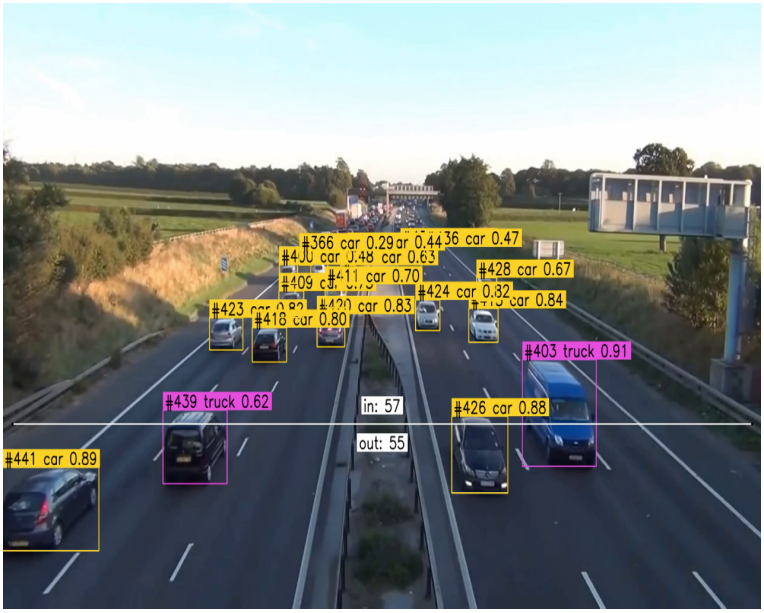
Universidad del País Vasco
Euskal Herriko Unibertsitatea
BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

GRADO EN INGENIERÍA INFORMÁTICA DE GESTIÓN Y SISTEMAS DE INFORMACIÓN

TRABAJO FIN DE GRADO

SISTEMA SLAM PARA UN VEHÍCULO FORMULA STUDENT



Estudiante: García Justel, Alan

Director/Directora: Zubizarreta Pico, Asier

Curso: 2023-2024

Fecha: 25 de junio de 2024

Resumen:
El equipo Formula Student Bizkaia, formado por alumnos de la Escuela de Ingeniería de Bilbao y que realiza un monoplaza de competición cada año, decidió no quedarse atrás en el sector de la automoción y dar el salto al vehículo autónomo. La complejidad de sustituir todas las funciones de un piloto se han redirigido a la implementación de algoritmos de control, toma de decisiones, percepción del entorno, modelización de sistemas y procesos de localización.

El proceso de localización y entendimiento del entorno es crucial para un vehículo que se desplaza sin la necesidad de intervención humana. Para lograr esto, es necesario el desarrollo y la coordinación de una serie de módulos que imiten la visión y procesamiento de un conductor. El presente trabajo tiene como objetivo el desarrollo del sistema de localización y mapeo simultáneos del vehículo; se establecerán los requerimietos del módulo y se presentarán los diseños software realizados.

El diseño comenzará con un estudio de los algoritmos y técnicas utilizadas en la industria para afrontar el problema de la localización en un entorno desconocido. Después, se realizará un primera implementación y se buscarán puntos de mejora para el sistema. Para finalizar, este proyecto pretende ir un paso más allá y no estancar el diseño en un plano teórico, por lo que se realizarán pruebas de validación que verifiquen el buen funcionamiento del sistema.

Palabras Clave: FSB, Sistema autónomo, SLAM, IA, Visual Tracking

Abstract:
The Formula Student Bizkaia team, comprised of students from the Bilbao School of Engineering and which produces a single-seater racing car every year, decided not to lag in the automotive sector taking the leap to the autonomous vehicle. The complexity of replacing all the functions of a driver has been redirected towards the implementation of control algorithms, decision-making, environment perception, system modeling, and localization processes.

The process of localization and understanding of the environment is crucial for a vehicle that moves without the need for human intervention. To achieve this, it is necessary to develop and coordinate a series of modules that mimic the vision and processing capabilities of a driver. This project aims to develop the vehicle's simultaneous localization and mapping system; the requirements for the module will be established, and the software designs will be presented.

The design will begin with a study of the algorithms and techniques used in the industry to address the problem of localization in an unknown environment. Subsequently, an initial implementation will be carried out, and improvement points for the system will be identified. Finally, this project aims to go a step further and not remain on a theoretical level, so validation tests will be conducted to verify the proper functioning of the system.

Key Words: FSB, Autonomous System, SLAM, AI, Visual Tracking

Laburpena:
Formula Student Bizkaia taldeak, Bilboko Ingeniaritza Eskolako ikasleek osatua, zeinek urtero lehiaketarako plaza bakarreko autoa egiten duen, automobilgintzaren sektorean atzean ez geratzea erabaki zuen ibilgailu autonomora salto egiteko pausoa emanez. Gidari baten funtzio guztiak ordezkatzeak konplexutasun handia dauka. Horrela, taldeak kontrol algoritmoak ezartzera, gidatze erabakiak hartzera, ingurunearen pertzepziora, sistemen modelizaziora eta lokalizazio-prozesuak garatzera bideratu ditu bere esfortzuak.

Autoaren eta bere inguruaren egoeraren berri izatea funtsezkoa da ibiligailu autonomo batentzat. Hori lortzeko, beharrezkoa da gidari baten ikusmena eta prozesamendua imitatuko duten modulu batzuk garatzea eta koordinatzea. Lan honen helburua aldi berean ibilgailua aurkitzeko eta mapa egiteko sistema garatzean datza; horretarako moduluaren eskakizunak ezarriko dira eta egindako software-diseinuak aurkeztu.

Diseinuaren hasieran, ingurune ezezagun batean kokatzearen arazoari aurre egiteko industrian erabiltzen diren algoritmoak eta teknikak aztertuko dira. Ondoren, lehen inplementazioa egingo da, eta sistemarako hobekuntza-puntuak bilatuko dira. Amaitzeko, proiektu honek urrats bat harago joan nahi du eta ez du diseinua plano teoriko batean gelditu nahi; beraz, sistemak ondo funtzionatzen duela egiaztatzeko balidazio-probak egingo dira.

Gako-hitzak: FSB, Sistema autonomoa, SLAM, IA, Visual Tracking

Índice

Índice de figuras

Índice de tablas

Abreviaturas

ADAS Advanced Driver Assistance Systems

ADS Automated Driving System

ADS-DV Autonomous Driving Systems – Dedicated Vehicle

API Application Programming Interface

AS Autonomous System

ASSI Autonomous System Status Indicator

DDT Dynamic Driving Task

EBS Emergency Brake System

EKF Extended Kalman Filter

EKF-SLAM Extended Kalman Filter SLAM

EOL End Of Life

FOV Field Of View

FS Formula Student

FSAE Formula Society of Automotive Engineers

FSB Formula Student Bizkaia

FSG Formula Student Germany

FSUK Formula Student UK

FS-AI Formula Student Artificial Intelligence

GNSS Global Navigation Satellite System

IMechE Institution of Mechanical Engineers

IMU Inertial Measurement Unit

KF Kalman Filter

LiDAR Light Detection and Ranging

MOT Multi Object Tracking

MRM Minimum Risk Maneuver

NNDA Nearest Neighbor Data Association

ODD Operational Design Domain

ODS Objetivos de Desarrollo Sostenible

OEDR Object and Event Detection and Response

RES Remote Emergency System

ROS Robot Operating System

RTK Real Time Kinematic

R2DA Ready to Drive Alone

SAE Society of Automotive Engineers

SDK Software Development Kit

SLAM Simultaneous Localization And Mapping

TFG Trabajo Fin de Grado

1. Introducción

Durante las últimas décadas, los vehículos autónomos han sido objeto de estudio debido a sus numerosas aplicaciones: desde la robótica industrial hasta los vehículos de pasajeros. Concretamente, en el sector automotriz, estos vehículos tienen que ser capaces de tomar decisiones y realizar tareas complejas a la vez que se garantiza la seguridad de las personas. Además, en el contexto de los vehículos estilo Formula Student, estos vehículos tienen que trabajar en unas condiciones muy exigentes en cuanto a velocidad y tiempo de respuesta.

La arquitectura típica para un vehículo autónomo divide la lógica del sistema en tres grandes grupos: percepción, planificación y control (Figura 1). De esta manera, el módulo de percepción es el encargado de recoger y procesar datos de los sensores y, por lo tanto, es una pieza fundamental para determinar el estado y situación del entorno.

Este Trabajo Fin de Grado (TFG) busca realizar el diseño, desarrollo e implementación de un sistema Simultaneous Localization And Mapping (SLAM) para un vehículo eléctrico automatizado de tipo Formula Student, con el que el vehículo será capaz de generar un mapa «online» del entorno trabajando con coordenadas globales al mismo tiempo que se estima la posición del vehículo en el mapa. Por ello, se tratarán diversos conceptos como el procesamiento de imágenes con inteligencia artificial; «tracking» de objetos en imágenes; fusión de sensores con métodos probabilísticos y algoritmos de localización y mapeo para resolver el problema de SLAM. Finalmente, se realizará una etapa de evaluación de los sistemas implementados tanto en simulaciones como haciendo uso de datos reales.

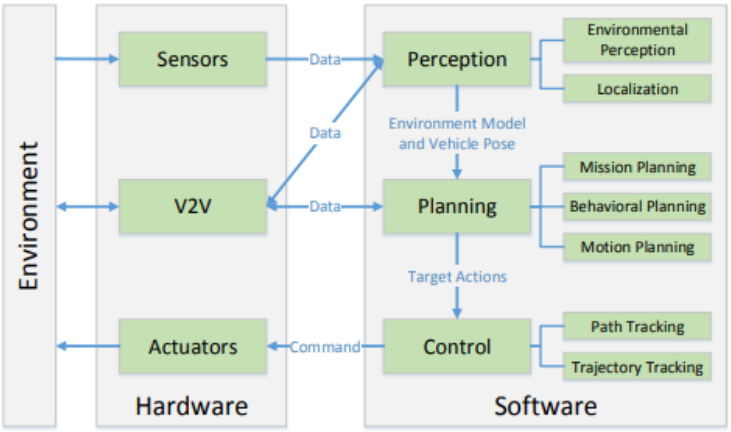


Figura 1: Módulos de un sistema autónomo típico

1.1. Repositorio

Todo el material de apoyo de este proyecto está disponible en el siguiente repositorio de GitHub: https://github.com/alanglk/tfg_perception_slam. El material incluye el código LaTeX del presente documento, el código fuente de los programas realizados o modificados y todos los datos generados en la evaluación de resultados.

2. Contexto

La automatización de vehículos en el sector automotriz no es una novedad. En 1988 se desarrolló el «NAVLAB 1» en la universidad de Carnegie Mellon, capaz de hacer un seguimiento de carril, [3] y en los años 2004 y 2005 se llevaron a cabo los «DARPA Grand Challenges» [4], competiciones cuyo objetivo era alcanzar la mayor distancia recorrida sin intervención humana. Estos acontecimientos catapultaron la investigación y el desarrollo en la robótica en general y en la conducción automatizada en particular.

Sin ir más lejos, desde los últimos 20 años es muy común contar con los denominados ADAS o sistemas avanzados de asistencia a la conducción, un conjunto de tecnologías que contribuyen a una conducción más sencilla y segura. Entre estos sistemas podemos encontrar desde el sistema de dirección asistida hasta sistemas más avanzados como el frenado de emergencia autónomo, que detiene el vehículo en caso de detectar una colisión inminente, o el seguimiento de carril automatizado [5]. Todo ello con el principal objetivo de minimizar el número de accidentes, el impacto ambiental y la congestión en las carreteras.

En este ámbito, en 2014 la Society of Automotive Engineers (SAE) definió 6 niveles de automatización que se han convertido en un referente de la conducción autónoma.

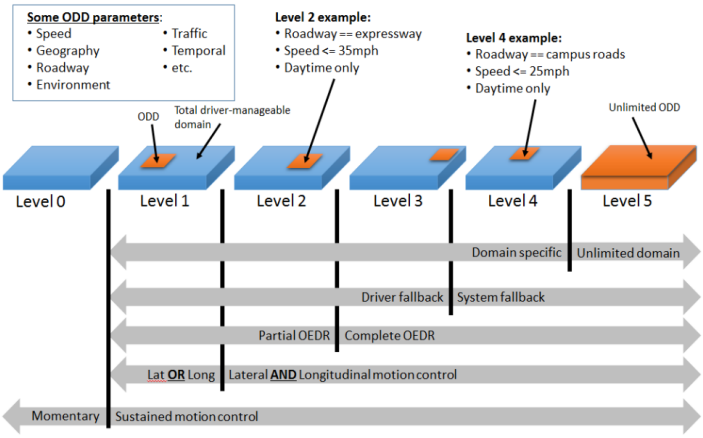


Figura 2: Niveles de automatización SAE

Actualmente, nos encontramos entre los niveles 3 y 4 ya que todavía existen numerosos impedimentos para conseguir una conducción automatizada completa: desde aspectos técnicos hasta éticos, sociales y económicos. En primer lugar, estas tecno-

logías hacen un gran uso de sensores, los cuales pueden fallar o aportar información imprecisa que comprometería el comportamiento del vehículo. Además, las situaciones externas como el estado de la carretera o el clima también afectan a los sistemas de percepción y de control de los vehículos. De esta manera, es muy importante definir el «dominio de diseño operativo» (ODD por sus siglas en inglés) en el que se recogen todos los estados, tanto internos como externos, para los que es seguro mantener el sistema activo y, en caso de que ya no se cumpla el ODD poder realizar las conocidas como Minimum Risk Maneuver (MRM) [6].

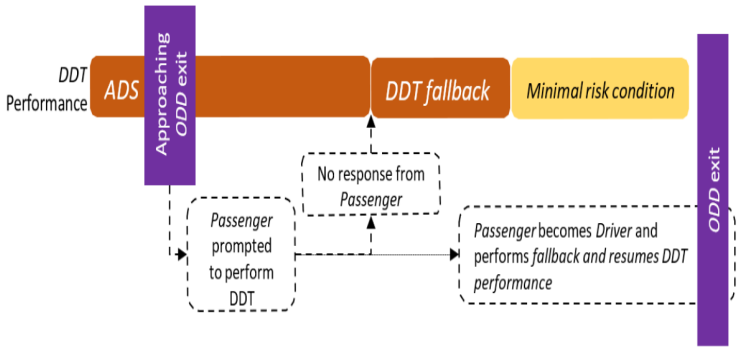


Figura 3: Diagrama de flujo MRM en un vehículo nivel 4 SAE

Así, podemos ver como todavía es necesario un gran esfuerzo de investigación y desarrollo en el sector de la conducción autónoma, sector que no pasa desapercibido para la competición Formula Student: la mayor competición de estudiantes de ingeniería entre diferentes universidades de todo el mundo.

2.1. Formula Student

Formula Student (FS) es una competición ampliamente conocida entre las diferentes universidades de ingeniería de todo el mundo. En ella, cada año equipos formados por alumnos de los distintos centros formativos diseñan, fabrican, montan y compiten con un monoplaza del tipo Formula 1 en eventos organizados en circuitos como Montmeló, Silverstone, Ricardo Palleti, o Hockenheimring.

La competición surgió en la década de 1980 cuando un profesor de la Universidad de Houston, Texas, contactó con la SAE para crear la que entonces se denominó la llamada «BAJA SAE». De esta iniciativa nació la Formula SAE, que se celebró por primera vez en 1981 y contó con la participación de cuarenta estudiantes divididos en seis equipos de la propia Universidad de Houston.

Actualmente la Formula SAE se ha expandido a lo largo de todo el mundo (Figura 4) dando lugar a más de 100 equipos únicamente en Europa, los cuales están formados por alrededor de 3000 estudiantes que toman parte en 22 competiciones distintas en todo el globo.

Los equipos que se presenten a cualquier competición de FS pueden elegir entre presentar un monoplaza de combustión interna, eléctrico, o driverless (autónomo), que puede ser tanto de combustión como eléctrico. Cada equipo puede presentar única y exclusivamente un monoplaza, y en caso de que se vayan a presentar dos coches de distinta modalidad tendrán que hacerlo bajo entidades de distinto nombre.

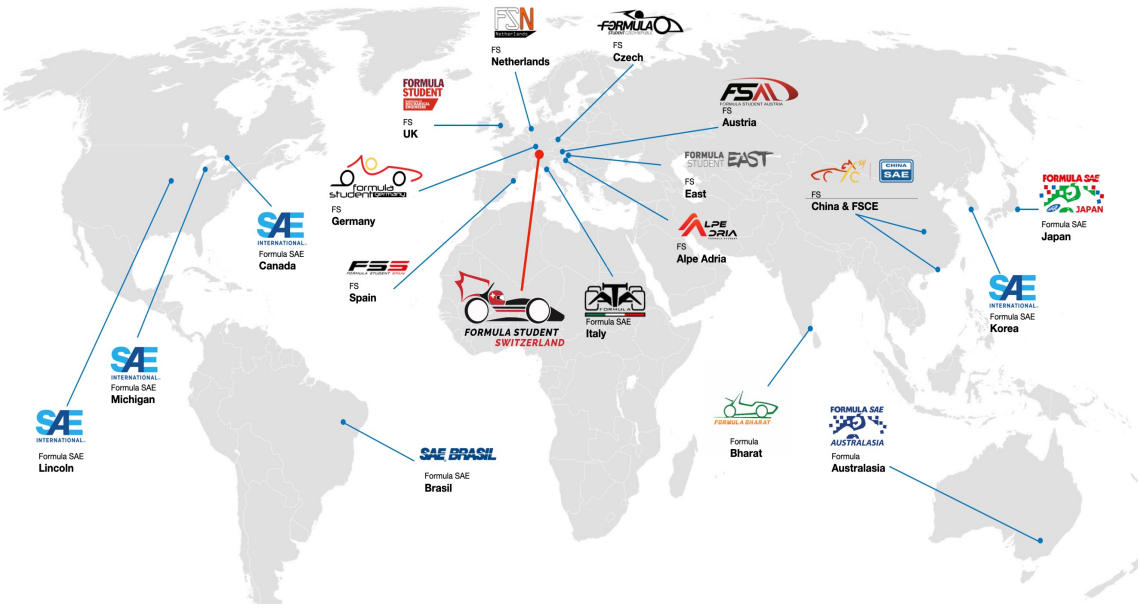


Figura 4: Principales competiciones de FS

2.2. Formula Student AI

Formula Student Artificial Intelligence (FS-AI) es una es una competición anual para estudiantes de todo el mundo que tiene lugar en el circuito de Silverstone, Inglaterra. Esta, tiene el objetivo de diseñar y construir vehículos de carreras autónomos. Esta competición comenzó en 2018 como una extensión de la tradicional Formula Student que se lleva a cabo desde 1981. FS-AI está diseñada para acelerar el desarrollo de habilidades en inteligencia artificial y tecnología de conducción autónoma, brindando a los estudiantes la oportunidad de aplicar sus conocimientos en un entorno práctico y desafiante. Esta competición, se divide en 2 categorías,

ADS y DDT, ambas compitiendo por la misma clasificación tanto en las pruebas estáticas como en las dinámicas:

- **Automated Driving System (ADS):** Los equipos compiten con vehículos autónomos diseñados y construidos completamente por ellos mismo. Su objetivo es adaptar vehículos FS existentes añadiendo capacidades de conducción automatizada o contruirlos desde cero.
- **Dynamic Driving Task (DDT):** Los equipos pueden comprar una plataforma automatizada, Autonomous Driving Systems – Dedicated Vehicle (ADS-DV), desarrollada exclusivamente para la competición FS-AI. El vehículo se entrega «listo para la autonomía» lo que permite a los equipos elegir sus propios algoritmos de control y sensores. Esta opción elimina la necesidad de diseñar y construir el vehículo, lo que permite a los estudiantes concentrarse en el desarrollo de software. También existe la posibilidad de compartir un prototipo ADS-DV propiedad de IMechE. El presente TFG se enfoca en esta última modalidad.

Figura 5: Plataforma ADS-DV de IMechE

Todas las competiciones de FS parten de la normativa de la Formula Society of Automotive Engineers (FSAE) que es la base para las normativas locales. Actualmente, las normas predominantes en Europa son FSUK y FSG, de las que existen apartados específicos para Driverless. Esta normativa [7] no solo representa un reto de diseño adicional para los equipos, sino que también pretende garantizar la máxima seguridad y fiabilidad en los monoplazas de competición. Tal y como se indica en la introducción de la normativa, esta competición tendrá un máximo de 1000 puntos y se dividirá en pruebas dinámicas y estáticas. Las primeras, se desarrollan con el coche en pista rodando, y las segundas, jueces pertenecientes a la industria de la automoción y equipos de la propia Formula 1 y Formula E tratan de poner a prueba la capacidad de diseño, marketing, fabricación y planificación

de los equipos participantes, además de valorar el conocimiento en el mundo de la inteligencia artificial y la conducción automatizada.

2.2.1. Pruebas estáticas

Las pruebas estáticas, en línea con el espíritu de la competición, analizan los aspectos de la ingeniería que van más allá del rendimiento del vehículo. Los jueces, expertos en el campo, son responsables de evaluar y los estudiantes deben interactuar con ellos para discutir las decisiones tomadas en el diseño, implementación y simulación de sus sistemas. Hay varias pruebas estáticas, cada una con su respectiva puntuación para la clasificación final.

Pruebas estáticas		
Business Plan Presentation	100 puntos	El equipo tendrá que desarrollar un plan de negocios sobre el monoplaza que ha diseñado, y ha de presentarlo a jueces expertos en el mundo del marketing de la automoción.
Real World AI	100 puntos	El equipo deberá demostrar que comprende los desafíos existentes para que un vehículo autónomo pueda operar en las carreteras públicas y privadas, y presentar un análisis creativo de cómo la industria actual y futura busca resolver estos problemas.
Engineering Design	150 puntos	El concepto del Design Event es evaluar el proceso de ingeniería del estudiante y el esfuerzo realizado en el diseño y desarrollo del sistema de IA y su integración en el vehículo, cumpliendo con la intención de la competición.
Simulation Development	100 puntos	El propósito de este evento es evaluar la comprensión de la simulación dentro del flujo de trabajo de desarrollo de vehículos autónomos, y demostrar el funcionamiento de los algoritmos sin hardware.

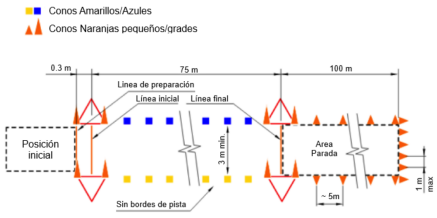
Tabla 1: Puntuaciones de las pruebas estáticas

2.2.2. Pruebas dinámicas

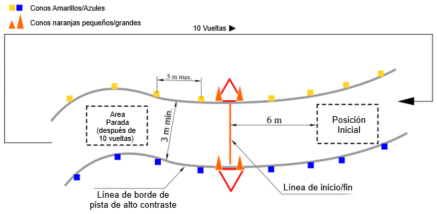
En cambio, las pruebas dinámicas son aquellas en las que se evalúa el Autonomous System (AS) que ha implementado cada equipo por su rendimiento, es decir, hay distintas pruebas donde el coche tiene que circular de forma automatizada con diferentes objetivos.

Pruebas dinámicas		
Acceleration	100 puntos	La prueba trata de medir la capacidad de aceleración del monoplaza en un tramo de 75 metros con una anchura de pista de 3 m (Figura 6a).
Skidpad	100 puntos	Consiste en poner a prueba la capacidad de giro del vehículo dando hasta cinco vueltas alrededor de un «8» (Figura 6c).
Autocross	100 puntos	Los equipos compiten en una pista cerrada con curvas y rectas de hasta 80 metros de longitud y 3 metros de ancho. Cada equipo tendrán dos intentos para terminar una vuelta completa al circuito y la pista no es conocida de antemano.
Trackdrive	150 puntos	Los equipos compiten en un circuito no conocido de 1km por 10 vueltas (Figura 6b). Los equipos pueden examinar la pista antes del evento sin utilizar equipo electrónico.

