universidad de león

Escuela de Ingenierías Industrial, Informática y Aeroespacial

GRADO EN INGENIERÍA EN ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

Trabajo de Fin de Grado

LiDAR 2D y cámara de profundidad: Análisis y fusión de datos para la diferenciación de objetos

2D LiDAR and depth camera: Analysis and data fusion for object differentiation

Autor: Isabel Montilla Rojo

Tutor: Natalia Prieto Fernández

(Septiembre, 2022)

UNIVERSIDAD DE LEÓN

Escuela de Ingenierías Industrial, Informática y Aeroespacial

GRADO EN INGENIERÍA EN ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

Trabajo de Fin de Grado

ALUMNO: Isabel Montilla Rojo

TUTOR: Natalia Prieto Fernández

TÍTULO: LiDAR 2D y cámara de profundidad: Análisis y fusión de información para diferenciación de objetos

TITLE: 2D LiDAR and depth camera: Analysis and data fusion for object differentiation

CONVOCATORIA: Septiembre, 2022

RESUMEN:

El objetivo de este proyecto es ofrecer un sistema de visión artificial para la detección e identificación de objetos a partir del empleo de un sensor lídar y una cámara de profundidad, basándose en la fusión de los datos procedentes de ambos y sirviéndose del entorno de programación de ROS. Previo al desarrollo de la parte práctica, se presenta el marco teórico introduciendo la visión artificial y se estudian los sensores más empleados en este ámbito, haciendo hincapié en el lídar, la cámara de profundidad, el radar y las combinaciones más recurrentes de estos tres. A continuación, se introduce la fusión de sensores, sus distintas clasificaciones, métodos y algoritmos. Asimismo, se profundiza en el análisis de las tecnologías empleadas: características técnicas y funcionamiento del lídar "RPLIDAR A1M8" y de la

cámara de profundidad “Intel RealSense Depth Camera D435”, y se explican los conceptos generales, las herramientas específicas y los diferentes paquetes prediseñados del software utilizado “ROS”, como “YOLO” o “obstacle-detector-fusion”. Tras poner en práctica tres ensayos distintos y abordar los problemas surgidos durante su desarrollo, se concluye que el método óptimo para alcanzar los objetivos propuestos consistiría en una adaptación y combinación de una serie de paquetes prediseñados. De esta manera, se proporciona un resultado final que cumple fielmente con las expectativas, aunque presenta debilidades de precisión y velocidad que se explican por las limitaciones del equipo disponible, sobre todo en la parte correspondiente a la detección de YOLO.

ABSTRACT:

The aim of this project is to offer an artificial vision system for the detection and identification of objects using a lidar sensor and a depth camera, based on the fusion of data from both and using the ROS programming environment. Prior to the development of the practical part, a theoretical framework is presented introducing artificial vision and studying the most commonly used sensors in this field, focusing on the lidar, the depth camera, the radar and the most recurring combinations of these three. This is followed by an introduction to sensor fusion, its different classifications, methods and algorithms. Furthermore, the technologies used are analysed in depth: technical characteristics and performance of the lidar "RPLIDAR A1M8" and the depth camera "Intel RealSense Depth Camera D435". Likewise, general concepts, specific tools and different pre-designed used packages of the "ROS" software, such as "YOLO" or "obstacle-detector-fusion", are also explained. After implementing three different trials and addressing the problems that arose during their development, it is concluded that the optimal method to achieve the proposed objectives would consist of an adaptation and combination of a number of pre-designed packages. This provides a final result that faithfully meets the expectations, although it shows weaknesses in accuracy and speed that are explained by the limitations of the available equipment, especially in the YOLO detection part.

Palabras clave: LiDAR, cámara de profundidad, ROS, fusión de datos, detección e identificación de objetos, visión artificial	
Firma del alumno:	**VºBº Tutor/es:**

Resumen

El objetivo de este proyecto es ofrecer un sistema de visión artificial para la detección e identificación de objetos a partir del empleo de un sensor lídar y una cámara de profundidad, basándose en la fusión de los datos procedentes de ambos y sirviéndose del entorno de programación de ROS. Previa a la realización práctica del proyecto, se presenta el marco teórico introduciendo la visión artificial y sus posibles aplicaciones en diferentes sectores. A continuación, se expone un estudio de los diferentes sensores más empleados en este ámbito, haciendo hincapié en el sensor lídar, la cámara de profundidad y sus tipos, el radar y finalmente las combinaciones más recurrentes de estos tres. Asimismo, se introduce la fusión de sensores, sus distintos tipos de clasificaciones, los diversos métodos y algoritmos que se emplean, diferenciando técnicas de asociación de datos, de estimación del estado y de fusión de decisiones. Profundizando en el desarrollo de la parte práctica, se analizan las tecnologías empleadas: primeramente, el lídar "RPLIDAR A1M8" con sus características y su principio de funcionamiento; se sigue con las características técnicas que ofrece la cámara de profundidad "Intel RealSense Depth Camera D435"; y se concluye con la explicación del software empleado "ROS", incluyendo los conceptos generales de este, las herramientas específicas que ofrece y que se emplean en este proyecto y los diferentes paquetes prediseñados como "YOLO" o "obstacle-detector-fusion". Todos estos conocimientos se ven finalmente reflejados en los resultados de los ensayos efectuados. Tras realizar tres ensayos distintos y abordar los diferentes problemas que surgen durante el desarrollo, se concluye que el método óptimo para alcanzar los objetivos que se plantean en un principio consistiría en una adaptación y combinación de una serie de paquetes prediseñados. De esta manera, se proporciona un resultado final que cumple fielmente con las expectativas, pero que, naturalmente, presenta debilidades de precisión y velocidad, sobre todo en la parte correspondiente a la detección de YOLO, y que se explican por las limitaciones del equipo disponible.

Abstract

The aim of this project is to offer an artificial vision system for the detection and identification of objects using a lidar sensor and a depth camera, based on the fusion of data from both and using the ROS programming environment. Prior to the practical implementation of the project, a theoretical framework is presented, introducing artificial vision and its possible applications in different sectors. This is followed by a study of the different sensors most commonly used in this field, focusing mainly on the lidar sensor, the depth camera and its types, the radar and finally the most common combinations of the three of them. Likewise, it is also introduced the fusion of sensors, their different types of classifications, the various methods and algorithms that are used, and differentiating techniques of data association, state estimation and decision fusion. Moving on to the development of the practical part, there is an analysis of the technologies used: firstly, the lidar "RPLIDAR A1M8" with its characteristics and operating principle; followed by the technical characteristics offered by the depth camera "Intel RealSense Depth Camera D435"; and concluding with an explanation of the used software "ROS", including its general concepts, the specific tools it offers that are used in this project and the different pre-designed packages such as "YOLO" or "obstacle-detector-fusion". All this knowledge is finally reflected on the results of the tests carried out. After executing three different trials and addressing the different problems that arise during their development, it is concluded that the optimal method to achieve the objectives set out at the beginning would consist of an adaptation and combination of a series of pre-designed packages. This provides a final result that faithfully meets the expectations, but obviously presents weaknesses in accuracy and speed, especially in the YOLO detection part, which can be explained by the limitations of the available equipment.

