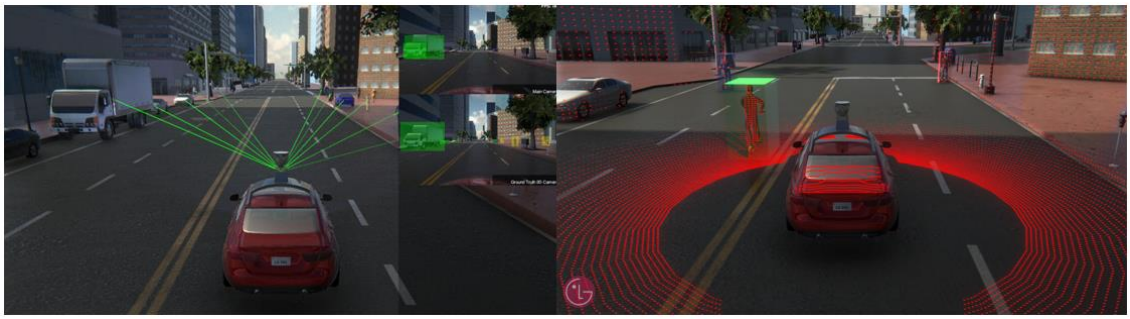

Ilustración 10: SVL Simulator

NVIDIA DRIVE Sim [11] es una plataforma de simulación integral, diseñada desde cero para ejecutar simulaciones multisensoriales de gran escala y precisión física. Es abierta, escalable y modular, y admite el desarrollo y validación de vehículos autónomos desde el concepto hasta la implantación, lo que mejora la productividad de los desarrolladores. Con Omniverse Kit, los usuarios y socios pueden crear extensiones para DRIVE Sim de modelos de sensores, modelos de tráfico, contenido 3D y herramientas de validación. Al mismo tiempo, NVIDIA dispone una colección de base de datos reales que pueden ser comparados con los datos simulados ofreciendo una mayor posibilidad de trabajo.

Ilustración 11: NVIDIA DRIVE Sim

CARLA [12] se ha desarrollado desde cero para apoyar el desarrollo, la formación y la validación de los sistemas de conducción autónoma. Además del código y los protocolos de código abierto, CARLA proporciona activos digitales abiertos (trazados urbanos, edificios, vehículos) creados para este fin y que pueden utilizarse libremente. La plataforma de simulación admite la especificación flexible de conjuntos de sensores, condiciones ambientales, control total de todos los actores estáticos y dinámicos, generación de mapas y mucho más. Además, ofrece una amplia gama de características como la escalabilidad a través de una arquitectura servidor-multicliente, planificación y control de la simulación, generación de mapas reales o escenarios con tráfico. En el artículo [13] se analiza el simulador CARLA en distintos escenarios para visualizar su comportamiento entre un escenario real y el escenario simulado para su validación.

Ilustración 12: Simulador CARLA: Visión global / Visión desde el vehículo

INVESTIGACIÓN DE PROCESAMIENTO DE IMÁGENES

Continuando, antes de realizar un estudio sobre los posibles procesamientos de imagen es importante conocer detalladamente las características principales tanto de la cámara como de las imágenes.

La cámara es un sensor crucial en los vehículos autónomos por toda la información que es capaz de recoger gracias a la capacidad que tiene de capturar el entorno para detectar objetos, personas y obstáculos. Se pueden distinguir dos tipos de cámara. La cámara estenopeica (*en inglés, Pinhole Camera*) se trata de una cámara fotográfica sin lentes en la cual la luz es enfocada en un solo punto y es dirigida a la película de la cámara. Sin embargo, esto limita los niveles de luz que se pueden medir. En el otro lado, la cámara con lentes enfocará la luz en un punto del plano convergiendo en el punto focal donde el mayor de los problemas será controlar el concepto de distorsión. Por lo tanto, con la utilización de esta última cámara será necesaria la calibración de ella y obtener la matriz de parametrización de la cámara para reducir la distorsión al multiplicarla por los pixeles de la imagen.

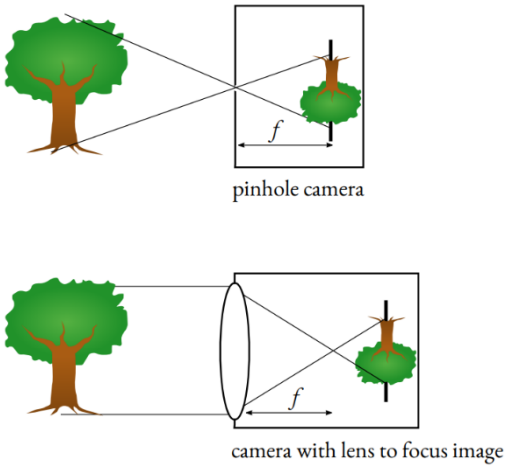

Ilustración 13: Tipos de cámara

Por consiguiente, cuando se captura una imagen con una cámara, la luz pasa a través del objetivo y cae sobre el sensor de imagen. Dicho sensor está formado por elementos de la imagen, también llamados píxeles, que registran la cantidad de luz que incide sobre ellos convirtiendo la cantidad de luz recibida en un número correspondiente de electrones. Cuanto más intensa es la luz, más electrones se generan. Posteriormente, los electrones se convierten en tensión y luego se transforman en números mediante un convertidor A/D. La señal constituida por los números es procesada por los circuitos electrónicos del interior de la cámara.

En la actualidad, hay dos tecnologías principales que pueden utilizarse para el sensor de visión de una cámara: CCD (Charge-Coupled Device) y CMOS (Complementary Metal-Oxide Semiconductor).

En un sensor CCD, la luz que incide en los píxeles del sensor se transfiere desde el chip a través de un nodo de salida o a través de sólo unos pocos nodos de salida. Las cargas se convierten en niveles de tensión, se atenúan y se envían como una señal analógica. Esta señal se amplifica y se convierte en números mediante un convertidor A/D fuera del sensor.

A diferencia del sensor CCD, el sensor CMOS incorpora amplificadores y convertidores A/D, lo que reduce el coste de las cámaras, ya que contiene toda la lógica necesaria para producir una imagen. Además, cada píxel del CMOS contiene una electrónica de conversión. Sin embargo, esta adición de circuitos dentro del chip puede conducir a un riesgo de ruido más estructurado; aunque, por otro lado, también tienen una lectura más rápida, un menor consumo de energía, una mayor inmunidad al ruido y un menor tamaño del sistema.

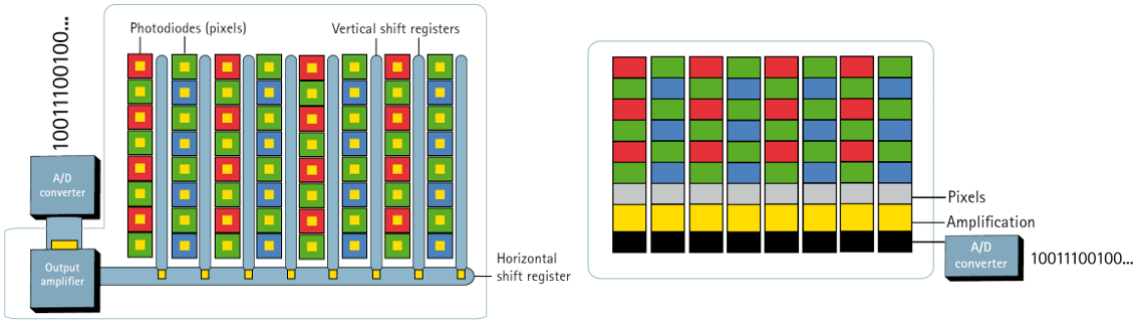

Ilustración 14: CCD / CMOS

Luego, una vez conocidas las principales características tanto de la cámara como de la imagen, se ha investigado cómo es posible obtener información de las imágenes obtenidas, es decir, como obtener información de los pixeles de las imágenes. Para ello, después de una exhausta búsqueda se ha encontrado que en la mayoría de los escenarios de procesamiento de imágenes en el sector automovilístico para procesar las imágenes se utiliza la odometría visual (*en inglés, Visual Odometry*).

ODOMETRÍA VISUAL

La odometría visual (VO) es un proceso de estimación de la posición de un elemento, como un vehículo, un ser humano o un robot, utilizando solo la entrada de una o más cámaras conectadas a él. El término VO fue propuesto por Nister en un artículo histórico publicado en 2004 [14]. Se eligió este término porque es similar a la odometría de la rueda, que estima el movimiento del vehículo integrando el número de rotaciones de la rueda a lo largo del tiempo. De manera similar, el principio de funcionamiento de la VO es estimar gradualmente la posición del vehículo mediante el estudio del movimiento de la imagen de la cámara a bordo. Para que la VO funcione de manera efectiva, debe haber suficiente iluminación en el entorno y las escenas estáticas deben tener suficiente textura para extraer un movimiento obvio. Además, se deben capturar fotogramas consecutivos para garantizar una superposición suficiente de la escena.

Por consiguiente, el problema de estimar la posición relativa de la cámara y la estructura tridimensional (3D) a partir de un conjunto de imágenes de cámara se llama Estructura de Movimiento (SFM) (*en inglés, Structure From Motion*) y sus orígenes se remontan a trabajos como [15]. Por otro lado, el problema de estimar visualmente la posición del

vehículo comenzó a principios de la década de 1980 y fue propuesto por Moravec. Curiosamente, la mayor parte de la investigación inicial sobre la VO se realizó en robots planetarios y fue promovida por el Programa de Exploración de Marte de la NASA, que intentó equipar a los vehículos todoterreno con la capacidad de medir su movimiento en presencia del deslizamiento de ruedas en terreno irregular.

Procediendo, para analizar el desplazamiento relativo realizado por el vehículo entre dos fotogramas consecutivos, existen tres técnicas estándar diferentes que permiten calcular la matriz de transformación basadas en la especificación de puntos en 2D o 3D [16].

- **Correlación 3D-3D:** el movimiento de la cámara puede calcularse a partir de dos conjuntos de cohesiones especificadas en 3D, es decir, mediante la captura de dos pares de imágenes estéreo, extrayendo y emparejando los puntos característicos entre ellos para triangular los puntos coincidentes en 3D para cada par estéreo. La transformación se calcula con una escala absoluta minimizando la distancia entre los dos conjuntos de puntos 3D.
- **Correlación 2D-2D:** la matriz de transformación se calcula utilizando la matriz esencial. La matriz esencial define la relación geométrica entre dos imágenes consecutivas y se calcula a partir de las relaciones de características 2D utilizando la restricción epipolar. Un enfoque sencillo y común para calcular la matriz esencial es el uso del algoritmo de cinco puntos [17]. En este método, se utiliza un conjunto de cinco puntos característicos para determinar la escala relativa entre fotogramas consecutivos. Por otro lado, otra manera de calcular la matriz esencial es utilizando el algoritmo de ocho puntos presentado por Fischler y Firschein [18].
- **Correlación 3D-2D:** el concepto principal de este método es calcular la matriz de transformación minimizando el error de reproyección 2D a partir de las relaciones 2D y 3D. Este problema también se conoce como perspectiva-n-puntos (PnP), que estima la posición de una cámara utilizando un conjunto de N números de puntos 3D. La solución mínima para recuperar la posición de la cámara requiere tres correlaciones 3D-2D, lo que se conoce como perspectiva-3-puntos (P3P).

Por otro lado, las técnicas de VO pueden clasificarse en función de la información relevante obtenida de las imágenes, la posición de la cámara y/o el tipo/número de cámaras. En primer lugar, según la información relevante sobre la que se realiza la odometría pueden ser mediciones directas en bruto analizando los píxeles de la imagen, o mediante características indirectas de la imagen como esquinas y bordes, o una combinación de ellas, es decir, enfoque híbrido.

Técnicas de VO basadas en información relevante

En el análisis directo se utilizan mediciones visuales, en términos de píxeles, para estimar la posición del vehículo. Para ello, se tiene en cuenta el cambio en la apariencia de las imágenes capturadas, es decir, la intensidad de los píxeles de la imagen, donde se analiza la imagen para estimar la posición [19]. Basándose en las imágenes capturadas, se utiliza un algoritmo de flujo óptico (OF*) (en inglés, Optical Flow*) para determinar los cambios entre fotogramas, el cual utiliza la intensidad de los píxeles para calcular el vector de desplazamiento 2D que muestra el movimiento de los puntos entre dos fotogramas consecutivos. En la odometría visual, los algoritmos de OF se clasifican en esquemas densos y dispersos. En el OF denso, todos los píxeles se optimizan utilizando varias técnicas basadas en una suposición de suavidad global. Un ejemplo de un algoritmo de OF denso es el método Horn-Shunck [20] que calcula el desplazamiento de cada píxel en un fotograma resolviendo la constancia de brillo. Por otro lado, en los algoritmos dispersos, por ejemplo, Lucas-Kanade [21] explotan la suposición de que el flujo en una imagen es localmente suave. Así, el OF disperso sólo procesa algunos píxeles de toda la imagen resolviendo la ecuación de constancia de brillo.

En cambio, los estudios basados en características o indirectos extraen puntos de interés de cada imagen capturada utilizando detectores de características como detectores de esquinas o de bordes. Las esquinas son uno de los puntos clave más singulares de la imagen, ya que muestran un cambio de intensidad bidimensional, que las distingue bien de los puntos vecinos [22]. En consecuencia, varios métodos propuestos se basan en las esquinas, por ejemplo, el detector Harris [23], SIFT (*en inglés, Scale-Invariante Feature Transform*) [24] y SURF (*en inglés, Speeded-Up Robust Features*) [25]. Por otro lado, los bordes en las imágenes son áreas con fuertes contrastes de intensidad donde la mayoría de los detectores de bordes se basan en el gradiente o en el laplaciano [26]. El detector de bordes Laplaciano utiliza un núcleo para buscar los cruces de cero en la segunda derivada de la imagen. A diferencia del detector laplaciano, el detector de bordes por gradiente, como el detector de bordes de Canny [27], utiliza dos núcleos para detectar los bordes buscando el máximo y el mínimo en la primera derivada de la imagen. Las imágenes consecutivas se preprocesan utilizando diferentes técnicas de detección y comparación de características para generar una representación intermedia, es decir, correspondencias de puntos. Posteriormente, se realiza un proceso de optimización minimizando el error geométrico para calcular la matriz de transformación. Una de las principales ventajas del método basado en características es que es robusto frente a las distorsiones geométricas y las inconsistencias de brillo. Sin embargo, descarta información valiosa de la imagen capturada al extraer sólo los puntos de interés fuertes.

Cabe destacar que los enfoques basados en características no son robustos en entornos de baja textura, porque sólo se pueden detectar y rastrear unas pocas características. Por otro lado, los métodos directos explotan toda la información clave de las imágenes,

incluidas las variaciones débiles de intensidad, lo que permite obtener resultados más sólidos y eficaces en estos entornos. Sin embargo, los enfoques directos son computacionalmente más exigentes. Para resolver estos problemas se utiliza una combinación de métodos directos y basados en características, es decir, un enfoque híbrido. Por ejemplo, Scaramuzza y Siegwart [28] proponen un método híbrido de VO para estimar la posición de un vehículo terrestre. El desplazamiento se estima utilizando un método basado en características, mientras que la orientación se determina eficientemente basándose en un método directo. En [29] se ha presentado un método de odometría visual semidirecta (SVO) para eliminar la necesidad de una costosa extracción de características en cada fotograma. Este enfoque utiliza la correspondencia de características para aumentar la precisión, y realiza la extracción de características sólo en los fotogramas clave seleccionados.

Técnicas de VO basadas en el tipo/número de cámaras

En segundo lugar, la odometría visual se puede clasificar en función del tipo y el número de cámaras utilizadas en el algoritmo. Hay cuatro tipos de cámaras que se utilizan principalmente en los enfoques de odometría visual: monocular, estéreo, RGB-D y omnidireccional.

Odometría visual monocular

Los sistemas de mono visión utilizan una cámara para estimar la distancia basándose en puntos de referencia en el campo de visión de la cámara [30]. El método propuesto por Scaramuzza y Fraundorfer [31] para estimar el movimiento de una cámara monocular basado en correspondencias 2D-2D encuentra una matriz de transformación Tt = {Rt,tt} que minimiza el error de reproyección de los puntos coincidentes en dos fotogramas consecutivos. Las características se extraen de la imagen actual y de la anterior y luego se emparejan para generar un conjunto de puntos de correspondencia. Las correspondencias generadas son usadas para calcular la matriz esencial, necesita un mínimo de 5 puntos de correspondencia, que se descompone en una matriz de rotación Rt y un vector de traslación tt.

En el artículo [32] se presenta una localización absoluta de un objeto en el sistema de coordenadas de la cámara utilizando la estimación de la distancia entre el punto principal y el punto característico basado en el área calculada en el sistema. La realización de este proceso se descompone en tres partes. Primero se realiza la calibración de la cámara, es decir, la calibración de los parámetros intrínsecos. Luego, se construye un modelo para la medición de la distancia en la dirección del eje óptico de acuerdo con la relación de mapeo entre los objetos en el sistema de coordenadas de la cámara y su proyección en el sistema de coordenadas del píxel. Y, finalmente, se realiza la estimación de la distancia absoluta. Por otro lado, en [33] se propone un sistema de evitación de obstáculos y medición de distancias utilizando el método de mono visión.

Miden la distancia entre el vehículo y el obstáculo basándose en técnicas de calibración de la cámara y en la variación de los píxeles en fotogramas de vídeo consecutivos utilizando los puntos clave extraídos por los algoritmos SIFT y SURF. También, en [34] se propone un sistema de mono visión con segmentación de instancias y distancia focal de la cámara para detectar la distancia de los coches frente al coche actual. Para este sistema propuesto se extraen las localizaciones de los coches que son clasificados para obtener sus tipos y sus valores de máscara utilizando un modelo entrenado, donde al mismo tiempo, se utiliza un nuevo modelo de segmentación de instancias para obtener la máscara de los coches. Por último, se calculan las distancias de los coches basándose en la relación entre la información del tamaño de los diferentes tipos de coches y sus valores de máscara.

Odometría visual estereoscópica

El sistema de visión estereoscópica es un sistema de visión que se basa en técnicas de alcance estereoscópico para calcular la distancia. Este sistema utiliza dos cámaras como una sola, tratando de dar la impresión de profundidad y utilizar la disparidad de los objetos entre las cámaras para calcular la distancia con alta precisión. La Ilustración 15 presenta los principales elementos de la odometría estereoscópica. En la ilustración, un punto, que se ve desde dos cámaras totalmente calibradas y alineadas con sus respectivos centros de proyección, puede reconstruirse con una escala absoluta utilizando una técnica de triangulación y dos conjuntos de correspondencias de puntos. La idea principal es buscar características correspondientes en dos imágenes alineadas a lo largo de la misma línea epipolar. Esta técnica, también llamada geomática epipolar reduce el tiempo de búsqueda de características de las imágenes 2D a las líneas epipolares. Una de las principales desventajas del uso de cámaras estereoscópicas para la localización es que requieren una calibración extrínseca precisa para proporcionar resultados exactos. Debido a la variación de las condiciones, por ejemplo, golpes, vibraciones, etc., dicha calibración extrínseca se degrada con el tiempo y es necesario recalibrarla periódicamente para una estimación eficaz de la postura. Además, las cámaras estereoscópicas suelen tener una distancia base fija, es decir, la distancia entre las dos cámaras, lo que afecta a la precisión de la estimación de la profundidad en diferentes escenarios.

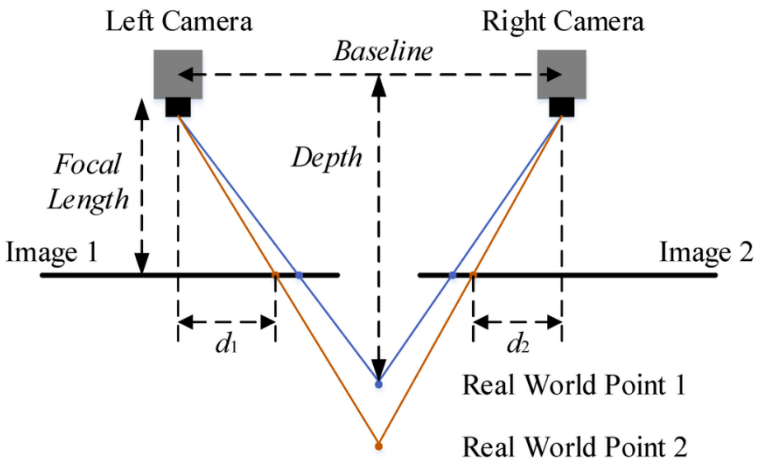

Ilustración 15: Odometría visual estereoscópica

En el artículo [35] se presenta un método de estimación de la distancia basado en la medición estereoscópica utilizando ecuaciones trigonométricas, su método se divide en tres partes. La primera parte consiste en aplicar algunos métodos de procesamiento de imágenes para mejorar la velocidad de cálculo, como la reducción de la resolución de la imagen de entrada y la conversión de la imagen de entrada del dominio RGB al dominio en escala de grises. La segunda parte consiste en extraer la posición del objeto de las dos cámaras. La tercera parte consiste en averiguar el estado en el que se encuentra el objeto, en función de la posición del mismo y estimar la distancia mediante la ecuación de estado basada en el método trigonométrico.

En [36] se propone un sistema de visión estereoscópica para estimar la distancia de un objeto basándose en la distancia focal de las cámaras y la disparidad entre las imágenes. El método propuesto se compone de cuatro etapas: la primera consiste en aplicar métodos de preprocesamiento a las imágenes para reducir la velocidad de cálculo, así como la reducción del tamaño de las imágenes a un determinado nivel. La segunda etapa es la segmentación de la región, en la que se dividen las imágenes en pequeños bloques y se aplica el algoritmo de selección de umbrales locales para aislar los objetos del fondo. La tercera etapa consiste en buscar la información de disparidad de cada objeto entre las dos imágenes extrayendo sus características y comparándolas con descriptores específicos. La última etapa consiste en calcular la distancia entre los objetos de las dos imágenes extrayendo sus características y comparándolas. En el libro [37] se basan en el método de visión estéreo para estimar la distancia y utilizan un método de coincidencia de imágenes para aumentar la precisión del sistema. Se utiliza el algoritmo del Filtro de Kalman Simulado (SKF) para la comparación de fotogramas debido a que es más eficiente para resolver el problema de la medición de la distancia. En [38] estiman la distancia a partir de las diferencias entre las imágenes tomadas por las dos cámaras y los datos técnicos adicionales como la distancia focal y la distancia entre las cámaras.