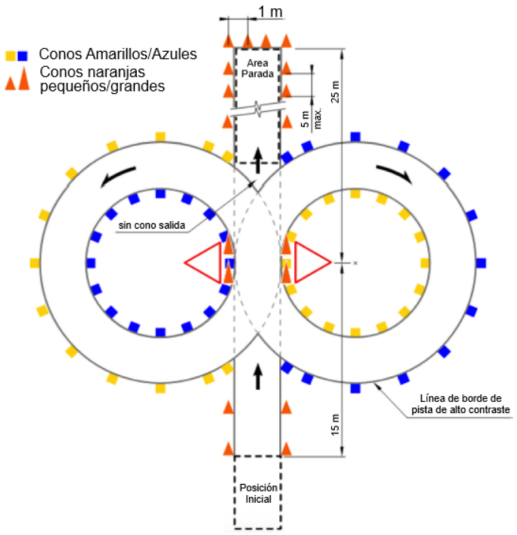
Tabla 2: Puntuaciones de las pruebas dinámicas



(a) Layout de la pista de Acceleration



(b) Normas para las pistas de Autocross y Trackdrive



(c) Layout de la pista de Skidpad

Figura 6: Diagramas de las pistas de las pruebas dinámicas

2.2.3. IMechE ADS-DV

El vehículo que ofrece la competición, mas concretamente el ADS-DV, es la plataforma de vehículos que ha desarrollado Institution of Mechanical Engineers (IMechE) para los equipos que deseen competir en la clase DDT. Este vehículo, ofrece a los equipos con presupuestos, mano de obra u otros recursos limitados la oportunidad de iniciarse en el desarrollo de sistemas autónomos y perfeccionar sus habilidades.

Al igual que la modalidad «Concept» de la competición FS, la clase DDT proporciona un punto de partida para que los equipos se involucren en la competición, progresando eventualmente hasta construir su propio vehículo desde cero en la clase ADS. Esto encaja con el espíritu de Formula Student como competición educativa que debe ser lo más accesible posible. Además, Formula Student UK es la única competición Formula Student del mundo que ofrece esta plataforma a los equipos de estudiantes.

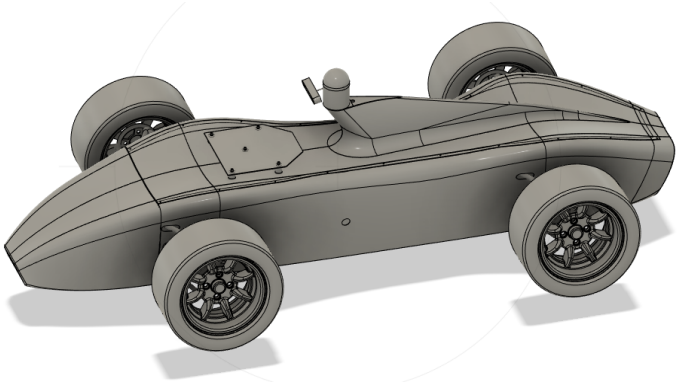


Figura 7: Render del ADS-DV

Este vehículo, está completamente equipado con todo lo necesario para que se pueda realizar una primera implementación de un AS. Para poder tener el control del pilotaje al igual que lo hacemos con un conductor, esta plataforma de conducción autónoma viene preparada con una unidad de procesamiento y diferentes sensores y actuadores para el sistema de frenos y de dirección. El control y lectura de los sensores y actuadores se realiza a través de un bus CAN[1].

Entre los sensores que incluye el vehículo se encuentran los «Wheel Encoders», un «GPS» e «IMU» y una cámara estereoscópica ZED. Sin embargo, también

[1]CAN es un protocolo de comunicación diferencial muy resistente a interferencias y que es muy utilizado en el mundo de la automoción

se ofrece la opción de que los equipos participantes instalen sensores adicionales propios en el vehículo siempre y cuando se cumpla con la norma impuesta (Figura 7).

Para garantizar la seguridad de los participantes en competición, en caso de un comportamiento inadecuado, descontrolado o peligroso, este vehículo tiene un sistema de seguridad llamado Emergency Brake System (EBS). Este sistema, tiene el objetivo de bloquear las ruedas si se detecta un fallo en la maquina de estados del coche, si hay algún fallo de alimentación en el vehículo o se pide su activación ya sea desde el propio AS o de forma externa empleando el Remote Emergency System (RES) (Figura 8).

Figura 8: Fotografía del control remoto del RES

Además de todo lo mencionado, la competición también ofrece facilidades a la hora de establecer la comunicación con el vehículo mediante una API [8] que permite leer los sensores y enviar comandas de actuación de una forma sencilla.

2.3. Formula Student Bizkaia

En el año 2006, un grupo de estudiantes de la Escuela de Ingeniería de Bilbao se unió con el objetivo de diseñar un monoplaza para competir en eventos deportivos. De este modo, se creó el proyecto de Formula Student Bilbao, que más tarde cambió su nombre a Formula Student Bizkaia (FSB). Desde su inicio, el equipo ha participado en más de 25 competiciones en España (FSS) y en Inglaterra (FSUK), obteniendo resultados destacados de manera constante. En los últimos años, el equipo ha logrado mantenerse en el top 10 en ambas competiciones, lo que actualmente lo sitúa como el mejor equipo español y en la posición 47 del ranking mundial.

El equipo está compuesto por 80 estudiantes que se dividen en diferentes grupos o departamentos especializados en diferentes áreas del proyecto. Por un lado,

los grupos *Management* y *Organization* tienen un papel más enfocado en la organización y gestión. Por otro lado, los grupos *Dynamics*, *Powertrain*, *Chassis*, *Aerodynamics*, *Simulation*, *Electronics* y *Driverless* son los encargados de la parte técnica del proyecto, siendo el grupo Driverless responsable de la implementación del sistema autónomo mencionado anteriormente.

Además de los diferentes departamentos, los miembros del equipo también se distinguen por los cargos que ocupan, según las responsabilidades que tienen. Cada grupo cuenta con un Manager que se encarga de todas las tareas de su departamento y coordina a los miembros. También hay varios Technical Managers, que son veteranos del equipo encargados de supervisar los aspectos técnicos y gestionar los diferentes departamentos. Todo ello es liderado por el Team Leader, el cual es el principal responsable del proyecto y el representante del equipo en las competiciones. En la Figura 9 se puede apreciar a los miembros que han formado parte de la temporada 2023-2024.

Figura 9: Fotografía de todos los miembros de FSB en 2024

Los primeros años del proyecto, se enfocaron únicamente en el desarrollo de monoplazas de combustión interna, hasta que, en el año 2012, se dio un paso hacia la movilidad eléctrica y la sostenibilidad, implementando desde entonces un monoplaza de forma anual cuyo sistema de tracción este impulsado únicamente por motores eléctricos.

Desde entonces, se vivieron saltos tecnológicos muy significativos, tales como el del FSB2016, cuando se paso de uno a dos motores; el del FSB2017, cuando se implemento el primer paquete aerodinámico y el del FSB2022 en el que se estrenó la tecnología AWD añadiendo un motor a cada rueda. A todas estas inovaciones hay que añadirle el salto que se ha dado durante la temporada pasada, implementando el primer sistema autónomo para Formula Student de la Universidad del País Vasco

y consiguiendo un primer puesto en la prueba «Engineering Design», suponiendo un gran impulso en la temporada actual.

Con la intención de empezar a competir en las emergentes competiciones de Formula Student Driverless, en el año 2021, el equipo FSB decidió adentrarse en un nuevo reto: la implementación del primer sistema autónomo, teniendo que abordar todos los retos de investigación previa, diseño, programación y testeo que requiere un proyecto de este calibre. Este proyecto, se inicio con un total de 5 personas que asentaron las bases del diseño en 2021 que, tras el desarrollo durante la temporada pasada, permitió participar por primera vez en la competición FS-AI en julio de 2023 como un grupo con identidad propia (Figura 10).

Actualmente, el grupo FSB Driverless se compone de 7 personas, todas ellas enfocadas en perfeccionar los sistemas y desarrollar soluciones para volver a competir en la modalidad DDT de la competición FS-AI en julio de 2024 y conseguir un buen desempeño en las pruebas dinámicas.

Figura 10: Logo de FSB Driverless

2.4. GidarIAv2

A fin de preparar la competición FS-AI 2024 y partiendo de un desarrollo anterior, el grupo ha implementado y optimizado los diferenes subsistemas que son necesarios para una operatividad autónoma: sistemas de percepción, aprendiendo a manejar sensores como los LiDAR y las cámaras estereoscópicas; algoritmos de planificación, para guiar al vehículo dentro de esa percepción, además de diferentes estrategias de control para la trayectoria y velocidad planificadas. Al sistema desarrollado, se le ha llamado «GidarIAv2», nombre decidido entre todos los integrantes del grupo. En la siguiente Figura 11 se pueden ver las funcionalidades programadas para este sistema autónomo.

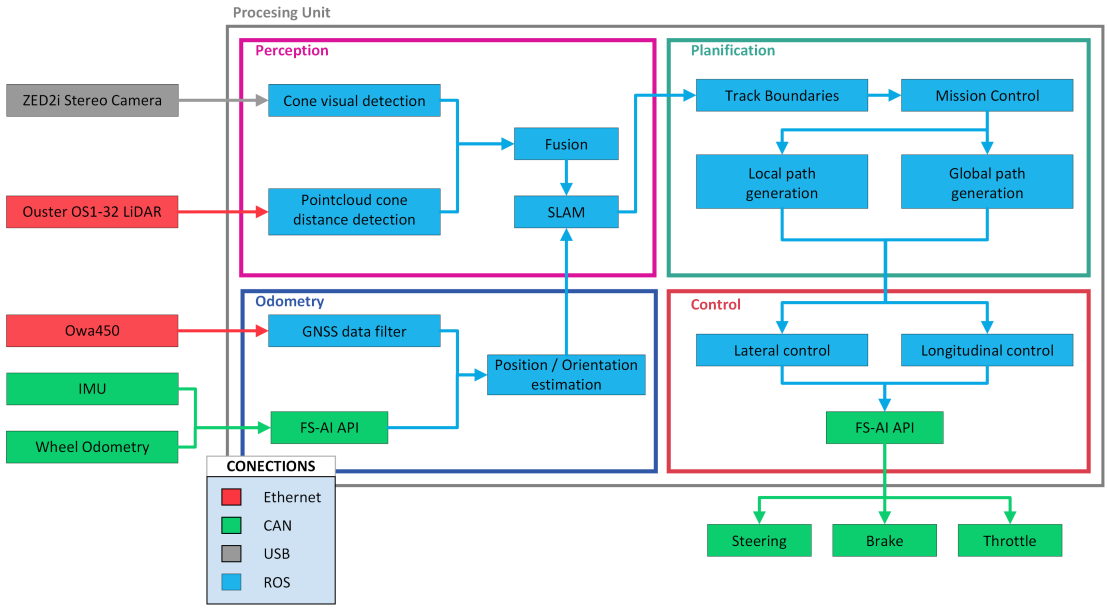


Figura 11: Estructura base del AS

El presente Trabajo Fin de Grado está enfocado en este grupo de trabajo y, en concreto, en el módulo de percepción, cuyo hardware y funcionalidades básicas se detallan a continuación.

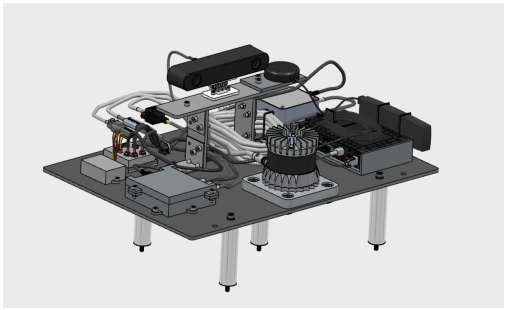
2.4.1. Placa de sensores

Como se ha mencionado previamente, el equipo está enfocado en participar de nuevo en la modalidad DDT de FS-AI. Es por ello, que el vehículo automatizado para el que hay que desarrollar los algoritmos del sistema autónomo es el ADS-DV, vehículo que ya cuenta con una serie de sensores como «Wheel Encoders», un «GPS» integrado, un IMU o una cámara estéreo «ZED1».

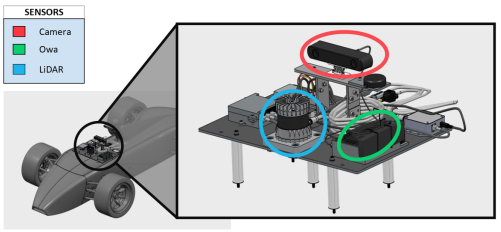
Sin embargo, la competición también ofrece la posibilidad de montar los sensores propios de cada equipo en el coche, permitiendo no estar limitados únicamente a los ya instalados y pudiendo desarrollar y validar los algoritmos de dichos sensores antes de la competición. Por esta razón, se ha desarrollado una placa que sirve como soporte para todos los sensores adicionales acoplándose en la parte frontal del vehículo.

Con todo ello se ha llegado al diseño mostrado en la Figura 12, que incorpora un Global Navigation Satellite System (GNSS), un LiDAR y una cámara *ZED2i*, que

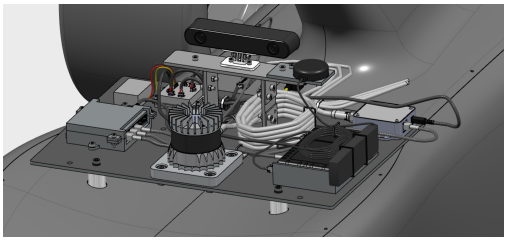
están interconectados mediante un switch Ethernet y alimentados mediante una placa diseñada para tal fin. A continuación se detallan las características generales de estos sensores.



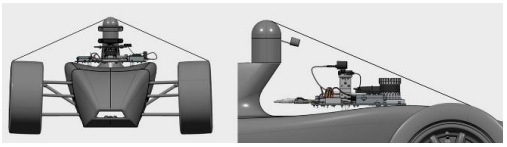
(a) Placa de sensores



(b) Visión global del coche



(c) Sensores montados en el ADS-DV



(d) Requerimientos de la placa

Figura 12: Placa de sensores adicionales

2.4.1.1 Owasys Owa450

La plataforma Owa450 de *Owasys* es un dispositivo embebido utilizado en maquinaria y vehículos industriales para llevar a cabo labores de monitorización. Sin embargo, en nuestro diseño del sistema autónomo es utilizada principalmente por su módulo GNSS, que nos permite localizar el vehículo con una gran precisión. Esto se lleva a cabo utilizando información proporcionada por varios satélites y corrigiéndola con tramas RTK obtenidas de estaciones base conocidas y cercanas al circuito. Además, pese a que existen equipos que construyen sus propias estaciones base para proporcionar las tramas RTK, nosotros, debido a la facilidad de uso y a la normativa, hemos decidido recurrir a servicios web para obtenerlas.

Figura 13: Owasys Owa450

Este dispositivo se comunica con el vehículo mediante una conexión de red Ethernet y es una de las entradas para el módulo de odometría detallado más adelante.

2.4.1.2 LiDAR Ouster OS1

El LiDAR es un dispositivo clave en la percepción dentro de un contexto de vehículo autónomo, ya que permite detectar con gran precisión la distancia a los obstáculos situados en los alrededores del coche, en este caso conos. Existen varios tipos de LiDAR que emplean diferentes técnicas para conseguir una nube de puntos 3D del entorno, entre los que se encuentran los LiDAR de estado sólido y los rotativos. Los primeros emplean una matriz de emisores y receptores fijos para realizar mediciones, lo que resulta en un diseño compacto y sin partes móviles. En contraposición, el segundo emite una serie de haces laser en un determinado FOV vertical de manera rotativa. En resumen, ambas son tecnologías de teledetección óptica que permite medir la distancia desde un punto sensor a diferentes posibles obstáculos utilizando el tiempo de vuelo (ToF) entre un punto de emisión y reflexión láser. Como se ha mencionado, el resultado es una nube de puntos tridimensional en la que se reflejan los distintos obstáculos del entorno.

Debido al apoyo de una empresa patrocinadora del proyecto, su uso por otros equipos de Formula Student, y la documentación existente, la opción escogida es la gama OS1 de Ouster. La gama OS1 corresponde con sensores rotativos de medio alcance y existen varias configuraciones en cuanto al número de capas y su disposición. Estas son las características del sensor utilizado:

Característica	Valor
Número de capas	32
Rango mínimo	0.3m
Rango eficaz	36m
Precisión	<5cm
FOV Vertical	29°(+13°a-16°)
Resolución angular vertical	(0.35°±2.5°)
Resolución horizontal	512, 1024 o 2048
FOV Horizontal	360°
Puntos por segundo	655.360
Frame rate	(5,10,20)Hz
Protección	IP68
Consumo	(14-20)W
Voltaje	24V
Peso	455g
IMU	Sí

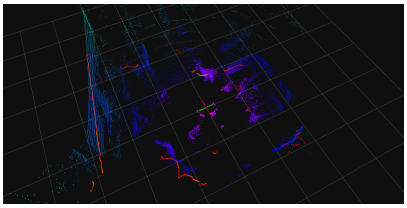


Figura 14: Especificaciones Ouster OS1 LiDAR

2.4.1.3 Cámara estereo ZED2i

La cámara es una pieza clave en nuestro diseño del sistema automático. Su misión es identificar los obtáculos que componen la pista así como el color de los conos y estimar una posición aproximada de los mismos que se corregirá con las posiciones más precisas aportadas por el LiDAR.

Los conos que delimitan el carril por el que se debe circular son pequeños, azules en el margen izquierdo y amarillos en el derecho. Las entradas y las salidas están marcadas por conos pequeños de color naranja y los conos naranjas grandes indican el comienzo y el final de las zonas de cronometraje. La cámara toma imágenes de lo que se encuentra por delante del coche y luego estos datos son enviados a la unidad de procesamiento correspondiente para procesarlos con una red neuronal e identificar los conos y sus características.

Después de hacer un estudio de las diferentes alternativas, el equipo se ha decantado por la cámara ZED2i, la cual destaca por tener su propio software de desarrollo (SDK). Además, es compatible de forma nativa con varios OS como Ubuntu, Windows, subsistemas como Jetson Nano (distro Linux), con ROS y con Python, programas con los que se quiere trabajar, por lo que esto facilitaría y evitaría muchos posibles problemas de incompatibilidad. Estas son sus características principales:

Característica	Valor
Depth range	20cm to 20m
Depth FPS	Up to 100Hz
Desrcipción	Estereoscópica con sensores de Profundidad
Sensores de movimiento	Acelerómetro y giroscopio (40Hz)
Software development	SDK compatible con Ubuntu 22.04, ROS2 y Python
Sistemas compatibles	Ubuntu 22,20,18/ Jetson L4T/ Windows 11,10,8
Profundidad FOV	120ž
Dimensiones	175.25x30.25x43.10mm
Peso	166g
Comunicación	USB 3.0 tipo-C
Protección	IP66
Power	USB 5v/380mA
SLAM	6-DoF visual-inertial stereo

Figura 15: Especificaciones cámara ZED2i

2.4.2. Módulo de percepción

Entre los diferentes subconjuntos del sistema automático, se encuentra el módulo de percepción, que es el objeto de este TFG. Está representado por la sección rosa en la Figura 11 y se encarga de recibir y procesar distintos tipos de datos con el objetivo de detectar los obstáculos y límites de la pista para que, posteriormente, el módulo de planificación sea capaz de definir las trayectorias adecuadas que debe seguir el vehículo.

El módulo de percepción recibe los datos que provienen de los sensores citados anteriormente 2.4.1, así como la odometría del vehículo. Esta odometría está compuesta por un filtro para el GNSS, la API de comunicación con los sensores del ADS-DV y un módulo de fusión de sensores (sección azul de la Figura 11). En definitiva, este módulo proporciona los datos de movimiento que sirven de entrada para el subconjunto SLAM de la percepción.

Por otro lado, como anteriormente se ha indicado, el módulo de percepción tiene la función de definir los elementos del entorno del vehículo, así como la posición del mismo en un sistema de referencia global. Estos datos serán utilizados para la definición de la trayectoria a seguir por el vehículo, de la que se encargará el módulo de planificación.

El módulo de percepción está dividido en 4 submódulos principales: la detección visual; la detección en nubes de puntos; la fusión entre cámara y LiDAR y el mapeo y localización (SLAM). Los dos primeros submódulos se encargan de identificar los conos que conforman los límites del circuito. Una vez se han identificado los obstáculos en el instante actual, se fusionan los datos para obtener conos clasificados y sus respectivas posiciones relativas al vehículo. Con estos datos unidos a la estimación de la posición proporcionada por el subconjunto de odometría, el módulo de SLAM se encarga de generar un mapa de la pista, situar el vehículo en dicho mapa y proporcionar información como el número de vueltas que se ha dado al circuito.

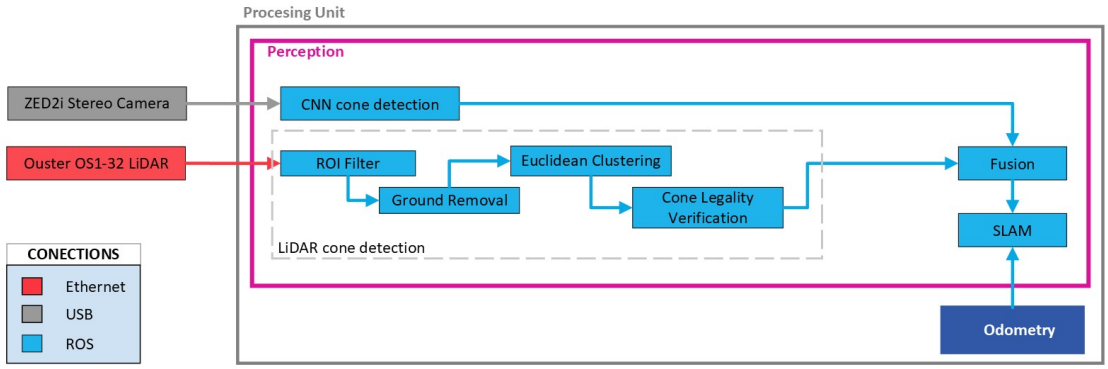


Figura 16: Imágenes de cada uno de los módulos por separado

2.4.2.1 Detección visual

Este sistema emplea «deep learning», específicamente el modelo «You Only Look Once» (YOLOv5), para analizar las imágenes capturadas por la cámara y detectar la presencia de conos y su color. El uso de redes neuronales permite que la máquina sea capaz de reconocer patrones en las imágenes en lugar de depender de ecuaciones predefinidas para obtener resultados imposibles con métodos tradicionales. Para entrenar el modelo YOLOv5, es necesario disponer de un conjunto de datos variado de conos, obtenido mediante la colaboración con otros equipos de FS y la participación en el conjunto de datos FSOCO [9]. Las posiciones de los objetos identificados se estiman y se utilizan para la fusión de sensores, que se logra utilizando el SDK de ZED debido a que ofrece resultados óptimos en comparación con otros algoritmos.

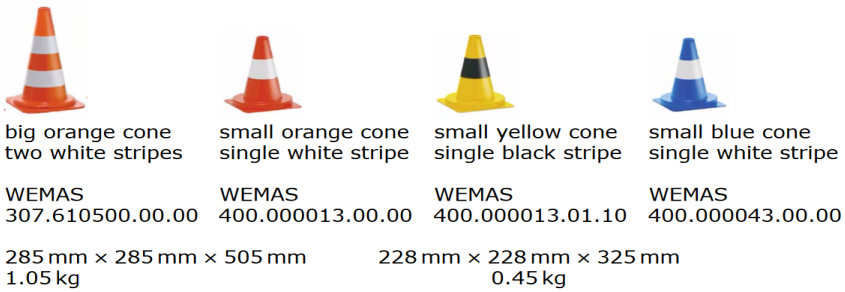


Figura 17: Tipos de conos y su clasificación en Formula Student Driverless

2.4.2.2 Detección en nubes de puntos

La nube de puntos generada por el lidar se procesa utilizando la librería Open3d, que proporciona objetos de nube de puntos y funciones predefinidas para filtrar los puntos. La Figura 16 ilustra el flujo seguido para obtener grupos de objetos en la nube. Una vez obtenida la nube de puntos, se filtra la región de interés y se restringen los ejes. Luego, se eliminan los puntos correspondientes al suelo y se agrupan los restantes. Por último, se verifica que estos grupos cumplan con ciertas condiciones de tamaño para confirmar que puedan ser conos. Las coordenadas resultantes se envían al módulo de fusión, donde se combinan con la información obtenida de la cámara.