Índice de contenidos

Índice de figuras

Índice de tablas

Glosario

Big data: "conjuntos de datos extremadamente grandes que pueden ser analizados computacionalmente para revelar patrones, tendencias y asociaciones, especialmente relacionadas con el comportamiento y las interacciones humanas" [1].

Computación en la nube ("Cloud Computing"): "práctica de uso de servicios informáticos, incluidos servidores, almacenamiento, bases de datos, redes, software, análisis e inteligencia, a través de Internet ("la nube") para ofrecer una innovación más rápida, recursos flexibles y economías de escala" [2].

Internet de las cosas (IoT, "Internet of Things"): "término que describe la red de objetos físicos que están integrados con sensores, software y otras tecnologías con el fin de conectar e intercambiar datos con otros dispositivos y sistemas a través de Internet" [3].

Inteligencia Artificial (AI, "Artificial Intelligence"): "teoría y desarrollo de sistemas informáticos capaces de realizar tareas que normalmente requieren inteligencia humana, como la percepción visual, el reconocimiento del habla, la toma de decisiones y la traducción entre idiomas" [4].

Machine Learning: "es un tipo de inteligencia artificial que permite a las aplicaciones de software ser más precisas en la predicción de resultados sin estar explícitamente programadas para ello. Los algoritmos de aprendizaje automático utilizan datos históricos como entrada para predecir nuevos valores de salida" [5].

1. Introducción

En la búsqueda por una sociedad superinteligente y conectada que se sirva de la digitalización, la innovación y la transformación tecnológica para afrontar sus problemas y mejorar la calidad de vida, se publica en 2015 el "V Plan Básico de Ciencia y Tecnología para 2016-2021" por parte del Gobierno de Japón. En este se define un nuevo concepto de sociedad, la *"Sociedad 5.0"* [6].

Esta nueva concepción de la sociedad trata de ir más allá de la ya conocida como Industria 4.0 y se enfoca en una sociedad centrada en el ser humano, que fusiona el ciberespacio y el espacio físico a través de herramientas que hasta ahora se habían utilizado únicamente en la industria, como el Internet de las cosas (IoT), la Inteligencia Artificial (IA), el Big Data, el Cloud Computing, el Machine Learning, la visión artificial, la robótica, etc.

Esta fusión del espacio físico y el ciberespacio implica la recolección y fusión de todo tipo de datos procedentes de sensores instalados en el espacio físico a través del IoT, acumulándolos en el ciberespacio para ser analizados por una IA que supera las capacidades humanas, y para posteriormente servirse de sus resultados, nuevamente, en el espacio físico (en coches autónomos, robots, aplicaciones militares, aplicaciones de teledetección, equipos de supervisión y de diagnóstico de dispositivos, equipos biomédicos, sistemas de transporte inteligentes, etc.) [7].

Mientras que los sensores proporcionan la información necesaria sobre ciertas características de interés del entorno, las técnicas de fusión de datos sensoriales van a permitir percibir, analizar y comparar información para obtener la interpretación óptima del entorno, proporcionando datos más significativos y fiables que los obtenidos mediante sistemas basados en una sola fuente de información.

Son muchas las ventajas derivadas de utilizar sistemas de múltiples sensores frente a los tradicionales mono-sensores, entre las que destacan: el aumento de la confiabilidad en caso de error o fallo del sensor; la reducción del ruido y la incertidumbre; la mejora de la precisión con la que el sistema percibe el entorno; la posibilidad de obtener información en el sistema de características independientes, proporcionando una visión más completa del entorno; la mejora de la resolución y las dimensiones de las medidas [8].

El área interdisciplinar de investigación de sensores e integración de multisensores abarca conocimientos en teoría de control, procesamiento de señales, inteligencia artificial, probabilidad y estadística, etc. En los últimos 70 años se han investigado diversos métodos de fusión de datos en este área, aplicándolos a una serie de disciplinas como la fiabilidad de Von Neumann, el reconocimiento de patrones de Chow, las redes neuronales de Hashem, la toma de decisiones y la representación de entradas/salidas de los procesos de fusión de Dasarathy y Vaeshney, la estimación estadística de Breiman, Juditsky y Nemirovski y la predicción del tiempo de Granger [9].

2. Objetivos

La meta principal de este trabajo consiste en el empleo de un sensor lídar 2D y una cámara de profundidad para la implementación de un sistema de visión artificial, que lleve a cabo la diferenciación e identificación de objetos a través de la fusión de datos procedentes de ambos sensores. Para alcanzar correctamente este objetivo principal, se han fijado otra serie de objetivos:

- Analizar y comparar los sensores del mercado para reconocer los más apropiados para la detección de objetos, comprendiendo sus principios de funcionamiento y los datos que proporcionan cada uno de ellos.
- Identificar los distintos métodos de fusión de datos, su clasificación y los algoritmos más recurrentes, para posteriormente inferir en el más adecuado en el ámbito de la detección de objetos, para los sensores disponibles: la cámara y el lídar 2D.
- Aprender a desenvolverse en el entorno de programación de ROS y todo lo que esto conlleva: programar en ROS; comprender la estructura de este entorno; estudiar el funcionamiento de los distintos paquetes prediseñados disponibles, tanto para la configuración de los sensores como para la detección e identificación de objetos; comunicarse con otros creadores de paquetes de la comunidad de ROS para intercambiar información, etc.
- Diseñar el propio paquete de ROS, ajustándose al equipo disponible y entendiendo las limitaciones de este.
- Evaluar el resultado final del proyecto e identificar posibles mejoras para un futuro.
- Estudiar las diferentes aplicaciones del servicio de la visión artificial, detección e identificación de objetos, en el panorama actual y las nuevas líneas de investigación existentes.

3. Estado del arte

En el concepto de la *Cuarta Revolución Industrial* o *Industria 4.0* que acuña Klaus Schwab en 2016 se habla de difuminar las fronteras entre las esferas físicas, biológicas y digitales a partir de la fusión de técnicas vanguardistas como pueden ser el Internet de las cosas, el Big Data, la Inteligencia Artificial o el Machine Learning. Estas dos últimas técnicas se encuentran estrechamente relacionadas con la visión artificial.

Análoga a la visión humana, que utiliza los ojos para recibir información de su entorno y procesarla, la visión artificial es la disciplina científica que trata de dotar a los dispositivos de esa misma capacidad y proporcionar orientación operativa en la ejecución de sus funciones basándose en la captura y el procesamiento de imágenes [10]. Los sistemas de visión artificial se basan en equipos de sensores, que incluyen desde dispositivos de posicionamiento global hasta cámaras o sensores de detección y alcance por luz, y métodos de procesamiento de los datos procedentes de estos.