Odometría visual RGB-D

Las cámaras RGB-D son una solución óptima para proporcionar información sobre la profundidad real en comparación con las configuraciones estereoscópicas y monoculares comentadas anteriormente. Un análisis de cámara estereoscópica realiza una costosa búsqueda de líneas epipolares para obtener la información de profundidad, y si las cámaras estereoscópicas no están alineadas, se necesita un proceso adicional para alinear las líneas epipolares de ambas cámaras horizontalmente. Por otro lado, una configuración monocular no puede obtener la información de profundidad de los alrededores a escala real.

Ilustración 16: Fotograma de visión por RGB-D

Odometría visual omnidireccional

Una cámara omnidireccional, también conocida como cámara de 360, es una cámara con un campo de visión (FoV) (*en inglés, Field of Views*) de 360º en azimut y de 90º a 140º en elevación. Además, puede lograr una estimación de la posición más precisa en comparación con las cámaras tradicionales con un FoV pequeño, porque puede capturar más información del entorno. En [28], los autores presentan un sistema de VO basado en una única cámara omnidireccional para obtener la posición y orientación de los vehículos. El sistema consta de dos módulos: uno basado en la traslación y otro en la dirección. El primer módulo utiliza un método basado en la traslación para detectar y rastrear características del suelo. El segundo módulo utiliza un método directo para estimar la rotación del vehículo.

Ilustración 17: Fotograma de visión por cámara omnidireccional

Técnicas de VO basadas en la posición y orientación de la cámara

En tercer lugar, en función de la posición y orientación de la cámara, los sistemas de localización de VO existentes pueden dividirse en tres categorías: orientados hacia delante, orientados hacia abajo e híbridos.

Una configuración orientada hacia delante proporciona más información, pero es una solución subóptima para detectar pequeños movimientos. Además, puede verse oscurecida por las sombras y los cambios en el entorno, como el viento y la luz del sol. Por otro lado, los sistemas de localización basados en cámaras orientadas hacia abajo se han utilizado con éxito para el posicionamiento en entornos preexplorados. Sin embargo, estos sistemas son imprecisos cuando los vehículos se mueven rápidamente, ya que es difícil encontrar buenos puntos de coincidencia entre dos imágenes consecutivas. Por lo tanto, se utiliza un enfoque híbrido, es decir, una combinación de las configuraciones de cámaras orientadas hacia delante y hacia abajo, para abordar las limitaciones de cada esquema. Por ejemplo, en [39] proponen un sistema de localización en exteriores basado en la VO utilizando una configuración de cámara híbrida para un robot de dirección deslizante. Los datos de visión de la cámara orientada hacia abajo se utilizan para localizar el robot a bajas velocidades. En cambio, a altas velocidades, las imágenes capturadas por la cámara orientada hacia abajo no se utilizan en el proceso de VO, ya que es difícil rastrear las características basadas en ellas, y por lo tanto los datos proporcionados por la cámara orientada hacia adelante se utilizan para localizar el vehículo.

Por último, aunque los enfoques de odometría visual tienen mucho éxito en la localización de plataformas en entornos interiores, todavía hay muchos retos que deben ser abordados para utilizar la VO como un método de localización preciso en entornos exteriores. Los principales retos de la localización basada en la VO están relacionados con la complejidad computacional, la ambigüedad de la escala y las condiciones de la imagen, como la iluminación, las regiones con poca textura y la borrosidad de la imagen. Además, sufre problemas de deriva, ya que se basa en el cálculo incremental de la trayectoria de la cámara, lo que lleva a una acumulación gradual de errores introducidos por cada nuevo fotograma a lo largo del tiempo.

6. DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA

Una vez observados los antecedentes y analizadas las diferentes propuestas que se han desarrollado hasta el momento, en este proyecto se ha optado por la siguiente solución. Para ello, se va dividir la solución propuesta en 3 apartados.

SIMULADOR CARLA v: 0.9.9

Debido a las múltiples alternativas encontradas y al gran avance realizado en los simuladores de vehículos autónomos, en este proyecto se ha optado por utilizar el simulador CARLA [12]. Entre las distintas opciones encontradas respecto a los simuladores, el simulador CARLA es el simulador más avanzado y que más se ajusta a los requisitos que se deben de cumplir gracias a la gran facilidad de manejo, la gran variedad de posibilidades de inserción de los distintos sensores, así como la posibilidad de diseñar el propio escenario.

Ilustración 18: Simulador CARLA

CARLA es un simulador abierto de conducción urbana que se ha desarrollado desde cero para apoyar la formación, la creación de prototipos y la validación de modelos de conducción autónoma, incluyendo tanto la percepción como el control. Además, CARLA es una plataforma abierta donde el contenido de los entornos urbanos que ofrece es gratuito. Incluye trazados urbanos, multitud de modelos de vehículos, edificios, peatones, señales de tráfico, etc. La plataforma de simulación admite una configuración flexible de los conjuntos de sensores y proporciona señales que pueden utilizarse para entrenar estrategias de conducción, como las coordenadas GPS, los datos de los vehículos o los datos detallados sobre colisiones y otras infracciones. Se puede especificar una amplia gama de condiciones ambientales, incluyendo el clima y la hora del día.

Ilustración 19: Distintas condiciones ambientales en CARLA

CARLA simula un mundo dinámico y proporciona una interfaz sencilla entre el mundo y el agente que interactúa con él. Para soportar esta funcionalidad, CARLA está diseñado como un sistema servidor-cliente donde el servidor ejecuta la simulación y renderiza la escena. En el cual, por otro lado, la API del cliente se implementa en Python y es la responsable de la interacción entre el agente autónomo y el servidor a través de un intercambio de datos. El cliente envía comandos y metacomandos al servidor y recibe a cambio las lecturas de los sensores. Los comandos controlan el vehículo incluyendo la dirección, la aceleración y el frenado, mientras que los metacomandos controlan el comportamiento del servidor y se utilizan para reiniciar la simulación, cambiar las propiedades del entorno o modificar el conjunto de sensores. Entre las propiedades del entorno se incluyen las condiciones meteorológicas, la iluminación o la densidad de coches y peatones. Por consiguiente, cuando se reinicia el servidor, el agente se reinicia en una nueva ubicación especificada por el cliente.

Entorno de CARLA: se compone de modelos 3D de objetos estáticos, como edificios, vegetación, señales de tráfico e infraestructuras, así como de objetos dinámicos como vehículos y peatones. Todos los modelos 3D comparten una escala común, y sus tamaños reflejan los de los objetos reales.

Ilustración 20: Entorno de CARLA

Sensores de CARLA: permite una configuración flexible del conjunto de sensores del agente. Los sensores de cámara se limitan a las cámaras RGB y a los sensores secundarios que proporcionan tanto la profundidad como la segmentación semántica. El cliente puede especificar el número de cámaras, su tipo y posición. Los parámetros de la cámara incluyen la ubicación 3D, la orientación 3D con respecto al sistema de coordenadas del coche, el campo de visión y la profundidad de campo. El sensor de segmentación semántica proporciona 12 clases semánticas: carretera, marca de carril, señal de tráfico, acera, valla, poste muro, edificio, vegetación, vehículo, peatón y otros.

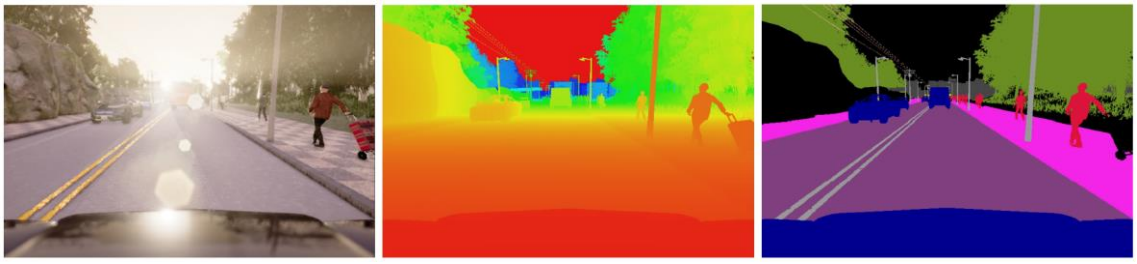

Ilustración 21: Visión normal / Visión de profundidad / Visión segmentada

Al mismo tiempo, CARLA proporciona información sobre otros sensores como IMU, LIDAR, LIDAR segmentado o Radar. Además de las lecturas de los sensores, CARLA proporciona una serie de medidas asociadas con el estado del vehículo y el cumplimiento de las normas de tráfico. Por último, CARLA proporciona acceso a las ubicaciones exactas y a los recuadros delimitadores de todos los objetos dinámicos del entorno. Estas señales desempeñan un papel importante en la formación y evaluación de las políticas de conducción.

Por otro lado, CARLA ha sido elegido debido a la gran cantidad de documentación que ofrece. En [40] y [41] se muestra toda la documentación necesaria para empezar, construir y trabajar con los distintos agentes de CARLA. Al mismo tiempo, debido a que el proceso de procesamiento de imágenes se va a realizar mediante MATLAB se dispone de [42] donde se explica con todo tipo de detalle los pasos a seguir para la correcta comunicación entre el simulador CARLA y el entorno de trabajo de MATLAB mediante el interfaz PYTHON. Toda esta comunicación será detallada en la sección “CÁLCULOS Y ALGORITMOS”.

TIPO, POSICIONAMIENTO Y ORIENTACIÓN DE LAS CÁMARAS

En segundo lugar, después de una exhausta investigación, como bien ha sido analizada en la sección “ANÁLISIS DEL ESTADO DEL ARTE”, en la odometría visual existen 4 tipos principales de cámaras: monocular, estereoscópica, RGB-D y omnidireccional. Por lo tanto, se ha realizado un estudio continuo de pruebas entre las posibilidades comentadas.

Inicialmente, se abordó el análisis del posicionamiento mediante la utilización de una única cámara con orientación hacia adelante en la parte superior del vehículo para obtener tanto el ángulo de giro como el desplazamiento en las coordenadas x e y. En él, los resultados obtenidos fueron exitosos en el control del ángulo de giro, pero tanto en el desplazamiento lateral como longitudinal no fue posible obtener resultados de manera exitosa.

Ilustración 22: Distribución de cámaras: 1º opción

Continuando, se optó por la colocación de dos cámaras con orientación hacia adelante con una separación entre ellas para obtener un sistema de odometría visual estereoscópica. Dicho sistema fue descartado debido a que, en primer lugar, se debían de detectar los elementos fijos de la imagen actual para reconocerlos y almacenarlos con el objetivo de ser detectados en la imagen del instante posterior. Por consiguiente, mediante dicho sistema se extraería en cada uno de los dos instantes consecutivos la distancia del vehículo a cada uno de los elementos fijos detectados para que, calculando, se obtuviese, mediante la diferencia, el desplazamiento real del vehículo. Este proceso conllevaba una gran complejidad computacional debido a que se debía de añadir la visión segmentada para poder detectar los elementos fijos del entorno lo que suponía un retraso considerable para la obtención del posicionamiento del vehículo en tiempo real.

Ilustración 23: Distribución de cámaras: 2º opción

Luego, se analizó el posicionamiento de 4 cámaras con orientación delantera, trasera, lateral derecha y lateral izquierda en la parte superior del vehículo para obtener una visión aproximada a la visión omnidireccional. Mediante dicho sistema se obtuvieron resultados decentes tanto para el caso del ángulo de giro como para el desplazamiento en x e y, pero seguían apareciendo distintos errores en diferentes entornos como cruces. Esto era debido a que el desplazamiento longitudinal se calculaba mediante los datos obtenidos por las cámaras con orientación lateral y en las situaciones como cruces u objetos a distancias variables, el desplazamiento relativo de la cámara variaba en función de la distancia a la que se encontraban los objetos del entorno, dando errores no solucionables.

Ilustración 24: Distribución de cámaras: 3º opción

Por ello, continuando con este mismo principio de visión omnidireccional, se analizó el mismo sistema, pero en esta ocasión realizando una inclinación de las cámaras laterales de 45º respecto al suelo para poder evitar dichos errores comentados, pero resultó que la solución mejoraba, pero no era suficiente para la eliminación completa de los errores. Por la tanto, se descartó el sistema de visión omnidireccional y se volvió al sistema monocular, pero realizando un sistema hibrido. Donde, dentro de este sistema de posicionamiento de las cámaras, se fueron probando distintas alturas de colocación con distintos ángulos de inclinación de las cámaras tanto delantera como laterales para obtener la mejor combinación y el menor número de errores para los distintos entornos.

Finalmente, la solución adoptada ha resultado en la utilización de dos cámaras laterales colocadas en los retrovisores laterales izquierdo y derecho con orientación hacia abajo para la obtención tanto del ángulo de giro como del desplazamiento relativo. Numéricamente, las dos cámaras laterales están posicionadas a una altura de 1 metro respecto del suelo, a una distancia lateral de 1 metro sobre el eje del vehículo respectivamente y con una inclinación perpendicular a la carretera. A continuación, se muestra la solución final adoptada.

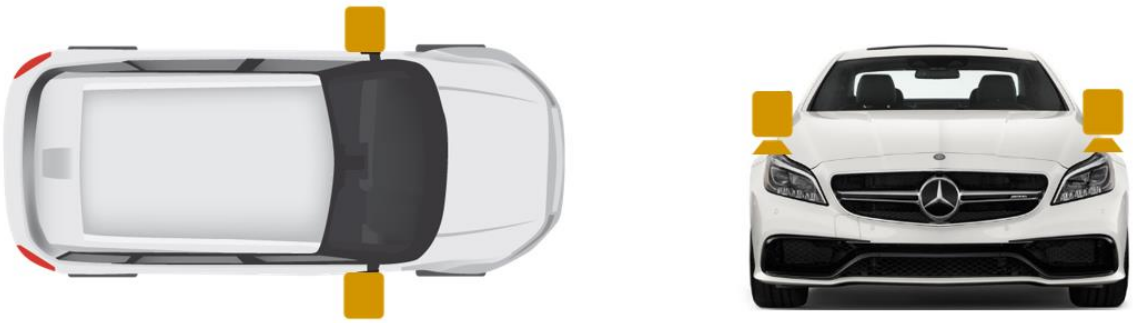

Ilustración 25: Distribución de cámaras: Solución final adoptada

PROCESAMIENTO DE IMÁGENES Y DATOS

En tercer lugar, una vez obtenidas las imágenes de las cámaras seleccionadas se deben de procesar estas para extraer la información que permita obtener tanto el ángulo de giro como los desplazamientos relativos longitudinal y lateral. Para ello, como bien se ha expuesto, existen dos métodos de procesamiento de imágenes para una correlación 2D-2D: análisis directo o análisis de características. En este proyecto se van a analizar las imágenes mediante el análisis de características, el cual se basa en la extracción de puntos de interés de cada imagen capturada utilizando detectores de características como detectores de esquinas o de bordes. Entre los distintos sistemas de detección de puntos se ha utilizado el algoritmo SURF (Speed-Up Robust Features) debido a la consecución de mejores resultados para extraer la información detallada y especifica del contenido. Conjuntamente, mediante un algoritmo de coincidencia de características obtenidas entre dos fotogramas consecutivos, el algoritmo presentado, a través del algoritmo RANSAC, elimina los puntos característicos no válidos y estima la matriz esencial sin parámetros erróneos de estimación gracias a la eliminación de los valores no válidos. Más tarde, obtenida la matriz esencial Tt={Rt,tt}, se puede obtener la orientación y el desplazamiento relativo de la cámara entre los dos fotogramas consecutivos. Para ello, se ha realizado un proceso muy detallado que será explicado paso a paso en la sección “CÁLCULOS Y ALGORITMOS” donde, al mismo tiempo, una vez obtenidos los datos de las imágenes procesadas, dichos datos serán procesados para eliminar cualquier valor erróneo que se haya podido obtener y combinar los datos de las distintas cámaras para obtener el resultado de ángulo de giro y desplazamiento relativo más preciso.

METODOLOGÍA SEGUIDA

7. DESCRIPCIÓN DE TAREAS

Análisis del estado del arte

- El análisis se centrará en la búsqueda de publicaciones científicas e información acerca de algoritmos de procesamiento de imágenes desarrollados en el marco de la conducción autónoma obtenidas a través del sensor de visión. De igual modo, se estudiarán distintas metodologías de obtención de base de datos referentes a los vehículos autónomos.
- Recursos: bases de datos bibliográficas de artículos científicos.
- Duración: 60 horas.

Configuración de CARLA

- Desarrollo del algoritmo de configuración del escenario sobre el cual se van a extraer información y realizar las validaciones del algoritmo desarrollado. Desarrollo del algoritmo de inserción de sensores para la obtención y extracción de datos necesarios para cada uno de ellos. Y, obtención de la base de datos real realizada por el vehículo autónomo.
- Recursos: simulador “open-source” CARLA y software y herramientas de Matlab junto con todas sus “toolbox” necesarias.
- Duración: 20 horas.

Desarrollo del algoritmo de localización

- Creación y desarrollo del algoritmo de procesamiento de imágenes para la extracción de información entre dos fotogramas consecutivos para la obtención del ángulo de giro y el desplazamiento longitudinal y lateral relativo entre las dos imágenes analizadas.
- Recursos: simulador “open-source” CARLA y software y herramientas de Matlab junto con todas sus “toolbox” necesarias.
- Duración: 100 horas.

Validación y obtención de resultados

- Desarrollado el algoritmo de localización, se realizarán distintas pruebas en distintos escenarios para comprobar su validez ante diferentes situaciones. Una vez validado, se obtendrán los resultados de cada uno de ellos para su análisis y comparación para su posterior mejora.
- Recursos: simulador "open-source" CARLA y software y herramientas de Matlab junto con todas sus "toolbox" necesarias.
- Duración: 60 horas.

Redacción del documento

- Validado el algoritmo y obtenidos los resultados, se realizará un documento detallado mostrando toda la información y metodología utilizada para la obtención de los resultados finales.
- Recursos: herramientas informáticas de redacción.
- Duración: 60 horas.

8. DIAGRAMA DE GANTT

Se muestra un detallado Diagrama de Gantt con las fechas de inicio y final del proyecto en base a los apartados desarrollados en la descripción de tareas.

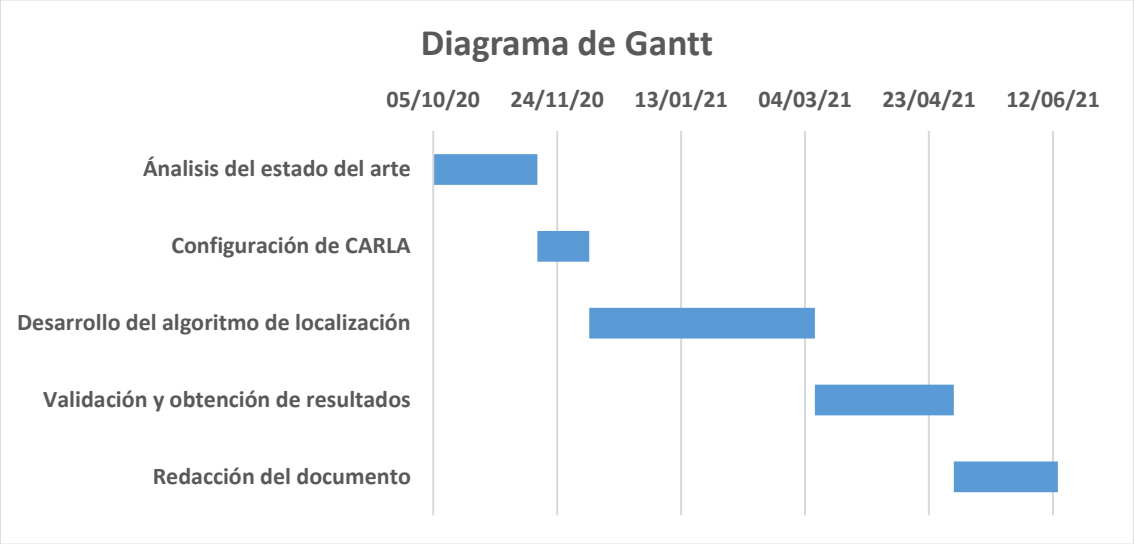

Gráfico 1: Diagrama de Gantt

9. CÁLCULOS Y ALGORITMOS

En este apartado se va a explicar detalladamente la solución propuesta tanto teóricamente, los algoritmos utilizados, como los pasos ejecutados para la obtención de los resultados finales. Existen 3 códigos principales a explicar donde, mediante el uso de diagramas de bloques, se va a explicar el proceso del código utilizado. Para ello, en primer lugar, se debe de iniciar el simulador CARLA y el entorno de trabajo MATLAB.

ADQUISICIÓN DE DATOS DE CARLA

Mediante este primer algoritmo de adquisición se va poder obtener la información necesaria de cada uno de los sensores, así como las imágenes de las cámaras simuladas en CARLA. Todo ello junto con la obtención de la base de datos real de la simulación realizada.