2.4.2.3 Fusión entre cámara y LiDAR

Para integrar la información de ambos sensores, los datos publicados por los dos módulos anteriores se recopilan y sincronizan mediante una función de ROS2. Una vez los mensajes se han relacionado temporalmente, se compara su contenido. Para determinar que un cono detectado por la cámara corresponde a algún clúster del lidar, se verifica que sus respectivas coordenadas no difieran más allá de una cierta tolerancia.

2.4.2.4 Localización y mapeo

Pese a que es posible conseguir un sistema que sea capaz de generar trayectorias y seguirlas con la información instantánea del entorno, es decir haciendo uso únicamente de los tres primeros módulos descritos, un AS que solo tiene en consideración el estado actual del vehículo no será capaz de anticipar obstáculos o planificar una ruta que optimize tiempos y proporcione una conducción suave y

fiable. Nos encontraríamos ante un robot puramente reactivo. Es aquí donde reside la importancia del cuarto módulo y el objetivo de el presente TFG.

En definitiva, la entrada de este módulo consiste en los datos descritos de la fusión entre cámara y LiDAR 2.4.2.3 y los datos de movimiento relativo procedentes del conjunto de odometría. Y, la salida del sistema debe consistir en un mapa de posiciones de conos además del posicionamiento del vehículo en el mapa. Con esta información y, como se ha comentado, el conjunto de Planificación de trayectorias será capaz de definir las rutas que debe seguir el vehículo.

3. Conexión con los ODS

La implementación de un sistema SLAM en un vehículo autónomo de Formula Student Driverless, ofrece una amplia gama de beneficios tanto a nivel social como técnico, alineándose con varios Objetivos de Desarrollo Sostenible (ODS) establecidos por las Naciones Unidas.

El proyecto Formula Student Bizkaia, de hecho, está reconocido como proyecto de referencia en la EHUAgenda 2030, que desarrolla los ODS en el marco de las actividades de la UPV/EHU. Formula Student Bizkaia tiene como objetivo primordial el ODS 8, trabajo decente y crecimiento económico. A tal fin, su enfoque de proyecto docente se centra en el ODS 4, educación de calidad. El propio desarrollo de vehículos eléctricos, se enmarca en los ODS 7 (Energía asequible y no contaminante) y el ODS 12 (Producción y Consumo Responsable). Por último, en el desarrollo de este proyecto se trabajan objetivos transversales, como las acciones para la igualdad de género (ODS5) y las alianzas con entidades y empresas para lograr este objetivo (ODS17).

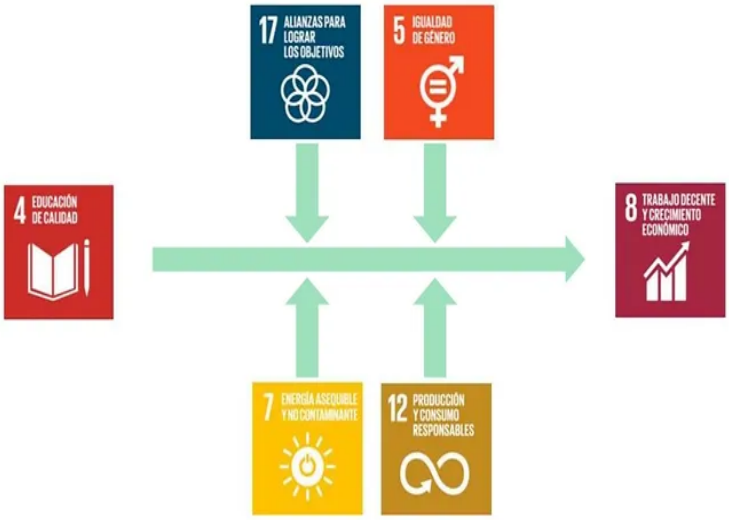


Figura 18: Objetivos de Desarrollo Sostenible de FSB

Adicionalmente a estos objetivos, por la naturaleza del presente trabajo, también se alinean:

- **Seguridad Vial (ODS 3):** Los vehículos autónomos reducen la incidencia de accidentes de tráfico causados por errores humanos, salvaguardando la vida y la integridad de las personas.

- **Acceso a la Movilidad (ODS 9):** Los coches autónomos ofrecen la posibilidad de transporte sin barreras para personas con movilidad reducida, favoreciendo la igualdad de oportunidades y la integración social.
- **Eficiencia Energética (ODS 7):** La detección inteligente del entorno y el mapeo del entorno en los vehículos automatizados permite ejecutar estrategias para una conducción más eficiente y a un menor consumo de energía.

Figura 19: ODS adicionales de FSB Driverless

La formación y experiencia adquirida durante los desarrollos de estos sistemas eleva un peldaño el conocimiento con el que los miembros de FSB dejan la escuela. De este modo, se fortalecen las habilidades con las que se enfrentan al entorno laboral, lo que resulta beneficioso tanto para las empresas como para los proyectos en los que se involucran. Como resultado, se mejorará el tejido empresarial y el producto final que se genera.

4. Objetivos

El objetivo principal de este trabajo es el del diseño, implementación y validación de un sistema de localización y mapeo simultáneos (SLAM) para el monoplaza automatizado ADS-DV. Este es un subconjunto crucial para el correcto funcionamiento del sistema automatizado, ya que permite minimizar el error en la localización del vehículo y la construcción de un mapa de pista en tiempo real.

Con el fin de desarrollar el objetivo anterior, se proponen los siguientes objetivos parciales:

- Diseño de un sistema SLAM que se adecúe a las necesidades de la competición.
- Adaptación de los subconjuntos del módulo de Percepción para la integración del SLAM
- Validación del sistema desarrollado con datos simulados.
- Validación con datos reales.

Con el fin de lograr los mencionados objetivos, se requerirá la utilización de herramientas de software estándares dentro del contexto del proyecto FSB Driverless. Estas herramientas incluyen Python, que se empleará para llevar a cabo la programación integral del módulo; ROS2 Humble, utilizado para facilitar la comunicación entre los diversos nodos que conforman el sistema y Docker, una herramienta que permite el empaquetamiento de entornos software en «contenedores» para agilizar el desarrollo y el despliegue de aplicaciones.

El propósito último de este proyecto consiste en desarrollar y presentar una percepción completamente funcional y validada para finales de la temporada 23/24. Por tanto, el alcance de este proyecto abarcará no solo el diseño e implementación de todos los nodos necesarios, si no también la evaluación del desempeño del conjunto en entornos simulados y reales. Esta fase de validación permitirá asegurar el correcto funcionamiento del AS que se utilizará en la competición FS-AI 2024.

5. Beneficios

En futuras temporadas, el equipo de Formula Student Bizkaia quiere dar el salto al monoplaza eléctrico driverless, y es por ello que este año se ha invertido en desarrollar y testear un sistema autónomo software funcional.

Cuando se habla de los beneficios de este proyecto se pueden diferenciar dos grupos principales enmarcados dentro del equio FSB. Por un lado resaltan las ventajas a nivel técnico, al ser los vehículos autónomos un área en constante desarrollo y crecimiento. Y por otro, también destacan las ventajas a nivel económico, siendo este tema de principal interés para muchas empresas en el sector de la mobilidad.

Además, el propio proyecto Formula Student Bizkaia (FSB) promueve la investigación y generación de conocimiento para futuros desarrollos. De igual modo, es una motivación para aumentar la participación del alumnado.

5.1. Beneficios técnicos

La automatización de un vehículo ofrece una serie de beneficios técnicos importantes que pueden mejorar significativamente su rendimiento y eficiencia. Teniendo en cuenta que en esta aplicación se trabaja con un coche de Formula Student (FS), se pueden generalizar los beneficios para este tipo de vehículos.

En primer lugar, los sistemas de percepción y toma de decisiones que se diseñan para estos coches, pueden mejorar la precisión y la velocidad de las respuestas del vehículo. Esto se debe a que los sistemas autónomos pueden procesar información más rápidamente y tomar decisiones en tiempo real basadas en una gran cantidad de datos, lo que reaccionar de manera más eficiente ante cualquier situación.

La automatización del monoplaza también puede mejorar su capacidad de rendimiento en el circuito ya que pueden ser programados para controlar la dirección, la velocidad y la posición del vehículo de manera más precisa. Estos conjuntos pueden mejorar la estabilidad y la maniobrabilidad del vehículo. Además, el sistema autónomo puede permitir una mayor precisión en la ejecución de estrategias de carrera, lo que puede mejorar el rendimiento general del vehículo. De hecho, ya hay monoplazas de FS que han conseguido mejorar sus tiempos cuando el coche conduce en modo autónomo.

En segundo lugar, la creación de un sistema autónomo robusto puede mejorar la eficiencia del vehículo en términos de consumo de combustible y energía eléctrica. La precisión en la toma de decisiones y la optimización del rendimiento pueden

reducir la necesidad de aceleraciones y frenadas bruscas, lo que conduce a un menor consumo de combustible o energía eléctrica. En el caso del monoplaza eléctrico de FSB, se podría optimizar la autonomía al reducir el consumo de energía.

Por último, la implementación de un AS para vehículos Formula Student (FS) puede promover la innovación tecnológica en la industria automotrizya que requiere el uso de tecnologías avanzadas y puede motivar la investigación, el desarrollo de nuevas soluciones tecnológicas y la mejora de las tecnologías existentes.

Por todos estos motivos, la automatización de un vehículo FS es una prioridad en la industria automotriz actual y es esencial para mejorar su rendimiento en la pista y en la competencia en general.

5.2. Beneficios económicos

La investigación y el desarrollo de los conocimientos del equipo FSB en el área de los coches autónomos consigue formar a ingenieros en un ámbito nuevo. Además, también abre puertas a otras competiciones europeas en las que cada vez es más relevante que los monoplazas que compiten sean automatizados. Si bien es cierto que el material necesario para realizar un sistema autónomo aumenta el presupuesto del monoplaza, también puede llamar la atención de nuevas empresas especializadas en áreas relacionadas con esta parte de la industria automotriz. Empresas que pueden estar interesadas y dispuestas a promocionar y subvencionar estos costes.

En resumen, la automatización del vehículo de Formula Student (FS) puede mejorar la imagen y la reputación del equipo en la industria, haciendo que haya más empresas interesadas en invertir dinero o material en el proyecto. Además, la utilización de tecnologías avanzadas puede hacer que aumente la visibilidad y el reconocimiento del proyecto.

6. Requerimientos

Debido a que el desarrollo del presente proyecto se centra en el sistema de percepción del AS de Formula Student Driverless, éste se tiene que adecuar a las necesidades y requerimientos de la competición y del resto de subconjuntos.

6.1. Normativa Formula Student UK

Todos los sistemas diseñados e implementados tienen que cumplir una normativa impuesta por la propia competición de FSUK que es actualizada de forma anual. Esta norma dedica un apartado exclusivamente a la competición *Driverless* o modalidad autónoma, que cuenta con un gran número de regularizaciones [7]. Es por esta razón que se ha llevado a cabo un exhaustivo estudio encontrando varias pautas que interfieren de forma directa o resultan relevantes para el desarrollo del presente TFG:

Referencia	**Cumplimiento de Normativa**
T1.3.3	Todos los sensores deben estar posicionados dentro de la superficie definida por la parte superior del TSAL montado en la «aleta principal» y los bordes externos de los cuatro neumáticos o dentro la carrocería original del ADS-DV con un margen de 100mm.
T4.4.2	No está permitido guardar información tras haber acabado un intento de una prueba dinámica y hay que demostrar que no se tienen mapas del entorno precargados para la prueba de autocross.
D6.3.2	Un examinador se encargará de asegurar que los equipos eliminan información previa antes de la prueba de autocross.

6.2. Internos

6.2.1. ROS2 Humble

La comunicación entre los distintos nodos o submódulos que componen el sistema autónomo, incluyendo a los sensores y actuadores, se realiza empleando ROS2 Humble. Robot Operating System (ROS) es un «middleware» que permite implementar una capa de abstracción a la hora de comunicar los diferentes elementos de un sistema robótico y facilita la modularización y paralelización de sistemas. Además, ROS es un software de código abierto que cuenta con varias herramientas y librerías para facilitar la creación de aplicaciones en el campo de la robótica: desde

«drivers» para sensores hasta implementaciones de algoritmos punteros (*Cartographer* [11], *Navigation2* [12], *TF2* [13]...).

Actualmente existen dos líneas de desarrollo de este «middleware»: ROS1 y ROS2, siendo la segunda la versión más reciente y a la que están migrando todos los desarrolladores. Es por ello que la versión seleccionada para el AS es ROS2 Humble, que cuenta con un End Of Life (EOL) en mayo de 2027 y está disponible para Ubuntu 22.04LTS, el sistema operativo escogido.

ROS define un sistema de paquetes en los que se contienen funcionalidades de código y los denominados «nodos», los cuales se comunican siguiendo dos estrategias principales: una arquitectura cliente-servidor denominados «ros services» y una arquitectura basada en publicadores y subscriptores.

Los «ros services» siguen un patrón solicitud-respuesta, donde donde un nodo (cliente) envía una solicitud a otro nodo (servicio) y espera una respuesta, lo que proporciona una comunicación síncrona y confiable. Este patrón es muy común a la hora de solicitar información del vehículo, solicitar una parada de emergencia o hacer un request de inicio de misión autónoma. Sin embargo, a la hora de comunicar sistemas de procesamiento de sensores es necesario conseguir el mayor tiempo de actualización posible, por lo que la arquitectura «Publisher/Subscriber» se amolda mejor, de forma que un nodo se encarga de enviar datos y pueden haber varios nodos que se subscriban a los «tópicos» de envío (similar a una arquitectura *MQTT*).

Para estandarizar y facilitar el intercambio de datos entre los nodos se utilizan los tipos de mensaje: estructuras de datos que establecen qué tipos de datos se van a publicar. Concretamente, el desarrollo de este TFG se tiene que amoldar a los tipos de mensajes empleandos empleandos por el resto de submódulos del AS.

6.2.2. EUFS Simulator

Durante el desarrollo de los algoritmos es necesario simular los sensores y el comportamiento del vehículo en un entorno controlado para que, una vez se hayan conseguido buenos resultados en simulaciones, sea más fiable y seguro probarlos en hardware físico. Es por ello que se ha recurrido al simulador EUFSsim [14], un simulador open-source desarrollado por el equipo de FS de la universidad de Edimburgo basado en el entorno de simulación *Gazebo*.

Este simulador está diseñado para que los equipos que compiten en FS-AI puedan probar sus desarrollos en circuitos prestablecidos que cumplen la norma y forman parte de la competición. Además, las comunicaciones están implementadas sobre

ROS2 haciéndolo compatible con el AS, el cual hace uso de los mensajes de ROS del simulador (eufsmsgs).

El simulador también ofrece la opción de simular varios módulos de forma independiente. Este es el caso de la simulación del módulo de percepción y odometría que nos ayuda a generar los datos de entrada necesarios para desarrollar el módulo de SLAM, cuya salida es recibida por otros submódulos y reflejada en la actuación del vehículo simulado.

6.2.3. Frecuencias de operación

Para garantizar un funcionamiento eficiente y seguro del sistema de control del AS, es fundamental que el sistema de percepción opere a una frecuencia que supere el mínimo requerido para el sistema de control de la actuación del vehículo. En este sentido, se establece que el sistema de percepción deberá operar a una frecuencia superior a 10Hz, asegurando así una actualización y respuesta adecuadas ante las condiciones del entorno en tiempo real.

Además, otro aspecto relacionado y que resulta crítico es la sincronización de los datos en el tiempo. Debido a que tenemos sistemas que toman datos de sensores y módulos que funcionan a distintas frecuencias, hay que garantizar que dichos datos estén en un marco de tiempo común para no obtener medidas erróneas.

6.2.4. Requisitos de cómputo

El software que se desarrollará para el AS a lo largo de este TFG estará sujeto a un hardware específico, el cual consiste en dos ordenadores con características similares. Estos ordenadores proporcionarán la capacidad de procesamiento necesaria para ejecutar los algoritmos requeridos por el sistema autónomo.

En primer lugar se encuentra el InCarPC CQ77G, instalado en el vehículo de la competición. Este es el ordenador que dará soporte al AS durante las pruebas dinámicas de FS-AI.Y en segundo lugar nos encontramos con el OnLogic Karbon 803, utilizado para realizar desarrollos y validaciones en un entorno similar al InCarPC. Estas son las características de ambas unidades de procesamiento:

Características	**InCarPC CQ77G**	**OnLogic Karbon 803**
Procesador	Intel Core i7-8700T (4,0GHz)	Intel Core i7-13700TE (4,8GHz)
GPU	NVIDIA GeForce GTX 1060	NVIDIA T1000 8GB
RAM	32GB	32GB SO-DIMM DDR4 2666
WLAN	Internal 802.11ac	Internal 802.11ac
Bluetooth	BT v5.1	-
CAN Module	Dual CAN module	CAN interface

Tabla 3: Especificaciones técnicas del hardware

Es importante tener en cuenta que, a pesar de que el hardware descrito es potente, se debe prestar atención a la eficiencia de los algoritmos ya que el software del sistema autónomo final estará compuesto por varios módulos concurrentes que funcionarán en paralelo. Por lo tanto, la implementación de los algoritmos debe ser eficiente para garantizar un rendimiento óptimo del sistema.

7. Estado del Arte: Algoritmos de SLAM

El Simultaneous Localization And Mapping (SLAM) es un aspecto fundamental en el desarrollo de la conducción autónoma. Aunque ahora se esté trabajando intensamente en este campo, sus raíces se encuentran en la robótica, donde durante muchos años ha sido un desafío de referencia. La mayoría de los algoritmos SLAM se han diseñado inicialmente para robots móviles, proporcionando una base sólida sobre la cual se han construido los sistemas de conducción autónoma modernos. Por ello, durante el desarrollo de este TFG, se englobará el marco de la conducción automatizada dentro dentro del ámbito de la robótica.

En las últimas décadas, la robótica ha experimentado un cambio significativo en su enfoque, pasando de modelos deterministas a una perspectiva probabilística. Este cambio ha sido impulsado por diversos factores, desde la naturaleza incierta del entorno en el que operan los robots, hasta la incertidumbre impuesta por el ruido presente en las mediciones de los sensores y actuadores. En este nuevo paradigma, los robots, y por extensión los vehículos autónomos, no pueden depender únicamente de datos precisos y modelos rígidos; en cambio, deben ser capaces de lidiar con la incertidumbre y adaptarse a distintos entornos. Por ende, la adopción de enfoques probabilísticos en la robótica y la conducción autónoma permite codificar la información en distribuciones de probabilidad que abarcan todo un espacio de posibles hipótesis, dejando de lado un único *best guess*.

En este contexto se encuentra uno de los problemas que más atención está ganando en el mundo de la conducción automatizada: el Simultaneous Localization And Mapping (SLAM). El SLAM implica la creación de un mapa de un entorno desconocido mientras se intenta determinar la ubicación del vehículo en dicho mapa. Este concepto es el que se refleja en la Figura 20, donde se representan cuatro instantes temporales de un robot y cinco elementos de referencia. El robot comienza en la posición x_{t-1} y observa los dos primeros objetos. Luego se mueve en la dirección u_t hasta la posición x_t para observar de nuevo uno de los objetos de la iteración anterior y dos objetos nuevos. El proceso continúa sucesivamente y poco a poco se va construyendo el mapa del entorno.

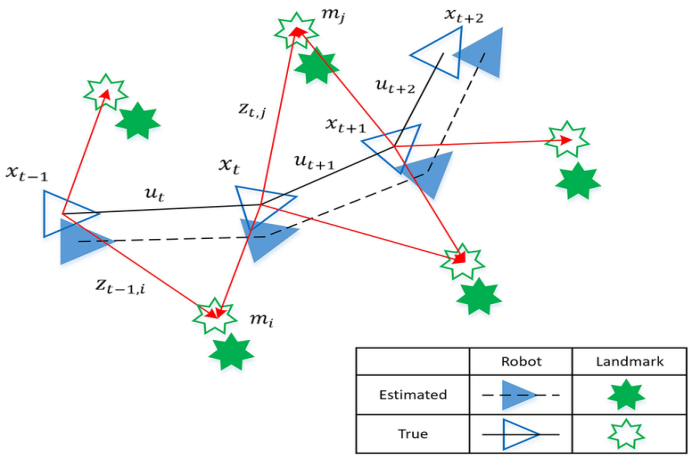


Figura 20: Esquematización del problema

Este es un problema difícil ya que se asemeja a la paradoja de «el huevo y la gallina»: para lograr una localización precisa, se requiere un mapa confiable; sin embargo, para construir dicho mapa, es necesaria una localización precisa. En respuesta a este desafío, se han desarrollado numerosos enfoques y técnicas para abordar el problema del SLAM. De esta manera, el SLAM no se refiere a una única técnica, sino más bien a un conjunto de diversos métodos.

Estas técnicas pueden ser agrupadas de diversas maneras, pero dada la naturaleza específica del caso de uso, que implica vehículos autónomos navegando en entornos exteriores donde predominan los conos como elementos de referencia, nos centraremos en tres clasificaciones principales: según la representación del entorno, según los sensores utilizados y según la metodología de estimación.

7.1. Técnicas SLAM según la representación del entorno

Las técnicas SLAM pueden ser clasificadas según la manera en que representan el entorno en el que operan los vehículos autónomos. Existen dos enfoques principales: las matrices de ocupación y los sistemas basados en landmarks.

7.1.1. Matrices de Ocupación

Estos métodos representan el entorno mediante una matriz que indica la ocupación de cada celda en un espacio discretizado. Cada celda tiene predefinido un tamaño específico ($1m \times 1m$ por ejemplo) que determinará la resolución de todo el mapa.

Además, como se ha indicado previamente, se emplea un enfoque probabilístico por lo que en cada celda se representa una distribución de probabilidad que indica si hay presente un obstáculo o no.

Lo interesante de este enfoque es que se integra directamente toda la información de los sensores para generar un mapa complejo en el que pueden aplicarse algoritmos de «pathplaning» como *RRT** de forma directa o con un «postprocesado» sencillo.

(a) 2D LiDAR accumulated input (b) Matriz de ocupación

Figura 21: SLAM basado en matrices de ocupación

Esta técnica es muy común para modelar entornos complejos y dinámicos destacando en áreas como el mapeo de habitaciones, parkings, cuevas o minas. Sin embargo, pueden requerir una gran cantidad de memoria y procesamiento para almacenar y procesar la información de la cuadrícula.

7.1.2. Basado en Landmarks

En este enfoque, el entorno se representa mediante puntos de referencia distintivos, conocidos como landmarks (Figura 20). Estos landmarks pueden ser cualquier característica del entorno que sea fácilmente identificable y distinguible, como esquinas, bordes, puntos de referencia o cualquier otro rasgo único.

Los sistemas SLAM basados en landmarks son adecuados para entornos donde las características del entorno son fácilmente distinguibles y estables en el tiempo, lo que facilita la detección y asociación de landmarks. Sin embargo, pueden ser más sensibles a la presencia de landmarks ambiguos o dinámicos, lo que puede requerir técnicas adicionales para lidiar con estos desafíos.

Algunas implementaciones de sistemas SLAM basadas en landmarks son *MRPT SLAM* [15], un *toolkit* que cuenta con implementaciones para ROS 1, o *Real-Time Appearance-Based Mapping* [16], un sistema SLAM que detecta landmarks partiendo de imágenes y cuenta con nodos para ROS 2.

7.2. Soluciones SLAM según los sensores utilizados

Se pueden diferenciar varios tipos de SLAM en función de los sensores de entrada que presente el sistema. De esta manera hay sistemas que hacen uso de un único sensor o llevan a cabo una fusion de sensores para resolver el problema del mapeo y localización simultáneos. En esta sección vamos a cubrir los tipos principales de SLAM haciendo mención a módulos robustos ya implementados para ROS.