Si bien en los próximos apartados quedan explicados estos sensores y los correspondientes métodos de análisis de sus datos, conviene repasar las posibles aplicaciones de estos sistemas de visión artificial:

- ***Aplicación en la industria electrónica*:** en los procesos de fabricación de obleas o de inspección de placas de circuito impreso, la velocidad y la calidad son esenciales. Es por esto por lo que se sirven de estos sistemas de visualización y detección para realizar inspecciones ópticas automatizadas, clasificar defectos, detectar presencia o ausencia de componentes, verificar ensamblajes, medir componentes, medir la posición de ciertos componentes, etc.
- ***Aplicaciones en la industria de la automoción*:** dejando al margen las muchas aplicaciones de estos sistemas a nivel de fábrica, destaca el auge de su aplicación en los vehículos autónomos. Estos vehículos se sirven de estos sistemas de visualización para percibir el medio que les rodean y navegar de manera autónoma, imitando las capacidades humanas de manejo y control. Generalmente integran sensores radar, lídar, cámaras, sistemas de posicionamiento global, unidades de medición inercial, etc. Así mismo, se definen acorde a un sistema de clasificación

dictado por la SAE International [11] dependiendo de su nivel de autonomía, que va desde sistemas totalmente manuales (nivel 0) hasta los totalmente automatizados (nivel 5) , como se observa en la Figura 3.1 :

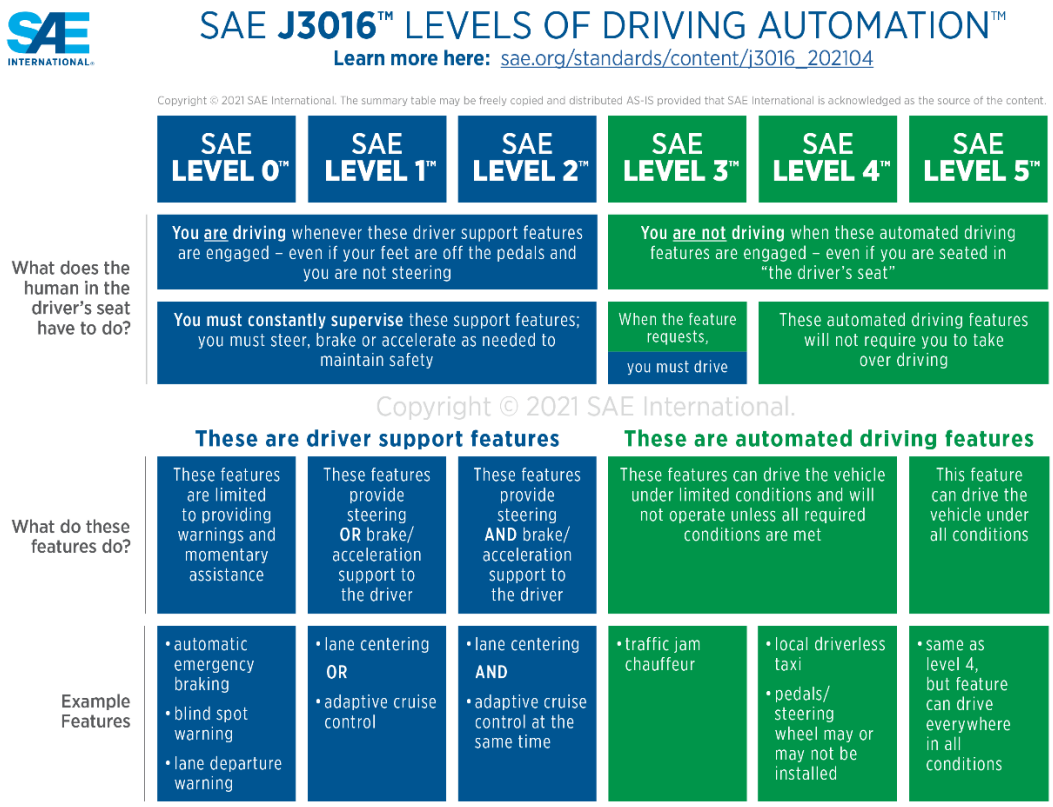


Figura 3.1 Niveles de conducción autónoma (Fuente:[11]).

- ***Aplicaciones en la salud:*** como en otros sectores, la industria sanitaria se sirve de estos sistemas para sus procesos de fabricación; sin embargo, resulta más interesante analizar otras nuevas líneas de investigación, como es la asistencia a personas con discapacidad visual. Estos sistemas tienen como objetivo ayudar a las personas discapacitadas a llevar una vida normal gracias a la clasificación de escenas a través de la detección de objetos por diferentes sensores. De tal manera que, si se detecta un obstáculo en el entorno del usuario, este quedará notificado [12]. Recientemente, *Apple* ha incluido en sus dispositivos "*pro*" un sensor lídar que junto con la aplicación móvil "*Supersense*" [13], asiste a los invidentes. *Supersense*

detecta automáticamente lo que el usuario está tratando de escanear y le guía a apuntar la cámara, leyendo el contenido en el formato correcto.

Así mismo, estos sistemas de visión artificial se emplean en lugares donde se requiere hacer de forma automática mediciones para control de acceso de personas. A raíz de la pandemia del *Covid-19*, muchos establecimientos (desde grandes almacenes, fábricas hasta estaciones de tren) han regulado la confluencia de personas instalando cámaras y otros sensores en sus entradas.

- ***Aplicaciones en los procesos de embalaje y logística*:** se implementan estos sistemas para procesos de etiquetación y verificación de productos, para analizar las trayectorias de los operarios en las fábricas y calcular las trayectorias más eficientes, para robots de pick and pack, paletizadores, reconocimiento de números de serie, etc.
- ***Cobots* (*Collaborative robots*):** son robots creados para interactuar físicamente con los humanos en entornos de trabajo y que integran estos sistemas de detección basados en la visión artificial. Estos robots se extienden en muchos sectores como el de la automoción, la electrónica, la química, la electrónica, la logística, la industria alimentaria, y en general en procesos que requieren trabajos repetitivos y manuales en cadena. Destacan por su versatilidad, su reducido tamaño y sus ventajas frente a sus antecesores, como la facilidad de conexión, la facilidad de programación de sus operaciones, su multifuncionalidad (pueden desempeñar varias tareas) , la seguridad que ofrecen (por ejemplo, se pueden configurar su parada de funcionamiento en función de presencia humana en la zona de trabajo), etc [14].

Figura 3.2 Cobot en funcionamiento (Fuente: [15]).

3.1 SENSORES

Se entiende como sensor al dispositivo que capta magnitudes físicas u otras alteraciones de su entorno y responde en consecuencia, transformando las magnitudes físicas o químicas en magnitudes eléctricas[16]. Es posible hacer una ligera diferenciación entre los sensores que miden los valores internos de los sistemas dinámicos (sensores de estado internos, conocidos como sensores propioceptivos) y los que miden los valores externos (sensores de estado externos, sensores exteroceptivos): los sensores de estado internos son dispositivos como codificadores, giroscopios, magnetómetros, acelerómetros que conforman sistemas de posicionamiento global (del inglés *Global Positioning System,* **GPS**) o Unidades de Medición Inercial (del inglés *Inertial Measurement Unit*, **IMU**); mientras que, los sensores **ultrasónicos**, los sensores de detección y alcance de luz (**LiDAR**), las **cámaras**, los sensores de alcance y detección radio (**Radar**) son ejemplos de los sensores de estado externos.