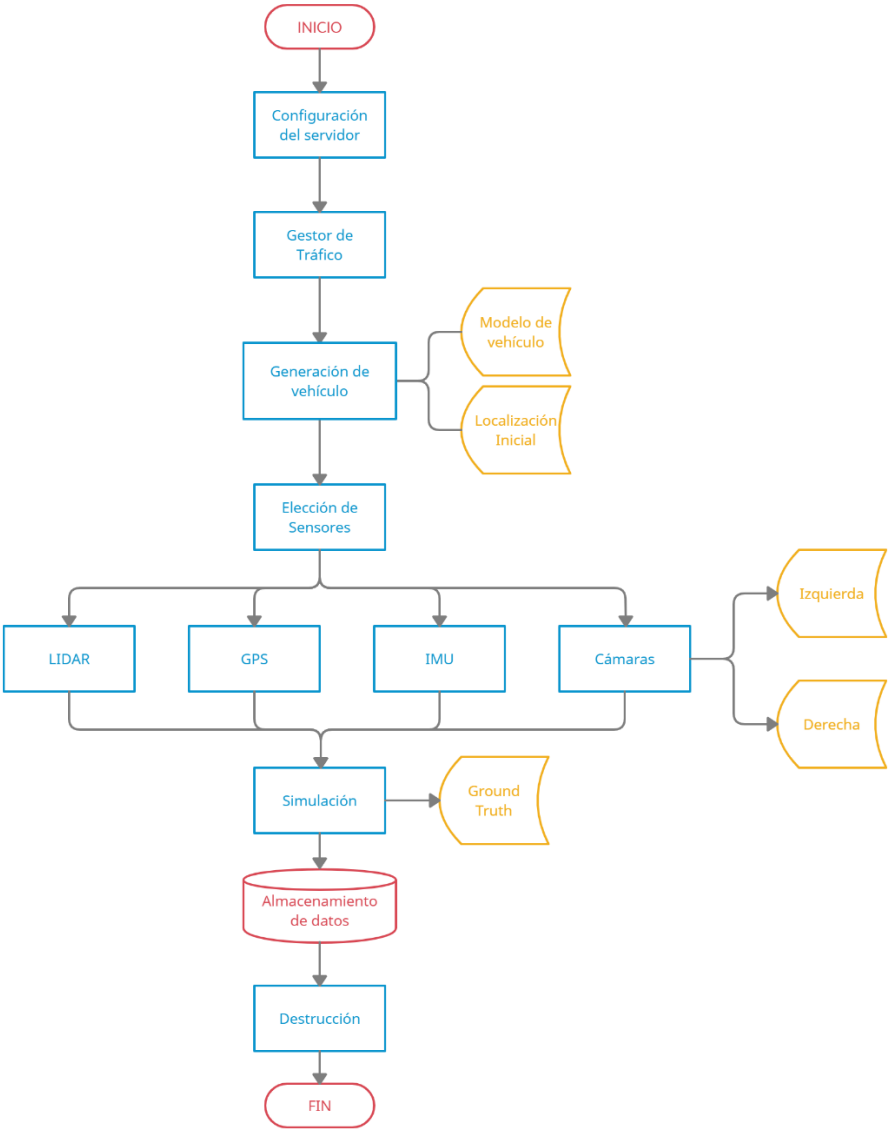

Gráfico 2: Diagrama de bloques "DataAdquisition_sync_CARLA"

Observando el diagrama de bloques anterior, primeramente, se debe de configurar el servidor que tal y como se ha comentado, CARLA se comunica a través de interfaz de comunicación Python. Aunque, en cierta medida, una característica interesante de MATLAB es que permite la utilización de Python desde el propio entorno.

Por consiguiente, antes de comenzar con la explicación del algoritmo es importante entender cómo trabaja CARLA. Básicamente, CARLA puede dividirse en 3 principales variables:

- Actor: es cualquier elemento que desempeña un papel en la simulación y que pueda moverse como vehículos, peatones o sensores.
- Blueprint: antes de generar un actor es necesario especificar sus atributos, y para eso están los blueprints. CARLA proporciona una biblioteca de blueprints con las definiciones de todos los actores disponibles.
- Mundo: representa el mapa actualmente cargado y contiene las funciones para convertir un blueprint en un actor vivo, entre otras. También proporciona acceso al mapa de carreteras y a las funciones para cambiar las condiciones meteorológicas.

Prosiguiendo, se realiza la conexión con CARLA mediante el puerto por defecto 2000 con un tiempo de espera de conexión de 12 segundos, permitiendo de dicha manera una conexión satisfactoria. Al mismo tiempo, se selecciona el mundo sobre el cual se va a realizar la simulación, coexistiendo la posibilidad de utilizar uno de los distintos mundos proporcionados por CARLA o trabajar sobre mundos creados por el usuario. En esta ocasión se ha elegido el “Mundo 3” que se puede visualizar a continuación en formato CARLA y formato MATLAB respectivamente.

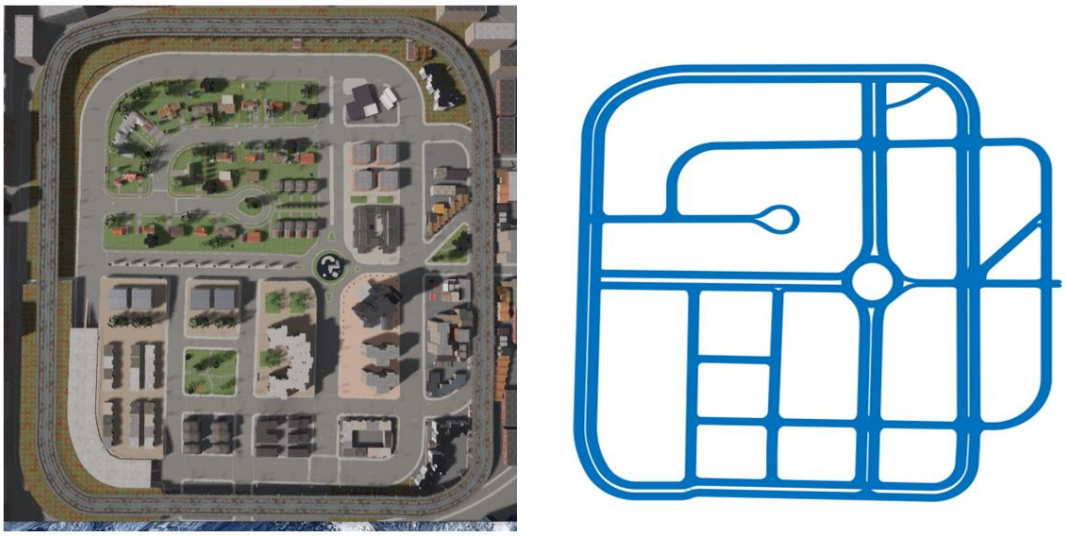

Ilustración 26: Mapa del mundo seleccionado: CARLA / MATLAB

Procediendo, se dispone del gestor de tráfico TM que se encarga de controlar los vehículos dentro de la simulación. Su objetivo es poblar la simulación con condiciones de tráfico urbano realistas que permitan a los usuarios personalizar los comportamientos.

Después de seleccionar el mundo, se escanean los blueprints y se insertan los vehículos necesarios para la simulación. Para este ejemplo, se elige el Tesla Modelo 3. Continuando, el siguiente paso es definir el punto inicial donde se va a posicionar el vehículo en el mundo. Para ello, primero, se obtiene una posible ubicación aleatoria dentro del mundo para que posteriormente sea modificada con el fin de obtener la posición deseada del vehículo habiendo cambiado únicamente los atributos de la ubicación. Conjuntamente, se configura el vehículo en piloto automático permitiendo de dicha manera que el vehículo tome sus propias decisiones en situaciones de múltiples trayectorias.

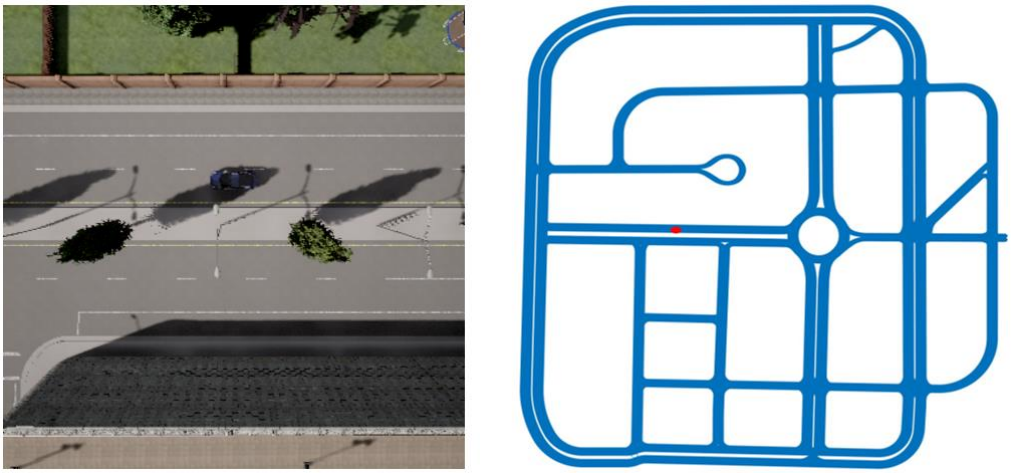

Ilustración 27: Posicionamiento inicial del vehículo: CARLA / MATLAB

El siguiente paso es añadir los sensores necesarios para el proyecto. Los sensores van unidos a otro actor, en este caso, el vehículo creado, donde al mismo tiempo se posicionan de manera similar al vehículo. Se comienza buscando el sensor correspondiente para cada caso dentro de la librería de blueprints donde, por ejemplo, para el caso de las cámaras RGB se pueden modificar los atributos de dimensión de la imagen obtenida, el intervalo de tiempo entre una captura y otra o la posición y orientación de la cámara. Por consiguiente, como Python y MATLAB se ejecutan de forma asíncrona, no es tan sencillo intercambiar datos entre ellos. Por ello, la solución a adoptar es hacer variables de alcance global en el lado de Python y dejar que MATLAB lea de él de forma asíncrona. De esta manera, la parte de Python es responsable de escribir en el buffer mientras que MATLAB sólo lee de él.

Luego, una vez definidos todos los parámetros de todos los sensores necesarios, se

realiza la simulación correspondiente para obtener los datos de la simulación, y en este caso, también las imágenes correspondientes a las cámaras planteadas. Se muestran las vistas de las cámaras seleccionadas en la posición inicial del vehículo.

Ilustración 28: Vista Lateral Izquierda / Vista Frontal / Vista Lateral Derecha

En último lugar, se realiza el almacenado de la información obtenida en la simulación y se destruyen todos los actores utilizados en la simulación. Se observa, comparándolo con la posición inicial del vehículo visualizada en la Ilustración 27, como el recorrido real realizado por el vehículo no analiza las 1000 primeras posiciones debido al tiempo de ejecución del posicionamiento del vehículo en la posición inicial descrita.

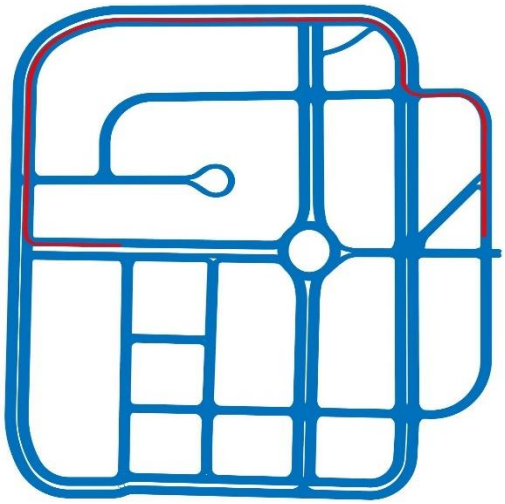

Ilustración 29: Recorrido real realizado por el vehículo

PROCESAMIENTO DE IMÁGENES

Una vez obtenidas las imágenes mediante el simulador CARLA, se procede a realizar el procesamiento correspondiente para cada una de las cámaras simuladas. Para ello, tal y como se ha comentado, la solución adoptada ha sido la utilización de visión monocular, es decir, estudiar cada una de las cámaras individualmente, realizando su procesamiento basado en un análisis de características de detección de esquinas o bordes de la imagen. A continuación, se explica teóricamente cada paso realizado en el algoritmo de procesamiento para la obtención de los resultados correspondientes.

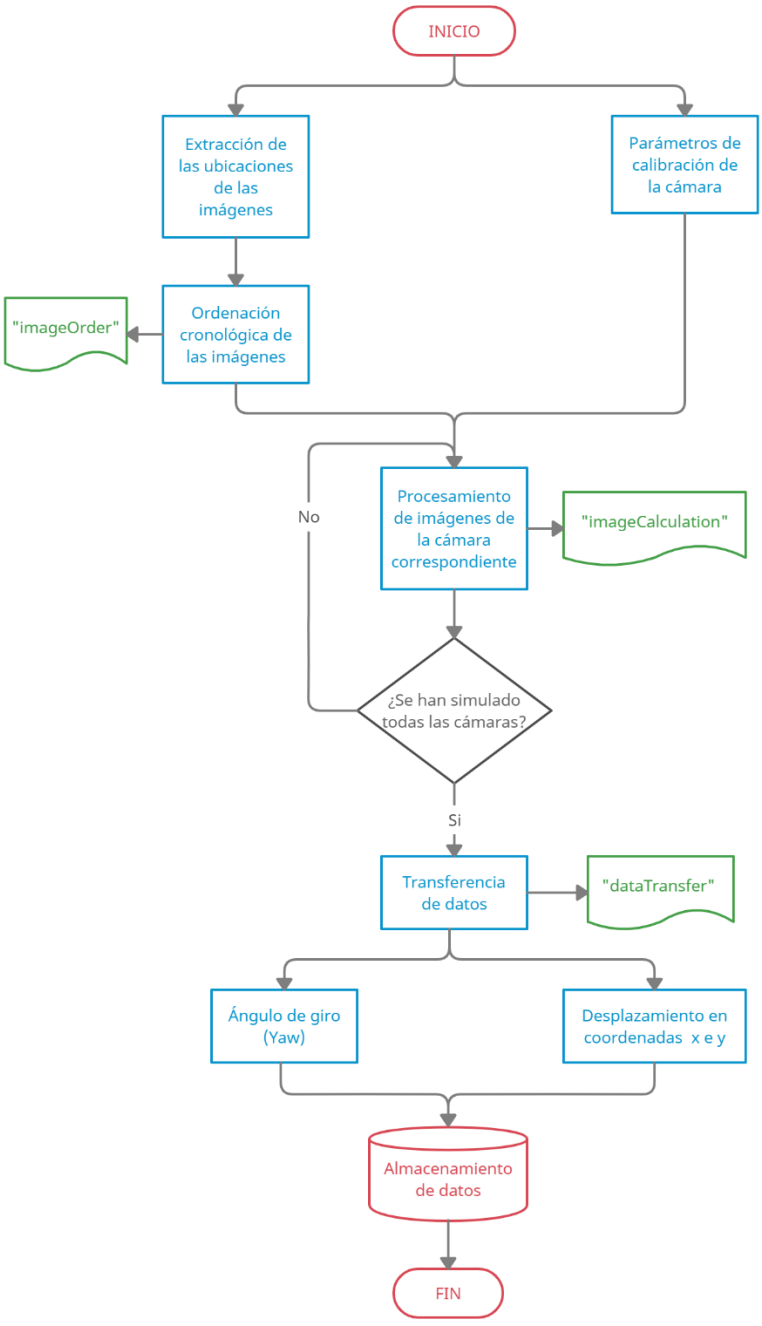

Gráfico 3: Diagrama de bloques "Main_images_CARLA"

Se presenta el algoritmo principal, el cual va a extraer y cargar las ubicaciones de las imágenes en el entorno de trabajo de MATLAB, comunicar todos los algoritmos de procesamiento y transferir y guardar los datos obtenidos del procesamiento para posteriormente ser analizados y procesados para obtener el ángulo de giro relativo y el desplazamiento relativo longitudinal y lateral entre dos fotogramas consecutivos. Para ello, en primer lugar, se parte de que las imágenes están separadas en distintas carpetas en función de la cámara utilizada para que, posteriormente, las ubicaciones de cada una de ellas sean transferidas al entorno. Conjuntamente, debido al formato de lectura de MATLAB, las imágenes deben ser ordenadas cronológicamente mediante la función "imageOrder" creada para dicho acometido. Al mismo tiempo, se especifica la matriz de parámetros de calibración de las cámaras que será transferida al algoritmo de procesamiento. Continuando, se procesan las imágenes de las cámaras individualmente mediante la intervención de la función "imageCalculation" expuesto en el siguiente diagrama de bloques.

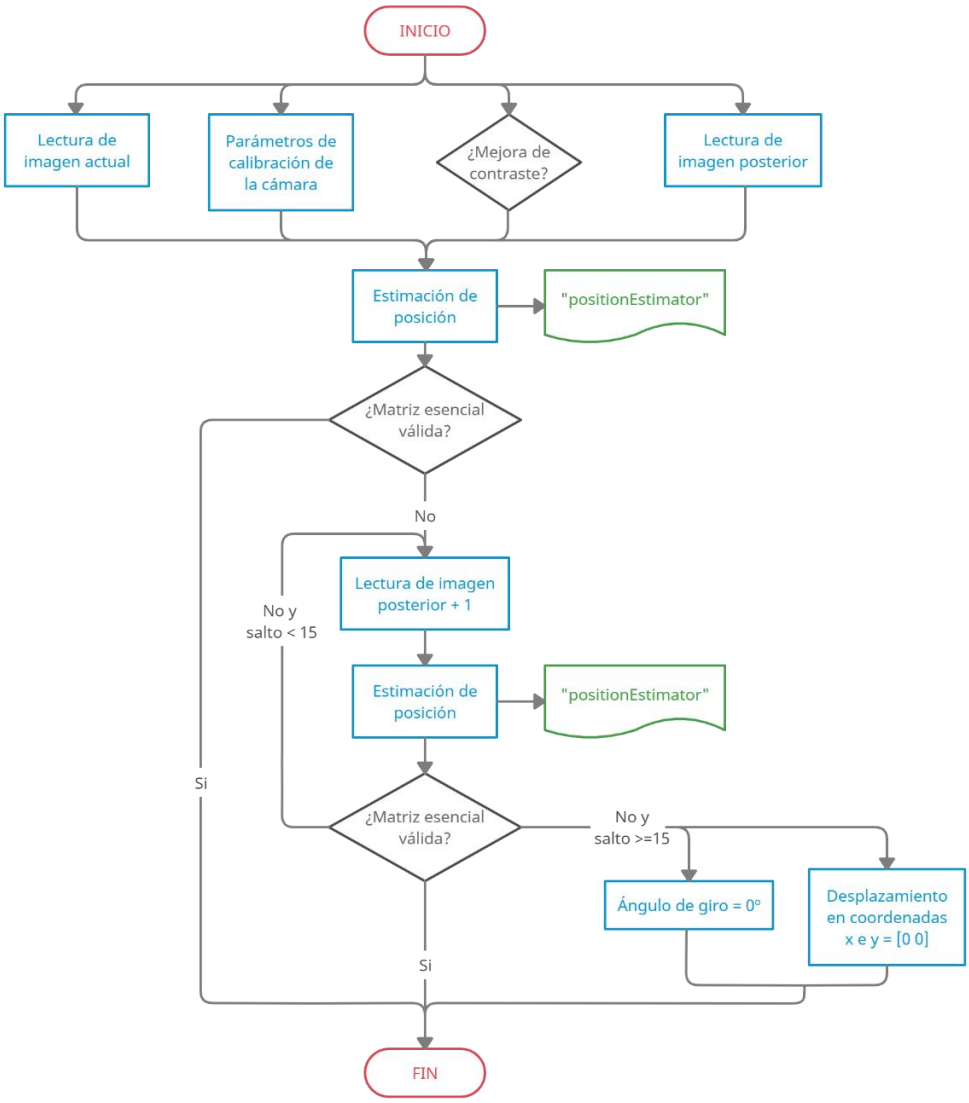

Gráfico 4: Diagrama de bloques "imageCalculation"

La principal función de este algoritmo va a ser realizar los saltos necesarios a fotogramas más alejados una vez la matriz esencial haya sido computada pero el resultado obtenido no haya sido válido debido a un factor de validación que será posteriormente explicado. Para ello, se realizará, primeramente, la lectura de las dos imágenes consecutivas que serán procesadas para obtener la matriz esencial y si el factor de validación correspondiente no es válido, este algoritmo entrará en un bucle de saltos de fotogramas consecutivos, es decir, siempre que no encuentre una solución válida, la segunda imagen pasará a ser el siguiente fotograma. Esto se realizará hasta un máximo de 15 saltos de fotograma. Si una vez completados los 15 saltos de fotograma, el factor de validación de la matriz esencial no ha sido válido, este algoritmo devolverá como solución un ángulo de giro y un desplazamiento nulo debido a que se presupone que en este intervalo de cálculo el vehículo se encuentra detenido al no detectar movimiento en 15 fotogramas consecutivos.