7.2.1. Laser scan

Estos métodos se basan principalmente en datos obtenidos por sensores LiDAR o similares que proporcionan mediciones tridimensionales de distancia y, en algunos casos, intensidad de retorno de luz. Existen muchas soluciones SLAM robustas que hacen uso de este tipo de sensores ya que suelen tienen una alta fiabilidad tanto en interiores como en exteriores y entornos con baja visibilidad. Es por ello que es bastante común encontrar este tipo de sensores en vehículos automatizados, y normalmete se suele registrar una nube de puntos acumulada con todos los frames para aplicar técnicas de «postprocesado» y «feature detection».

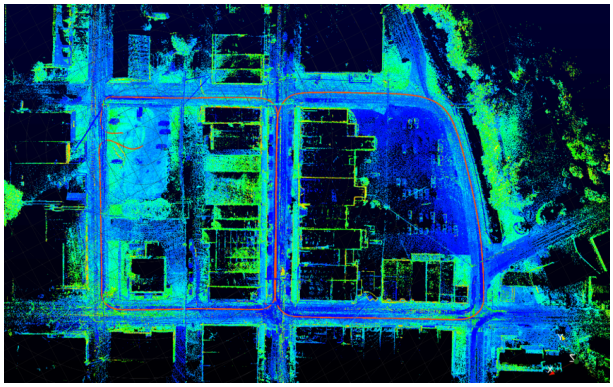


Figura 22: Pointcloud acumulado utilizando LiDAR SLAM

Destacan implementaciones como *HectorSLAM* [17], que necesita información de un LiDAR y de odometría; las soluciones *LOAM* [18] que solo requieren del «point-cloud» como input ya que estiman el movimiento comparando frames consecutivos [19]; *GMapping* [20] o el ya mencionado *Cartographer* [11].

7.2.2. Visual

En este tipo de SLAM, se utilizan cámaras para obtener información visual del entorno. Los algoritmos de Visual SLAM pueden basarse en características visuales como puntos de interés, líneas o características estructurales.

Uno de los mayores beneficios del Visual SLAM es su capacidad para funcionar en entornos altamente variables y dinámicos. Sin embargo, el Visual SLAM suele proporcionar resultados mediocres en entornos homogéneos o con características visuales limitadas. Además, factores como la oclusión y el movimiento rápido pueden dificultar el seguimiento continuo de las características visuales.

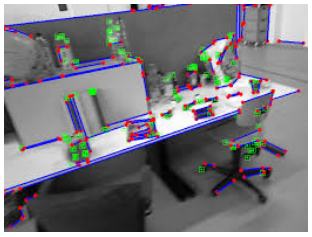


Figura 23: «feature extraction» en Visual Slam [21]

Entre las implementaciones estudiadas destacamos el visual SLAM implentado por el SDK de la ZED2i; *ORB-SLAM* [22], disponible tanto para cámaras monoculares como estéreo, o *DPPTAM* [23].

7.3. Técnicas SLAM según la metodología de estimación

En este subapartado, trataremos los tres paradigmas principales que se adoptan en técnicas SLAM. Cada uno de estos paradigmas ofrece un marco conceptual y metodológico único para abordar los aspectos fundamentales del problema SLAM, estimando de distintas formas las posiciones del agente y los obstáculos.

7.3.1. Basado en Grafos

Este paradigma trata el problema SLAM como un grafo probabilístico, donde los nodos representan las poses del robot o de obstáculos en diferentes momentos o ubicaciones, y las aristas representan las relaciones de movimiento u observación entre las poses y las características del entorno. Los métodos basados en grafos

utilizan técnicas de optimización para estimar la trayectoria del robot y la ubicación de las características del entorno, minimizando la discrepancia entre las observaciones del sensor y las predicciones del modelo [24].

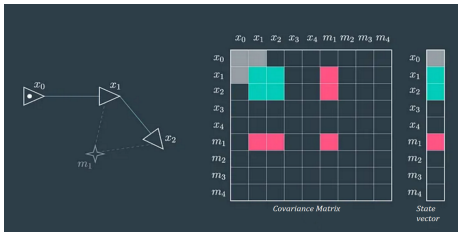


Figura 24: Estructuras de datos GraphSLAM

Los métodos *GraphSLAM* ofrecen ventajas significativas en términos de representación global del entorno, capacidad de integración de datos y están demostrando proporcionar mejores resultados que técnicas SLAM más tradicionales [25]. Sin embargo, es importante reconocer sus posibles limitaciones en cuanto a eficiencia computacional a la hora de optimizar grafos de gran tamaño.

7.3.2. SLAM como problema de estimación de estado

El enfoque SLAM como problema de estimación de estado se basa en formular el problema SLAM como un problema de estimación estadística. En lugar de construir un mapa explícito del entorno y luego optimizarlo, como es el caso del *GraphSLAM*, este enfoque se centra en estimar el estado del sistema, que incluye la posición y orientación del robot, así como la ubicación de las características del entorno, a partir de las observaciones del sensor y los comandos de control.

En este enfoque, se utiliza un modelo probabilístico para representar la incertidumbre asociada con la estimación del estado del sistema. El estado del sistema se representa como una distribución de probabilidad, que se actualiza continuamente a medida que se reciben nuevas observaciones del sensor y se ejecutan comandos de control. Esta actualización se realiza utilizando técnicas de fusión de datos, como el Extended Kalman Filter (EKF) o el filtro de partículas.

7.3.3. Deep Learning

Los sistemas SLAM basados en «deep learning» están experimentando un crecimiento significativo en los últimos años. Ejemplos de estos sistemas incluyen *RatSLAM*, *LIFT-SLAM* o *EnvSLAM*. En términos generales, los enfoques SLAM

basados en el aprendizaje profundo representan una técnica emergente que ha demostrado resultados prometedores en diversas aplicaciones.

Sin embargo, es importante destacar que estos sistemas también enfrentan importantes limitaciones, como los elevados requisitos computacionales necesarios para el entrenamiento y la ejecución de los modelos; la complejidad de su implementación y la generabilidad de estos algoritmos en entornos diversos y en condiciones variables.

7.4. Selección de algoritmos a implementar

Dada esta información, se nos presentan dos opciones: diseñar un sistema SLAM basado en una implementación ya hecha en ROS y que tome la información directamente de los sensores, lo que se conoce como un sistema «early fusion», o implementar un sistema donde se realice un preprocesado de los sensores para luego fusionar la información.

Además de esto, no hay que olvidar la salida basada en conos (Sección 2.4.2.4) que debe tener el sistema y el caso de uso, ya que el vehículo operará en exteriores. Es por ello que las aproximaciones SLAM basadas en «landmarks» suponen una mejor opción al contar con elementos facilmente distintivos. De igual manera, en el caso de recurrir a un sistema SLAM ya implementado, las opciones basadas en LiDAR prevalecerían sobre las opciones *Visual SLAM*, ya que, en general, presentan mejores resultados en exteriores [26].

Sin embargo, la primera opción descrita queda descartada, ya que desde un primer momento se diseñó el módulo de percepción (Sección 2.4.2) como un sistema «late fusion», a lo que hay que sumarle la complejidad del postprocesado para la detección de «features» en este sistema. De igual forma, también queda descartado el uso de técnicas «deep learning» debido a su alta complejidad y escasa documentación.

Con todo ello, la solución propuesta consiste en un SLAM basado en filtros y landmarks, descartando el *GraphSLAM* aunque también puede ser una buena opción. Concretamente se va a optar por dos implementaciones distintas: un «FastSLAM» y un «EKFSLAM». De esta manera, se pretende conseguir un sistema computacionalmente eficiente y que proporcione estimaciones en «tiempo real».

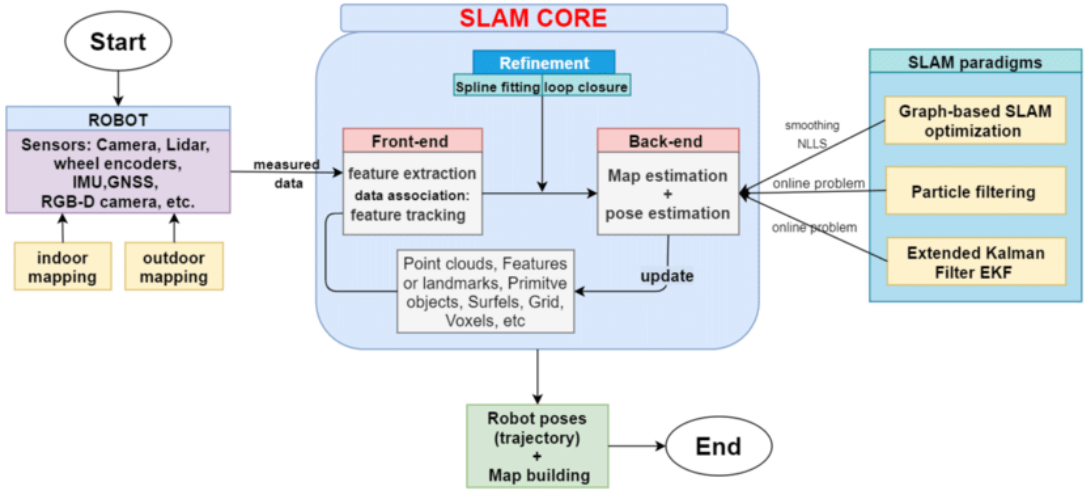


Figura 25: Esquema general SLAM

8. FastSLAM

Se ha elegido realizar una implementación del algoritmo «FastSLAM» como una primera aproximación al problema del SLAM. FastSLAM es un algoritmo que hace uso de un filtro de partículas adaptado para estimar la predicción a posteriori del estado del robot. Cada partícula se puede interpretar como una estimación independiente del estado que, tras unos procesos de predicción y corrección, convergen para devolver una estimación más fiable. Este algoritmo destaca por ser una implementación bastante eficiente que, unido a la documentación existente, lo vuelven una buena opción a considerar.

En el enfoque considerado de SLAM, se ha optado por utilizar solo dos dimensiones y la orientación del vehículo, lo que simplifica enormemente la implementación de los algoritmos. Además, el estado del sistema no se restringe únicamente a la posición y orientación del vehículo, sino que también abarca las posiciones de todos los puntos de referencia (landmarks), aunque solo se toman en cuenta sus coordenadas, las cuales están todas referenciadas al «frame» del mapa global (8.2). Así, llegamos a la siguiente expresión:

$$X = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \quad Z = \begin{bmatrix} lx_1 \\ ly_1 \\ \vdots \\ lx_n \\ ly_n \end{bmatrix} \tag{1}$$

Como se ha mencionado, FastSLAM hace uso de un filtro de partículas, cuya misión es la de estimar un conjunto de vectores de estado $\{x_k\}$ para un sistema dinámico, donde $k \in \mathbf{N}$ es una sucesión de marcas de tiempo. Para lograrlo es necesario que existan dos aportes de información. En primer lugar, necesitamos las acciones de control que se han llevado a cabo desde el estado anterior, las cuales denotaremos como u_k y provienen del módulo de odometría (Sección 2.4.2). Con estos datos se lleva a cabo la primera etapa del filtro de partículas: la **predicción** del nuevo estado. En segundo lugar, necesitamos las observaciones de los landmarks para el instante actual z_k. Esta información proviene de la fusión del módulo de percepción (Sección 2.4.2) y gracias a ella se lleva a cabo la segunda etapa del filtro de partículas: la **corrección** de x_k. Finalmente, hay una etapa de «remuestreo» de las partículas, la cual se explicará con detalle más adelante. De manera general, todos estos procesos se resumen en el siguiente diagrama de flujo:

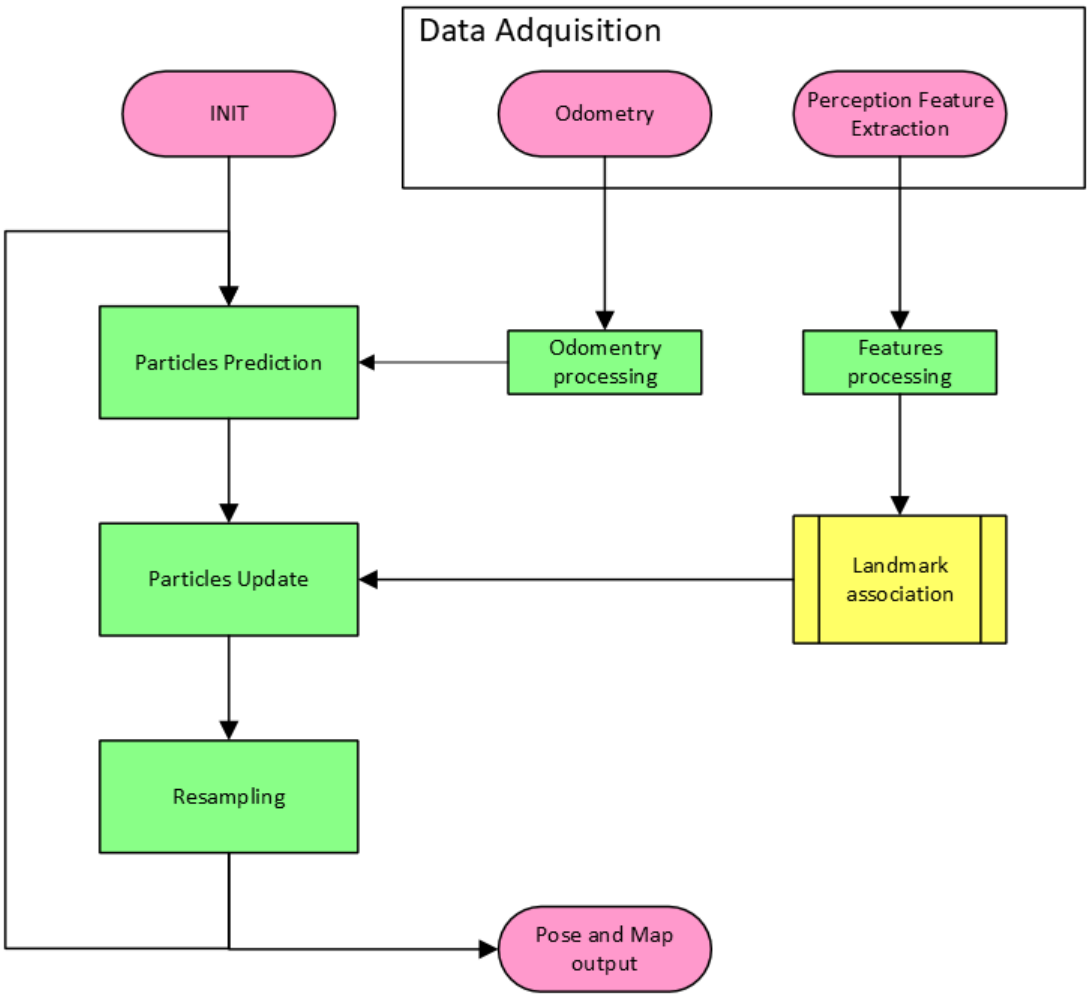


Figura 26: Diagrama de flujo del FastSLAM

El sistema se inicializa generando un número determinado de vectores de estado (Ecuación 1), también conocidos como partículas, siguiendo una distribución normal en un espacio acotado de $1m \times 1m$ alrededor del origen.

8.1. Predicción del nuevo estado

La entrada de datos de odometría define las acciones de control que se han llevado a cabo desde el último instante registrado. Sin embargo, esta información no es suficiente para estimar el nuevo estado: es necesario definir un modelo cinemático f_k que dado un estado anterior x_{k-1} sea capaz de predecir el nuevo estado x_k

tras una evolución temporal. Considerando la información de entrada, podemos ejemplificar el proceso de predicción con el siguiente diagrama:

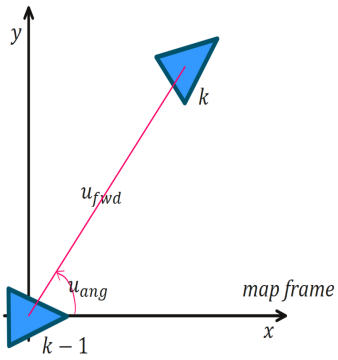


Figura 27: Esquema del movimiento del vehículo

Como se puede observar, el modelo f_k resultante no sería lineal debido a que la relación entre las variables de estado y las acciones de control u_k involucra funciones trigonométricas. Esto es particularmente importante ya que la popularidad de los filtros de partículas radica en su capacidad para manejar modelos no lineales sin la necesidad de realizar linearizaciones como es el caso de métodos más tradicionales. De esta manera, llegamos a la siguiente expresión:

$$f_k(x_{k-1}, u_k) = \underbrace{\begin{bmatrix} x \\ y \\ \theta \end{bmatrix}_k}_{x_k} = \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\text{A}} \underbrace{\begin{bmatrix} x \\ y \\ \theta \end{bmatrix}_{k-1}}_{x_{k-1}} + \underbrace{\begin{bmatrix} t\cos(\theta_{k-1}) & 0 \\ t\sin(\theta_{k-1}) & 0 \\ 0 & t \end{bmatrix}}_{\text{B}} \underbrace{\begin{bmatrix} u_{\text{fwd}} \\ u_{\text{ang}} \end{bmatrix}}_{u_k} \tag{2}$$

Este modelo cinemático se aplica para cada una de las partículas instanciadas, pero, como es de presuponer, si aplicamos las mismas entradas para todas las partículas, el sistema tendrá muy poca variabilidad. Es por ello que es muy común introducir algo de ruido, mostrando un comportamiento de las partículas más variado (Figura 28) y garantizando una exploración más efectiva del espacio de estados. En este caso, se ha empleado ruido proveniente de una distribución normal que tiene por media la acción de control u_k y por covarianza la matriz Q, hiperparámetro del sistema y que debe ser ajustada según el caso de aplicación. Además, es importante mencionar que a esta sucesión de predicciones de estado sin la corrección con observaciones se le conoce como «dead reckoning».

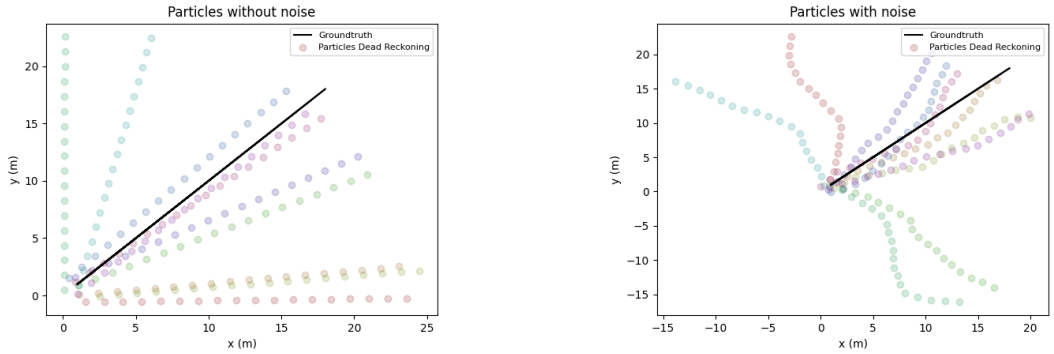


Figura 28: Dead Reckoning de las partículas

8.2. Asociación de observaciones

La segunda entrada de datos en nuestro sistema son las observaciones de los conos provenientes del módulo de fusión (Sección 2.4.2). Sin embargo, antes de poder realizar la corrección del estado con dichas observaciones, es necesario preprocesar los datos. Esta etapa de preprocesado desempeña un papel crucial al asignar identificadores a las observaciones, lo que nos permite determinar si cada visualización corresponde con un landmark ya incluido en el mapa o se trata de una nueva observación. Para lograr esto, es esencial trabajar en un marco de coordenadas común.

Los «frames de coordenadas», también conocidos como «sistemas de coordenadas» o «marcos de referencia», son estructuras utilizadas para definir la posición y orientación de objetos en un espacio tridimensional. En nuestro contexto, estos frames de coordenadas juegan un papel fundamental en el preprocesamiento de datos. En particular, necesitamos transformar las observaciones de los conos al frame del mapa global antes de poder asignarles identificadores y continuar con la etapa de corrección.

A lo largo de este TFG, los frames de coordenadas y las transformaciones entre marcos de referencia toman gran importancia debido a la necesidad de procesar los datos tomados de los sensores en un sistema de referencia común. Por ello, es necesario hacer mención al «arbol de transformadas» definido para nuestro sistema (Figura 29), en el que se detalla la relación entre los distintos frames de coordenadas de los sensores, actuadores, uniones físicas, las coordenadas del GNSS y el sistema de coordenadas globales para nuestro mapa del circuito. Este enfoque nos permite integrar de manera efectiva las observaciones y realizar dichas transformaciones con ayuda de funciones de ROS (Sección 6.2.1).

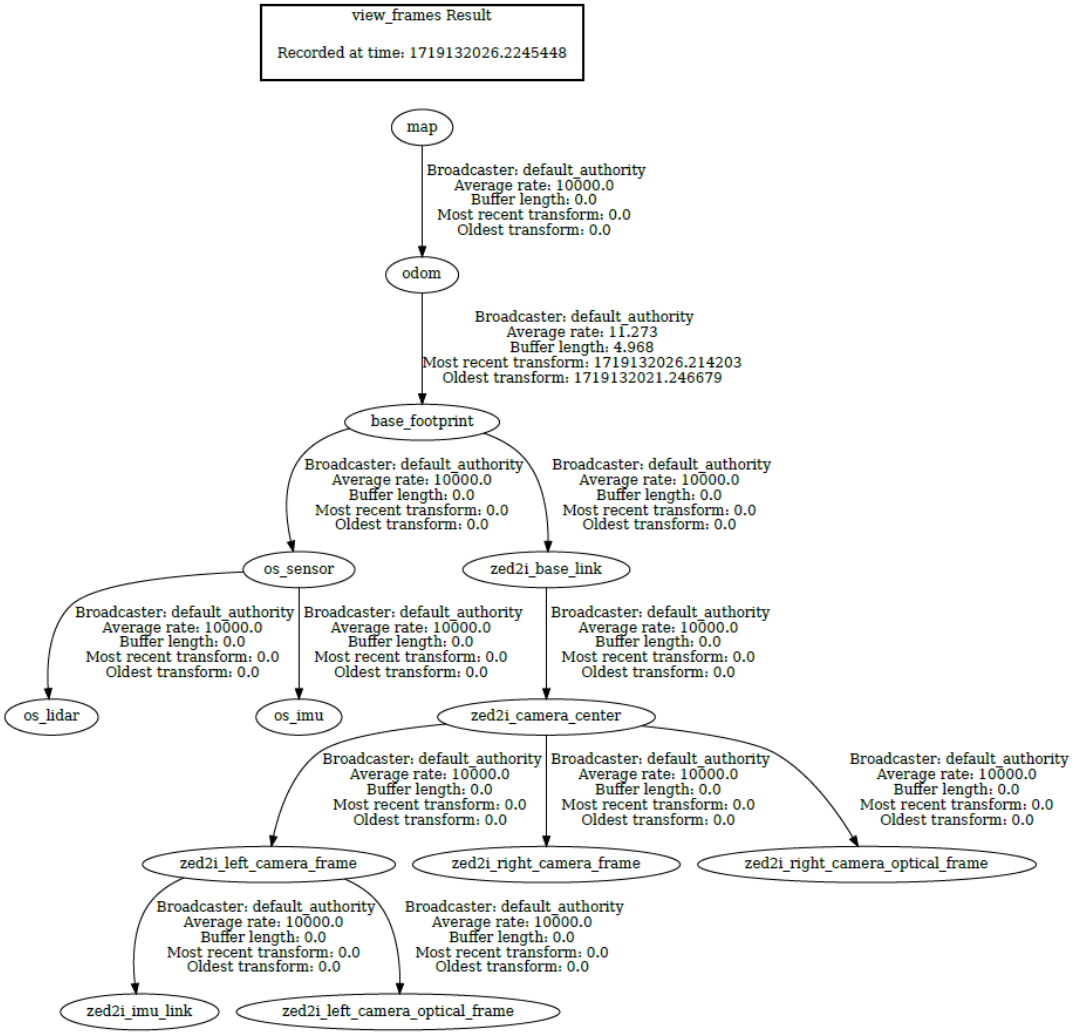


Figura 29: Árbol de transformadas del AS

Con la transformación de las coordenadas de las observaciones al frame del mapa, se puede proceder con la asignación de identificadores. En este proceso, se adopta un enfoque iterativo que implica explorar todas las partículas y obtener los mapas locales o matrices de landmarks asociadas a cada una. El objetivo es determinar si la distancia euclidiana entre la observación y algún landmark de alguna partícula es menor que un cierto umbral predefinido, siempre y cuando las observaciones sean del mismo color. Si esta condición no se cumple para ningún landmark, se interpreta la observación como nueva y se le asigna un identificador único, incrementando así el conteo total de landmarks en el sistema.