3.1.1 IMU

Una Unidad de Medición Inercial, comúnmente conocido como IMU (acrónimo del inglés *Inertial Measurement Unit*) es un dispositivo electrónico de medición que proporciona datos de aceleración, orientación, velocidades angulares, velocidad lineal y fuerzas gravitacionales de un cuerpo. Generalmente se basa en una combinación de, como mínimo, un acelerómetro tridireccional y un giróscopo de tres ejes [17]. Los giroscopios detectan los cambios en la velocidad de rotación y los acelerómetros la aceleración lineal, ocasionalmente se incluyen también magnetómetros, como referencia de rumbo.
Gracias a esta información, permiten al sistema seguir la posición del aparato, usando el método conocido como "navegación por estima" (*dead reckoning*). Consiste en estimar la posición actual y la dirección a seguir, partiendo de una posición de origen y basándose en la dirección y la velocidad que se ha seguido durante cierto tiempo[18]. El dispositivo IMU es clave para los sistemas de guía inercial de los vehículos aéreos, marinos, aplicaciones robóticas, naves espaciales, etc.

Habitualmente estos sensores de navegación inercial se integran como complemento a la navegación satélite **GPS**, otorgando una mayor independencia al sistema en caso de que las señales GPS no estén disponibles, como en zonas interiores o en caso de interferencia electrónica. El sistema de posicionamiento global (GPS) proporciona geolocalización de un receptor del sistema GPS sobre la Tierra, basándose en la recepción de señales procedentes de como mínimo tres satélites. Estas señales proporcionan la información horaria e identificación de cada satélite de tal manera que, el receptor sincroniza su propio reloj con el de los satélites GPS y calcula el tiempo que tardan en llegar las señales al equipo y mide la distancia al satélite [19].

3.1.2 SENSORES ULTRASÓNICOS

Los sensores ultrasónicos facilitan la detección de proximidad de objetos al propio dispositivo. El principio de los sensores ultrasónicos consiste en medir el tiempo que transcurre entre la emisión de una onda ultrasónica desde el sensor emisor y la recepción del eco producido por esa onda en el mismo emisor, al propagarse hacia cierto objetivo. La

velocidad de propagación de la señal es conocida, de tal manera que, estos sensores miden la distancia al objeto calculando el tiempo entre la emisión y la propia recepción de la onda.

3.1.3 LiDAR

LiDAR (acrónimo del inglés, *Light Detection and Ranging* o *Laser Imaging Detection and Ranging*), se traduce en castellano como sistema de medición y detección de objetos mediante láser, también conocido como escáner láser.

Este sensor basa su funcionamiento en el principio de emisión de pulsos de un haz infrarrojo que se refleja en los objetos cercanos. El sistema detecta los objetos reflejados midiendo el tiempo que tarda el pulso emitido en llegar al objetivo y volver, lo que permite calcular la distancia recorrida y dada la elevada velocidad a la que viaja la luz, acelerar el proceso de medición, convirtiéndolo en un sistema muy eficiente y preciso.

Al escanear su entorno, el lídar genera la representación de la escena con datos de salida como una serie de puntos, también conocidos como datos de nube de puntos en espacios 1D, 2D o 3D. Estos datos de la nube de puntos contienen las coordenadas x, y, z (dependiendo de si es 3D, 2D, o 1D) y la información de intensidad de los obstáculos dentro del espacio de trabajo.

Todas estas mediciones requieren los componentes clave de un sistema lídar: un GPS y una unidad de medición interna (IMU).

3.1.4 CÁMARA DE PROFUNDIDAD

El principal fundamento de las cámaras de profundidad se puede explicar partiendo de una serie de conocimientos generales de las tradicionales cámaras digitales.

Las imágenes emitidas por una cámara digital estándar son de la forma de cuadrículas de 2 dimensiones de píxeles. Cada píxel se asocia con una serie de valores; por lo general, pensamos en ellos como colores rojo, verde y azul. Esto es lo que se conoce como el sistema de representación de colores en términos de la intensidad de los colores primarios, RGB (del inglés: *red, green and blue* lo que corresponde en castellano: rojo, verde y azul) [20].

En base al modelo de síntesis aditiva de color, el rojo, el verde y el azul se pueden combinar en varias proporciones para obtener cualquier color del espectro visible. Dependiendo de

la proporción de color, los niveles de R (rojo), G (verde) y B (azul) pueden oscilar entre 0 y 100 por ciento de la intensidad total, de manera que un valor de "0" significa que no interviene en la mezcla y un aumento de ese valor se asocia con un mayor aporte de intensidad a la mezcla.

Cada nivel está representado por el rango de números decimales del 0 al 255 (256 niveles para cada color), equivalente al rango de números binarios de 00000000 a 11111111. El número total de colores disponibles es 256 x 256 x 256, o 16.777.216 colores posibles [21]. Las fotografías como comúnmente se conocen, no son más que miles o millones de píxeles de esta forma juntos. En la Tabla 3.1 quedan recogidas las correspondencias numéricas de los colores principales:

Tabla 3.1 Valores numéricos correspondientes a los colores principales (Fuente: propia).

Negro	0	0	0
Rojo	**255**	0	0
Verde	0	**255**	0
Azul	0	0	**255**
Blanco	**255**	**255**	**255**

Ahora bien, una cámara de profundidad cuenta con píxeles que tienen un valor numérico diferente asociado con ellos, siendo ese número la distancia desde la cámara, o "profundidad". Algunas cámaras de profundidad tienen un sistema RGB y otro de profundidad, proporcionando píxeles con los cuatro valores, o lo que se conoce como RGBD (donde la “D” se refiere al término en inglés “*Depth”* que se traduce al castellano como profundidad).