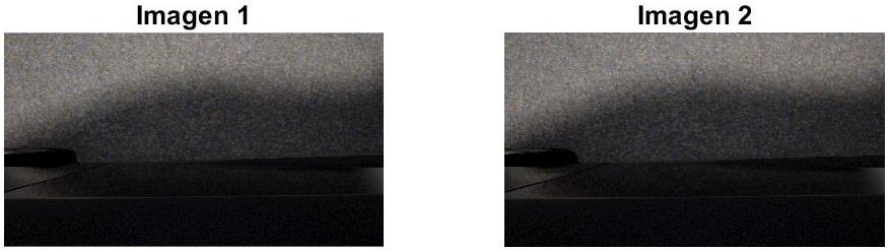

Ilustración 30: Imágenes lateral derecho iniciales consecutivas cargadas

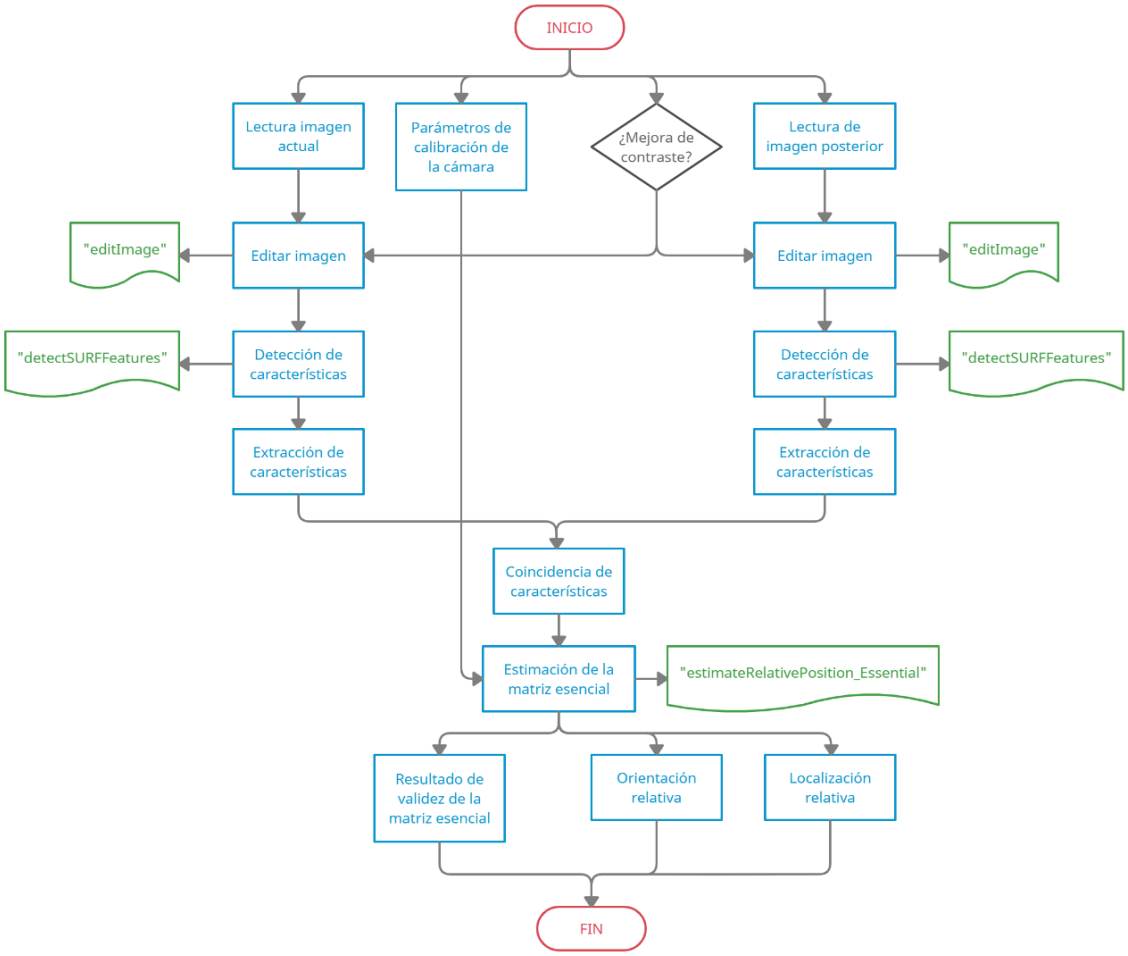

Gráfico 5: Diagrama de bloques "positionEstimator"

Explicados los algoritmos anteriores, se procede a detallar el algoritmo de estimación de posición relativa entre dos imágenes paso por paso. Para ello, en primer lugar, se modifican las imágenes en color a imágenes en escala de grises para poder transferirlas al algoritmo SURF de detección de características debido a que dicho detector trabaja con imágenes en escala de grises.

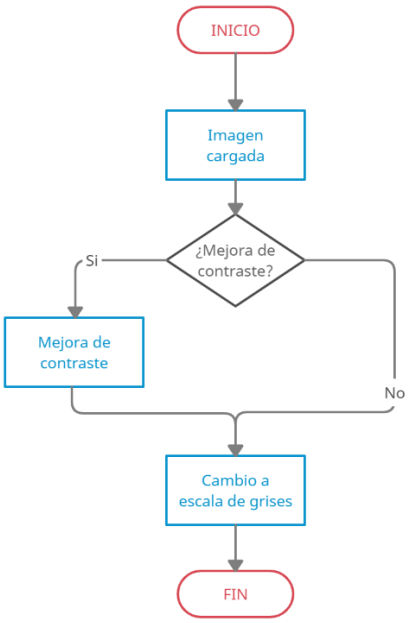

Gráfico 6: Diagrama de bloques "editImage"

Dicha edición se realiza mediante la función de “editImage” creada donde coexisten dos ediciones diferentes. En primer lugar, únicamente se modificaría la imagen en color a imagen en escala de grises y, por otro lado, la segunda variante realizaría el mismo proceso, pero entre medias realizaría una mejora de contraste para resaltar los puntos más característicos de la imagen. En este caso, se utiliza la segunda variante.

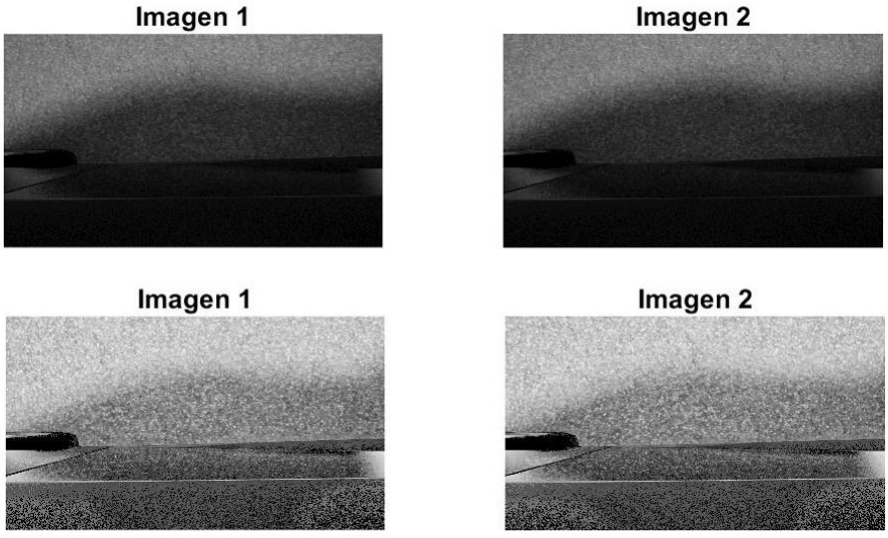

Ilustración 31: Sin Contraste (Superior) / Con Contraste (Inferior)

Volviendo al algoritmo “positionEstimator”, el siguiente paso es obtener información de la imagen basándose en las características de borde o esquinas. Para ello, en este proyecto se va a utilizar el algoritmo SURF para dicho acometido.

SURF (Speed Up Robust Features)

El método SURF (Speeded Up Robust Features) es un algoritmo rápido y robusto para la representación y comparación local e invariante de la similitud de las imágenes. El principal interés del método SURF radica en su rápido cálculo de operadores mediante filtros cuadrados, lo que permite aplicaciones en tiempo real como el seguimiento y el reconocimiento de objetos. El algoritmo SURF está compuesto por dos apartados: extracción de característica y descriptor de característica.

En primer lugar, para la detección de puntos de interés se utiliza una aproximación muy básica de la matriz hessiana. Para dicho acometido se emplea la imagen integral como una forma rápida y eficaz de calcular la suma de los valores (valores de los píxeles) en una imagen dada o un subconjunto rectangular de una cuadrícula de la imagen dada. Donde, también puede aplicarse para calcular la intensidad media dentro de una imagen dada.

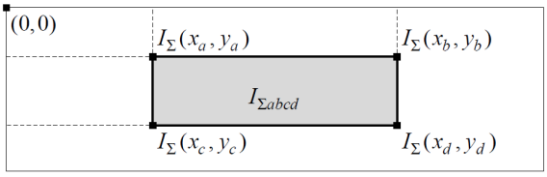

Ilustración 32: Cuadrícula de la imagen integral

Por consiguiente, permite calcular rápidamente los filtros de convolución de tipo cuadrado. Donde la entrada de una imagen integral $I_\Sigma(x)$ en un lugar $x = (x,y)^T$ representa la suma de todos los píxeles de la imagen de entrada I dentro de una región rectangular formada por el origen.

$$I_\Sigma(x,y) = \sum_{x=1}^{X} \sum_{y=1}^{Y} I(x,y)$$

Ecuación 1

Luego, una vez calculado I_Σ, sólo se necesitan cuatro adiciones para calcular la suma de las intensidades sobre cualquier área rectangular, independientemente de su tamaño.

Para ello, el algoritmo SURF utiliza la matriz hessiana por su buen rendimiento en tiempo de cálculo y precisión. En lugar de utilizar una medida diferente para seleccionar la ubicación y la escala (detector de Hessian-Laplace), SURF se basa en el determinante de la matriz hessiana para ambas. Por lo que, dado un píxel, la matriz hessiana de este píxel quedaría como:

$$H(f(x,y)) = \begin{bmatrix} \partial^2 f/\partial x^2 & \partial^2 f/\partial x \partial y \\ \partial^2 f/\partial x \partial y & \partial^2 f/\partial y^2 \end{bmatrix}$$

Ecuación 2

Donde para adaptarla a cualquier escala, se filtra la imagen mediante un núcleo gaussiano, de modo que dado un punto p=(x,y), la matriz hessiana H(p,σ) en p a escala σ se define como:

$$H(p,\sigma) = \begin{bmatrix} L_{xx}(p,\sigma) & L_{xy}(p,\sigma) \\ L_{xy}(p,\sigma) & L_{yy}(p,\sigma) \end{bmatrix}$$

Ecuación 3

donde $L_{xx}(p,\sigma)$ es la convolución de la derivada gaussiana de segundo orden con la imagen I en el punto x, $\frac{\delta^2}{\delta x^2} g(\sigma)$, y de igual manera calculada para $L_{xy}(p,\sigma)$ y $L_{yy}(p,\sigma)$. Las gaussianas son óptimas para el análisis del espacio de escala, pero en la práctica hay que discretizarlas y recortarlas. Esto lleva a una pérdida de repetibilidad bajo rotaciones de la imagen alrededor de múltiplos impares de π/4. Esta debilidad es válida para los detectores basados en el hessiano en general. Sin embargo, los detectores siguen funcionando bien, y la ligera disminución del rendimiento no neutraliza la ventaja de las convoluciones rápidas que aportan la discretización y el recorte.

Luego, para calcular el determinante de la matriz hessiana, primero hay que aplicar la convolución con el núcleo gaussiano y luego la derivada de segundo orden. Donde SIFT consigue la solución mediante logaritmos, SURF lleva la aproximación tanto la convolución como la derivada de segundo orden aún más lejos con los filtros cuadrados. Éstos se aproximan a las derivadas gaussianas de segundo orden y pueden evaluarse a un coste computacional muy bajo utilizando imágenes integrales e independientemente del tamaño, y ésta es parte de la razón por la que SURF es rápido.

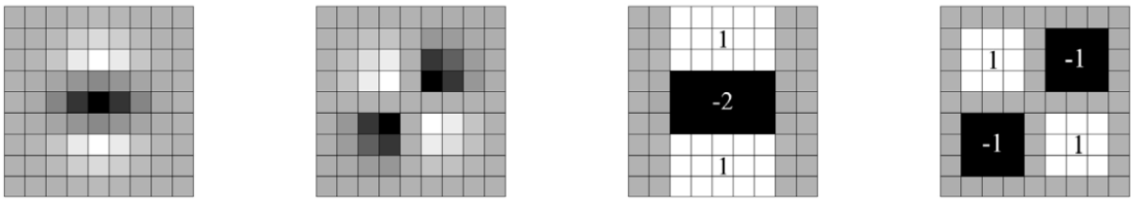

Ilustración 33: Derivada de 2º orden Gaussiana / Aproximación filtros cuadrados

Los filtros de caja de 9×9 en las imágenes anteriores son aproximaciones para derivadas gaussianas de segundo orden con un σ=1,2 donde estas aproximaciones son denotadas como D_{xx}, D_{yy} y D_{xy}. Luego, representando el determinante del hessiano con dichos valores donde w es un coeficiente de ponderación igual a w=1-1/8=0.875≈0.9 quedaría:

$$det(H(x,y)) = D_{xx}D_{yy} - \left(wD_{xy}\right)^2$$

Ecuación 4

Continuando, los espacios de escala suelen implementarse como pirámides de imágenes. Las imágenes se suavizan repetidamente con una gaussiana y posteriormente se sub-muestran para conseguir un nivel superior de la pirámide. Gracias al uso de filtros cuadrados e imágenes integrales, el algoritmo SURF no tiene que aplicar iterativamente el mismo filtro a la salida de una capa previamente filtrada, sino que puede aplicar dichos filtros de cualquier tamaño a la misma velocidad directamente sobre la imagen original, e incluso en paralelo. Por lo tanto, el espacio de escala se analiza aumentando el tamaño del filtro (9×9 → 15×15 → 21×21 → 27×27, etc) en lugar de reducir iterativamente el tamaño de la imagen. Por lo tanto, para cada nueva octava, el aumento del tamaño del filtro duplica simultáneamente los intervalos de muestreo para la extracción de los puntos de interés (σ) permitiendo de dicha manera el aumento de la escala del filtro a un coste constante. Luego, para localizar los puntos de interés en la imagen y sobre las escalas, se aplica una supresión no máxima en una vecindad de 3×3×3.

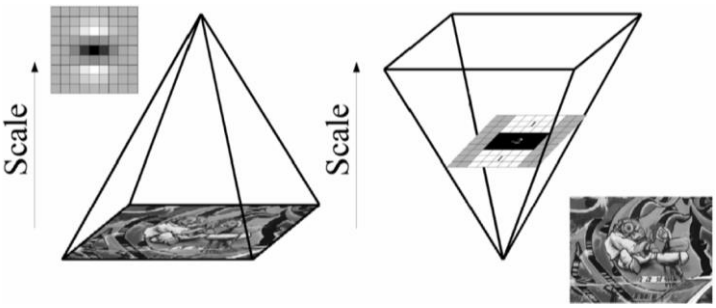

Ilustración 34: Cálculo del espacio de escala por SIFT y SURF

Por consiguiente, en segundo lugar, se encuentra el descriptor de característica que tiene lugar en dos pasos. El primer paso consiste en fijar una orientación reproducible basada en la información de una región circular alrededor del punto clave. Donde, en el segundo paso, se construye una región cuadrada alineada con la orientación seleccionada extrayendo de ella el descriptor SURF.

Para ser invariante a la rotación, el algoritmo SURF intenta identificar una orientación reproducible para los puntos de interés donde para lograrlo sigue los siguientes pasos:

1. En primer lugar, se calculan las respuestas de las ondículas de Haar en las direcciones x e y en una zona circular de radio 6s alrededor del punto clave, siendo s la escala a la que se detectó el punto clave. Además, el paso de muestreo depende de la escala y se elige para s, donde las respuestas de las ondículas se calculan a esa escala actual s. En consecuencia, a escalas altas el tamaño de las ondículas es grande. Por lo tanto, las imágenes integrales se utilizan de nuevo para el filtrado rápido.

2. Continuando, se calcula la suma de las respuestas de las ondículas verticales y horizontales en un área de exploración. Donde cambiando pausadamente la orientación de la exploración (añadiendo π/3) y volviendo a calcular, se encontrará la orientación con el mayor valor de la suma, quedando por tanto esta orientación como la orientación principal del descriptor de características.

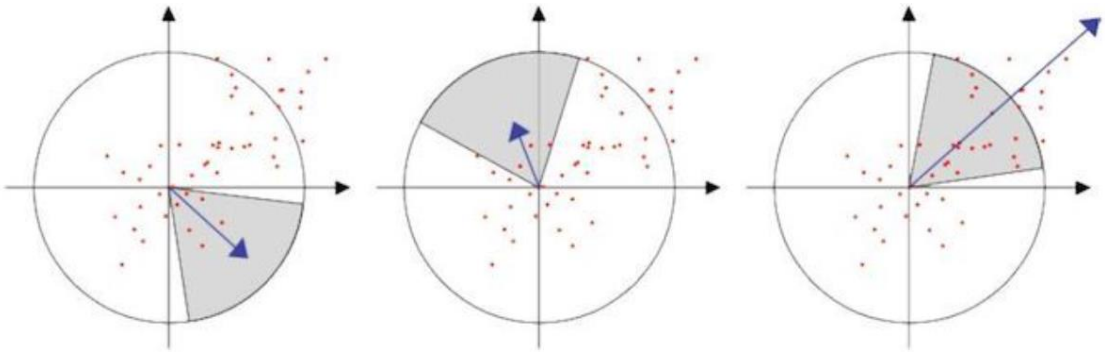

Ilustración 35: Asignación de la orientación del punto característico

Continuando, se extrae el descriptor de característica. Para ello, el primer paso consiste en construir una región cuadrada centrada en el punto clave y orientada según la orientación obtenida donde el tamaño de esta ventana es de 20s.

Seguidamente, la región se divide en subregiones cuadradas más pequeñas de 4×4 donde para cada subregión, se calculan algunas características simples en puntos de muestreo espaciados regularmente 5×5. Por razones de simplicidad, se llama d_x a la respuesta de la ondícula de Haar en la dirección horizontal y d_y a la respuesta de la ondícula de Haar en la dirección vertical. Al mismo tiempo, para aumentar la solidez frente a las deformaciones geométricas y los errores de localización, las respuestas d_x y d_y se ponderan primero con una gaussiana (σ=3,3s) centrada en el punto clave.

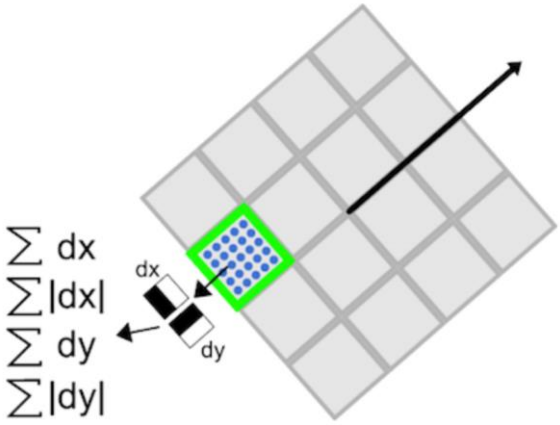

Ilustración 36: Respuesta de una ondícula de Haar

Prosiguiendo, las respuestas de las ondículas en d_x y d_y se suman en cada subregión y forman un primer conjunto de entradas del vector de características. Para aportar información sobre la polaridad de los cambios de intensidad, también se extrae la suma de los valores absolutos de las respuestas, $|d_x|$ y $|d_y|$. Por lo tanto, cada subregión tiene un vector descriptor de cuatro dimensiones v para su estructura de intensidad subyacente v = ($\sum d_x$, $\sum d_y$, $\sum|d_x|$, $\sum|d_y|$). Esto da como resultado un vector descriptor para todas las subregiones de 4×4 de longitud 64.

Luego, una vez descrito el funcionamiento del algoritmo SURF, se muestran los resultados obtenidos aplicando el algoritmo sobre la imagen mostrada anteriormente donde se le ha determinado la zona de búsqueda.

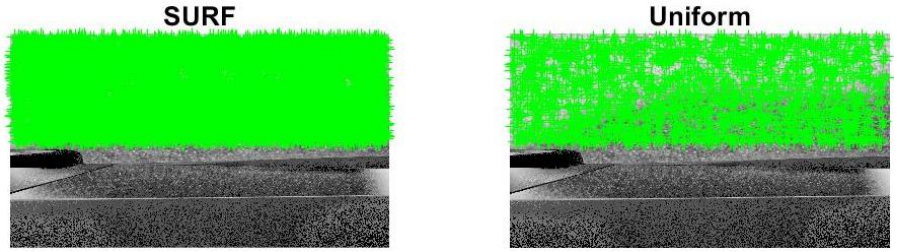

Ilustración 37: Extracción de características)

Empleado el algoritmo SURF para obtener la posición de los puntos característicos y los detectores de cada una de las imágenes, los puntos de características repetidos obtenidos de cada una de las imágenes deben ser coincidentes. Es decir, la coincidencia de los puntos característicos es la comparación de las distancias entre los vectores de los puntos característicos de cada una de ellas. La fórmula de la distancia vectorial es:

$$d = \sqrt{\sum_{i=0,1,\ldots} [V_1(i) - V_2(i)]^2}$$

Ecuación 5

donde V_1 y V_2, respectivamente, representan los descriptores de características de los dos puntos de características SURF. La distancia entre dos descriptores de características puede reflejar un grado de similitud. Cuanto menor sea la distancia d, mayor será el grado de similitud, y más representativo será el par de puntos correcto. Se observa la coincidencia de puntos entre las dos imágenes consecutivas mostradas.