Además, al igual que en la etapa de predicción donde se define un modelo cinemático para relacionar el estado anterior con el nuevo estado, también hay que establecer un modelo de observaciones que vincule las mediciones con el estado actual. Este modelo de observaciones h_k (Ecuación 4) cobra especial importancia en la etapa de corrección del filtro de partículas. Sin embargo, es necesario hacer mención a él, ya que la salida del módulo de asociacion depende de la representación del entorno considerada por dicho modelo. Por lo tanto, para simplificar el sistema, el modelo de observaciones se constituye como un sensor de distancia y orientación, considerando únicamente la distancia y el ángulo del landmark con respecto al vehículo en el estado x_k.

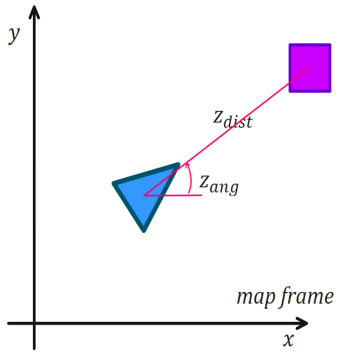


Figura 30: Esquema del modelo de observación

Un aspecto muy importante que no debe pasarse por alto en los sistemas SLAM es lo que comúnmente se conoce como «loop closure» o cierre de lazo. En esta aplicación, el cierre se produce al detectar los conos naranjas grandes, que indican el inicio de la pista, una vez que el vehículo ha completado una vuelta completa en el circuito. A diferencia de los sistemas SLAM basados en grafos en los que la detección del «loop closure» desencadena métodos de optimización del grafo, en nuestro caso el sistema se optimiza mediante la asociación de identificadores. A pesar de ello, es esencial registrar este evento para llevar un conteo preciso del número de vueltas que ha dado el vehículo en el circuito.

Con todo ello, la etapa de asociacion de observaciones se resume con el siguiente algoritmo:

Algorithm 1 Procesamiento de observaciones de landmark

$z \leftarrow \{\}$
for each observation c in *cone_list* **do**
  $c \leftarrow$ transform_to_map_frame(c)
  $id \leftarrow -1$
  for each particle p in *particles* **do**
    for each landmark i in p_z **do**
      $d_i \leftarrow \sqrt{(c_x - p^i_{z,x})^2 + (c_y - p^i_{z,x})^2}$
      if $d_i < BIAS$ and $l_{\text{color}} = c_{\text{color}}$ **then**
        $id \leftarrow i$
      end if
    end for
  end for
  if $id = -1$ **then**
    continue
  end if
  dist $\leftarrow \sqrt{c_x^2 + c_y^2}$
  $\alpha \leftarrow \text{atan2}(c_y, c_x)$
  $\mathbf{z}_i \leftarrow \begin{bmatrix} \text{dist} \\ \alpha \\ id \end{bmatrix}$
  $z \leftarrow z \cup \{z_i\}$
end for
return z

8.3. Corrección con observaciones

La esencia de los filtros de partículas radica en la estimación de una probabilidad a posteriori, la cual se representa mediante una función de densidad de probabilidad discreta que se aproxima mediante la suma de muestras ponderadas. Estas muestras están constituidas por las partículas previamente mencionadas. Por consiguiente, cada partícula no solo representa una configuración del estado del vehículo autónomo (incluyendo su posición y landmarks) dentro del espacio de estados posibles, sino que también posee un peso asociado que refleja la probabilidad de que esa instancia corresponda con el estado real. En resumen, el objetivo final es el de calcular x_k sabiendo que se han realizado las observaciones z_k, lo que puede expresarse de la siguiente manera:

$$p(x_k|z_k) \approx \sum_{i=1}^{Np} w_k^i \delta(x_k - x_k^i) \tag{3}$$

Donde Np es el número de partículas instanciadas; w_k^i es el peso o verosimilitud de la partícula y la delta de Dirac $\delta(x_k - x_k^i)$ representa el estado estimado de la partícula i en el instante k. Así, aquellas partículas que tengan un peso más elevado tendrán una mayor participación en la estimación final del estado. Cabe destacar que $\sum_{i=1}^{Np} w_k^i = 1$, lo que implica que todos los pesos deben estar normalizados.

Con ello, inicia la primera etapa de corrección en el proceso de FastSLAM: el cálculo de los pesos de las partículas. Para ello, se busca calcular la probabilidad de que la obseración real z_k se corresponda con las observaciones estimadas de las partículas $\hat{z}_k$. Esta estimación se realiza empleando el modelo de observaciones h_k el cuál, como se ha mencionado anteriormente, se constituye como un sensor de distáncia y ángulo y se puede formular de la siguiente manera:

$$h_k(\hat{x}_k, z_k) = \begin{bmatrix} z_{dist} \\ z_{ang} \end{bmatrix} = \begin{bmatrix} \sqrt{(z_x - \hat{x}_x)^2 + (z_y - \hat{x}_y)^2} \\ \arctan 2(\frac{z_y - \hat{x}_y}{z_x - \hat{x}_x}) \end{bmatrix}_k \tag{4}$$

De esta forma y teniendo en cuenta el desarrollo matemático del Teorema de Bayes [27] se obtiene la ecuación 5, donde se pretende ajustar el peso en función de la concordancia entre la observación real y la predicción de la partícula.

$$w_k^i \propto w_{k-1}^i p(z_k|x_k^i) \tag{5}$$

Como se puede apreciar, la verosimilitud de la partícula es directamente proporcional a la verosimilitud en el instante anterior por la probabilidad de la concordancia. Para calcular dicha probabilidad es necesario recurrir a una distribución de probabilidad auxiliar que recibe el nombre de «proposal distribution» y depende de la naturaleza de los sensores de medición. En este caso, se emplea una distribución normal con covarianza R y media z_k que es evaluada en la estimación $\hat{z}_k$ de la partícula.

$$p(z_k|x_k^i) = \prod_{j=1}^{Nl} \mathcal{N}(\begin{bmatrix} \hat{z}_{dist}^i \\ \hat{z}_{ang}^i \end{bmatrix}_k^j ; \begin{bmatrix} z_{dist}^i \\ z_{ang}^i \end{bmatrix}_k^j, R) \tag{6}$$

Finalmente, se normalizan todos los pesos y se calcula el estado del vehículo con la ecuación 3.

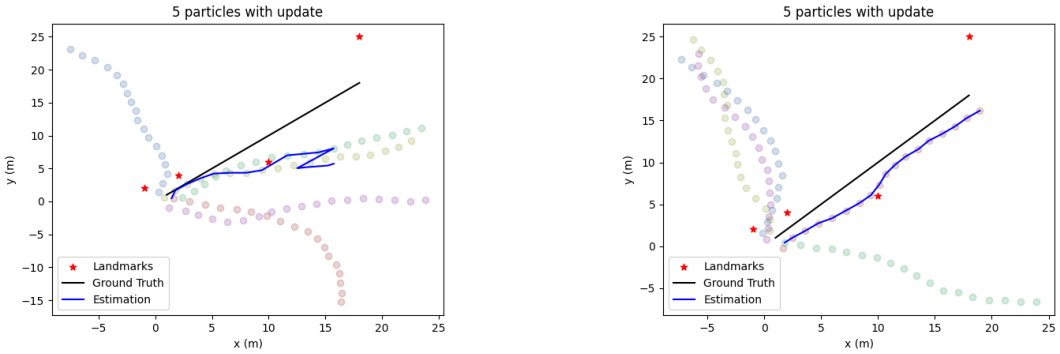


Figura 31: Corrección del estado con partículas

8.4. Remuestreo de partículas

En este punto, al analizar detenidamente la Figura 31, nos encontramos con uno de los problemas más significativos de los filtros de partículas: la degeneración de los pesos. Conforme avanza el proceso de iteración, el peso de una sola partícula tiende a predominar sobre el resto, imponiendo su estado como la estimación final. Incluso cuando una partícula exhibe una gran divergencia y tiene un peso extremadamente bajo, sigue siendo simulada, a pesar de su escasa contribución al cálculo del estado final. Por lo tanto, la etapa de remuestreo se convierte en un paso crítico para abordar los valores atípicos y mitigar el determinismo impuesto por una sola partícula.

El propósito fundamental del remuestreo es reinicializar las partículas, minimizando aquellas con pesos bajos. Para lograr esto, se implementa un enfoque estocástico en el cual se seleccionan nuevas partículas al azar, con reemplazo, del conjunto existente de partículas ponderadas por sus pesos asociados. Las partículas con pesos más altos tienen una mayor probabilidad de ser duplicadas, mientras que las de menor peso tienen más probabilidades de ser eliminadas. Una vez completado el proceso de remuestreo, se restablecen los pesos de todas las partículas a $1/Np$.

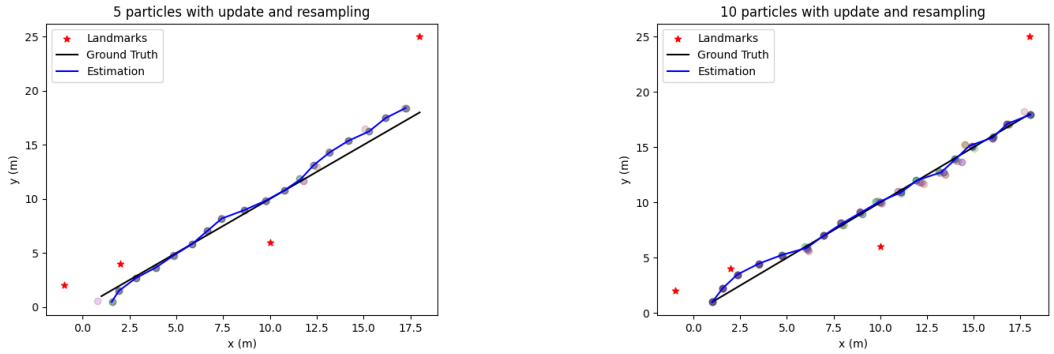
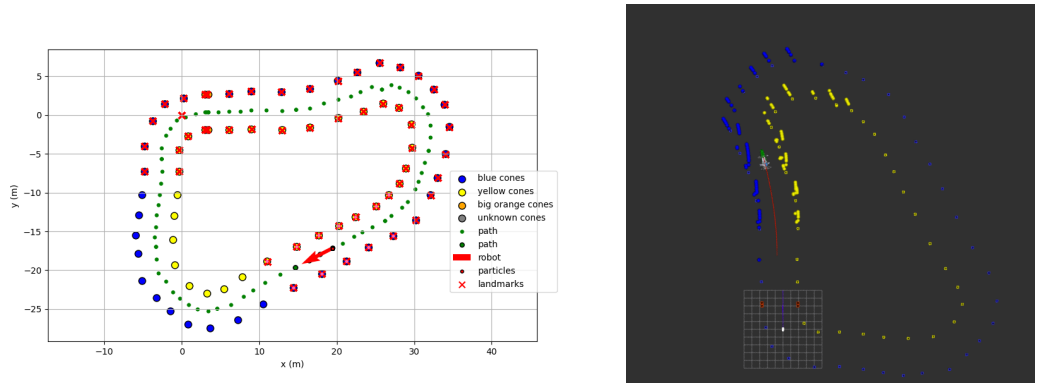


Figura 32: Estimación de las partículas con remuestreo

8.5. Resultados

A la hora de desarrollar el módulo de FastSLAM, se ha modelizado un entorno en Python en el cual se carga un mapa de un circuito, con sus correspondientes conos clasificados, y una trayectoria a seguir por el vehículo autónomo, la cual corresponde con los puntos medios del circuito ya precargados. Además, se ha modelizado la entrada del input del sistema de una forma muy básica con el fin de agilizar el proceso de desarrollo, lo que ha resultado en la creación de un sistema muy simplificado para poder probar todos los algoritmos antes de enfrentarse a sistemas más complejos.

Tras optimizar varios parámetros de FastSLAM, se han obtenido muy buenos resultados en este entorno. Las partículas convergen y el módulo es capaz de mapear el entorno y localizar el vehículo de forma precisa, ya que la simulación de la entrada de datos es ideal y no presenta ruido (Figura 33a). Sin embargo, al utilizar un entorno de simulación más complejo, como el simulador EUFS Simulator (Sección 6.2.2), el sistema FastSLAM ya no funciona correctamente.



(a) Entorno de simulación Python (b) EUFS Simulator

Figura 33: FastSLAM en simulación

Concretamente, se presentan dos tipos de errores. Por un lado, el módulo de asociación no puede asignar correctamente los identificadores debido a la gran divergencia de las partículas. Esto provoca un mal posicionamiento, y al utilizar solo distancias euclídeas, los landmarks previamente registrados se añaden como nuevas visualizaciones, es decir, como nuevos landmarks al sistema (Figura 33b). Esto resulta en una cantidad excesiva de datos que afecta al rendimiento del módulo. El ruido añadido por los retrasos retroalimenta este problema, empeorando cada vez más la asociación de landmarks hasta que el sistema colapsa.

Por otro lado, la divergencia de partículas, común en los sistemas basados en filtros de partículas, hace que una vez el sistema se ha desviado demasiado del ground truth, sea muy difícil que pueda volver a ajustarse.

Para abordar esta problemática, se han seguido dos vías principales de mejora: mejora del módulo de asociación y uso de un Filtro de Kalman.

En primera instancia, se pretende mejorar el módulo de asociación. Esta mejora se llevará a cabo por dos frentes. El primero consiste en modificar la detección de conos y la fusión del sistema de percepción, con el fin de realizar un tracking visual que asigne IDs a los conos para luego utilizar esos IDs locales en nuestro módulo de asociación, facilitando así la identificación de landmarks. Esto se aplicará en un ámbito local. Para el ámbito global e identificar nuevos landmarks, se pretende utilizar distancias de Mahalanobis en lugar de distancias euclídeas, ya que además de las coordenadas x e y, la distancia de Mahalanobis tiene en cuenta la varianza de las variables aleatorias.

En segunda instancia, se pretende utilizar un Extended Kalman Filter SLAM (EKF-SLAM), que tiene una implementación más sencilla y que también puede

proporcionar muy buenos resultados en el contexto en el que estamos trabajando. Además, de esta manera, se podrá contar con dos implementaciones distintas de sistemas SLAM, lo que permitirá elegir la más adecuada en cada contexto.

A continuación, se analizarán cada una de estas mejoras en sus correspondientes capítulos.

9. Tracking Visual

El tracking o seguimiento de objetos en video es una técnica que engloba todo un campo de estudio en el ámbito de la visión por ordenador. El tracking visual recoge un conjunto extenso de técnicas para determinar la posición de un objeto de interés en cada frame de un video, incluso si el objeto se encuentra parcial o totalmente oculto durante algunos frames. Concretamente, en el caso de uso de un sistema de visión por ordenador para un vehículo tipo Formula Student (FS), es de principal interés el Multi Object Tracking (MOT) debido a que se es necesario llevar el seguimiento de varios conos al mismo tiempo.

El objetivo final al hacer uso de un sistema de tracking visual de conos es poder conseguir una asociación de landmarks robusta. Sin embargo, para el sistema SLAM también es crucial que no existan falsas detecciones de colores de conos, ya que si se añade un cono mal clasificado al mapa, sería muy complejo eliminarlo, lo cual se escapa del alcance de este TFG. Es por ello, que las técnicas de tracking visual a considerar son técnicas basadas en detección, en las que primero se identifican los objetos con un clasificador visual y luego se emplean métodos de asociación locales.

Por ello, en el contexto del conjunto de percepción del sistema autónomo (definido en la Sección 2.4.2), será necesario realizar modificaciones en los subconjuntos de la cámara y fusión entre cámara y LiDAR.

9.1. Rediseño del módulo de detección visual

Hasta este momento y cómo se ha introducido en la Sección 2.4.2.1, la detección visual de la cámara está basada en YOLOv5. Sin embargo, se ha decidido actualizar a la versión YOLOv8 del modelo, ya que presenta mejores carácterísticas en cuanto al rendimiento (Figura 34) y, reentrenando el modelo con un conjunto más extenso de datos, se espera reducir el número de falsos positivos. Sobre todo entre los conos grandes naranjas y los conos naranjas normales (Figura 17), que es un factor crítico para detectar el «loop closure» del circuito. Además, YOLOv8 proporciona herramientas que facilitan enormemente la implementación del módulo de tracking y, al ser la versión del modelo que actualmente tiene soporte y una comunidad activa, cuenta con una muy buena documentación.

A la hora de trabajar con modelos de inteligencia artificial y «deep learning», resultan de gran importancia las conocidas como métricas de bonanza: criterios o medidas utilizadas para evaluar la calidad, efectividad y eficiencia de un modelo

de aprendizaje automático. Estas métricas proporcionan una manera cuantitativa de determinar qué tan bien está funcionando un modelo y, por ende, son esenciales para comparar dierentes modelos y optimizarlos. En el ámbito del entrenamiento de modelos de detección visual, son comunes el mAP o mean Average Precision, el Recall o la curva F1-score. Por otro lado, en el ámbito del tracking son comunes métricas como MOTA, HOTA o IDF1. Estas métricas se explicarán más adelante con mayor detalle.

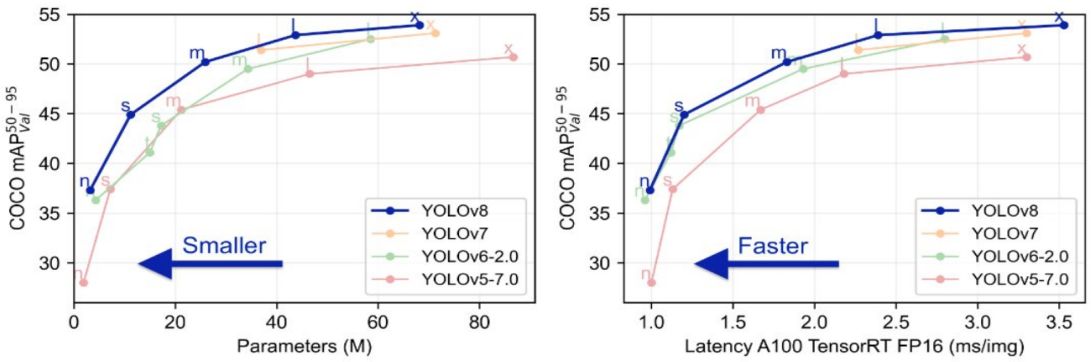


Figura 34: Comparación de rendimiento de YOLO

Los ejes horizontales indican el número de parámetros (pesos ajustables de las neuronas) y la latencia respectivamente y el vertical es mAP.

En cuanto al sistema de tracking MOT, se han considerado varios trackers para la integración con YOLO, como Deep OC-SORT [29], StrongSORT [28], BoxMOT [30], y otros tipos de trackers conocidos por su precisión y robustez en aplicaciones de detección de objetos. Después de evaluar las opciones, finalmente se ha decidido emplear ByteTrack [31]. Esta elección se ha tomado debido a su sencillez y facilidad de implementación, lo que facilita enormemente su integración y manejo. Además, ByteTrack ha demostrado proporcionar buenos resultados en términos de precisión y eficiencia, haciendo que sea una opción óptima para nuestro sistema de tracking visual.

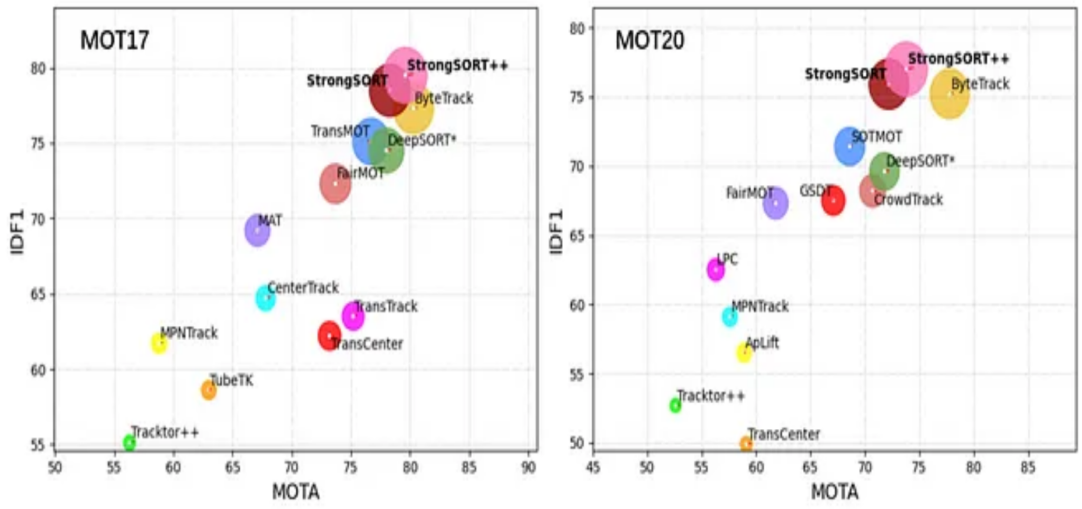


Figura 35: Comparación de trackers sobre los datasets MOT17 y MOT20

El eje horizontal es MOTA, el vertical IDF1 y el radio de los circulos HOTA.

Con todo ello, se ha rediseñado completamente el módulo de detección de conos con la cámara, resultando en el diagrama de flujo mostrado en la Figura 36. Como se ha comentado, se distinguen dos etapas muy marcadas: la detección de *bounding boxes* [2] en las imágenes de la cámara y el seguimiento de dichos *bounding boxes* con su correspondiente asignación de IDs. Por lo tanto, es importante abordar las etapas de entrenamiento de YOLOv8 y el tracking con ByteTrack. Además, esta implementación se ha llevado a cabo siguiendo un paradigma de orientación a objetos (Figura 54, ver anexo).

[2] Un *bounding box* es un cuadro delimitador empleado para describir la ubicación y tamaño de un objeto en una imágen

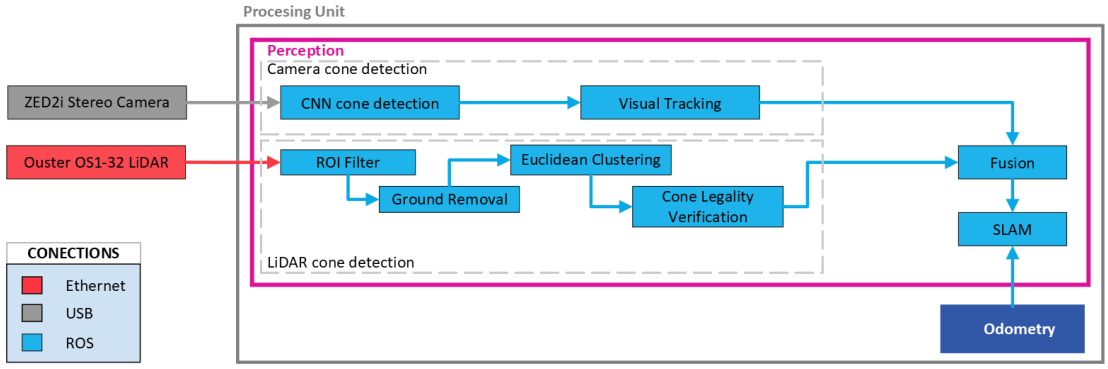


Figura 36: Diagrama de flujo final del conjunto de percepción

9.1.1. Entrenamiento YOLOv8

YOLOv8 cuenta con varios tamaños de modelo que definen arquitecturas con más o menos número de capas, como YOLOv8s, YOLOv8m o YOLOv8l. La elección del tamaño del modelo depende del equilibrio entre el rendimiento de la detección y la fiabilidad deseada. En este caso, se busca una confianza en detecciones de como mínimo un 75% y una frecuencia de operación superior a $10Hz$.

Es necesario reentrenar los modelos con datos para reconocer y clasificar los conos que forman el circuito. Es por ello que es de vital importancia contar con un dataset de imágenes de conos clasificadas con el que ajustar los modelos para realizar dicha tarea. El dataset escogido es FSOCO [9], una base de datos de conos clasificados según la normativa de Formula Student que cuenta con la participación de numerosos equipos de todo el mundo. Por ello, al contar con imágenes de conos desde distintos ángulos y entornos de iluminación, facilitará evitar casos de *overfitting* o sobre ajuste del modelo a los datos de entrenamiento.