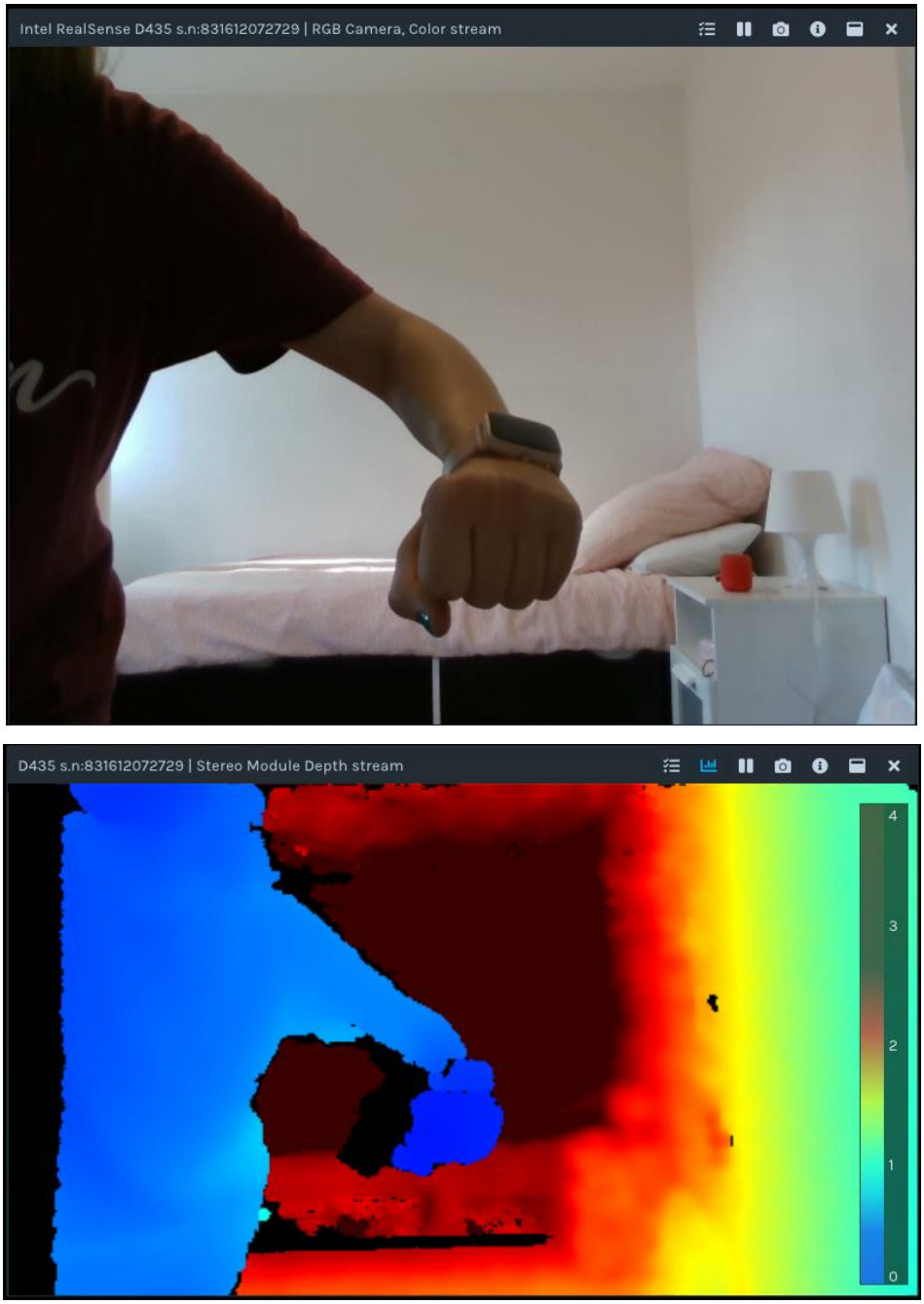


Figura 3.3 Salida de una cámara de profundidad (Fuente: propia).

En la Figura 3.3 se observa la salida de información que proporciona una cámara de profundidad: en la imagen superior, se muestra la imagen en RGB y en la inferior, la imagen de profundidad, donde cada color del mapa de profundidad representa una distancia diferente a la cámara. En este caso, la profundidad sigue la escala de la derecha, donde los azules más oscuros representan objetos más cercanos y los rojos más oscuros, los objetos más lejanos.

3.1.4.1 Tipos de cámaras de profundidad

Son muchos los métodos para calcular la profundidad en estos dispositivos, dentro de las técnicas de escaneados 3D actuales se hace una distinción entre aquellas técnicas en las que se produce contacto con el objeto y las que no. Mientras que las primeras implican un proceso lento y no resultan muy interesantes más allá de su aplicación en el control de procesos de fabricación; las técnicas sin contacto con el objeto resultan clave para el desarrollo de estas cámaras de profundidad y así mismo, se dividen entre técnicas activas o pasivas.

Las activas se basan en la emisión de cierta señal que será reflejada en el objeto para posteriormente analizarla con objeto de capturar la geometría del objeto. A diferencia de estas, en las técnicas pasivas no se emite ningún tipo de señal, sino que se fundamentan en observar la radiación del ambiente que refleja el objeto.

Dentro de oferta del mercado actual, las cámaras de profundidad basadas en las técnicas activas utilizan principalmente la técnica de luz estructurada o tiempo de vuelo mientras que las que se basan en técnicas pasivas se reducen principalmente a la técnica de profundidad desde el estéreo.

a) ***Luz estructurada***

   Las cámaras de profundidad de luz estructurada se fundamentan en la proyección de un patrón de luz, generalmente luz infrarroja, en el objeto desde emisor y el posterior análisis de la deformación del patrón producido por la forma del objeto. Debido a que se conoce el patrón proyectado, la forma en que el sensor de la cámara ve el patrón en la escena proporciona la información de profundidad. Usando la disparidad entre la imagen esperada y la imagen real que llega a la cámara, se puede calcular la distancia desde la cámara para cada píxel.

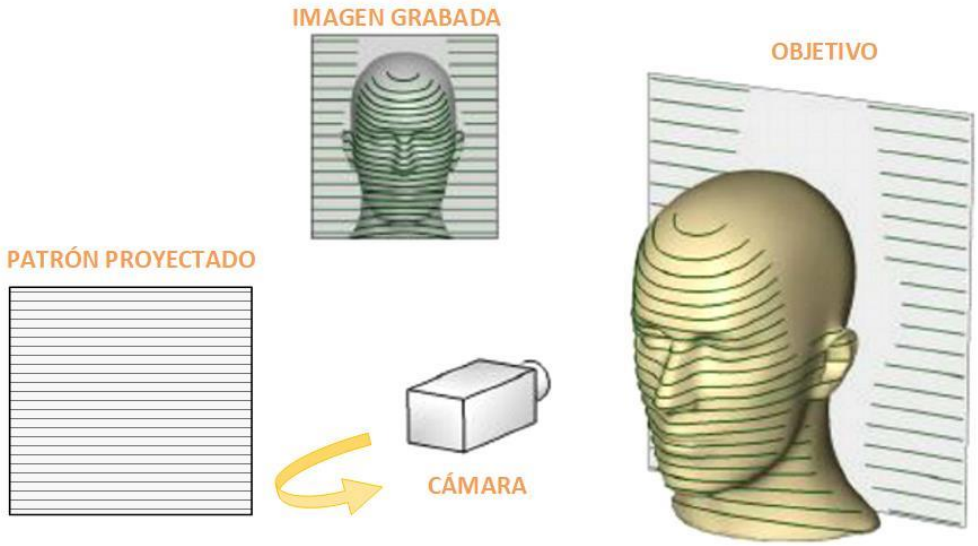


Figura 3.4 Cámara 3D basado en sistema de luz estructurada (Fuente: propia).