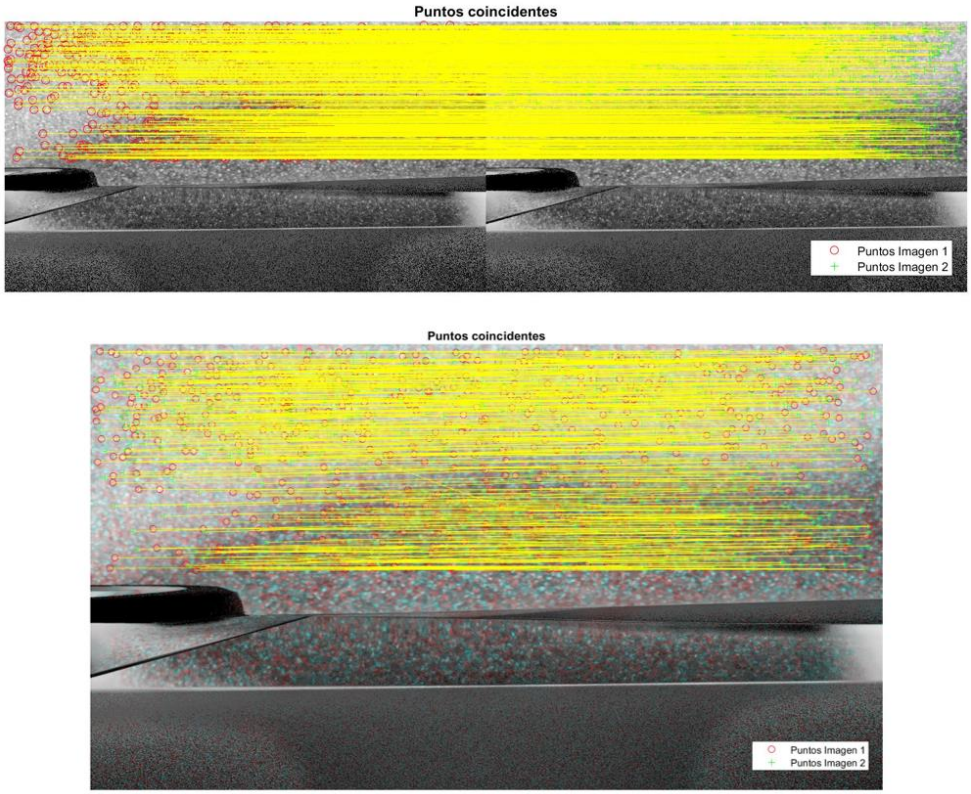

Ilustración 38: Extracción de puntos coincidentes

Luego, una vez obtenida la coincidencia de puntos entre las dos imágenes consecutivas, es estimar la matriz esencial para obtener el desplazamiento relativo entre las dos imágenes consecutivas. Para ello, disponemos de la función "EstimateRelativePosition_Essential" creada.

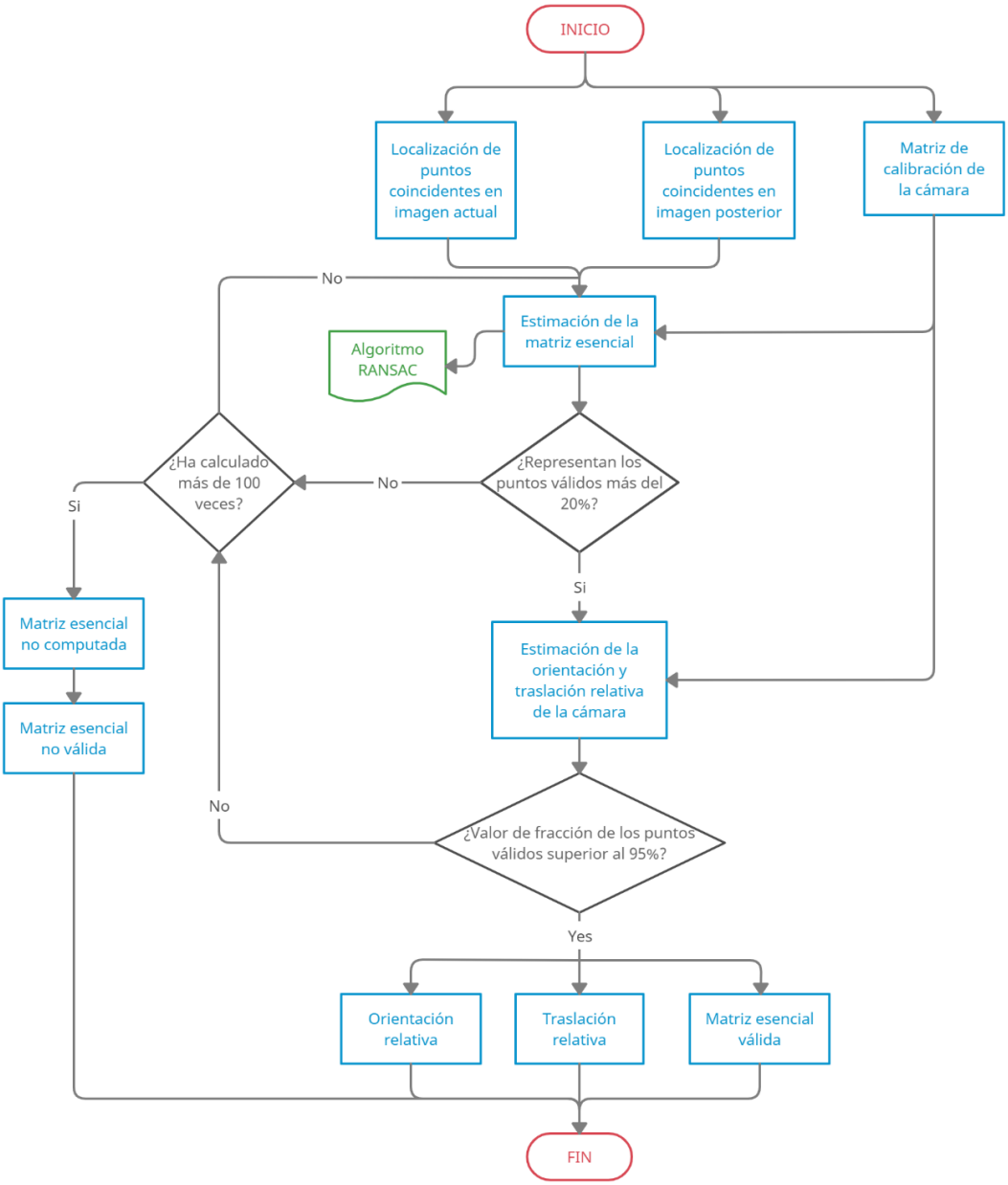

Gráfico 7: Diagrama de bloques "estimateRelativePosition_Essential"

Dentro de este algoritmo se va a realizar el proceso de estimación de la matriz esencial. Para ello, se va a detallar teóricamente cada uno de los elementos que intervienen en el algoritmo. En primer lugar, se debe de comprender que es la matriz de parámetros de calibración de la cámara.

Matriz de parámetros de calibración

La calibración de la cámara tiene como finalidad obtener todos los parámetros que son necesarios para la formación de la imagen. Por lo tanto, mediante los algoritmos de calibración se obtienen tanto los parámetros geométricos como los radiométricos. A grandes rasgos los parámetros geométricos se pueden clasificar en dos tipos: parámetros intrínsecos y parámetros extrínsecos.

En general, puede afirmarse que la relación existente entre cualquier punto p del espacio y su proyección p' en el plano de la imagen se puede simplificar con la siguiente ecuación:

$$\begin{pmatrix} p' \\ 1 \end{pmatrix} = K \begin{pmatrix} & R & & t \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} p \\ 1 \end{pmatrix}$$

Ecuación 6

donde K es una matriz 3x3 que representa los parámetros intrínsecos de la cámara, R es una matriz de rotación 3x3 y t es un vector de traslación de la cámara con sistema de referencia global. R y t son los parámetros extrínsecos de la cámara.

Parámetros Intrínsecos

Los parámetros intrínsecos son todo aquellos que tienen relación con el conjunto de la cámara y la óptica, es decir, son los que están involucrados en la transformación de puntos 3D en el sistema de referencia de la cámara a puntos 2D del plano de la imagen. Dentro de este tipo se incluyen el desplazamiento del centro de la imagen, la distancia focal, los coeficientes de distorsión y el tamaño de cada píxel.

Los parámetros intrínsecos se pueden agrupar mayoritariamente dentro de una matriz, denominada matriz intrínseca de calibración para facilitar su posterior tratamiento matemático. Dicha matriz tiene la forma:

$$K = \begin{pmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}$$

Ecuación 7

donde f_x y f_y son las longitudes focales de la lente de la cámara en el eje x y en el eje y respectivamente. Estos parámetros se miden en pixeles y se obtienen según:

$$f_x = f S_x \quad ; \quad f_y = f S_y$$

Ecuación 8

donde f es la distancia focal de la lente de la cámara medida en unidades de longitud y S_x y S_y son el número de píxeles por unidad de longitud del sensor en el eje x y en el eje y respectivamente. Conjuntamente, s es un factor que indica el grado de perpendicularidad de las paredes de los píxeles del sensor. Tiene un valor inversamente proporcional a la tangente del ángulo que forma el eje x con el eje y, lo que implica que generalmente se puede asumir como nula su aportación en la matriz debido a que en la mayor parte de los sensores actuales los píxeles son rectangulares. Para terminar, c_x y

c_y determinan el desplazamiento del centro de la imagen.

Por consiguiente, el hecho de incorporar una lente a la cámara da pie a presentar el último parámetro intrínseco a tener en cuenta: la distorsión de la lente. Debido a ella, se deteriora la calidad geométrica de la imagen hasta el grado de que se pierde parte de la capacidad de medir las posiciones de los objetos con exactitud. Dicha distorsión surge porque los rayos de luz no emergen en las lentes en las direcciones previstas por las leyes ópticas. Luego, se pueden distinguir dos tipos: radial y tangencial.

La distorsión radial de la lente provoca que los puntos de las imágenes se desplacen de forma radial a partir del eje óptico. Es decir, favorece que los rayos más alejados del centro óptico se curven bastante más que aquellos que inciden directamente en las proximidades del centro de la lente. Para modelar matemáticamente con la máxima precisión la distorsión radial se requeriría una sucesión infinita de elementos. Por el contrario, en la práctica, y para la inmensa mayoría de las aplicaciones dentro de la visión artificial, con tener en cuenta los dos primeros términos de la serie, o incluso únicamente el primero, es suficiente. De esta manera, la ecuación que relaciona los píxeles con distorsión y los que carecen de ella, quedaría:

$$x_u = x_d(1 + k_1 r^2 + k_2 r^4)$$

$$y_u = y_d(1 + k_1 r^2 + k_2 r^4)$$

$$r = \sqrt{(x_d^2 + y_d^2)}$$

Ecuación 9

donde k_1 y k_2 son los coeficientes de distorsión, x_d e y_d son las coordenadas del píxel distorsionado y x_u e y_u son las coordenadas del píxel corregido. El grado de distorsión de la imagen se determina por el valor de los coeficientes de distorsión. Según sus signos se puede hablar de dos tipos de distorsión radial: positiva y negativa. La distorsión positiva se refleja en la formación de una imagen de tipo cojín, mientras que la negativa produce una imagen de tipo barril.

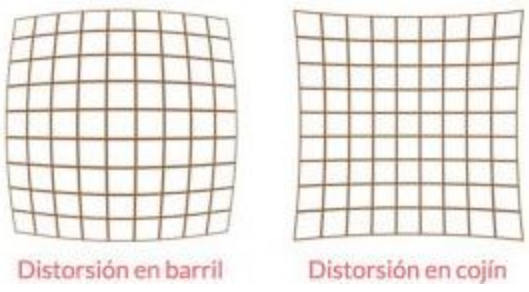

Ilustración 39: Distorsión radial: Tipo barril y tipo cojín

Por otro lado, la distorsión tangencial se produce perpendicularmente a las líneas radiales a partir del eje óptico. Su principal causa es un centrado defectuoso de todos los elementos que configuran el sistema de lentes. Los efectos de la distorsión tangencial son menos significativos que los de la distorsión radial. Por ello, en la mayor parte de las aplicaciones, se considerarán despreciables, teniendo únicamente en cuenta los radiales.

Simultáneamente, al igual que la radial, la forma más precisa de modelar la distorsión tangencial también es como una sucesión infinita de elementos. Aunque, de la misma manera que en la anterior, se puede aproximar matemáticamente, sin riesgo a cometer un grave error. Así, las ecuaciones que relacionan los píxeles quedarían:

$$x_u = x_d + (2p_1 y_d + p_2(r^2 + 2x_d^2))$$
$$y_u = y_d + (2p_1 x_d + p_1(r^2 + 2y_d^2))$$

Ecuación 10

donde p_1 y p_2 son los coeficientes de distorsión, y al igual que antes x_d e y_d son las coordenadas del píxel distorsionado y x_u e y_u son las coordenadas del píxel corregido. Todos los coeficientes de distorsión, radiales y tangenciales, se pueden agrupar en un único vector, denominado vector de distorsión, para trabajar matemáticamente de una manera más sencilla. En resumen, los parámetros intrínsecos de la cámara los conforman tanto los pertenecientes a la matriz de calibración como al vector de distorsión quedando completamente definidos si se conocen los términos de ambos.

Parámetros Extrínsecos

Los parámetros extrínsecos son aquellos que hacen referencia a la colocación física de la cámara, es decir, los que definen la posición y la orientación de la cámara con respecto al sistema de coordenadas global, es decir, no dependen de la propia cámara sino de su disposición espacial. Dentro de éstos se incluyen todos los componentes del vector de traslación y los de las matrices de rotación de la cámara con respecto a cada uno de los ejes.

Consecuentemente, la matriz de rotación R reproduce un giro de la cámara o de un objeto respecto de ella. Y, como se ha mencionado, adoptará distintas formas según el eje coordenado de referencia:

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\alpha & sin\alpha \\ 0 & -sin\alpha & cos\alpha \end{bmatrix} \quad R_y = \begin{bmatrix} cos\beta & 0 & sin\beta \\ 0 & 1 & 0 \\ -sin\beta & 0 & cos\beta \end{bmatrix} \quad R_z = \begin{bmatrix} cos\gamma & sin\gamma & 0 \\ -sin\gamma & cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Ecuación 11

Luego, en caso de que el giro se realizase con respecto a cualquiera de los tres ejes, se procedería de una manera similar para obtener las nuevas coordenadas, únicamente habría que cambiar la matriz de rotación empleada. Por ejemplo, si el giro tuviera lugar respecto al eje Z, la matriz utilizada sería R_z y las nuevas coordenadas quedarían:

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{bmatrix} cos\gamma & sin\gamma & 0 \\ -sin\gamma & cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

Ecuación 12

Por otro lado, el vector t se define como un vector de traslación que reproduce un cambio en la posición del sistema de coordenadas de la cámara o de un objeto con respecto a ella.

$$t = \begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix}$$

Ecuación 13

donde t_x, t_y y t_z son los desplazamientos que tienen lugar en cada uno de los ejes coordenados.

Por lo tanto, conocidos todos los parámetros de la matriz de calibración, en esta ocasión debido a que las imágenes son obtenidas mediante un simulador de vehículos autónomos y analizadas cada una de ellas por separado, únicamente se deben de tener en cuenta los parámetros intrínsecos de calibración despreciando de esta manera los parámetros de distorsión y parámetros extrínsecos.

Seguidamente, para realizar la estimación de la matriz esencial, son necesarias las localizaciones de los puntos característicos coincidentes en cada una de las imágenes. Luego, una vez obtenidos dichos datos se puede proceder a calcular la matriz esencial mediante la utilización del algoritmo RANSAC.

Algoritmo RANSAC

El algoritmo RANdom SAmple Consensus (RANSAC) propuesto por Fischler y Bolles es un enfoque general de estimación de parámetros diseñado para hacer frente a una gran proporción de valores atípicos en los datos de entrada.

RANSAC es una técnica de re-muestreo que genera soluciones candidatas utilizando el mínimo número de observaciones (puntos de datos) necesario para estimar los parámetros del modelo subyacente. Como señalan Fischler y Bolles, a diferencia de las técnicas convencionales de muestreo que utilizan la mayor cantidad de datos posible

para obtener una solución inicial y luego proceder a eliminar los valores atípicos, RANSAC utiliza el conjunto más pequeño posible y procede a ampliar este conjunto con puntos de datos consistentes.

El algoritmo básico se resume en:

1. Seleccionar al azar el número mínimo de puntos necesarios para determinar los parámetros del modelo.

2. Resolver los parámetros del modelo.

3. Determinar cuántos puntos del conjunto de todos los puntos se ajustan con una tolerancia predefinida.

4. Si la fracción del número de puntos válidos sobre el número total de puntos del conjunto supera un umbral predefinido τ, se vuelven a estimar los parámetros del modelo utilizando todos los valores típicos.

5. En caso contrario, se repiten los pasos 1 a 4 (un máximo de N veces).

El número de iteraciones, N, se elige lo suficientemente alto como para garantizar la probabilidad p (normalmente fijada en 0,99) de que al menos uno de los conjuntos de muestras aleatorias no incluya un valor atípico. Por consiguiente, sea u la probabilidad de que cualquier punto de datos seleccionado sea un punto válido y v=1-u la probabilidad de observar un valor atípico. Por lo tanto, se requieren N iteraciones del mínimo número de puntos denotados m, donde:

$$1 - p = (1 - u^m)^N$$

Ecuación 14

que despejando N quedaría:

$$N = \frac{\log(1-p)}{\log(1-(1-v)^m)}$$

Ecuación 15

Este resultado asume que los m puntos de datos se seleccionan independientemente, es decir, se sustituye un punto que ha sido seleccionado una vez y se puede seleccionar de nuevo en la misma iteración. A continuación, se puede observar la eliminación de los valores atípicos en comparación con la Ilustración 38 donde se pueden comparar los resultados obtenidos tanto en línea recta como en línea curva después del uso del algoritmo RANSAC.

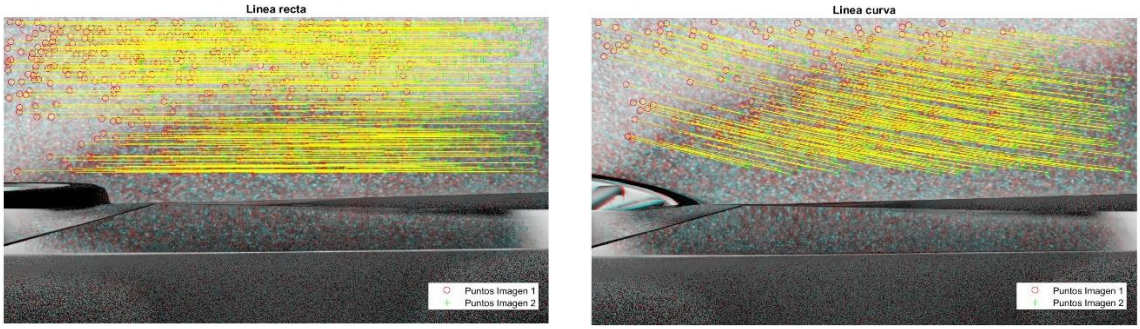

Ilustración 40: Eliminación de los valores atípicos mediante el algoritmo RANSAC

Una vez desechados los valores atípicos de la imagen, se puede estimar la matriz esencial.

Matriz esencial

La matriz esencial proporciona la relación que existe entre los dos instantes consecutivos de un punto cualquiera del espacio visto desde una misma cámara en distintos instantes o de dos cámaras en el mismo instante. Es decir, vincula las coordenadas de la proyección de un punto en el instante actual con las coordenadas de la proyección del mismo punto en el instante posterior. Habría que destacar una serie de propiedades de la matriz esencial:

- La matriz esencial tiene dimensiones 3x3 de rango 2. Por lo que su determinante es nulo.

- Los dos valores propios de la matriz esencial son iguales.

- La matriz esencial contiene cinco parámetros: tres correspondientes a la matriz de rotación y dos al vector de traslación.

Cabe destacar que, en este sistema, la forma más común de utilizarlo es para la relación matemática de los términos de la geometría epipolar, es decir, la correspondencia de las dos proyecciones de un mismo punto sobre los planos de las cámaras entre si de un sistema estéreo. Por consiguiente, en este caso, se va a considerar que los dos fotogramas consecutivos obtenidos por la misma cámara en instantes diferentes es equivalente a la obtención de dos imágenes en el mismo instante por dos cámaras en un sistema estéreo. Por lo tanto, se puede aplicar las reglas de geometría epipolar entre dos imágenes consecutivas.

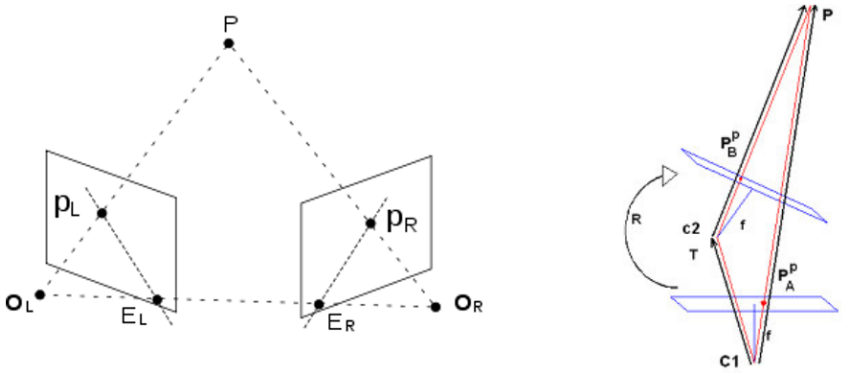

Ilustración 41: Geometría epipolar: Visión estereoscópica / Visión monocular

Seguidamente, sea P un punto 3D en el espacio y los correspondientes vectores de coordenadas en los dos fotogramas consecutivos sean P_A y P_B. P_A contiene las coordenadas del punto P en la referencia de la cámara en el fotograma actual y P_B contiene las coordenadas del punto P en la referencia de la cámara para el fotograma posterior. Si denotamos por $T=[tx,ty,tz]^T$ las coordenadas del vector de traslación de la referencia de la cámara en el fotograma anterior y por R la matriz de rotación entre los fotogramas consecutivos, obtenemos la siguiente relación:

$$P_B^T \cdot E \cdot P_A = 0$$

Ecuación 16

donde E = R·S y S es igual a:

$$S = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix}$$

Ecuación 17

Por lo tanto, la matriz esencial se puede calcular teniendo correspondencia de al menos 8 puntos entre dos fotogramas consecutivos utilizando el algoritmo de los ocho puntos.