Figura 37: Imagen clasificada por FSB para FSOCO

El proceso de entrenamiento supervisado involucra varias etapas cruciales para la optimización del modelo. Inicialmente, el dataset se divide en tres conjuntos distintos: entrenamiento, validación y test. La fase de entrenamiento es iterativa y se lleva a cabo a lo largo de un número predefinido de épocas o *epochs*.

Cada época comienza ajustando los pesos del modelo utilizando los datos del conjunto de entrenamiento. Este conjunto se subdivide en lotes (*batches*) más pequeños, permitiendo actualizaciones más frecuentes y estables de los pesos. Durante cada lote, el modelo realiza predicciones y calcula el error o desviación de estas predicciones con respecto a los valores reales. Utilizando este error, el modelo ajusta sus pesos para minimizar la desviación.

Este proceso de ajuste continúa a lo largo de la época hasta que el error en el conjunto de entrenamiento disminuye por debajo de un cierto límite. Una vez finalizada una época, se pasa a la etapa de validación, en la cual el modelo se evalúa utilizando el conjunto de validación, un conjunto de datos que no ha sido visto por el modelo durante el entrenamiento. Esta evaluación proporciona una medida de la capacidad del modelo para generalizar a datos no vistos.

Con base en los resultados de la validación, se pueden ajustar los hiperparámetros del modelo para mejorar su rendimiento en las siguientes épocas y evitar el mencionado *overfitting*. Este ciclo de entrenamiento y validación se repite hasta que el modelo alcanza un rendimiento satisfactorio o se ha alcanzado el límite predefinido de número de *epochs*. Finalmente, el conjunto de test se utiliza para evaluar el rendimiento final del modelo, proporcionando una medida objetiva de su capacidad para generalizar a nuevos datos.

El dataset de FSOCO cuenta con 11572 imágenes anotadas con *bounding boxes* en entornos de competición FS. Sin embargo, hay que realizar una etapa de preprocesado de las imágenes para descartar aquellas repetidas o de baja calidad. Además, es interesante aplicar técnicas de aumento de datos para aportar variabilidad y conseguir un dataset con un gran número de imágenes. Es por ello que, para trabajar con el dataset de FSOCO, se ha utilizado la plataforma de Roboflow. Roboflow agiliza el proceso de preparación de conjuntos de datos, facilitando el preprocesamiento y la anotación de datos, la gestión de versiones y la exportación de conjuntos de datos en formatos compatibles con marcos populares como YOLOv8. La división del dataset se define desde esta plataforma y se ha destinado un 75% para el conjunto de entrenamiento; un 15% para la validación y un 10% para el testing.

Los mejores resultados se han conseguido con el modelo YOLOv8m, 50 *epochs*, un *batch size* de 16 imágenes y un tamaño de imágen de 800 píxeles.Estos resultados se reflejan en la siguiente matriz de confusión y la curva F1-Confidence. En la

matriz de confusión se representa el ratio de acierto del modelo, mientras que en la gráfica de la curva se relacionan los niveles de confianza de las predicciones con el correspondiente puntaje F1. Así, considerado un valor de 0,75 F1-Score con una confianza de 0,7 y teniendo en cuenta que el modelo no suele confundir los tipos de conos, estos resultados son más que aceptables para el caso de uso descrito.

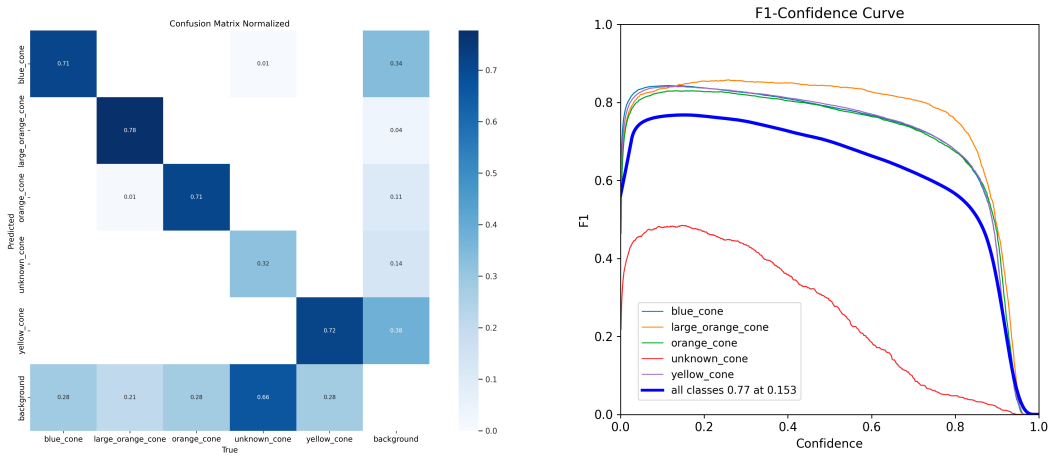


Figura 38: Métricas de bonanza del modelo YOLOv8 entrenado

9.1.2. Tracking con ByteTrack

ByteTrack es un MOT tracker para detecciones de video que tiene una implementación sencilla y aporta muy buenos resultados. Internamente hace uso de un filtro de Kalman y técnicas de interpolación para estimar las posiciones de los *tracklets*, o detecciones que tienen seguimiento, durante sucesivos frames. Esto permite asignar los mismos identificadores a objetos que han sido ocluidos durante algunos frames, permitiendo una asociación de IDs locales robusta.

Para implementar ByteTrack con los *bounding boxes* inferidos por el modelo YOLOv8 entrenado, se ha empleado la librería «Supervision». Esta es una librería que cuenta con herramientas y facilidades para trabajar con modelos de visión por ordenador. De esta manera, conseguimos una implementación sencilla y funcional.

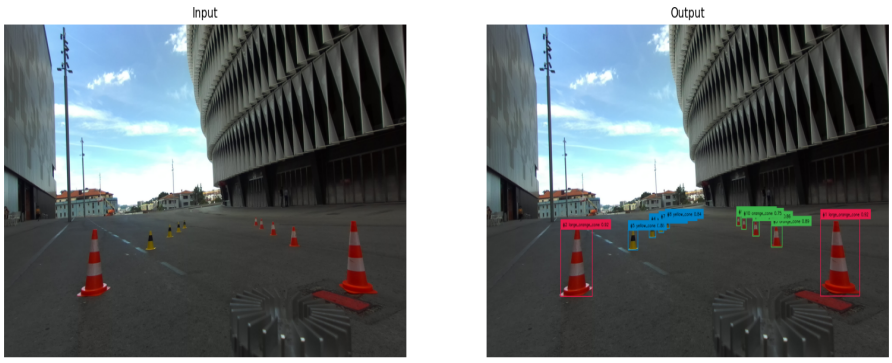


Figura 39: Inferencia y tracking de conos

9.2. Adaptación del módulo de fusión

Al igual que el módulo de la cámara, el subconjunto de la fusión entre cámara y LiDAR también tiene que ser modificado para aceptar los IDs del tracking visual. Además, debido a las características del LiDAR (Sección 14), el módulo de detección de conos en base al pointcloud sólo es capaz de visualizar objetos hasta una distancia máxima de $7m$, mientras que la cámara visualiza mucho más. Hasta ahora, sólo se tienen en cuenta las visualizaciones que se producen por la cámara y el LiDAR al mismo tiempo, descartando las demás. Es por esta razón, que se plantea el uso de un nuevo tipo de mensaje de ROS como salida del módulo de fusión: *ConeWithCovariancePlus*

```
1 // 2D cone locations with id, covariance and probabilities
2
3 int32 id
4 float64 blue_prob
5 float64 yellow_prob
6 float64 orange_prob
7 float64 big_orange_prob
8 float64 unknown_prob
9 geometry_msgs/Point point
10 float64[4] covariance
```

Listing 1: /fsb/msgs/ConeWithCovariancePlus

De esta manera, el objetivo del módulo de fusión es devolver los conos clasificados con su correspondiente ID y una covarianza que indique la fiabilidad de la detección en las coordenadas x e y. La coordenada x determina la profundidad en el plano y la distancia en el eje desde los sensores al cono, mientras que la coordenada y

representa la desviación lateral del cono con respecto al vehículo. Así, podemos distinguir un caso genérico y dos casos particulares. En el caso genérico, tanto la cámara como el LiDAR detectan un mismo cono. En este caso, la salida del módulo de fusión será una detección de cono, con su color e ID y una covarianza baja, ya que el LiDAR es muy fiable. Un caso particular se produciría cuando la cámara detecta un cono que el LiDAR no. En este caso, se produce el mismo output, pero con una covarianza alta. El último caso particular se produce cuando el LiDAR detecta un cono que la cámara no, en cuyo caso la detección queda descartada debido a que posiblemente se trate de ruido.

El cálculo de las covarianzas se realiza en función de las distancias de las detecciones al vehículo y de unos factores de ponderación específicos para la cámara y el LiDAR. Así, la distancia al objeto es directamente proporcional a la varianza de la observación. Estos factores de ponderación son hiperparámetros del sistema que deben ser ajustados adecuadamente con los sensores reales.

Finalmente, el módulo de fusión proporciona la salida de datos necesaria para alimentar el módulo *EKF-SLAM* detallado a continuación. Dicha salida de datos la podemos visualizar en la siguiente Figura:

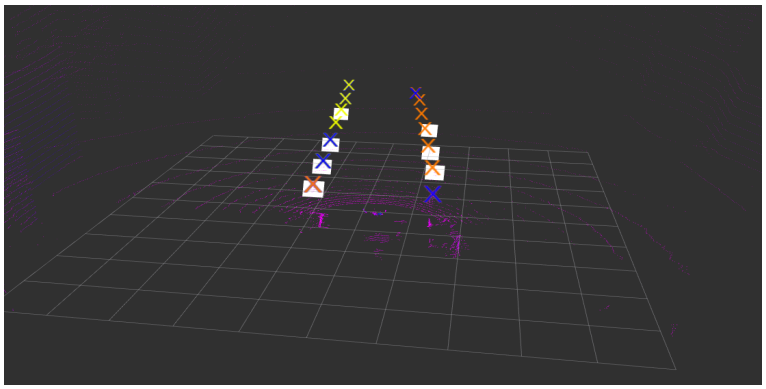


Figura 40: Salida del módulo de fusión adaptado

En cuadrados blandos las detecciones del LiDAR; en markers de colores las detecciones de la cámara y en markers azules el output de fusión. Fijarse en que las detecciones de fusión aparecen en todos los markers aunque estén ocluidos por otros markers.

10. EKF SLAM

Como se ha mencionado, se ha tomado la decisión de implementar un sistema Extended Kalman Filter SLAM como una segunda aproximación al sistema SLAM. Con este sistema, se espera crear un segundo módulo de SLAM que solucione los problemas descritos anteriormente para realizar las tareas de localización y mapeo en un vehículo tipo Formula Student. Además de conseguir un sistema funcional, esta implementación se enfoca en el uso de los *Filtros de Kalman*, un algoritmo que ha sido mencionado en múltiples ocasiones a lo largo de este TFG y que es ampliamente utilizado en diversos contextos de la robótica en general y de la conducción automatizada en particular.

Partiendo del contexto de los robots probabilísticos, nos encontramos con los Kalman Filter (KF). El KF es un algoritmo recursivo que, al igual que los filtros de partículas, cuenta con una etapa de predicción y otra etapa de corrección o actualización con observaciones. El KF asume que el ruido en el sistema y en las mediciones es gaussiano, y que las dinámicas del sistema son lineales. La representación del estado se basa en la media y la covarianza del mismo y este algoritmo se encarga de evaluar la estimación que minimiza el error cuadrático medio.

Sin embargo, y como se mencionó en el apartado de predicción del filtro de partículas 8.1, el caso de uso en un vehículo Formula Student es de un sistema no lineal. Es por ello que las implementaciones de un SLAM basado en un KF no se adecúan a la aplicación final. Una aproximación para resolver el problema en el marco de los filtros lineales, es el EKF. El EKF implementa un filtro de Kalman para una dinámica del sistema que resulta de la linearización de la dinámica original no lineal [32].

Durante la implementación del EKF-SLAM, se han considerado las mismas simplificaciones del entorno que en el *FastSLAM* (Sección 8). Por esta razón y teniendo en cuenta la representación del estado llevada a cabo por el KF, diferenciamos las dos matrices principales que compondrán nuestro sistema SLAM: el vector de estado y la matriz de covarianzas:

$$X = \begin{bmatrix} x \\ y \\ \theta \\ lx_1 \\ ly_1 \\ \vdots \\ lx_n \\ ly_n \end{bmatrix} \quad \Sigma = \begin{bmatrix} \sigma_x & \sigma_{xy} & \sigma_{x\theta} & \sigma_{xlx_1} & \sigma_{xly_1} & \cdots & \sigma_{xlx_n} & \sigma_{xly_n} \\ \sigma_{xy} & \sigma_y & \sigma_{y\theta} & \sigma_{ylx_1} & \sigma_{yly_1} & \cdots & \sigma_{ylx_n} & \sigma_{yly_n} \\ \sigma_{x\theta} & \sigma_{y\theta} & \sigma_\theta & \sigma_{\theta lx_1} & \sigma_{\theta ly_1} & \cdots & \sigma_{\theta lx_n} & \sigma_{\theta ly_n} \\ \sigma_{xlx_1} & \sigma_{ylx_1} & \sigma_{\theta lx_1} & \sigma_{lx_1} & \sigma_{lx_1ly_1} & \cdots & \sigma_{lx_1lx_n} & \sigma_{lx_1ly_n} \\ \sigma_{xly_1} & \sigma_{yly_1} & \sigma_{\theta ly_1} & \sigma_{lx_1ly_1} & \sigma_{ly_1} & \cdots & \sigma_{ly_1lx_n} & \sigma_{ly_1ly_n} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \sigma_{xlx_n} & \sigma_{ylx_n} & \sigma_{\theta lx_n} & \sigma_{lx_1lx_n} & \sigma_{ly_1lx_n} & \cdots & \sigma_{lx_n} & \sigma_{lx_nly_n} \\ \sigma_{xly_n} & \sigma_{yly_n} & \sigma_{\theta ly_n} & \sigma_{lx_1ly_n} & \sigma_{ly_1ly_n} & \cdots & \sigma_{lx_nly_n} & \sigma_{ly_n} \end{bmatrix} \tag{7}$$

La matriz $\mathbf{X}$ representa las medias de la posición del vehículo (coordenadas x, y y orientación) y de todos los landmarks identificados en el entorno. Esta matriz se actualiza continuamente conforme el vehículo se mueve y se detectan nuevos landmarks. Por otro lado, la matriz $\mathbf{\Sigma}$ contiene las covarianzas de las variables aleatorias y las relaciones entre ellas, proporcionando una medida de la incertidumbre asociada a cada estimación y a las correlaciones entre las distintas variables. A medida que se visualizan nuevos landmarks, tanto la matriz $\mathbf{X}$ como la matriz $\mathbf{\Sigma}$ aumentan de tamaño para incorporar la nueva información, manteniendo así una representación completa y actualizada del entorno y del estado del sistema.

A la hora de inicializar estas matrices o añadir nuevos elementos, se realiza un copiado de toda la matriz anterior y se añaden las nuevas variables como variables aleatorias independientes, es decir, las relaciones de covarianza entre las nuevas variables y las anteriores se establecen en 0 y se les asigna un valor de covarianza fijo para las nuevas adiciones.

En resumen, el proceso de estimación del estado del robot llevado a cabo por el EKF es similar al de un filtro de partículas y se puede ilustrar con la Figura 41. Sin embargo, presentan diferencias clave en su funcionamiento. El EKF utiliza una aproximación lineal del modelo de movimiento y observación para propagar las incertidumbres. Este enfoque permite al EKF ser más eficiente en términos de computación para problemas de pequeña escala y es adecuado para manejar distribuciones de probabilidad que pueden ser aproximadas como gaussianas, permitiendo una estimación precisa del estado del vehículo y sus landmarks en el contexto de SLAM.

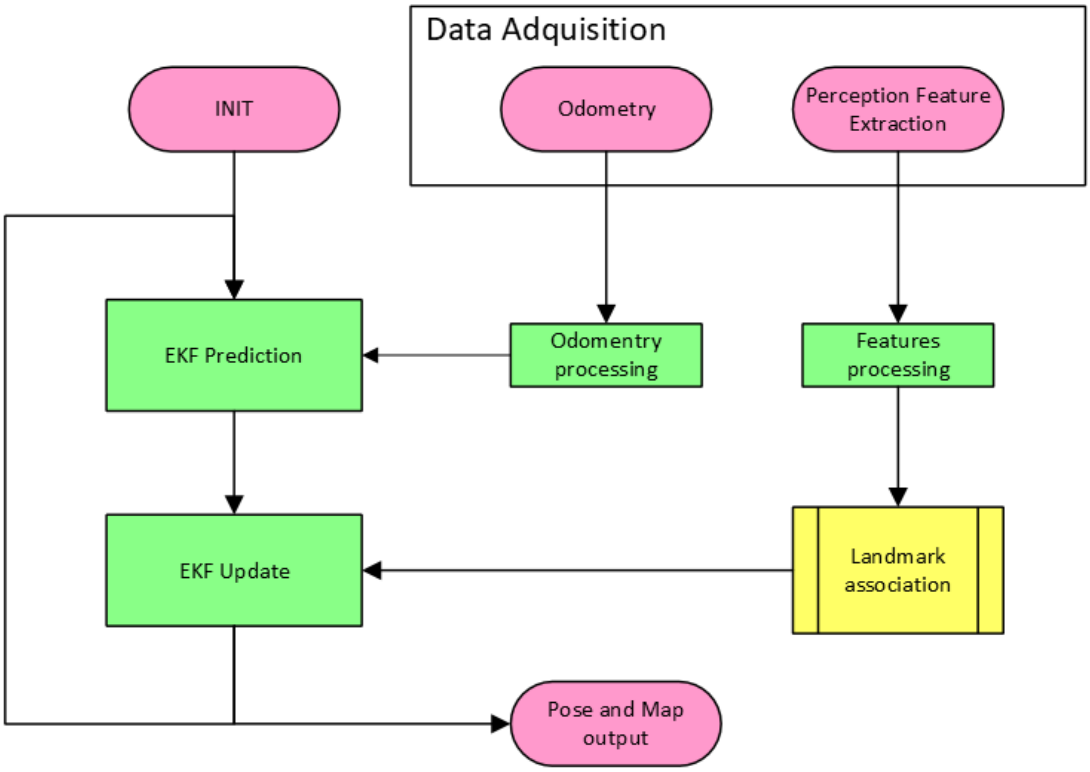


Figura 41: Diagrama de flujo del EKF SLAM

10.1. Predicción del nuevo estado

Para realizar la predicción del nuevo estado hay que hacer uso de un modelo cinemático del vehículo que tenga en cuenta la entrada de odometría u_k para calcular el nuevo estado x_k a partir del anterior x_{k-1}. Este modelo se ha definido como f_k y es mismo que se ha presentado anteriormente en la ecuación 2. Sin embargo, a diferencia del filtro de partículas, el EKF computa en cada iteración la media y la covarianza del estado del vehículo. El cálculo de la media se realiza utilizando el modelo cinemático f_k mencionado, pero para calcular la covarianza hay que hacer uso de la matriz de covarianzas anterior Σ_{k-1} y la matriz de ruido estimado Q para la entrada de odometria u_k. La matriz de ruido es un hiperparámetro del sistema que tendrá que ser ajustada en relación a los sensores utilizados.

En definitiva, la etapa de predicción se puede sintetizar con la siguiente expresión [32]:

$$\begin{aligned} &\text{Estimación } \textit{a priori} \text{ del estado} && \hat{x}_k = f_k(x_{k-1}, u_k) \\ &\text{Estimación } \textit{a priori} \text{ de la covarianza} && \hat{\Sigma}_k = F_x \Sigma_{k-1} F_x^T + F_u Q F_u^T \end{aligned} \tag{8}$$

Como se ha comentado, el KF solo puede funcionar con sistemas lineales. Es por esta razón que para realizar el cálculo de la covarianza es necesario linealizar el modelo cinemático f_k. Para ello, se toma la derivada parcial de f_k con respecto al estado del vehículo x_k y u_k, calculando las matrices jacobianas F_k y F_u. Estas matrices jacobianas se utilizan en el EKF para propagar las incertidumbres del estado en el proceso de predicción. Concretamente, F_x relaciona el estado del vehículo con su covarianza y F_u se encarga de computar la covarianza de las acciones de control teniendo en cuenta la matriz de ruido Q mencionada.

$$\begin{aligned} F_x &= \frac{\partial f}{\partial x_{k-1}} = \begin{bmatrix} 1 & 0 & -u_{fwd}\sin(\theta_{k-1}) \\ 0 & 1 & u_{fwd}\cos(\theta_{k-1}) \\ 0 & 0 & 1 \end{bmatrix} \\ F_u &= \frac{\partial f}{\partial u_k} = \begin{bmatrix} \cos(\theta_{k-1}) & 0 \\ \sin(\theta_{k-1}) & 0 \\ 0 & 1 \end{bmatrix} \end{aligned} \tag{9}$$

Finalmente, considerando la ecuación 8, se ha implementado la etapa de *dead reckoning* del EKF-SLAM que se muestra en la Figura 42. En dicha Figura se puede observar el *groud truth* del vehículo autónomo, el estado estimado por la predicción del EKF y las elipses de covarianza que demuestran como la incertidumbre aumenta al no contar con observaciones para corregir el estado.

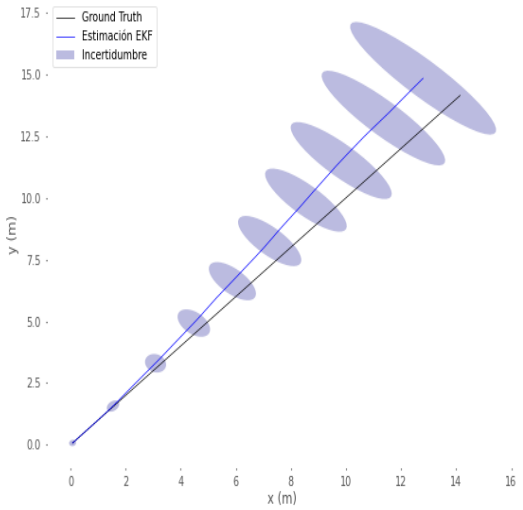


Figura 42: EKF dead reckoning

10.2. Asociación de observaciones

El módulo de asociación de observaciones se ha rediseñado completamente para incluir los IDs locales provenientes del tracking visual implementado (Sección 9).Así, los datos de entrada del módulo de asociación consisten en las detecciones de conos proporcionadas por el módulo de fusión adaptado (Sección 9.2). Sin embargo, estas detecciones tienen una etapa previa de preprocesado en la cual se desestiman aquellas observaciones con una incertidumbre muy elevada y se realiza la transformación de las detecciones al frame global del mapa. De esta manera, llegamos al submódulo de asociacion de observaciones del EKF-SLAM, encargado de asignar los identificadores a las observaciones para reconocer landmarks anteriormente visualizados o determinar si se tratan de nuevas observaciones. Este proceso se resume con el siguiente diagrama de flujo:

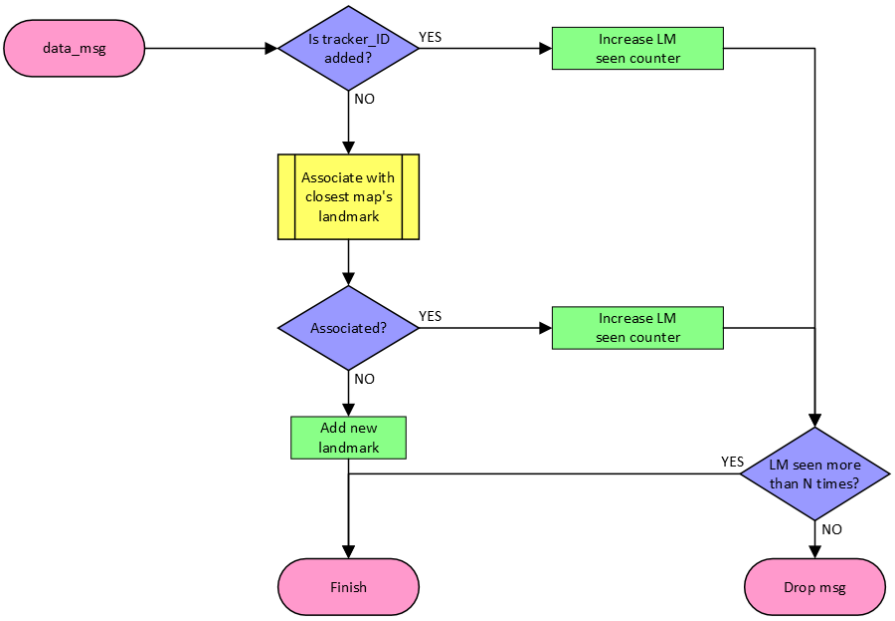


Figura 43: Diagrama de flujo de la asociación

Para implementar este módulo, se ha utilizado un sistema de tablas en el cual se relacionan los identificadores locales del tracking con los IDs o posiciones de los landmarks en la matriz de estado X. De esta manera, pueden existir varios *tracking IDs* que apunten al mismo landmark, pero no puede existir un *tracking ID* que apunte a dos landmarks distintos. Además, se ha utilizado una condición o *Validation Gate* [33], en la cual sólo se considerarán landmarks válidos a aquellos que hayan sido visualizados más de un cierto número de veces. Por ello, cada vez que se visualiza un landmark, se incrementa el número de veces que se ha visualizado. Esta es una técnica muy común en el contexto de asociación utilizando el método de Nearest Neighbor Data Association (NNDA) [27].