Las cámaras de luz estructurada funcionan mejor en interiores a distancias relativamente pequeñas y son vulnerables a otros ruidos en el entorno de otras cámaras o dispositivos que emiten infrarrojos [20].

b) *Tiempo de vuelo y lídar*

Estas cámaras basan su funcionamiento en la misma técnica comentada para el lídar en el "Apartado 3.1.3". La principal desventaja de las cámaras de tiempo de vuelo es que pueden ser susceptibles a otras cámaras que se encuentran el mismo espacio y también pueden funcionar peor en condiciones exteriores. En ocasiones la luz que incide en el sensor puede no haber sido una luz emitida por la cámara específica, sino que podría proceder de alguna otra fuente como el sol u otra cámara, degradando la calidad de la imagen de profundidad.

c) *Profundidad estéreo*

Las cámaras de profundidad estéreo disponen dos sensores separados por una pequeña distancia, cada uno de ellos toma una imagen y posteriormente se las compara. Dado que se conoce la distancia entre los sensores, estas comparaciones brindan información de profundidad. Este fundamento es precisamente el que utiliza el humano para la propia percepción de profundidad.

La distancia que estas cámaras pueden medir está directamente relacionada con la distancia entre los dos sensores, cuanto más ancha es la línea de base, más lejos puede ver la cámara [20].

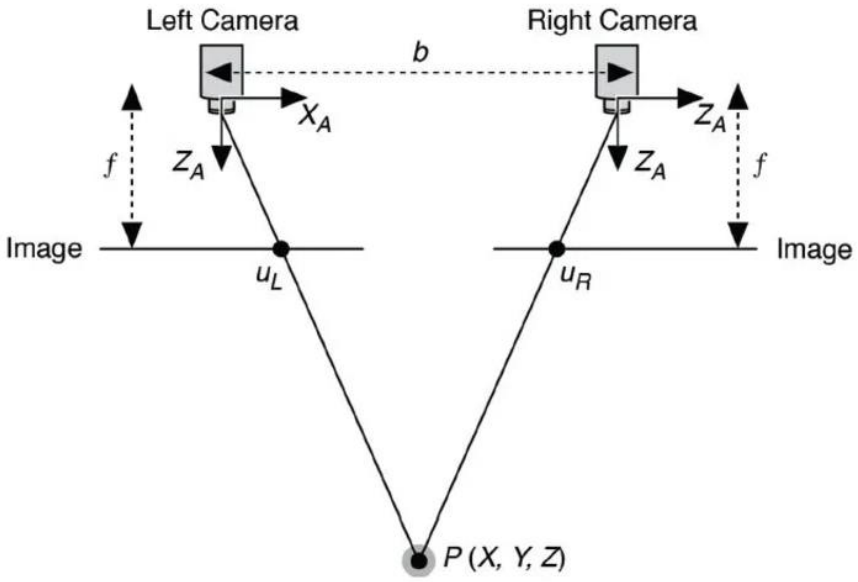


Figura 3.5 Simplificación del funcionamiento de visión de profundidad estéreo (Fuente: [22]).

Estas cámaras de profundidad estéreo también proyectan luz infrarroja en una escena para mejorar la precisión de los datos, pero a diferencia de las cámaras de luz estructurada, las cámaras estéreo pueden usar cualquier luz para medir la profundidad, considerándose como buen ruido a todo el ruido infrarrojo. La adición de un proyector de infrarrojos significa que, en condiciones de poca luz, la cámara aún puede percibir detalles de profundidad. Esto brinda un funcionamiento óptimo en la mayoría de las condiciones de iluminación, incluso en exteriores.

3.1.5 RADAR

Radar es un acrónimo del término inglés *Radio Detection and Ranging*, detección y localización por radio. Como su propio nombre indica, este sistema electromagnético utiliza ondas de radio para detectar y encontrar la distancia al objetivo. Se basa en el principio de

propagación de ondas electromagnéticas por el espacio de trabajo y la posterior recepción en el dispositivo de las reflexiones de los objetivos. Una vez que la onda reflejada llega al radar, se procesa, proporcionando la información necesaria para determinar la forma, el tamaño, la ubicación, el alcance, el ángulo, la velocidad de los objetos detectados y otras características, basándose en el efecto Doppler.

Aunque presenta algunos inconvenientes como una baja precisión angular o una generación de datos menor que otros sensores, se considera el más idóneo para condiciones meteorológicas adversas ya que la propagación de las ondas electromagnéticas es insensible a ellas y funciona independientemente de la condición de iluminación del entorno. Es por esto por lo que, para compensar las limitaciones del radar, su información suele fusionarse con los datos de otros sensores, como las cámaras y el LiDAR.

3.1.6 COMBINACIONES RECURRENTES

En el campo de los sistemas autónomos, más concretamente para la percepción del entorno y la detección de objetos, hay tres combinaciones que destacan: *Cámara-LiDAR* (CL), *Cámara-Radar* (CR) y *Cámara-LiDAR-Radar* (CLR). En la Tabla 3.2 , se comparan estos 3 sensores, basándose en características técnicas y otros factores externos [23]:

- **✓:** indica que el sensor funciona de forma competente bajo el factor específico.
- **~**: indica que el sensor funciona razonablemente bien bajo el factor específico.
- **x**: indica que el sensor no funciona bien bajo el factor específico.

Tabla 3.2 Comparación características sensores (Fuente: Traducción de [23]).

FACTORES	CÁMARA	LIDAR	RADAR	FUSIÓN
Alcance	~	~	✓	✓
Resolución	✓	~	x	✓
Precisión de la distancia	~	✓	✓	✓
Velocidad	~	x	✓	✓
Percepción del color	✓	x	x	✓
Detección de objetos	~	✓	✓	✓

Clasificación de objetos	✓	~	x	✓
Detección de rutas	✓	x	x	✓
Detección de bordes de obstáculos	✓	✓	x	✓
Condiciones de iluminación	x	✓	✓	✓
Condiciones meteorológicas	x	~	✓	✓

Según el estudio, "*Multi-Sensor Fusion in Automated Driving: A Survey*" [24], la combinación de sensores *CR* es la más empleada en los sistemas de fusión para la percepción del entorno, seguida de *CLR* y *CL*. La combinación de cámara y radar proporciona imágenes de alta resolución y obtiene información adicional sobre la distancia y la velocidad de los de los obstáculos circundantes.

Uno de los mercados más potentes dentro de los sistemas autónomos es el de los vehículos autónomos. Dentro de este, se puede llevar esta información a la práctica y hacer una comparación de las tecnologías empleadas por los grandes diseñadores:

- ***Tesla***

  El sistema de "*Autopilot*" que caracteriza a Tesla ha evolucionado recientemente hacia el nuevo sistema "*Tesla Vision*". Este cuenta con una combinación de ocho cámaras y un potente procesador de visión basado en un red neuronal desarrollada propiamente por Tesla, que ha permitido eliminar en 2022 el sensor radar de sus coches [25]. Reduciendo el sistema únicamente a las cámaras. Así mismo, en un intento de hacer sus coches asequibles al mayor público posible y haciendo mucho hincapié en la correlación de estos sistemas autónomos con la visión humana, Tesla ha ido en contra de la instalación de sensores lídar en sus vehículos, considerándolos innecesarios.