Prosiguiendo, se debe de conocer la posición relativa de la cámara entre los dos fotogramas consecutivos. Para ello, primeramente, se debe de estimar la rotación y la traslación a partir de la matriz esencial utilizando el método basado en la descomposición del valor singular (SVD). Donde sea $E \approx UDV^T$ la descomposición del valor singular de la matriz esencial, y U y V elegidos de forma que det(U)>0 y det(V)>0. Se denomina D como una matriz diagonal del estilo:

$$D = \begin{pmatrix} \varepsilon & 0 & 0 \\ 0 & \varepsilon & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

Ecuación 18

quedando el vector de traslación y de rotación:

$$R = U \cdot W^{-1} \cdot V^T$$

$$t = U \cdot W \cdot D \cdot U^T$$

Ecuación 19

donde el valor de M viene dado por:

$$M = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Ecuación 20

Por otro lado, se debe de superar un valor de fracción de puntos válidos que se proyectan sobre las dos cámaras para realizar otra confirmación de que la matriz esencial obtenida es válida.

Por consiguiente, antes de volver al algoritmo principal, obtenida la matriz esencial validada y los puntos válidos coincidentes entre las dos imágenes consecutivas, se procede a calcular los desplazamientos relativos longitudinales y laterales. Para ello, realizando la diferencia de posición de los puntos característicos válidos en pixeles en ambas coordenadas de la imagen, se obtiene en distancia de pixeles el desplazamiento longitudinal y lateral realizado por el vehículo autónomo. Luego, habiendo realizado un exhaustivo cálculo, y obtenida la relación tamaño/pixel, se llega a obtener la dimensión de cada pixel en ambas coordenadas. Donde, aplicando dicha relación, se obtiene el desplazamiento. Seguidamente, obtenido el desplazamiento en ambas coordenadas, se obtiene el ángulo de giro correspondiente.

Luego, volviendo al algoritmo principal “Main_images_CARLA”, y estando procesados todos los conjuntos de imágenes, se procede a transferir los datos correctamente mediante la función “dataTransfer” creada. Este paso es debido a que la solución devuelta ha podido realizar saltos a fotogramas más alejados para estimar la posición relativa de la cámara respecto al instante anterior y, por lo tanto, se debe de fraccionar para estimar el desplazamiento relativo entre dos fotogramas consecutivos. Para ello, en primer lugar, se resta el punto focal para cada una de las 3 coordenadas de traslación y si el resultado no ha sido posible obtenerse mediante fotogramas consecutivos y el

algoritmo ha tenido que saltar a fotogramas más alejados, se divide el resultado obtenido entre los saltos dados para que de dicha manera se consiga obtener el desplazamiento relativo entre dos instantes consecutivos.

Donde, finalmente, después de realizar el completo procesamiento de imágenes, se obtiene el ángulo de giro relativo y el desplazamiento relativo en las coordenadas x e y entre dos fotogramas consecutivos para cada uno de los instantes simulados por cada una de las cámaras utilizadas. Estos resultados son almacenados de manera independiente para cada cámara para su posterior procesado de datos.

PROCESAMIENTO DE DATOS

En tercer lugar, obtenidos los datos correspondientes a ángulo de giro y desplazamiento relativo para cada una de las cámaras, mediante el algoritmo que se plantea en el siguiente diagrama de bloques se realizará un procesado de las imágenes y su posterior combinación de datos de las distintas cámaras para obtener el ángulo de giro y desplazamiento del vehículo autónomo final entre dos fotogramas consecutivos.

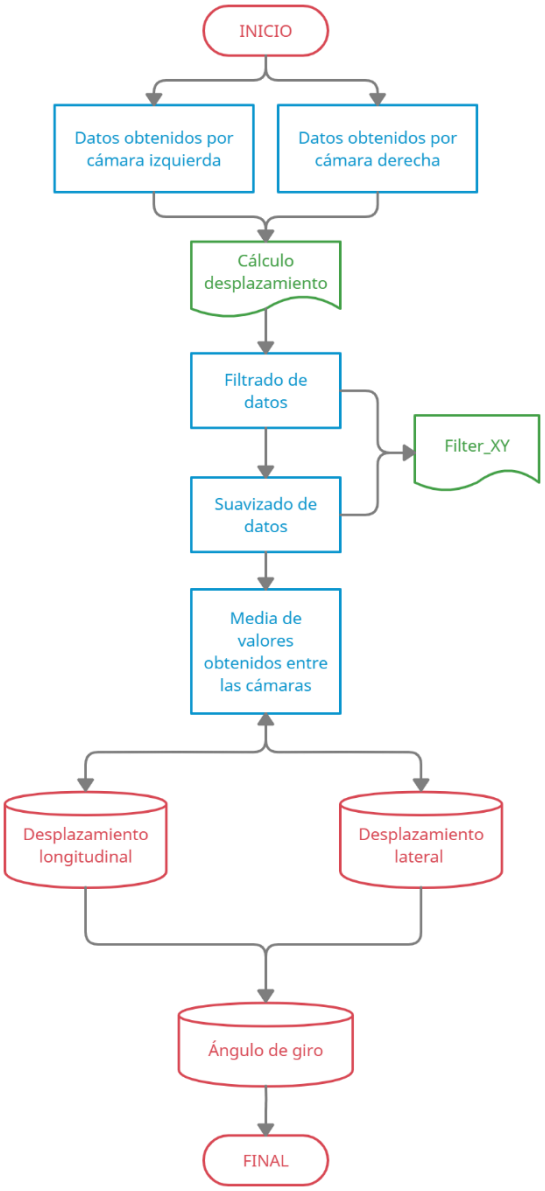

Gráfico 8: Diagrama de bloques "dataProcess"

Como se puede observar se trata de un procesamiento simple en el cual en primar lugar se realiza un tratamiento de los datos para filtrar errores no deseados en la señal y su posterior suavizado. Donde, luego, se realiza una media de los valores obtenidos por las cámaras respecto al desplazamiento y finalmente se obtiene el ángulo de giro a través

del desplazamiento longitudinal y lateral.

Por último, analizados todos los tramos del recorrido realizado por el vehículo autónomo, se obtiene de manera satisfactoria el desplazamiento tanto longitudinal como lateral relativo y su correspondiente ángulo de giro como se va a poder observar en la sección “ANÁLISIS DE LOS RESULTADOS” para los distintos escenarios.

10. ANÁLISIS DE LOS RESULTADOS

Después de explicar detalladamente los pasos dados para obtener y procesar los datos relacionados con al ángulo de giro y desplazamiento relativo entre dos fotogramas consecutivos, en esta sección se procede a mostrar los resultados obtenidos en distintos escenarios simulados. Conjuntamente, los distintos escenarios mostrados serán explicados detalladamente y comparados entre sí.

ESCENARIO 1

En primer lugar, se muestra el recorrido real realizado por el vehículo autónomo (posición inicial en punto naranja):

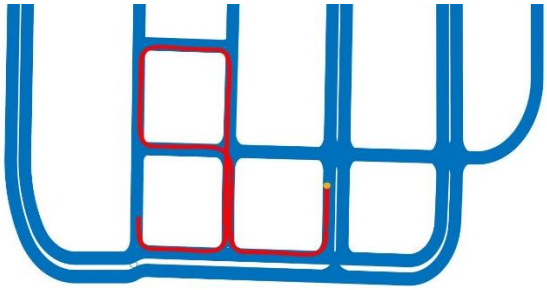

Ilustración 42: Recorrido real: Escenario 1

Continuando, se muestran las gráficas de ángulo de giro, desplazamiento longitudinal y lateral relativo entre fotogramas consecutivos y sus respectivos errores relativos del escenario 1:

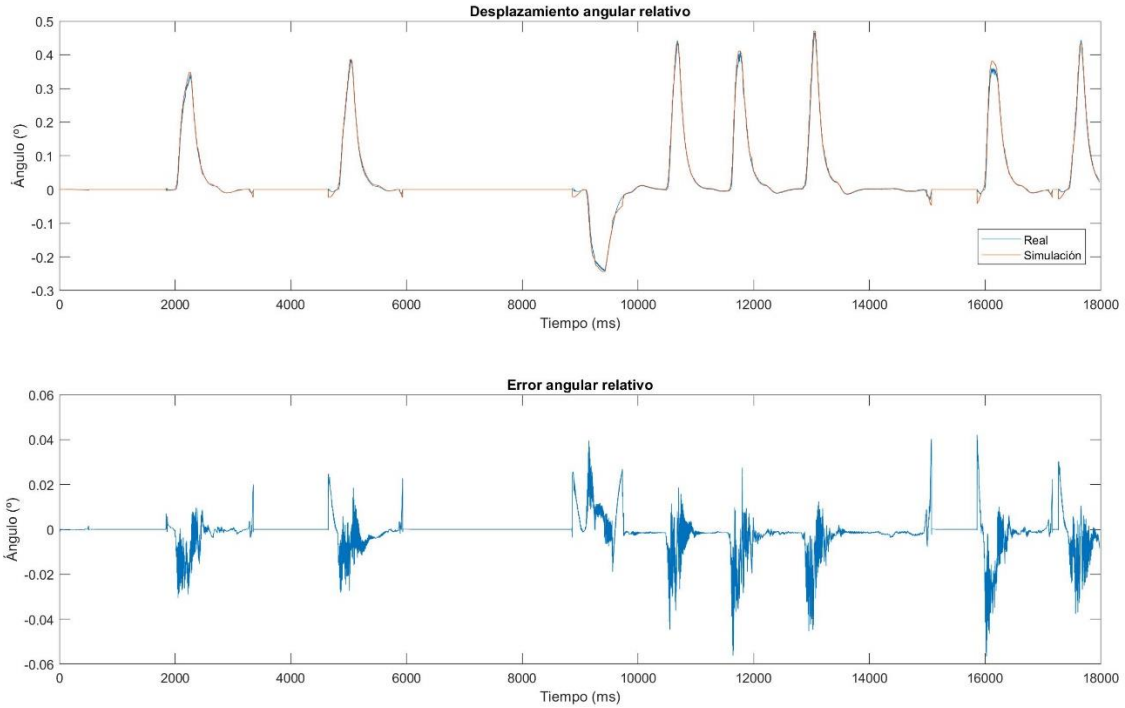

Gráfico 9: Desplazamiento y error angular relativo Escenario 1

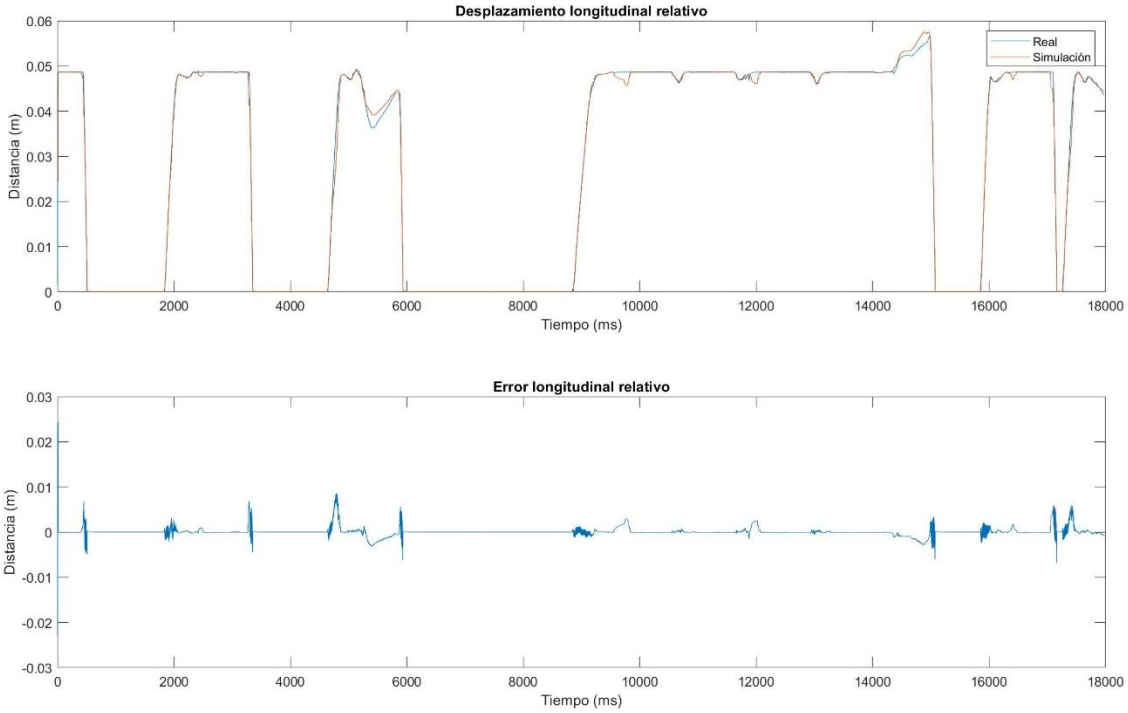

Gráfico 10: Desplazamiento y error longitudinal relativo Escenario 1

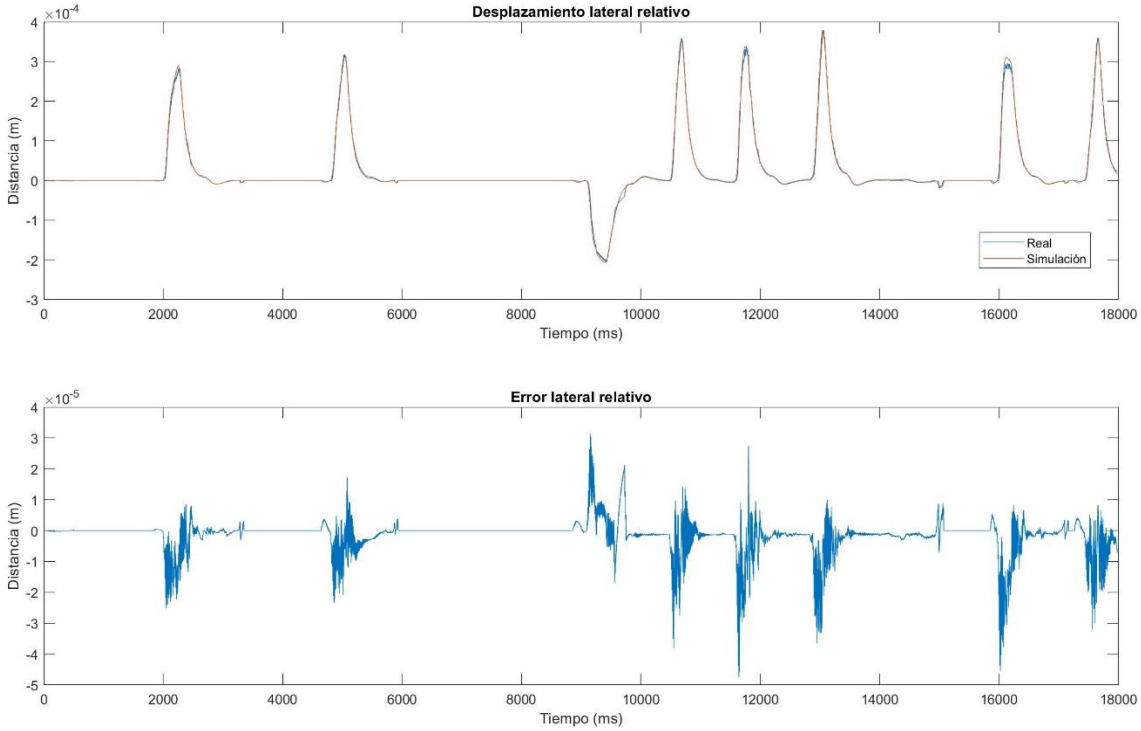

Gráfico 11: Desplazamiento y error lateral relativo Escenario 1

En este escenario 1, observando las distintas gráficas mostradas, se visualiza tanto un desplazamiento como un ángulo de giro relativo perfecto. No hay ningún problema aparente.

ESCENARIO 2

Se muestra el recorrido real realizado por el vehículo autónomo (posición inicial en punto naranja):

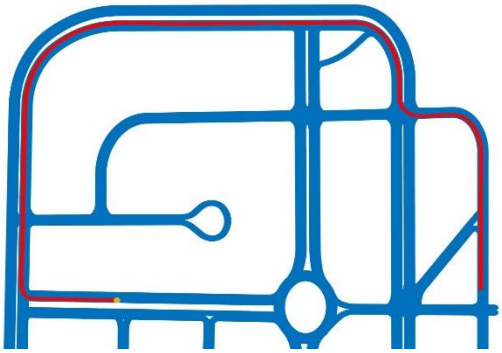

Ilustración 43: Recorrido real: Escenario 2

Se muestran las gráficas de ángulo de giro, desplazamiento longitudinal y lateral relativo entre fotogramas consecutivos y sus respectivos errores relativos del escenario 2:

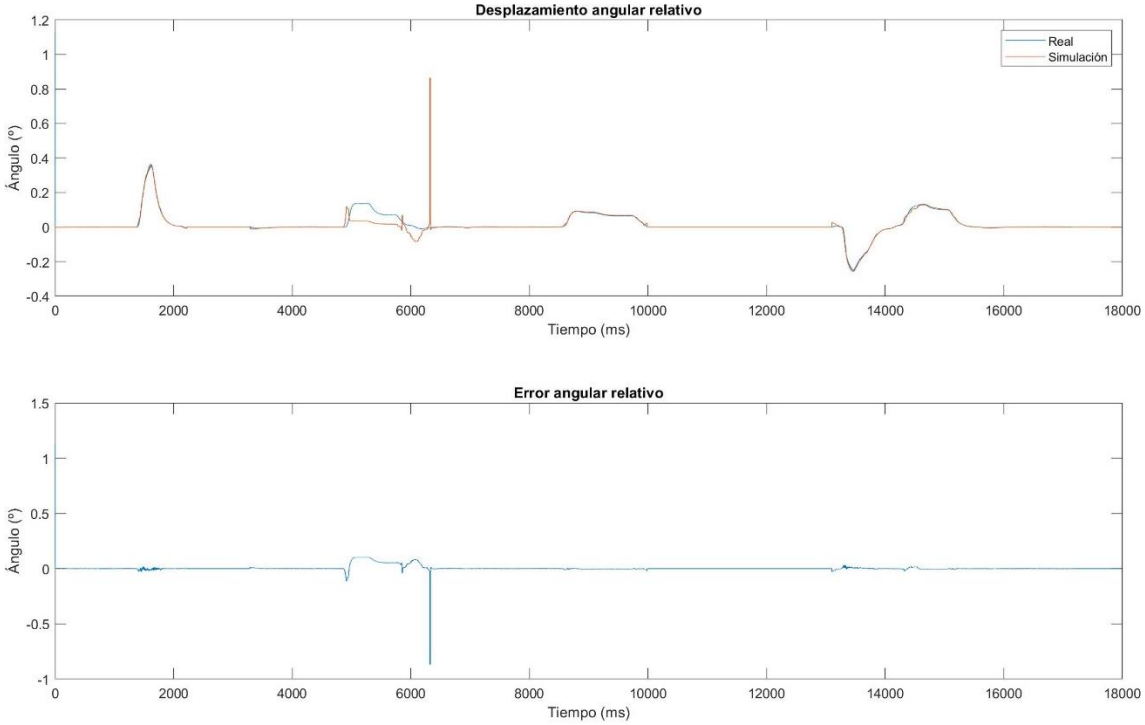

Gráfico 12: Desplazamiento y error angular relativo Escenario 2

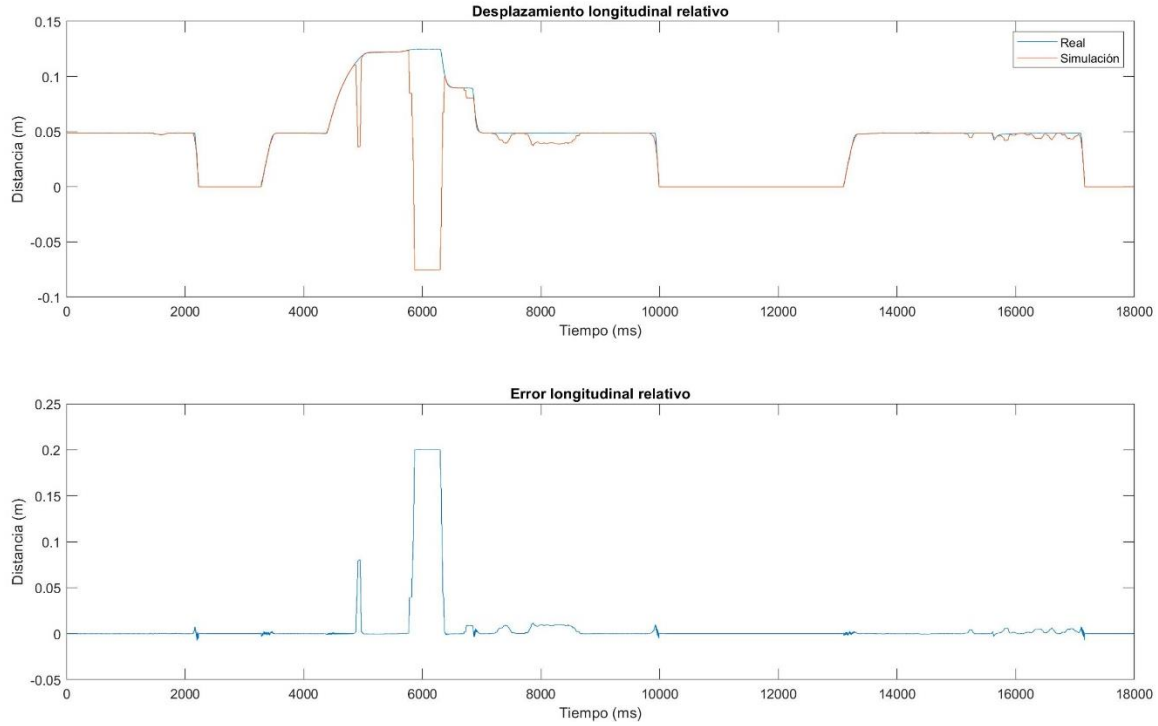

Gráfico 13: Desplazamiento y error longitudinal relativo Escenario 2

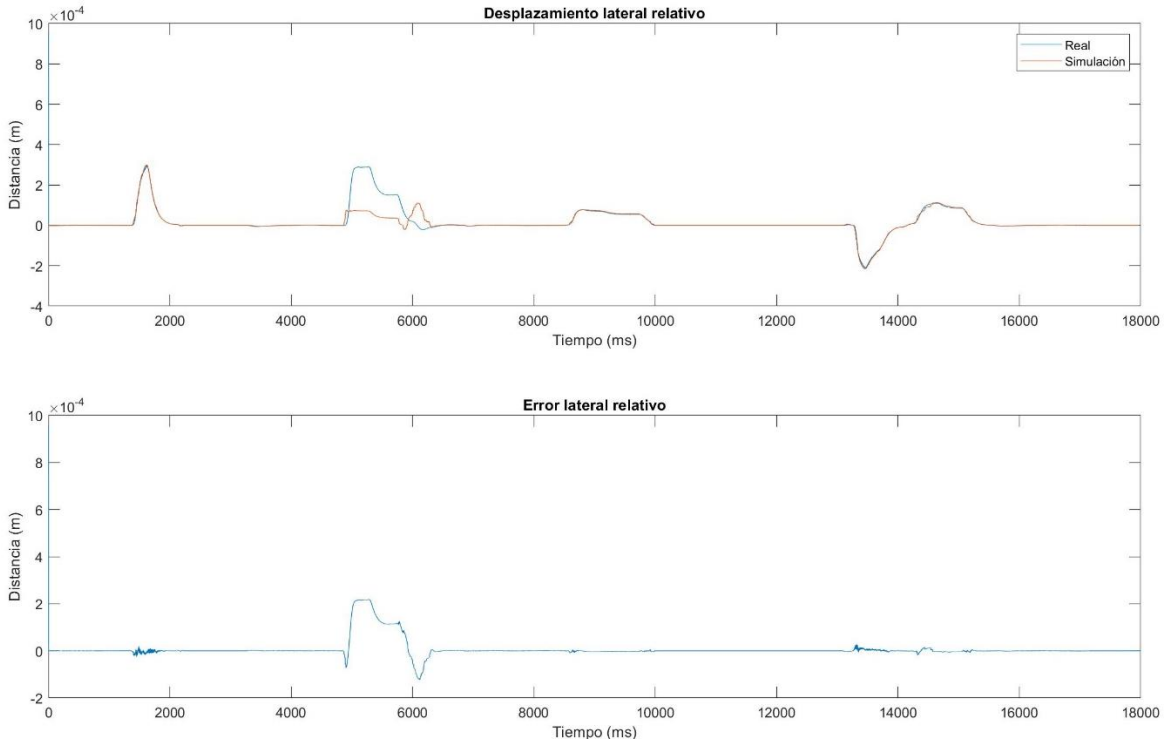

Gráfico 14: Desplazamiento y error lateral relativo Escenario 2

En este escenario 2, observando las distintas gráficas mostradas, se visualiza tanto un desplazamiento como un ángulo de giro relativo casi perfecto. Se registran dos errores, el primero de ellos ocurre en curva donde los tres parámetros no realizan el movimiento deseado, y el segundo, se puede observar en el desplazamiento longitudinal pequeños errores debido a sombras en la carretera.