En el caso de que el *tracking ID* no haya sido previamente añadido a la tabla, se realiza una búsqueda en el mapa del landmark más cercano empleando la distancia de Mahalanobis (Figura 55, ver anexo) y, si se supera un cierto umbral de distancia mínima, se considera que la observación corresponde a un landmark registrado. Este hecho se refleja incluyendo en la tabla el *tracking ID* apuntando al landmark asignado. Sin embargo, si no se encuentra ninguna asociacion válida para dicha observación, se considera como un nuevo landmark y se añade incrementando el número total de elementos en el mapa.

Luego de realizar la etapa de asociacion, se ha añadido un postprocesado para identificar el *loop closure* marcado por los conos grandes naranjas que indican el

inicio del circuito. Esta identificación se realiza considerando la distancia recorrida por el vehículo desde la última vuelta y la visualización de dichos conos.

10.3. Corrección con observaciones

Una vez se han realizado las etapas de predicción con la odometria y la adquisición y asociacion de datos de observaciones, se procede con la etapa de corrección. Con esta etapa, se pretende ajustar la distribución de probabilidad del estado del vehículo con el fin de adecuarse a las observaciones dadas y, por lo tanto, conseguir una estimación más fiable de su posición y los landmarks.

El proceso de corrección comienza realizando predicciones de las posiciones y covarianzas de los landmarks para después, contrastarlas con las observaciones reales de los mismos. Para ello, se emplea un proceso iterativo en el cual se realizarán tantas correcciones como observaciones de landmarks se hayan hecho en el instante k. Así, por cada landmark z_k^i visualizado, se realizará una predicción de su estado haciendo uso del modelo de observaciones h_k introducido anteriormente (ecuación 4). Sin embargo, de la misma manera que ha sido necesario linealizar el modelo cinemático, como el modelo de observaciones no es lineal también hay que obtener la matriz jacobiana H_x para poder realizar el cálculo de las covarianzas de las estimaciones:

$$H_k = \frac{\partial h}{\partial x}\Big|_{\hat{x}_k} = \begin{array}{c} \begin{matrix} x & y & \theta & lx_1 & ly_1 & lx_i & ly_i & lx_n & ly_n & \end{matrix} \\ \begin{bmatrix} \frac{\hat{x}_x - z_x^i}{r} & \frac{\hat{x}_y - z_y^i}{r} & 0 & 0 & 0 & -\frac{\hat{x}_x - z_x^i}{r} & -\frac{\hat{x}_y - z_y^i}{r} & 0 & 0 & \cdots \\ \frac{z_y^i - \hat{x}_y}{r} & \frac{z_x^i - \hat{x}_x}{r} & -1 & 0 & 0 & -\frac{z_y^i - \hat{x}_y}{r} & -\frac{z_x^i - \hat{x}_x}{r} & 0 & 0 & \cdots \end{bmatrix} \end{array}$$

$$\text{donde } r = \sqrt{(z_x^i - \hat{x}_x)^2 + (z_y^i - \hat{x}_y)^2} \tag{10}$$

Ya contando con la posición del landmark i estimada por el modelo de observación, que se denota como $\hat{z}_k^i$, se puede proceder con el cálculo del error o innovación. La innovación es la diferencia entre la estimación y la observación real y cuenta con una matriz de covarianza S_k asociada. Dicha matriz se calcula haciendo uso del modelo de observación linealizado H_k, de la matriz de covarianza estimada previamente en la etapa de predicciones $\hat{\Sigma}_k$ y de una matriz de ruido R. Esta última matriz R, que contiene las covarianzas del ruido asociado a la medición z_k^i, proviene del módulo de fusión adaptado anteriormente (sección 9.2).

Finalmente, se realiza el cálculo de la *Ganancia de Kalman*. Esta se trata de una matriz que determina cuánto se deben ajustar las estimaciones del estado y la

covarianza basándose en la innovación o error calculado [33]. Con esta ganancia se termina ajustando el estado final del vehículo y determinando las covarianzas asociadas a cada una de las variables aleatorias del sistema. El proceso completo de corrección con observaciones se puede resumir con el siguiente algoritmo:

$$\begin{aligned}
&\text{Cálculo de la innovación} && y_k = z_k^i - \hat{z}_k^i \\
&\text{Cálculo de la covarianza de la innovación} && S_k = H_k \hat{\Sigma}_k H_k^T + R \\
&\text{Cálculo de la Ganancia Kalman} && K_k = \hat{\Sigma}_k H_k^T S_k^{-1} \\
&\text{Actualización del estado} && x_k = \hat{x}_k + K_k y_k \\
&\text{Actualización de la covarianza} && \Sigma_k = (I - K_k H_k)\hat{\Sigma}_k
\end{aligned} \tag{11}$$

No hay que olvidar que este se trata de un proceso iterativo que se realizará por cada una de las observaciones dadas.

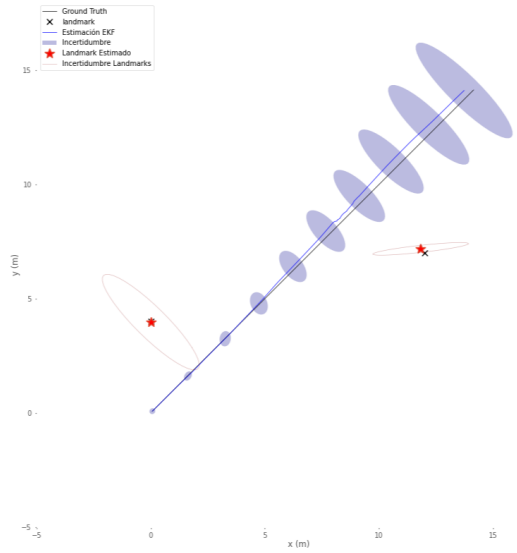


Figura 44: Actualización del estado del EKF SLAM

10.4. Validación en simulación

La validación del sistema EKF-SLAM en el simulador *EUFS Sim* comienza con la creación de un nodo de ROS2 encargado de comunicarse con el simulador pa-

ra proporcionar los datos de entrada necesarios al nodo del SLAM. Este nodo, denominado «data generator», tiene como función principal controlar el vehículo automático del simulador para que recorra el circuito, generando así un conjunto de datos de odometría y visualizaciones de conos.

Para facilitar esta tarea, se ha precargado el mapa del circuito en el nodo «data generator», permitiendo la generación de una trayectoria global inicial por el centro del carril. Posteriormente, se ha implementado un controlador de seguimiento de trayectorias basado en el algoritmo *pure pursuit*. Aunque el simulador *EUFS Sim* incluye opciones para simular la salida del módulo de fusión, carece de una opción para simular un sistema de tracking visual completo.

Por ello, con el fin de imitar la asignación de IDs locales del tracking, se ha incluido un sistema de generación de identificadores en el nodo descrito. Este sistema utiliza el mapa precargado del entorno y compara las observaciones con los landmarks del mapa. Si las distancias entre los landmarks del mapa y las observaciones son menores a un cierto umbral, se consideran como el mismo landmark y se les asigna un ID específico. Además, estos IDs tienen una probabilidad definida de cambiar mientras se mantiene la observación en frames de tiempo consecutivos y, en el caso de que un landmark no se visualize en un número determinado de frames de tiempo, se asignará un nuevo ID a la observación. Este enfoque permite replicar un caso de aplicación similar al real, facilitando la evaluación del funcionamiento del sistema EKF-SLAM implementado.

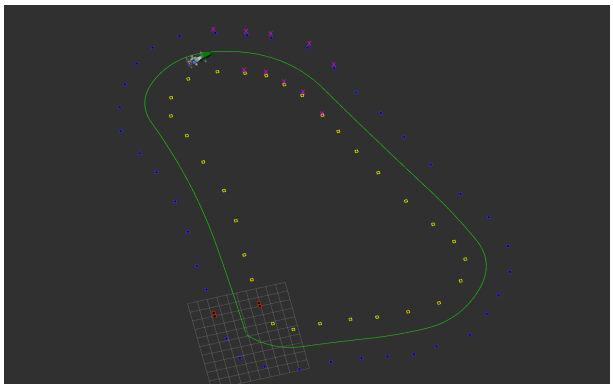


Figura 45: Nodo «data generator» controlando el vehículo

Además, es necesario implementar un sistema de procesamiento de datos para evaluar el rendimiento del EKF-SLAM. Este sistema se ha desarrollado en otro nodo de ROS2 llamado «metrics». Este nodo recopila datos tanto de la salida del SLAM como del estado del vehículo y del circuito del simulador. Con esta información,

se calcula el error cuadrático medio (*MSE*) para medir la precisión entre la localización real del vehículo y la estimada por el sistema SLAM. En la Figura 46 se pueden visualizar las comunicaciones entre todos los nodos mencionados.

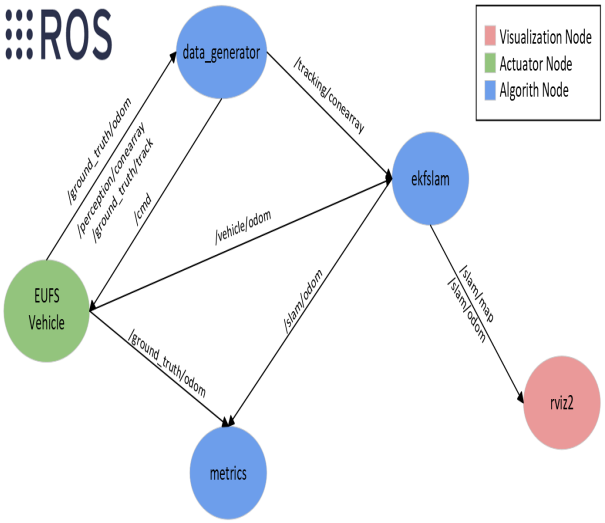


Figura 46: Diagrama de comunicaciones entre los nodos de ROS

Finalmente, tras un proceso de ajuste de las matrices de covarianzas del EKF-SLAM, se ha conseguido un sistema de localización y mapeo completamente funcional en simulación. El SLAM mantiene el error acotado durante sucesivas vueltas al circuito y, como se muestra en la Figura 47, es capaz de mapear todo el circuito con precisión. Sin embargo, los resultados obtenidos indican claramente que las situaciones de giros prolongados y cerrados representan un desafío crítico para el sistema, ya que en estos puntos el error tiende a incrementarse. A pesar de ello, este error se mantiene dentro de niveles aceptables y, durante las rectas o zonas con pocas curvas, el sistema tiende a corregir eficazmente la estimación de la posición.

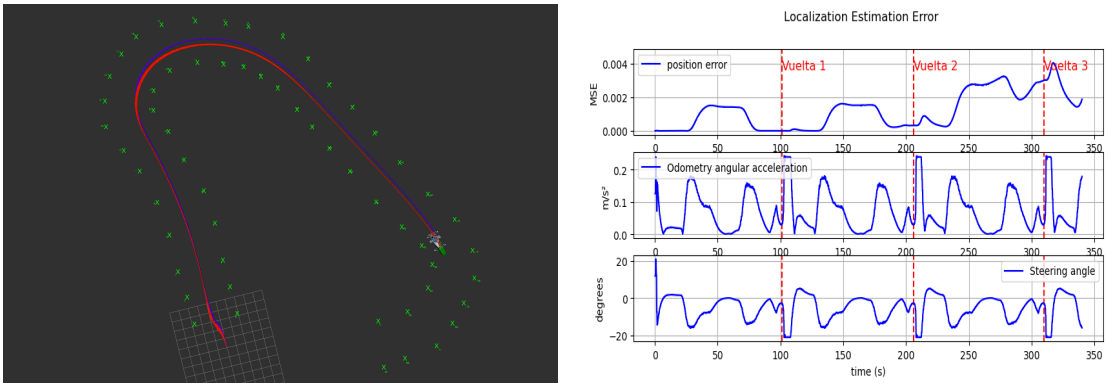


Figura 47: Simulación y métricas EKF SLAM

Otro factor a resaltar es la acumulación de errores en etapas de estimación prolongadas. Así, podemos observar cómo poco a poco el error aumenta, lo cual es un factor común en los sistemas SLAM. A pesar de ello, y como se ha mencionado, este error se mantiene en niveles aceptables y puede ser corregido tomando las curvas lentamente. Además, también cabe destacar la robustez del sistema ante acciones de control bruscas. Este hecho se puede observar en la Figura 47, donde al inicio de cada vuelta se produce un cambio repentino en la dirección del vehículo. Esto se debe a que el cierre de la trayectoria calculada previamente no está suavizado.

Considerando estos resultados, se puede dar por finalizada la etapa de validación en entornos de simulación, lo que da paso al siguiente reto: validar el sistema SLAM en entornos reales.

10.5. Validación en entornos reales

La validación del sistema SLAM implementado en entornos reales es una etapa crucial, ya que de ello dependerá el módulo final con el que se participe en la competición FS-AI. Además, el trabajo con datos reales supone la integración de todos los módulos del AS, lo que no solo demostrará posibles fallos y puntos de error en el sistema SLAM objeto de este TFG, sino también de todo el conjunto de software desarrollado.

La adquisición de datos para el módulo de percepción se lleva a cabo utilizando los sensores descritos en la sección 2.4.1. Sin embargo, para obtener los datos necesarios para el módulo de odometría (sección 2.4.2) y simular un entorno competitivo con una dinámica similar a la del ADS-DV, el grupo de trabajo *Driverless* (Sección 2.3) ha desarrollado un vehículo automatizado basado en un kart eléctrico durante esta temporada. Este vehículo, llamado Ready to Drive Alone (R2DA), cuenta con

una API implementada similar a la utilizada en la competición. Así, la placa de sensores, montada en la parte frontal del vehículo, y la API de R2DA proporcionan los datos necesarios para validar el SLAM implementado.

Figura 48: R2DA en pista

Unido a esto, también se ha desarrollado un sistema de toma de *logs* con el fin de crear una base de datos sobre la que lanzar los algoritmos en etapas de desarrollo para no estar limitados únicamente al simulador. Esta grabación de logs se realiza utilizando una herramienta conocida como *ros2 bag*. De esta forma se puede crear un «vídeo» de los datos proporcionados por cualquier nodo de ROS2. Esta herramienta es muy útil, pero como se ha mencionado, solo funciona para «grabar» publicaciones de mensajes de ROS2, hecho que impide registrar la salida sin procesar de los datos de la cámara. Esto se debe a que el nodo de detección visual (sección 2.4.2.1), carga el SDK de la ZED2i tomando los datos directamente. Además, el SDK sólo permite acceder a los datos de la cámara un único proceso, lo que impide crear un nodo de ROS2 externo para realizar la función de toma de logs. Por ello, la solución más asequible para minimizar las latencias ha sido grabar los datos de imágenes rgb y de profundidad en un fichero *SVO* desde el propio nodo de detección visual.

Finalmente, se han dispuesto dos tipos de pistas marcadas por conos: una pista recta imitando la disposición de la prueba dinámica *acceleration* (Figura 6a) y otra pista más exigente formada por una recta y una curva. En la siguiente Figura se puede apreciar cómo el sistema SLAM es capaz de mapear la pista y seguir el trazado del vehículo. Los conos se representan con una marca verde que tiene el identificador del landmark en la parte superior y la posición estimada del vehículo se representa con una traza azul.

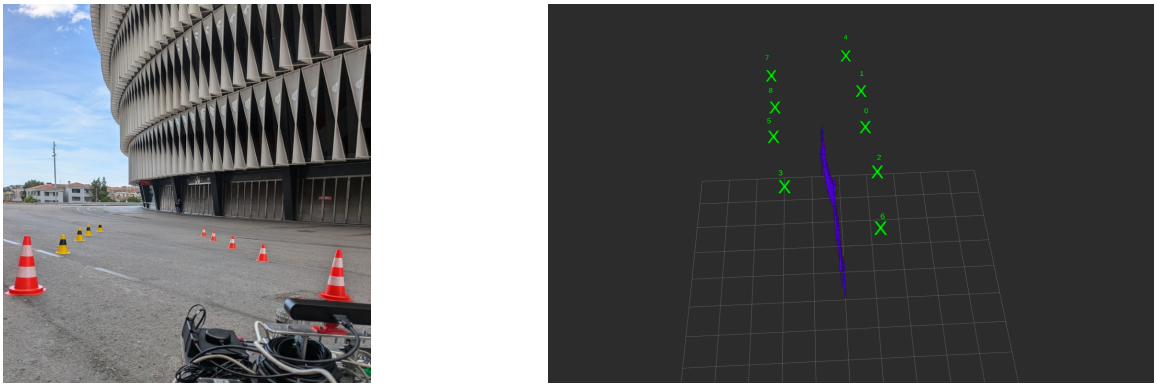


Figura 49: EKF SLAM en acceleration

En un entorno en el que el ángulo de giro no varía, es muy sencillo realizar el mapeo y localización del vehículo ya que solo hay que considerar su velocidad lineal. Sin embargo, como se vió en la etapa de simulación, resulta crítico evaluar el SLAM en entornos con curvas. Por ello y tras una ajuste de las covarianzas Q y R del EKF se ha conseguido validar este sistema en un entorno real con curvas:

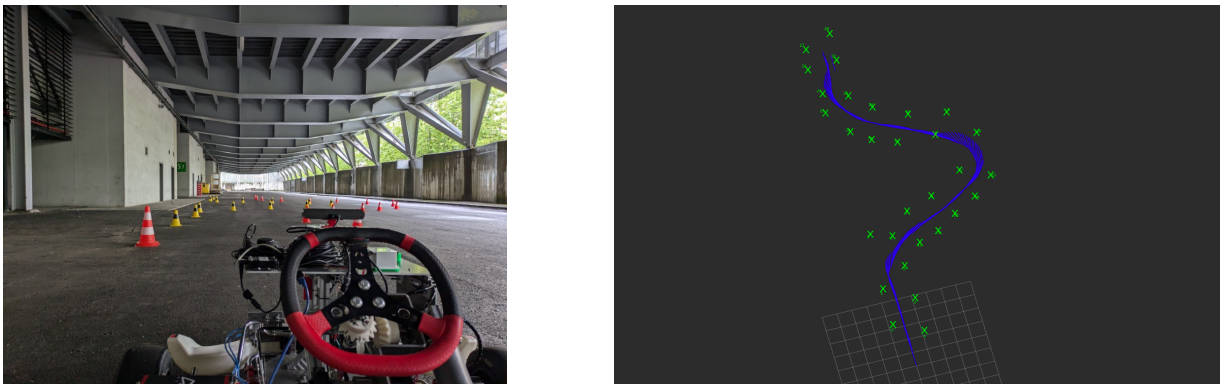


Figura 50: EKF SLAM en pista con curvas

11. Planificación del proyecto

La planificación de este proyecto se ha llevado a cabo teniendo en cuenta la duración de una temporada de Formula Student, dado que el presente TFG supone el desarrollo del módulo de SLAM que se llevará integrado con el resto de sistemas para la competición FS-AI en julio de 2024. Por una parte, en la etapa de diseños se toman todas las decisiones más relevantes y se desarrollan los diseños de todos los conjuntos. En la etapa de implementación, se programa el código de estos conjuntos previamente diseñados y se realizan *unit tests*, o pruebas aisladas, para garantizar el correcto funcionamiento de los módulos. Por último, todas las pruebas finales, integraciones y validaciones se realizan durante la etapa de testing.

Con el fin de coordinar el desarrollo de todos los conjuntos del sistema autónomo, en FSB se establecen varios hitos que sirven como puntos de control para todos los miembros. Los tres hitos principales del equipo se han adaptado para el sistema autónomo y son los siguientes:

1. **Fin del Diseño**. Marca el final de la etapa de diseños, dando comienzo a la etapa de implementación. En este punto, todos los diseños relacionados con el módulo de SLAM deberán de quedar cerrados y listos para empezar a implementar no solo este conjunto, si no todos los dependientes de él.
   - Fecha: 02/11/2023
   - Entregables:
     - Documento de diseño de SLAM
     - Instrucción de trabajo del LiDAR
     - Instrucción de trabajo de la cámara ZED
     - Instrucción de trabajo del simulador EUFS
2. **Fin de la Implementación**. Este hito establece el final de la implementación del software referente al conjunto de SLAM.
   - Fecha: 24/05/2024
   - Entregables:
     - Nodo de ROS2 para SLAM implementado
     - *Unit Tests* de SLAM
     - Simulación de SLAM funcional

3. **Fin del Proyecto**. Con este hito final, se da por finalizado el proyecto. Para este momento, deberá quedar validado el correcto funcionamiento de la totalidad del módulo de SLAM y su correcta integración con el conjunto de percepción y el resto de sistemas del AS.
    - Fecha: 05/07/2024
    - Entregables:
        - Conjunto de SLAM validado
        - Integración con todos los conjuntos del AS

Para facilitar el seguimiento del proyecto y la coordinación con el resto de desarrollos, estas tres etapas se han dividido en una serie de fases o tareas a realizar con sus correspondientes objetivos y tiempo a dedicar. La estimación de la duración del proyecto se ha realizado cuantificando el transcurso de cada tarea en días laborales, hecho que se refleja en la Figura 51 donde se puede observar el conjunto de la planificación temporal del proyecto. En cuanto a los recursos materiales necesarios, éstos han comprendido únicamente un ordenador, material de oficina y herramientas software como VS Code, Python, ROS2 y Gazebo, dejando los sensores y el vehículo automatizado para las últimas fases del desarrollo. En cuanto a los recursos humanos destinados a las tareas, sólo se ha contado con un Ingeniero Junior para realizarlas.

1. **Definición de requisitos y alcance (15 días)**. Esta primera etapa se centrará en estudiar las necesidades, requerimientos y puntos de mejora que tiene el AS para esta temporada.
2. **Análisis del estado del arte (15 días)**. Una vez se conocen las necesidades, es necesario realizar un estudio sobre las soluciones SLAM disponibles y seleccionar la mejor opción para el AS.
3. **Diseño del sistema software (13 días)**. Habiendo recogido información sobre las distintas opciones presentes y considerando los requerimientos del sistema se realizará todo el diseño del Software necesario. En esta fase se definirá el algoritmo, paradigma y flujograma del sistema de localización y mapeo simultáneos.
4. **Implementación del software (121 días o 4 meses)**. Una vez acabados los diseños, se tratará de conseguir una implementación funcional de cada uno de los subconjuntos relativos al módulo de SLAM. Se estudiarán distintas implementaciones con el fin de obtener un sistema robusto.
5. **Tests unitatios (25 días)**. En esta fase, se tratará de validar cada uno de los

submódulos con simulaciones y tests unitarios a fin de asegurar el correcto funcionamiento de la implementación realizada. Es en esta fase cuando se podrán obtener resultados previos que indiquen si el sistema implementado es prometedor o no.