- ***Waymo (Google)***

  El proyecto de vehículo autónomo de Google, *Waymo*, incorpora las tres tecnologías mencionadas: el lídar, la cámara y el radar [26]. Cada uno de estos sensores complementa a los demás y brinda uno de los sistemas de visión más potentes, permitiendo detectar objetos hasta 1 km en una variedad de condiciones [27].

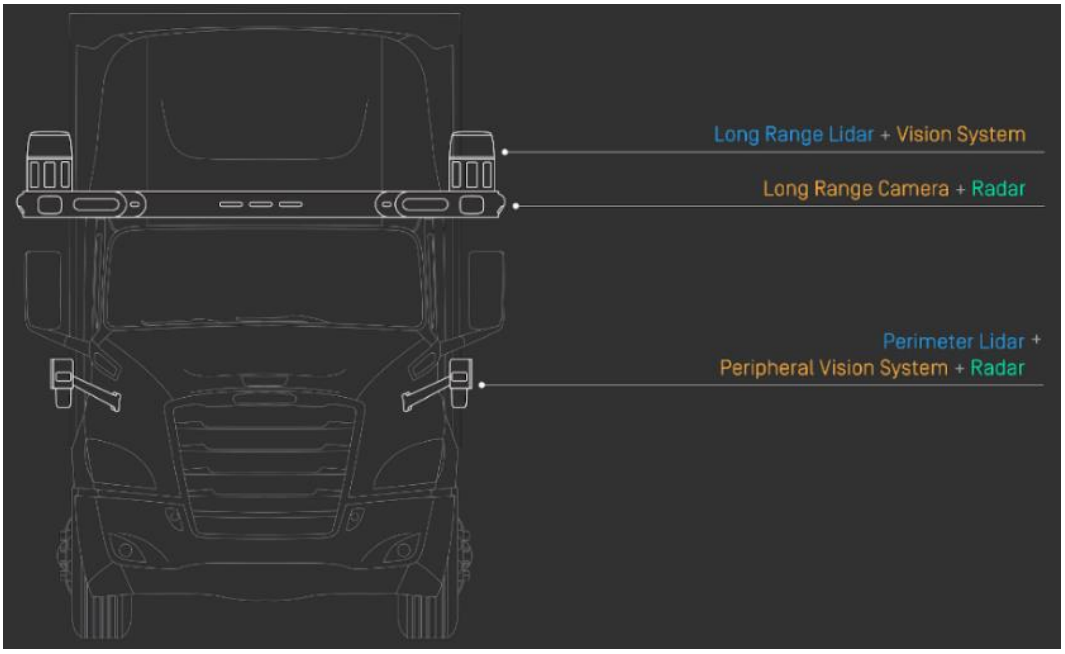


Figura 3.6 Hardware de los coches autónomos Waymo (Fuente:[27]).

- ***Aurora***

  Los coches autónomos de Aurora Driver emplean un sistema lídar propio, denominado "*FirstLight Lidar*", así como un radar de imágenes de largo alcance y cámaras de alta resolución, todo esto integrado gracias a "*Aurora Computer*" [28]. La empresa tiene como objetivo ofrecer sus servicios de conducción autónomo a Toyota, Volvo y PACCAR a finales de 2023.

- ***Lyft***

  Así mismo, Toyota ha adquirido los coches autónomos de nivel 5 de autonomía de Lyft. Estos vehículos incorporan nuevamente los sensores lídar, cámaras y radares, que, junto con todos los datos recopilados por la flota de taxis de Lyft, proporcionan una conducción autónoma muy completa [29].

3.2 FUSIÓN DE SENSORES

El "*Joint Directors of Laboratories*" (JDL) define la fusión de datos como "*Un proceso de varios niveles que trata de la asociación, correlación y combinación de datos e información de fuentes únicas y múltiples, para lograr una posición refinada, identificar estimaciones y evaluaciones completas y oportunas de situaciones, amenazas y su significado*" [30].

De tal manera que, se puede entender como un compendio de técnicas, análogo al proceso cognitivo que realizamos los humanos, que integra los datos procedentes de múltiples fuentes. El fin es realizar inferencias sobre el mundo exterior, alcanzando conclusiones que son imposibles de obtener a partir de una sola fuente y generando criterios para las tareas de decisión.

Desde un punto de vista más específico y entendiéndolo como una subdisciplina de la fusión de datos, la fusión de sensores se refiere a la producción de un resultado enriquecido a partir de la información de salida de sensores individuales o de los resultados de algoritmos específicos.

La fusión de sensores surge como resultado de la búsqueda de una optimización en los sistemas tradicionales de tratamiento de información procedente de una fuente. Estos presentan limitaciones como la incapacidad de reducir la incertidumbre a la hora de realizar observaciones, la provisión parcial de información del entorno o la falla completa del sistema en caso de error en un único sensor. Todo esto desaparece con la fusión de datos procedentes de múltiples sensores, que proporcionarán información redundante imposible de alcanzar de otra manera, incrementando la precisión y confiabilidad de los sistemas y reduciendo la incertidumbre.

3.2.1 CLASIFICACIÓN DE LA FUSIÓN DE SENSORES

Cuando se trata de hacer una clasificación de la fusión de sensores, resulta difícil establecer una única clasificación clara y estricta, se distinguen varias categorías, no exclusivas, dependiendo de diferentes criterios, se presentan a continuación las más conocidas [31]:

1. Clasificación por niveles de abstracción de los datos empleados
2. Clasificación de Dasarathy, basada en la entrada y la salida
3. Clasificación basada en la relación entre los datos de entrada
4. Clasificación basada en el nivel de centralización
5. Clasificación basada en los niveles de fusión del JDL

1. Clasificación por niveles de abstracción de los datos empleados

Los procesos de fusión suelen clasificarse en un modelo de tres niveles en función de los datos empleados: fusión de nivel bajo, intermedio o alto. Así mismo, estos tres niveles pertenecen a otros dos grupos, la fusión de nivel bajo se considera *fusión temprana* y las dos restantes pertenecen a la *fusión tardía*.

i. *Fusión de nivel bajo,* comúnmente llamada fusión de datos brutos. Consiste en producir nuevos datos a partir de la fusión de los datos brutos procedentes de varios sensores. Entendiendo por datos en bruto a los obtenidos directamente por el sensor sin haber realizado ninguna transformación.

ii. *Fusión de nivel intermedio* o fusión de características o rasgos. Consiste en combinar varias características (por ejemplo: forma, posiciones, texturas, líneas, etc.) detectadas independientemente en los datos de los sensores, en un mapa de características que puede utilizarse para la segmentación y la detección.

iii. *Fusión de alto nivel,* también conocida como fusión de decisiones. Consiste en basarse no únicamente en las detecciones de los sensores, sino también en las predicciones y el seguimiento, incluyendo lógica difusa, métodos estadísticos y votaciones [8].