ESCENARIO 3

Se muestra el recorrido real realizado por el vehículo autónomo (posición inicial en punto naranja):

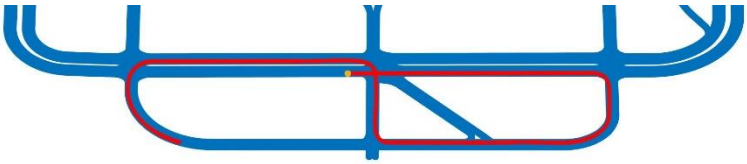

Ilustración 44: Recorrido real: Escenario 3

Se muestran las gráficas de ángulo de giro, desplazamiento longitudinal y lateral relativo entre fotogramas consecutivos y sus respectivos errores relativos del escenario 3:

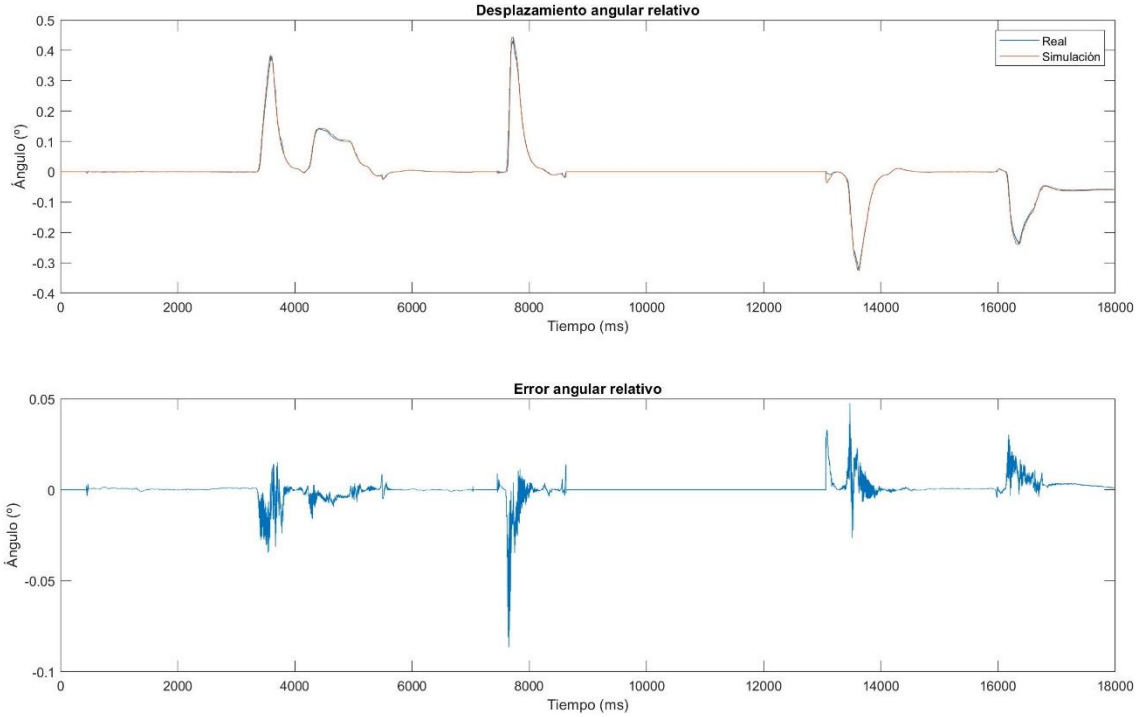

Gráfico 15: Desplazamiento y error angular relativo Escenario 3

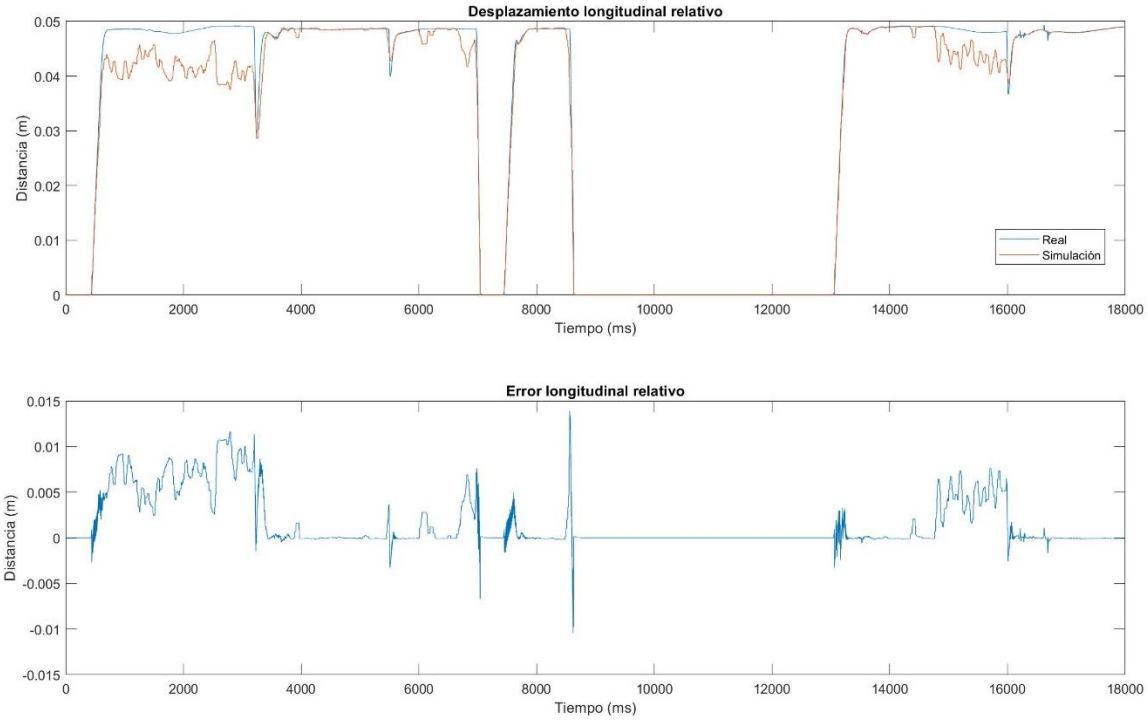

Gráfico 16: Desplazamiento y error longitudinal relativo Escenario 3

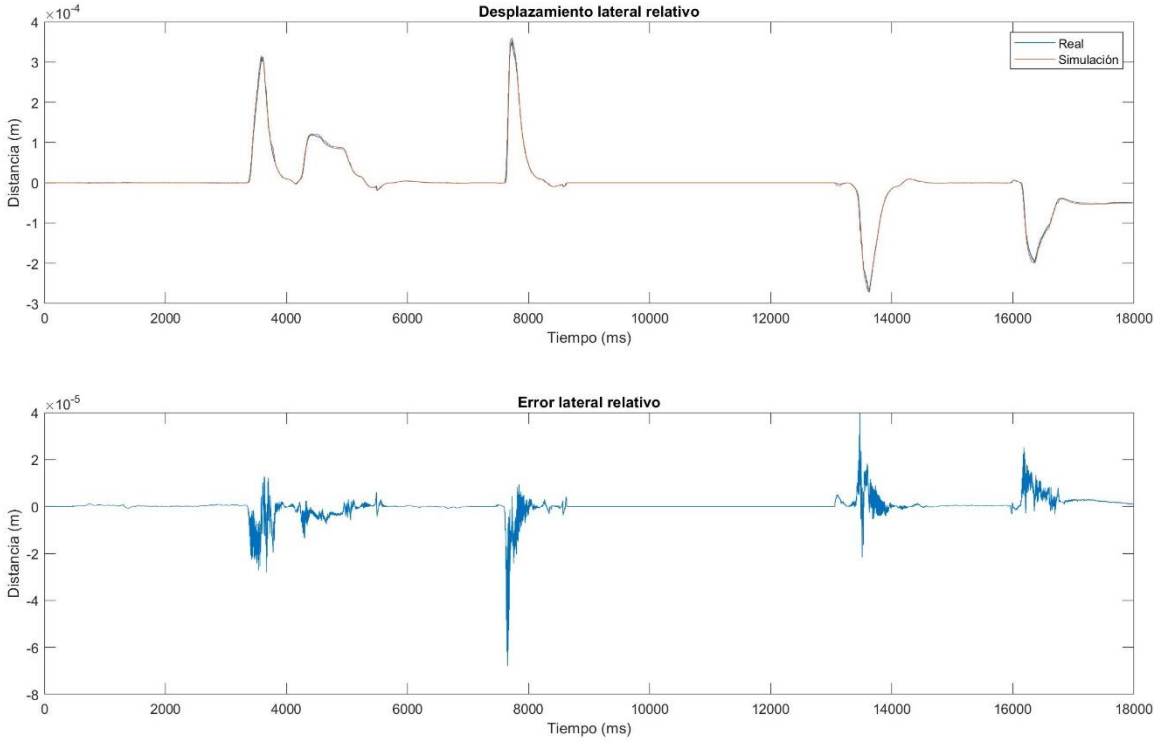

Gráfico 17: Desplazamiento y error lateral relativo Escenario 3

En este escenario 3, observando las distintas gráficas mostradas, se visualiza tanto un desplazamiento como un ángulo de giro relativo casi perfecto. Se registra un error en el desplazamiento longitudinal, donde debido a las sombras en la carretera las imágenes pierden calidad y se producen pequeños fallos.

Por otro lado, una vez visualizado y analizado detalladamente cada uno de los escenarios por separado, se procede a mostrar una tabla con los datos más destacables de cada uno de los escenarios.

Escenario	Distancia recorrida (m)	Error relativo					
		Ángulo (º)		Longitudinal (m)		Lateral (m)	
		Medio	Máximo	Medio	Máximo	Medio	Máximo
1	524.334	0.0016	0.0565	4.14e-5	8.32e-3	4.25e-6	4.75e-5
2	767.29	0.867	0.0036	0.0066	0.2	6.41e-6	2.15e-4
3	596.238	5.96e-5	0.086	0.0015	0.014	7.2e-9	6.8e-5

Tabla 1: Comparativa de resultados obtenidos en distintos escenarios

ASPECTOS ECONÓMICOS

11. DESCRIPCIÓN DEL PRESUPUESTO

A continuación, se muestra detallado el presupuesto del proyecto de investigación. El presupuesto se divide en dos conceptos principales: recursos humanos, salario del investigador, y recursos materiales, licencias de los entornos de trabajo y ordenadores utilizados.

Concepto	[ud]	Descripción	Ud. Total	Precio Partida (€)	Uso (h.)	Vida útil (h.)	Subtotal (€)	Total (€)
Recursos humanos								**15000**
Investigador	h.		300	50			15000	
Recursos materiales								**207,69**
MATLAB	Ud.	*Licencia anual MATLAB*	1	800	200	2080	76,92	
	Ud.	*Licencia anual "Computer Vision Toolbox"*	1	500	200	2080	48,08	
	Ud.	*Licencia anual "Image Processing Toolbox"*	1	400	200	2080	38,46	
	Ud.	*Licencia anual "Deep Learning Toolbox"*	1	460	200	2080	44,23	
CARLA	Ud.	*Licencia CARLA*	1	0	0	0	0,00	
Ordenador	Ud.		2	1000	300	5000	120,00	

TOTAL MATERIAL:	**15207,69**
Costes Indirectos (10%)	1520,77
Impuestos al Valor Añadido (21%)	3193,62
TOTAL:	**19922,08**

Tabla 2: Descripción del presupuesto

CONCLUSIONES

12. CONCLUSIONES

Concluyendo, se puede determinar que el algoritmo propuesto cumple con los objetivos de obtención del ángulo de giro y desplazamiento tanto longitudinal como lateral relativo entre dos fotogramas consecutivos obtenidos mediante el sensor de visión cámara posicionado en el vehículo autónomo. En modo resumen del proyecto, para la obtención de los datos finales, el proceso se inicia mediante la obtención de las imágenes correspondientes a través del simulador de vehículos autónomos CARLA. Donde, posteriormente, estas imágenes son analizadas para extraer características relevantes con el objetivo de lograr conocer la variación entre dos imágenes consecutivas. Y, finalmente, mediante un procesamiento de imágenes y datos llegar a obtener el resultado final del ángulo de giro y desplazamiento relativo.

Por otro lado, cabe recordar los objetivos marcados inicialmente y completados exitosamente:

- Estudio de los distintos proyectos realizados hasta la actualidad.
- Búsqueda e implementación del simulador apropiado de vehículos autónomos.
- Búsqueda e implementación de algoritmos de procesamiento de imágenes.
- Búsqueda e implementación de algoritmos de procesamiento para la eliminación de datos erróneos.
- Diseño del algoritmo final que permita conocer el desplazamiento relativo realizado por el vehículo autónomo entre dos fotogramas consecutivos.
- Aplicación del algoritmo en distintos escenarios para el análisis de datos en las diferentes situaciones.

Finalmente, para próximos trabajos con el sensor de visión, sería interesante analizar los objetos circundantes al vehículo autónomo mediante la cámara para obtener otra fuente de información con el objetivo de relocalización en el entorno. Por ello, este trabajo, es un pequeño paso para la obtención de un sistema de localización robusto basado en la fusión de distintos sensores en vehículos autónomos.

BIBLIOGRAFÍA

[1] "The Companies Developing Self-Driving Cars." https://www.techadvisor.com/feature/small-business/-companies-working-on-driverless-cars-3788696/

[2] "nuScenes dataset." https://www.nuscenes.org/

[3] H. Caesar *et al.*, "nuScenes: A Multimodal Dataset for Autonomous Driving." pp. 11621–11631, 2020.

[4] "Oxford RobotCar Dataset." https://robotcar-dataset.robots.ox.ac.uk/

[5] W. Maddern, G. Pascoe, C. Linegar, and P. Newman, "1 year, 1000 km: The Oxford RobotCar dataset," *International Journal of Robotics Research*, vol. 36, no. 1, pp. 3–15, Jan. 2017.

[6] D. Barnes, M. Gadd, P. Murcutt, P. Newman, and I. Posner, "The Oxford Radar RobotCar Dataset: A Radar Extension to the Oxford RobotCar Dataset," in *Proceedings - IEEE International Conference on Robotics and Automation*, May 2020, pp. 6433–6438.

[7] "The KITTI Vision Benchmark Suite." http://www.cvlibs.net/datasets/kitti/

[8] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237.

[9] "SVL Simulator by LG" https://www.svlsimulator.com/

[10] G. Rong *et al.*, "LGSVL Simulator: A High Fidelity Simulator for Autonomous Driving," Sep. 2020.

[11] "NVIDIA DRIVE Sim | NVIDIA Developer." https://developer.nvidia.com/drive/drive-sim

[12] "CARLA Simulator." https://carla.org/

[13] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An Open Urban Driving Simulator." PMLR, pp. 1–16, Oct. 18, 2017

[14] D. Nistér, O. Naroditsky, and J. Bergen, "Visual odometry," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2004, vol. 1.

[15] H. C. Longuet-higgins, "A computer algorithm for reconstructing a scene from two projections," *Nature*, vol. 293, no. 5828, pp. 133–135, 1981

[16] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision, Second Edition*. 2000. Accessed: Jun. 06, 2021.

[17] H. Li and R. Hartley, "Five-point motion estimation made easy," in *Proceedings - International Conference on Pattern Recognition*, 2006, vol. 1, pp. 630–633.

[18] M. Fischler and O. Firschein, "Readings in computer vision: issues, problems, principles, and paradigms," 1987.

[19] D. Valiente García, L. Fernández Rojo, A. Gil Aparicio, L. Payá Castelló, and O. Reinoso García, "Visual Odometry through Appearance- and Feature-Based Method with Omnidirectional Images," *Journal of Robotics*, vol. 2012, pp. 1–13, 2012.

[20] Y. Mizukami, T. Sato, and K. Tanaka, "A comparison study for displacement computation Horn and Shucnk's method versus March's method," *Pattern Recognition Letters*, vol. 22, no. 6–7, pp. 825–831, May 2001.

[21] N. Sharmin and R. Brad, "Optimal filter estimation for Lucas-Kanade optical flow," *Sensors (Switzerland)*, vol. 12, no. 9, pp. 12694–12709, Sep. 2012.

[22] M. M. El-Gayar, H. Soliman, and N. Meky, "A comparative study of image low level feature extraction algorithms," *Egyptian Informatics Journal*, vol. 14, no. 2, pp. 175–181, Jul. 2013.

[23] L. Ji-lin, "Harris Corner Detection-based Stereo Visual Odometry," 2007.

[24] L. C. Chiu, T. S. Chang, J. Y. Chen, and N. Y. C. Chang, "Fast SIFT design for real-time visual feature extraction," *IEEE Transactions on Image Processing*, vol. 22, no. 8, pp. 3158–3167, 2013.

[25] H. Bay, A. Ess, T. Tuytelaars, and L. van Gool, "Speeded-Up Robust Features (SURF)," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346–359, Jun. 2008.

[26] P. A. Mlsna and J. J. Rodríguez, "Gradient and Laplacian Edge Detection," in *The Essential Guide to Image Processing*, Elsevier Inc., 2009, pp. 495–524.

[27] J. Canny, "A Computational Approach to Edge Detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, no. 6, pp. 679–698, 1986

[28] D. Scaramuzza and R. Siegwart, "Appearance-guided monocular omnidirectional visual odometry for outdoor ground vehicles," *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 1015–1026, 2008.

[29] C. Forster, Z. Zhang, M. Gassner, M. Werlberger, and D. Scaramuzza, "SVO: Semidirect Visual Odometry for Monocular and Multicamera Systems," *IEEE Transactions on Robotics*, vol. 33, no. 2, pp. 249–265, Apr. 2017.

[30] J. A. Hesch, D. G. Kottas, S. L. Bowman, and S. I. Roumeliotis, "Consistency analysis and improvement of vision-aided inertial navigation," *IEEE Transactions on Robotics*, vol. 30, no. 1, pp. 158–176, 2014.

[31] D. Scaramuzza and F. Fraundorfer, "Tutorial: Visual odometry," *IEEE Robotics and Automation Magazine*, vol. 18, no. 4, pp. 80–92, Dec. 2011.

[32] Z. Zhang, Y. Han, Y. Zhou, and M. Dai, "A novel absolute localization estimation of a target with monocular vision," *Optik*, vol. 124, no. 12, pp. 1218–1223, Jun. 2013.

[33] U. S. V, "Obstacle avoidance and distance measurement for unmanned aerial vehicles using monocular vision," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 9, no. 5, pp. 3504–3511, 2019.

[34] L. Huang, Y. Chen, Z. Fan, and Z. Chen, "Measuring the absolute distance of a front vehicle from an in-car camera based on monocular vision and instance segmentation," *Journal of Electronic Imaging*, vol. 27, no. 04, p. 1, Jul. 2018.

[35] A. Zaarane, I. Slimani, W. al Okaishi, I. Atouf, and A. Hamdoun, "Distance measurement system for autonomous vehicles using stereo camera," *Array*, vol. 5, p. 100016, Mar. 2020.