6. **Integración (15 días)**. Contando con un sistema SLAM desarrollado y aparentemente funcional se introduce la etapa de integración. Esta es una de las fases más críticas del proyecto ya que se pondrá en funcionamiento con el resto de conjuntos y será necesario hacer modificaciones no solo al sistema desarrollado si no también a otros nodos del AS. Con esta etapa se pretende obtener un AS preparado para realizar validaciones en simulación y en entornos reales.

7. **Validación (15 días)**. Una vez se ha conseguido un AS con el módulo de SLAM integrado llega el momento de ponerlo a prueba alternando entre simulaciones completas y entornos reales. Con esta fase final del proyecto se pretende terminar de validar todo el desarrollo realizado.

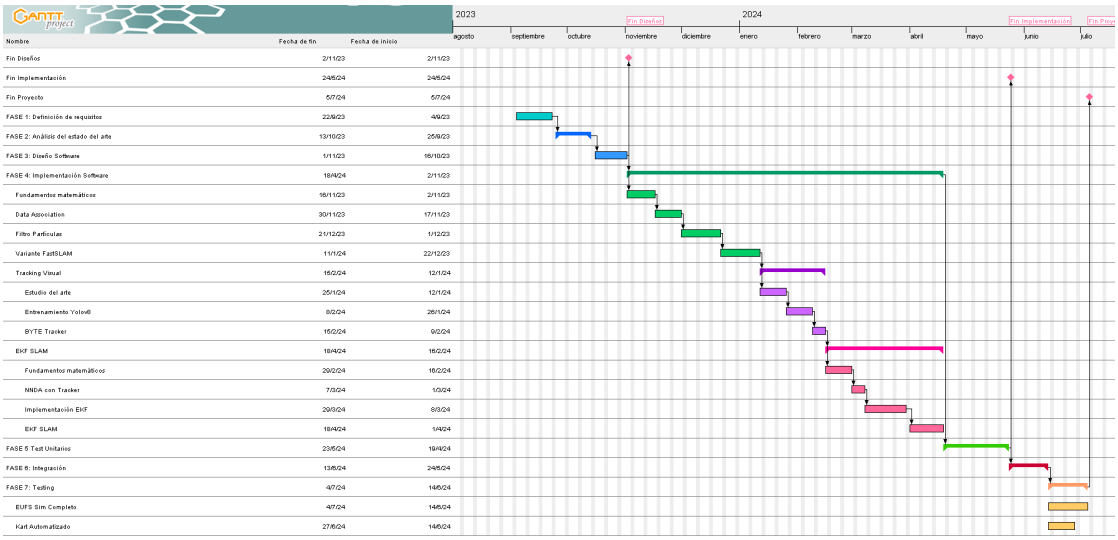


Figura 51: Diagrama de Gantt

12. Presupuesto

Uno de los puntos más importantes a la hora de llevar a cabo un proyecto es el coste económico que este va a tener. Detrás de este tipo de proyectos siempre habrá personas o entidades que querrán saber cuál es el capital necesario para poder desarrollar el sistema. Es por ello que, en este apartado se detallará el presupuesto necesario para poder desarrollar el sistema de percepción presentado en este trabajo.

Cabe destacar que el proyecto Formula Student Bizkaia es una entidad sin ánimo de lucro donde los miembros participan de forma voluntaria con el apoyo económico de los *sponsors* o patrocinadores. Por tanto, este proyecto ha podido llevarse a cabo con un reducido porcentaje del presupuesto que se va a presentar.

Antes de comenzar, es importante conocer qué tipos de costes existen:

- **Costes por material:** Engloba todos aquellos gastos por compras de sensores, chapas, tornillo, cables y consumibles necesarios para poder desarrollar el producto final.
- **Costes por Recursos Humanos:** Estos costes hacen referencia a las horas dedicadas al proyecto por trabajador.
- **Costes por herramientas, medios y recursos:** Dentro de este tipo entran todas las herramientas utilizadas para el desarrollo del trabajo como pueden ser las licencias del software empleado, el equipo de laboratorio o las herramientas de soldadura.
- **Costes indirectos:** Se trata de todos aquellos gastos de elementos generales no expresamente dedicados al proyecto, pero igualmente necesarios para su ejecución. Dentro de los costes indirectos se encuentran la luz de las instalaciones, el agua, las instalaciones, etc. Se establecerá como coste indirecto un 5% del total.
- **Costes por imprevistos:** En ocasiones ocurren eventos que inhabilitan parte de los componentes o dañan alguna herramienta, los gastos que supone el recambio de estos elementos es lo que engloba este tipo de costes. Se establece un coste del 10% en imprevistos.

12.1. Amortizaciones

Tal y como se ha mencionado anteriormente, el trabajo se ha llevado a cabo mediante el uso de diferentes herramientas software y equipo de laboratorio. Todo este material tiene un coste que no puede asignarse completamente al proyecto, ya que al terminar éste, podrá seguir utilizándose en futuros desarrollos. Para estimar este coste, se calculará la amortización de cada elemento. Se tomará como referencia que la duración del proyecto es de aproximadamente un año, tal y como se ha visto en la planificación del proyecto (sección 11).

Elemento	**Proveedor**	**Precio (€)**	**Vida útil**	**Usado**	**Total (€)**
Stealth 15 A13V	MSI	1.349,00	5 años	1 año	269,8
Dell Precision 5820	PcComponentes	3.299,00	5 años	2 años	1.319,6
Karbon 803	OnLogic	3.307,40	5 años	1 año	661,48
OS1 32	Ouster	5.529,00	4 años	2 años	2.764,5
ZED2i	Stereolabs	499,00	5 años	2 años	199,6
Owa450	Owasys	287,95	4 años	1 año	71,99
Kart eléctrico	Play&Drive	10.870,00	8 años	1 año	1.358,75
Microsoft 365	Microsoft	5,60	1 año	1 año	5,60
Coste Total					**6.651,32**

Tabla 4: Costes de las amortizaciones

12.2. Recursos Humanos

Teniendo en cuenta la planificación presentada en el apartado anterior y aplicando una jornada diaria de 3 horas. La duración del diseño, implementación y testing del modulo de planificación se ha calculado teniendo en cuenta todas la fases del proyecto que incumben a este conjunto. Sin tener en cuenta fines de semana ni festivos, el coste por la mano de obra de un ingeniero junior con titulación en «Ingeniería Informática de Gestión de Sistemas de Información» es el siguiente.

Concepto	**Tiempo**	**Coste/hora (€)**	**Precio Total (€)**
Ingeniero Informático	645h	35€/h	22.575€
Ingeniero Senior (Director del proyecto)	40h	60€/h	2.400€
Coste Total			**24.975€**

Tabla 5: Costes humanos

12.3. Costes totales

En la tabla que se presenta a continuación, se muestra el resultado total de los costes calculados en los apartados anteriores añadiendo los imprevistos y los costes indirectos.

Desarrollo	**Precio**
Amortizado	6.651,32€
Recursos humanos	24.975,00€
SUBTOTAL	31.626,32€
Imprevistos (5 %)	1.581,32€
Costes indirectos (8 %)	2.530,11€
COSTE TOTAL DEL PROYECTO	**35.737,74€**

Tabla 6: Costes totales

Con estos datos se puede concluir que lo más costoso a la hora de realizar el proyecto serán los recursos humanos asociados a este. Por la naturaleza del proyecto, su complejidad y falta de conocimiento previo, requiere tanto un esfuerzo individual para abordar las tareas especificas como colaborativas con el resto de participantes del proyecto. Esta complejidad común, reside en la dificultad de poner en conjunto la totalidad de todos los módulos que pertenecen al sistema, contribuyendo a la aumentar aun mas el precio total del proyecto. En la siguinte Figura 52 se puede ver como ha sido la división de todos los costes.

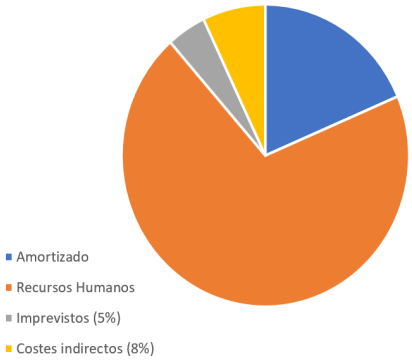


Figura 52: Diagrama de costes totales del proyecto

13. Conclusiones y desarrollo a futuro

El trabajo desarrollado se centra en el estudio, diseño e implementación del primer sistema SLAM para competir en FS-AI realizado por Formula Student Bizkaia. Este TFG se enmarca en el transcurso de una temporada, lo que supone un reto en cuanto al tiempo de desarrollo debido a la complejidad del módulo. El enfoque principal es lograr un sistema funcional que proporcione resultados aceptables, aunque no necesariamente optimizados, debido a los desafíos inherentes de desarrollar un diseño sin conocimiento previo.

El resultado final del trabajo ha sido la creación de un módulo SLAM funcional y, por ende, la mejora del módulo de percepción para un vehículo automatizado. Además, se han presentado diferentes alternativas a considerar sobre los sistemas SLAM y se ha profundizado en dos tipos de algoritmos muy utilizados, asentando una fuerte base para futuros desarrollos. Adicionalmente, los sistemas desarrollados han pasado por un proceso de simulación, integración y validación con datos reales no limitando este trabajo únicamente a un contexto teórico.

Sin embargo, a pesar de todo lo expuesto anteriormente, todavía existen numerosas configuraciones, algoritmos e implementaciones que pueden proporcionar un rendimiento superior que el sistema desarrollado. Es por ello, que el presente TFG se constituye como un buen punto de partida para explorar otras soluciones al problema del SLAM.

Entre estas soluciones, se proponen dos vías de desarrollo. La primera consiste en desarrollar un sistema SLAM que combine tanto el EKF-SLAM como el FastSLAM. El sistema EKF se emplearía para realizar las labores de localización y mapeo de la pista en una primera vuelta de reconocimiento y, posteriormente, se puede hacer uso de un filtro de partículas para localizar el vehículo en el entorno mapeado.

La segunda se presenta como un cambio radical en el módulo de percepción del AS y consiste en implementar un sistema «early fusion». Muchos otros equipos Formula Student *Driverless* han considerado sistemas de localización y mapeo basados únicamente en LiDAR que realizan el reconocimiento de la pista añadiendo una etapa de postprocesado y detección de *features* en el pointcloud acumulado resultante. Por ello, un sistema *LOAM* puede ser una buena opción a considerar en el caso de contar con un LiDAR de mayor resolución.

Otro aspecto a considerar de cara a optimizar los sistemas tanto de tracking visual como del SLAM, es la obtención de métricas en entornos reales. Para realizar esta tarea, es necesario generar un *ground truth* del entono. Esto se puede conseguir ha-

ciendo uso de herramientas como *WebLabel* que permiten la anotación de features en frames consecutivos de pointcloud y cámara. Con este *ground truth* generado se pueden usar sistemas como *TrackEval* [34] para obtener métricas sobre el tracking visual. En el caso del SLAM, se podría emplear el mismo sistema desarrollado para las simulaciones.

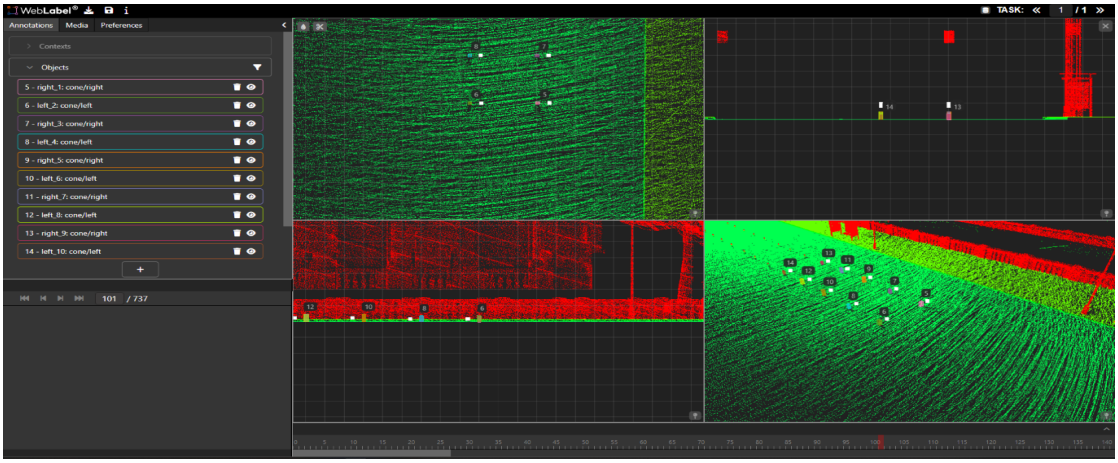


Figura 53: Anotación WebLabel

En resumen, se plantean diversas alternativas con el objetivo de alcanzar un sistema de SLAM y de percepción más preciso y eficiente, motivando la optimización de estos módulos para desarrollos futuros.

Agradecimientos

Me gustaría transmitir mi más sincero agradecimiento a todas aquellas personas que me han brindado su apoyo no sólo durante el desarrollo de este TFG, si no también durante mi periodo de estudio del grado en Ingeniería Informática de Gestión y Sistemas de Información. Desde luego, gracias a todos ellos he logrado crecer profesionalmente y cómo persona.

En primer lugar, a mi tutor y Faculty Advisor de Formula Student Bizkaia Asier Zubizarreta, por su ayuda durante el desarrollo del proyecto y que, junto con Mikel Diez, me ha apoyado y ha depositado su confianza en mí para gestionar el grupo de trabajo Driverless.

En segundo lugar, a todos los miembros de Fotmula Student Bizkaia Driverless, grupo del que he formado parte durante los últimos dos años y medio. Mario, Gorka, Irene, Adrián Arroyo, Unai, Luis, Adrián Zárate, Arkaitz, Eneko han sido mis compañeros durante estas últimas temporadas, cuyo esfuerzo y dedicación ha sido indispensable para el equipo. Gracias a ellos y al resto de miembros del proyecto en el ámbito eléctrico, por las horas y momentos compartidos.

Por último, y por ello no menos importante, no puedo terminar este documento sin agradecer a mi madre y mi abuela, quienes han sido el pilar fundamental sobre el que me podido apoyar y han tenido infinita paciencia conmigo durante el desarrollo de este TFG y mi transcurso por el grado. Gracias.

14. Anexos

14.1. Diagramas de clases

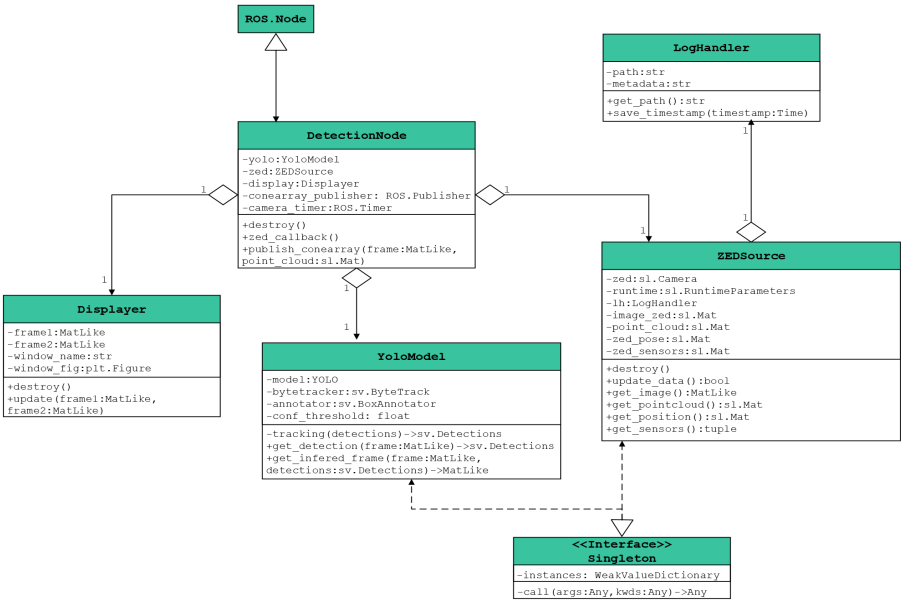


Figura 54: Diagrama de clases del módulo de detección visual y tracking

14.2. Distancia de Mahalanobis

$$d_m(\vec{x_{lm}}, \vec{z_i}) = \sqrt{(\vec{x_{lmx}} - \vec{z_{ix}})^T \Sigma^{-1} (\vec{x_{lmy}} - \vec{z_{iy}})} \tag{12}$$

Figura 55: Distancia de Mahalanobis en notación vectorial

Referencias

[1] LaTeX Project. *LaTeX Wikibook*. 2022. URL: https://en.wikibooks.org/wiki/LaTeX (visitado 18-02-2024).

[2] Scott Drew Pendleton. *Perception, Planning, Control, and Coordination for Autonomous Vehicles*. Machines. 2017. URL: https://doi.org/10.3390/machines5010006.

[3] Thorpe C.; Hebert M.H.; Kanade T.; Shafer S.A. *Vision and navigation for the Carnegie-Mellon Navlab*. 1988. URL: https://www.ri.cmu.edu/pub_files/pub2/thorpe_charles_1988_1/thorpe_charles_1988_1.pdf.

[4] Web Archive. *DARPA Challenges*. 2007. URL: https://web.archive.org/web/20080913175500/http://www.darpa.mil/GRANDCHALLENGE/.

[5] Rohit Bhagat; Priyanshu Singh; Prof. Ajay Kumar Srivastava; Er..Shilpi Khanna. *A Comprehensive Review on Advanced Driver Assistance Systems (ADAS)*. 2023. URL: https://ijaem.net/issue_dcp/A%20Comprehensive%20Review%20on%20Advanced%20Driver%20Assistance%20Systems%20(ADAS).pdf.

[6] SAE J3016. *SURFACE VEHICLE RECOMMENDED PRACTICE: Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles*. 2021. URL: https://www.sae.org/blog/sae-j3016-update.

[7] IMechE. *Formula Student AI 2024 Rules*. 2023. URL: https://www.imeche.org/docs/default-source/1-oscar/formula-student/2024/rules/fs-ai-2024-rules-v1-1.pdf?sfvrsn=2.

[8] IMechE. *Formula Student AI API*. 2021. URL: https://github.com/FS-AI/FS-AI_API.

[9] Niclas Vödisch, David Dodel y Michael Schötz. «FSOCO: The Formula Student Objects in Context Dataset». En: *SAE International Journal of Connected and Automated Vehicles* 5.12-05-01-0003 (2022).

[10] ROS Community. *ROS2 Humble Documentation*. 2022. URL: https://docs.ros.org/en/humble/index.html.

[11] Open-Source. *Cartographer ROS*. 2018. URL: https://github.com/cartographer-project/cartographer_ros.

[12] Open-Source. *Navigation2 ROS*. 2024. URL: https://github.com/ros-navigation/navigation2.

[13] Open-Source. *TF2 Transformations*. 2021. URL: https://github.com/ros-industrial/ros2_i_training/blob/main/workshop/source/_source/navigation/ROS2-TF2.md.

[14] EUFS. *Edinburgh University Formula Student Simulator*. 2019. URL: https://gitlab.com/eufs/eufs_sim.

[15] Vladislav Tananaev y Jose Luis Blanco Claraco. *Mobile Robot Programming Toolkit SLAM (MRPT SLAM)*. 2018. URL: https://github.com/mrpt-ros-pkg/mrpt_slam.

[16] IntRoLab. *Real-Time Appearance-Based Mapping*. 2023. URL: https://github.com/introlab/rtabmap.

[17] Johannes Meyer. *Hector mapping for ROS*. 2021. URL: http://wiki.ros.org/hector_mapping.

[18] Ji Zhang y Sanjiv Singh. *LOAM: Lidar Odometry and Mapping in Real-time*. 2014. URL: https://frc.ri.cmu.edu/~zhangji/publications/RSS_2014.pdf.

[19] Pierre Dellenbach et al. *CT-ICP: Real-time Elastic LiDAR Odometry with Loop Closure*. 2021. arXiv: 2109.12979 [cs.RO].

[20] Giorgio Grisetti; Cyrill Stachniss y Wolfram Burgard. *GMapping Rao-Blackwellized particle filer*. 2014. URL: https://openslam-org.github.io/gmapping.html.

[21] Albert Pumarola; Alexander Vakhitov; Antonio Agudo; Alberto Sanfeliu y Francesc Moreno-Noguer. *PL-SLAM: Real-Time Monocular Visual SLAM with Points and Lines*. 2015. URL: https://www.albertpumarola.com/publications/files/PLSLAM_ICRA17.pdf.

[22] Raúl Mur-Artal y Juan D. Tardós. «ORB-SLAM2: an Open-Source SLAM System for Monocular, Stereo and RGB-D Cameras». En: *IEEE Transactions on Robotics* 33.5 (2017), págs. 1255-1262. DOI: 10.1109/TRO.2017.2705103.

[23] Alejo Concha y Javier Civera. «Dense Piecewise Planar Tracking and Mapping from a Monocular Sequence». En: *Proc. of The International Conference on Intelligent Robots and Systems (IROS)*. 2015.

[24] Shiva Chandrachary. *A brief itroduction to GraphSLAM*. 2021. URL: https://shivachandrachary.medium.com/a-brief-introduction-to-graphslam-4204b4fce2f0.

[25] Kümmerle; Rainer et al. *G2o: A general framework for graph optimization*. 2011. URL: https://mengwenhe-cmu.github.io/Reading-Reports/Research/Localization/Graph_Optimization/g2o_A_General_Framework_for_Graph_Optimization/paper.pdf.

[26] Siti Sofiah Binti Mohd Radzi; Siti Sarah Binti Md. Sallah; Nazlee Azmeer bin Massuan ; Lee Ming Yi; Qamarul Aiman Bin Tajul Ariffin; Bayhaqi Bin Mohd Jailani y Hon Hock Woon. *Visual-based and Lidar-based SLAM Study for Outdoor Environment*. 2023. URL: https://robotics.pme.duth.gr/workshop_active2/wp-content/uploads/2023/06/12.-Visual-based-and-Lidar-based.pdf.

[27] Sebastian Thrun; Wolfram Burgard y Dieter Fox. *Probabilistic Robotics*. MIT press, 2005.

[28] Yunhao Du; Zhicheng Zhao; Yang Song; Yanyun Zhao; Fei Su; Tao Gong y Hongying Meng. *StrongSORT: Make DeepSORT Great Again*. 2023. URL: https://arxiv.org/pdf/2202.13514.

[29] Gerard Maggiolino; Adnan Ahmad; Jinkun Cao y Kris Kitani. *DEEP OC-SORT: MULTI-PEDESTRIAN TRACKING BY ADAPTIVE RE-IDENTIFICATION*. 2023. URL: https://arxiv.org/pdf/2302.11813.

[30] Gerard Maggiolino; Adnan Ahmad; Jinkun Cao y Kris Kitani. *BoxMOT: pluggable SOTA tracking modules for segmentation, object detection and pose estimation models*. 2024. URL: https://github.com/mikel-brostrom/yolo_tracking.

[31] Yifu Zhang; Peize Sun; Yi Jiang; Dongdong Yu; Fucheng Weng; Zehuan Yuan; Ping Luo; Wenyu Liu y Xinggang Wang. *ByteTrack: Multi-Object Tracking by Associating Every Detection Box*. 2022. URL: https://arxiv.org/pdf/2110.06864.

[32] Maria Ribeiro e Isabel Ribeiro. *Kalman and Extended Kalman Filters: Concept, Derivation and Properties*. Abr. de 2004.

[33] Soren Riisgaard y Morten Rufus Blas. *SLAM for Dummies*. 2012. URL: https://dspace.mit.edu/bitstream/handle/1721.1/36832/16-412JSpring2004/NR/rdonlyres/Aeronautics-and-Astronautics/16-412JSpring2004/A3C5517F-C092-4554-AA43-232DC74609B3/0/1Aslam_blas_report.pdf.

[34] Jonathon Luiten y Arne Hoffhues. *TrackEval*. 2020. URL: https://github.com/JonathonLuiten/TrackEval.