[36] T. S. Hsu and T. C. Wang, "An improvement stereo vision images processing for object distance measurement," *International Journal of Automation and Smart Technology*, vol. 5, no. 2, pp. 85–90, 2015.

[37] N. Q. Ann *et al.*, "SKF-based image template matching for distance measurement by using stereo vision," in *Lecture Notes in Mechanical Engineering*, vol. 0, no. 9789811087875, Pleiades Publishing, 2018, pp. 439–447.

[38] J. Mrovlje and D. Vrani, "Distance measuring based on stereoscopic pictures," 2008.

[39] L. Piyathilaka and R. Munasinghe, "Multi-camera visual odometry for skid steered field robot," in *Proceedings of the 2010 5th International Conference on Information and Automation for Sustainability, ICIAfS 2010*, 2010, pp. 189–194.

[40] "CARLA Simulator." https://carla.readthedocs.io/en/0.9.9/

[41] "GitHub - carla-simulator/carla: Open-source simulator for autonomous driving research." https://github.com/carla-simulator/carla

[42] "GitHub - darkscyla/MATLAB-Carla-Interface: Interfacing Carla with MATLAB using Python and ROS." https://github.com/darkscyla/MATLAB-Carla-Interface

ANEXO I

13. ALGORITMO: "DataAcquisition_sync_CARLA"

```
%% CONFIGURACIÓN SERVIDOR (WORLD)

client_config

%% VEHÍCULO

% Traffic Manager (TM)
tm = client.get_trafficmanager();
tm_port = tm.get_port();
tm.set_synchronous_mode(true)

% Configuración vehículo
blueprint_library = world.get_blueprint_library();
car_list = py.list(blueprint_library.filter("model3"));
car_bp = car_list{1};
spawn_point = py.random.choice(world.get_map().get_spawn_points());
map = world.get_map();
tesla = world.spawn_actor(car_bp, spawn_point);
tesla.set_autopilot(true,tm_port);
waypoint = map.get_waypoint(spawn_point.location);

% Extracción de los parámetros físicos del vehículo
physics_control = tesla.get_physics_control;
wheel_front = cell(physics_control.wheels(1));
wheel_back = cell(physics_control.wheels(3));
pos_front = wheel_front{1,1}.position;
pos_back = wheel_back{1,1}.position;
L = sqrt((pos_front.x - pos_back.x)^2 + (pos_front.y - pos_back.y)^2);
L = L/100;
max_steer = wheel_front{1,1}.max_steer_angle;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% SENSORES

world.tick();

%% LIDAR
(.............)
%% GPS
(.............)
%% IMU
(.............)
```

```
%% RGB Lateral Derecho

blueprint = world.get_blueprint_library().find('sensor.camera.rgb');
blueprint.set_attribute('image_size_x', '1920');
blueprint.set_attribute('image_size_y', '1080');
blueprint.set_attribute('sensor_tick','0.1');
blueprint.set_attribute('enable_postprocess_effects','true');
transform =
py.carla.Transform(py.carla.Location(pyargs('x',0.8,'z',1,y',1)),
py.carla.Rotation(pyargs('yaw',90.0,'pitch',-90.0)));
rgbr = world.spawn_actor(blueprint, transform, pyargs('attach_to',tesla));
rgbModuler = sensorBind(rgbr, "fir", "rgb", "array");

%% RGB Lateral Izquierdo

blueprint = world.get_blueprint_library().find('sensor.camera.rgb');
blueprint.set_attribute('image_size_x', '1920');
blueprint.set_attribute('image_size_y', '1080');
blueprint.set_attribute('sensor_tick','0.1');
blueprint.set_attribute('enable_postprocess_effects','true');
transform =
py.carla.Transform(py.carla.Location(pyargs('x',0.8,'z',1,'y',-1)),
py.carla.Rotation(pyargs('yaw',-90.0,'pitch',-90.0)));
rgbl = world.spawn_actor(blueprint, transform, pyargs('attach_to',tesla));
rgbModulel = sensorBind(rgbl, "fil", "rgb", "array");

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% SIMULACIÓN

player = pcplayer([-25 25],[-25 25],[-10 10]);
c=0; % Extracción de imágenes cada 10 interacciones

while t <= (190 * fps)
    world.tick();
    pause(0.1)

    % Valores reales
     yaw_real(t) = tesla.get_transform.rotation.yaw;
     pitch_real(t) = tesla.get_transform.rotation.pitch;
     roll_real(t) = tesla.get_transform.rotation.roll;
     pos_GT(t,:) = [tesla.get_location.x,tesla.get_location.y,
     tesla.get_location.z];
     vel_real(t,:) = [tesla.get_velocity.x,tesla.get_velocity.y,
     tesla.get_velocity.z];
     acc_real(t,:) = [tesla.get_acceleration.x, tesla.get_acceleration.y,
     tesla.get_acceleration.z];
     steer(t) = tesla.get_control.steer * max_steer;
     rot = rotz(yaw_real(t));
     vel_loc(t,:) = rot*vel_real(t,:)';
     steer_yaw(t) = (abs(vel_loc(t,1))*tan(steer(t)*pi/180))/L;

    % LIDAR
     (.............)
    % GPS
     (.............)
    % IMU
     (.............)
```

```
    % RGB Lateral Derecho

      c = c + 1;
      if c == 10
          currentImage = uint8(py.getattr(rgbModuler, 'array'));
          imageName = sprintf('r_%d.png',t);
          imwrite(currentImage, imageName)
          figure(3)
          imshow(currentImage)
      end

    % RGB Lateral Izquierdo

      if c == 10
          currentImage = uint8(py.getattr(rgbModulel, 'array'));
          imageName = sprintf('l_%d.png',t);
          imwrite(currentImage, imageName)
          figure(4)
          imshow(currentImage)
          c = 0;
      end

     t = t+1
end

close all
time_delta = time - time(1);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% POSPROCESO

%% Almacenamiento
steer_yaw = cumtrapz(1/fps, steer_yaw*180/pi);

% Borrado de los primeros 10seg (incluye arranque)
IMU_acc = IMU_acc(1001:end,:);
IMU_gyro = IMU_gyro(1001:end,:);
XYZ_tot = XYZ_tot(101:end);

% Cambio del yaw a rango [-360, 360]
for i=1+1:length(yaw_real)
    if yaw_real(i) < 0 && yaw_real(i-1) > 175
        yaw_real(i) = 360 + yaw_real(i);
    elseif yaw_real(i) > 0 && yaw_real(i-1) < -175
        yaw_real(i) = yaw_real(i) - 360;
    end
end

yaw_real = yaw_real(1001:end);
pitch_real = pitch_real(1001:end);
roll_real = roll_real(1001:end);
pos_GT = pos_GT(1001:end,:);
vel_real = vel_real(1001:end,:);
acc_real = acc_real(1001:end,:);
steer = steer(1001:end);
steer_yaw = steer_yaw(1001:end);
```

```
pos_GPS = pos_GPS(1001:end,:);
pos_GPS_enu = pos_GPS_enu(1001:end,:);
eGPS_1 = eGPS_1(1001:end);
eGPS_2 = eGPS_2(1001:end);
eGPS_abs = eGPS_abs(1001:end);

save('acc','IMU_acc')
save('gyro','IMU_gyro')
save('lidar','XYZ_tot','-v7.3')
save('real','yaw_real','pitch_real','roll_real','pos_GT','vel_real',
'acc_real','steer','steer_yaw','fps','L')
save('GPS','pos_GPS','pos_GPS_enu','eGPS_1','eGPS_2','eGPS_abs')

%% Comprobaciones

figure(5)
plot(pos_GPS(:,1), pos_GPS(:,2))
hold on
plot(pos_GT(:,1), pos_GT(:,2))
title('Posicion')
legend('GPS','GT')

%% DESTRUCCIÓN DE SENSORES

system('python destroy.py sensor.*')
clear lidar
clear imu
clear gnss
clear rgb
clear semantic
system('python destroy.py vehicle.*')
clear tesla
```

14. ALGORITMO: "Main_images_CARLA"

```
%% Datos iniciales (DATOS A MODIFICAR)

% Introducir directorio carpeta imágenes:
imagePath = 'C:\Users\Asier\Desktop\Codigo_(Carla-MATLAB)\3.7\
Sensores_Matlab\Sync\Conexion_CARLA\pXX\';

% Elección de carpetas:
imPath_right = 'pXX_right'; % Carpeta imágenes cámara lateral derecha
imPath_left = 'pXX_left'; % Carpeta imágenes cámara lateral izquierda

% Rango de cálculo de imágenes (3 minutos)
ini = 1; % Valor inicial de cálculo
fin = 1899; % Valor final de cálculo

%% La información de la cámara es extraída de la base de datos
set0_right = strcat(imagePath,imPath_right);
set0_right = set0_right(find(~isspace(set0_right)));
set0_right = imageDatastore(fullfile(set0_right));

set0_left = strcat(imagePath,imPath_left);
set0_left = set0_left(find(~isspace(set0_left)));
set0_left = imageDatastore(fullfile(set0_left));

%% Ordenamos las imágenes
[set0_right] = imageOrder(set0_right,imagePath,imPath_right,'r_');
[set0_left] = imageOrder(set0_left,imagePath,imPath_left,'l_');

%% Parámetros de calibración

% K = Matriz de calibración
K = [9.6e+02 0.0e+00 9.6e+02;
      0.0e+00 5.4e+02 5.4e+02;
      0.0e+00 0.0e+00 1.0e+00]';

%% Obtención de datos de las imágenes

% Cámara derecha
p = ini;
c = ini*10;
while p < fin
   [loc_right(c,1:2,1),angle_right(c,1),range_right(c,1),result_right(c,1)]
    = imageCalculation(set0_right,p,K,'cedit');
    p = p + 1;
    c = c + range_right(c,1);
    cr = c
end
```

```
% Cámara izquierda
p = ini;
c = ini*10;
while p < fin
    [loc_left(c,1:2,1),angle_left(c,1),range_left(c,1),result_left(c,1)]
    = imageCalculation(set0_left,p,K,'cedit');
    p = p + 1;
    c = c + range_left(c,1);
    cl = c
end

%% Transferencia de datos
[angle_l(:,1),loc_l(:,1:2,1)] =
dataTransfer(angle_left,loc_left,range_left,ini,size(range_left),
result_left);
[angle_r(:,1),loc_r(:,1:2,1)] =
dataTransfer(angle_right,loc_right,range_right,ini,size(range_right),
result_right);

%% Almacenamiento
save(append("angle_",imPath_right),'angle_r');
save(append("location_",imPath_right),'loc_r');
save(append("angle_",imPath_left),'angle_l');
save(append("location_",imPath_left),'loc_l');
```

15. ALGORITMO: "imageOrder"

```
function [set0] = imageOrder (set0,imagePath,imPath,cam)

set0 = set0.Files;
set0 = erase(set0,imagePath);
set0 = erase(set0,imPath);
set0 = erase(set0,"\");
set0 = erase(set0,cam);
set0 = erase(set0,".png");
set0 = str2double(set0);
set0 = sort(set0);
set0 = string(set0);
set0 = append(imagePath,imPath,"\",cam,set0,".png");
set0 = imageDatastore(fullfile(set0));

return;
end
```

16. ALGORITMO: "imageCalculation"

```
function [location,angle,range,result] = imageCalculation(set,p,K,img)

Pict1 = readimage(set,p);
Pict2 = readimage(set,p+1);

try % Si ocurre un error, se visualiza y continua con el siguiente
fotograma
    result = 2; % Inicialmente se supone error
    % Calculo de ángulo y desplazamiento
    [angle,location(1:2),result] = positionEstimator(Pict1,Pict2,K,img);
    catch ME
    fprintf('error in one frame: %s\n', ME.message);
end
range = 10;
if result == 2 % Se supone vehículo detenido
    angle = 0;
    location = [0 0];
    return;
end
return;
end
```

17. ALGORITMO: "positionEstimator"

```
function [angle,location,result]=positionEstimator(Pict1,Pict2,K,img)

format shortg

Pict1 = editImage(Pict1,img); % Mejora de contraste y cambio a grises
Pict2 = editImage(Pict2,img); % Mejora de contraste y cambio a grises

% Visualización de las imágenes
subplot(1,2,1), imshow(Pict1);
subplot(1,2,2), imshow(Pict2);

% Creación del objeto de parámetros de calibración (Matriz intrínseca)
cameraParams = cameraParameters('intrinsicMatrix',K);

% Detección de puntos característicos mediante el algoritmo SURF
Corners1 = detectSURFFeatures(Pict1,'MetricThreshold', 500,
'ROI',[1 1 1920 560]);
numPoints = 1000;
Points1 = selectUniform(Corners1, numPoints, size(Pict1));

% Extracción características
[Features1,validPoints1] = extractFeatures(Pict1, Points1,'Upright', true);

% Extracción y emparejamiento con la imagen posterior
[Points2, Features2, indexPairs, validPoints2] =
helperDetectAndMatchFeatures(Features1, Pict2);

% Obtención de la matriz esencial / ángulo / desplazamiento
[angle,location,result] = estimateRelativePosition_Essential(...
    Points1(indexPairs(:,1)), Points2(indexPairs(:,2)), cameraParams);

return;
end
```

18. ALGORITMO: "helperDetectAndMatchFeatures"

```
function [currPoints, currFeatures, indexPairs, validPoints2] =
helperDetectAndMatchFeatures(prevFeatures,I)

% Detección de puntos característicos mediante el algoritmo SURF
currPoints   = detectSURFFeatures(I,'MetricThreshold', 500,
'ROI',[1 1 1920 560]);
numPoints = 1000;
currPoints   = selectUniform(currPoints, numPoints, size(I));
[currFeatures,validPoints2] = extractFeatures(I, currPoints,'Upright',
true);

% Emparejamiento de características entre imágenes
indexPairs = matchFeatures(prevFeatures, currFeatures, 'Unique', true);

end
```

19. ALGORITMO: "estimateRelativePosition_Essential"

```
function [ang,loc,result] = ...
    estimateRelativePosition_Essential(matchedPoints1, matchedPoints2,
cameraParams)

if ~isnumeric(matchedPoints1)
    matchedPoints1 = matchedPoints1.Location;
end

if ~isnumeric(matchedPoints2)
    matchedPoints2 = matchedPoints2.Location;
end

% Se simula hasta 100 veces para estimar la matriz esencial
for i = 1:100

    % Estimación de la matriz esencial
    [E, inlierIdx] = estimateEssentialMatrix(matchedPoints1,
    matchedPoints2,cameraParams,'MaxNumTrials',1000);

    % Nos aseguramos de tener suficientes puntos válidos, sino recalcular
    if sum(inlierIdx) / numel(inlierIdx) < .3
        continue;
    end

    % Extracción de los puntos validos
    inlierPoints1 = matchedPoints1(inlierIdx, :);
    inlierPoints2 = matchedPoints2(inlierIdx, :);

    % Calculo de ángulo y desplazamiento
    loc = inlierPoints2 - inlierPoints1;
    loc = mean(loc);
    loc(1,1) = loc(1,1)*0.0010395;
    loc(1,2) = loc(1,2)*2.5941e-05;
    ang = atand(loc(1,2)/loc(1,1));

    % Obtención de la posición relativa de la cámara
    [orientation, location, validPointFraction] = ...
        relativeCameraPose(E, cameraParams, inlierPoints1(1:2:end, :),...
        inlierPoints2(1:2:end, :));

    if validPointFraction > .95
        result = 1;
        return;
    end
end

result = 2;
return;

% Después de 100 intentos el validPointFraction demasiado pequeño
error('Unable to compute the Essential matrix');
end
```

20. ALGORITMO: "dataTransfer"

```
function [ang,loc]= dataTransfer(ang,loc,range,ini,tam,result)

% Se divide por las interacciones
for j = ini:tam
    if result(j,1) == 1
        ang(j,1) = ang(j,1)/range(j,1);
        loc(j,1) = loc(j,1)/range(j,1);
        loc(j,2) = loc(j,2)/range(j,1);
    end
end

% Valores secundarios
for j = ini:tam
    if result(j,1) == 1
        for z = j-range(j,1)+1:j
            ang(z,1) = ang(j,1);
            loc(z,1) = loc(j,1);
            loc(z,2) = loc(j,2);
        end
    end
    if result(j,1) == 2
        for z = j-range(j,1)+1:j
            ang(z,1) = ang(j-range(j,1),1);
            loc(z,1) = loc(j-range(j,1),1);
            loc(z,2) = loc(j-range(j,1),2);
        end
    end
end
return;
end
```

21. ALGORITMO: "dataProcess"

```
%% Escenarios
inicio = 1001; % 1001 mínimo valor
final = 18980;  % 18980 máximo valor (3 minutos)

%% Carga DATOS REALES
imPath = 'pXX'; % Dato a modificar

load(append('real_',imPath,'.mat'))
load(append('location_',imPath,'_right.mat'))
load(append('angle_',imPath,'_right.mat'))
load(append('location_',imPath,'_left.mat'))
load(append('angle_',imPath,'_left.mat'))

% Datos reales: Coordenadas globales a coordenadas relativas
for j = inicio-1000:final-1000
   real_GT(j,1) = (pos_GT(j+1,1)-pos_GT(j,1));
   real_GT(j,2) = (pos_GT(j+1,2)-pos_GT(j,2));
end
for j = inicio:final
    angle_real(j,1) = yaw_real(1,j-1000);
    angle_diff(j,1) = angle_real(j,1) - angle_real(j-1,1);
end
xym(inicio,1) = pos_GT(1,1);
xym(inicio,2) = pos_GT(1,2);
for j = inicio:final-1
    xy_m1(j,1) = (abs(real_GT(j-1000+1,1)+real_GT(j-1000+1,2)*i))
    *cosd(angle_diff(j,1));
    xy_m1(j,2) = (abs(real_GT(j-1000+1,1)+real_GT(j-1000+1,2)*i))
    *sind(angle_diff(j,1));
    xym(j+1,1) = xym(j,1)+((abs(real_GT(j-1000+1,1)+real_GT(j-1000+1,2)*i))
    *cosd(angle_diff(j,1)));
    xym(j+1,2) = xym(j,2)+((abs(real_GT(j-1000+1,1)+real_GT(j-1000+1,2)*i))
    *sind(angle_diff(j,1)));
end

%% Filtrado de Parámetros
l=5;
s=3;
por = 5;

[loc_lx] = Filter_XY(loc_l,inicio,final,l,s,por,1);
[loc_rx] = Filter_XY(loc_r,inicio,final,l,s,por,1);
[loc_ly] = Filter_XY(loc_l,inicio,final,l,s,por,2);
[loc_ry] = Filter_XY(loc_r,inicio,final,l,s,por,2);

% Eliminacion de errores detenido
for z = inicio:final-1
    if xy_m1(z,1) > 0.001
        loc(z,1) = (-loc_lx(z,1)+loc_rx(z,1))/2;
        loc(z,2) = (-loc_ly(z,2)+loc_ry(z,2))/2;
        ang(z,1) = atand(loc(z,2)/loc(z,1));
```

```
    else
        loc(z,1) = 0;
        loc(z,2) = 0;
        ang(z,1) = 0;
    end
end

% Cálculo del error relativo
for z = inicio:final-1
    eloc (z,1) = xy_m1(z,1)-loc(z,1);
    eloc (z,2) = xy_m1(z,2)-loc(z,2);
    eang (z,1) = angle_diff(z,1) - ang(z,1);
end

%% Visualización
figure
subplot(2,1,1)
plot(xy_m1(inicio:final-1,1))
hold on
plot(loc(inicio:final-1,1))
legend('Real','Simulación')
ylabel('Distancia (m)')
xlabel('Tiempo (ms)')
title('Desplazamiento longitudinal relativo')
subplot(2,1,2)
plot(eloc(inicio:final-1,1))
ylabel('Distancia (m)')
xlabel('Tiempo (ms)')
title('Error longitudinal relativo')

figure
subplot(2,1,1)
plot(xy_m1(inicio+1:final-1,2))
hold on
plot(loc(inicio+1:final-1,2))
legend('Real','Simulación')
ylabel('Distancia (m)')
xlabel('Tiempo (ms)')
title('Desplazamiento lateral relativo')
subplot(2,1,2)
plot(eloc(inicio+1:final-1,2))
ylabel('Distancia (m)')
xlabel('Tiempo (ms)')
title('Error lateral relativo')

figure
subplot(2,1,1)
plot(angle_diff(inicio:final-1,1))
hold on
plot(ang(inicio:final-1,1))
legend('Real','Simulación')
ylabel('Ángulo (º)')
xlabel('Tiempo (ms)')
title('Desplazamiento angular relativo')
subplot(2,1,2)
plot(eang(inicio:final-1,1))
ylabel('Ángulo (º)')
xlabel('Tiempo (ms)')
```

```
title('Error angular relativo')

load('pm.mat')
figure
scatter(mapa(:,2),mapa(:,1))
hold on
plot(pos_GT(:,2),pos_GT(:,1),'r','LineWidth',4)
hold on
scatter(pos_GT(1,2),pos_GT(1,1),'filled')

%% Guardado
save(append("loc_",imPath),'loc');
save(append("ang_",imPath),'ang');
```

22. ALGORITMO: “Filter_XY”

```
function [location_2] = Filter_XY(location,inicio,final,l,s,por,j)

p = 1;

for z = inicio:final
    if location(z,j) ~= location(z-1,j)
        location1(p,1) = (location(z,j));
        pos(p,1) = z;
        p = p + 1;

    end
end

location1 = hampel(location1,l,s);
fil1 = movmean(location1,por);
fil2 = movmedian(fil1,por);

for z = 1:p-1
    location_2(pos(z,1),j) = fil2(z,1);
end

if location1(1,1) == 0
    for z = inicio:final
        location_2(z,j) = 0;
    end
else

    for z = inicio:size(location_2)
        if location_2(z,j) ~= 0
            location_2(z,j) = location_2(z,j);
        else
            location_2(z,j) = location_2(z-1,j);
        end
    end
    for z = size(location_2):final
        location_2(z,j) = location_2(z-1,j);
    end
end
end
```