2. Clasificación de Dasarathy, basada en la entrada y la salida

Uno de los sistemas de clasificación de fusión de datos más conocidos fue el que propone en 1997 Dasarathy [32] [4] y que se basa en la abstracción de la información de entrada y de salida. Se compone de cinco categorías, que se conocen por su abreviatura del término en inglés, de tal manera que: DAI hace referencia "*Data In*"; DAO a "*Data Out*"; FEO a "*Feature Out* "; FEI a "*Feature In*"; DEI a "*Decision In*"; DEO a "*Decision Out*".

i. *(DAI-DAO) Datos de entrada - Datos de salida:* la fusión procesa las entradas y salidas de datos en bruto y se lleva a cabo inmediatamente después de recoger los datos de los sensores, proporcionando resultados más fiables o precisos.

ii. *(DAI- FEO) Datos de entrada - Característica de salida:* la fusión emplea los datos brutos de las fuentes para extraer los rasgos o características que describen una entidad en el entorno.

iii. *(FEI-FEO) Característica de entrada - Característica de salida:* tanto la entrada como la salida del proceso de fusión de datos son características. De tal manera que, la fusión se dirige a un conjunto de características con el fin de mejorar, perfeccionar u obtener nuevas características.

iv. *(FEI- DEO) Característica de entrada - Decisión de salida:* se obtiene un conjunto de características como entrada y proporciona un conjunto de decisiones como salida.

v. *(DEI-DEO) Decisión de entrada – Decisión de salida:* fusiona las decisiones de entrada para obtener decisiones mejores o nuevas.

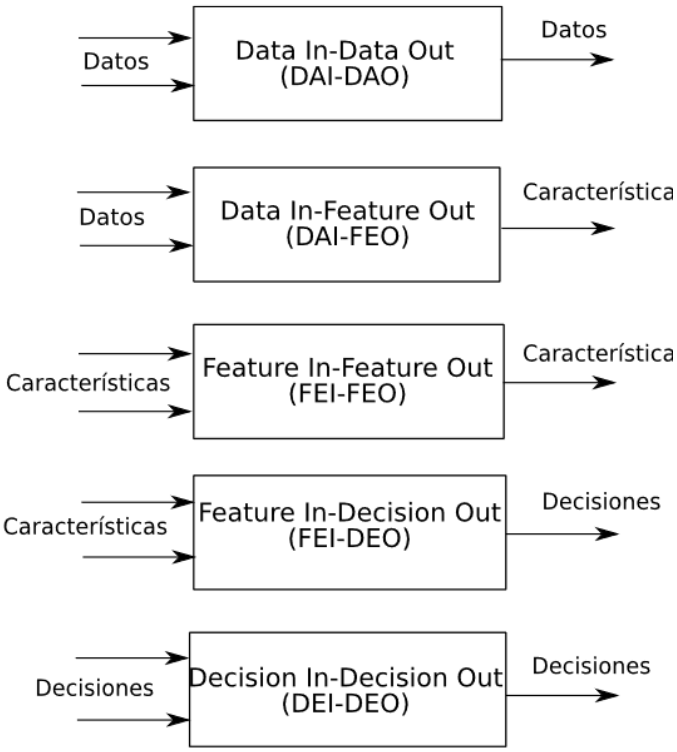


Figura 3.7 Clasificación de Dasarathy (Fuente: [9]).

3. Clasificación basada en la relación entre los datos de entrada

Durrant-Whyte propone una clasificación de la fusión de sensores también basada en la relación entre las fuentes de información [33].

i. *Fusión complementaria*: se considera cuando varios sensores se utilizan de forma independiente para observar diferentes escenas y se combinan para dar una visión más completa del fenómeno.

ii. *Fusión competitiva o redundante:* dos o más sensores proporcionan mediciones independientes sobre el mismo objetivo y, por tanto, podrían fusionarse para aumentar la confianza.

iii. *Fusión cooperativa o coordinada*: esta configuración consiste en utilizar la información proporcionada por dos o más sensores independientes y combinarla en una nueva información que suele ser más compleja que la original.

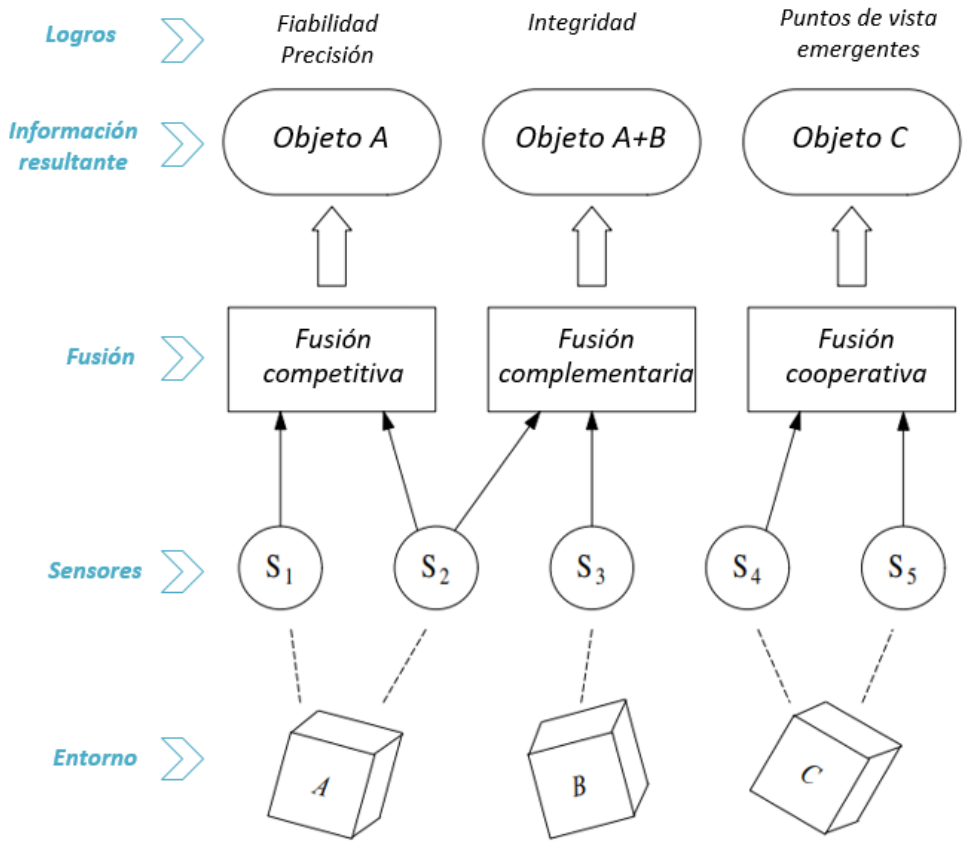


Figura 3.8 Clasificación basada en la configuración de los sensores (Fuente: Traducción de [9]).

4. Clasificación basada en el nivel de centralización

A la hora de diseñar un sistema de fusión de datos resulta clave determinar dónde se realiza la fusión, en base a ese criterio se puedes distinguir 3 clases, véase el esquema explicativo de la Figura 3.9.

i. *Fusión centralizada:* la fusión se realiza en un nodo central de fusión.

ii. *Fusión descentralizada*: la fusión se compone de una red de nodos en la que cada nodo tiene su propia capacidad de procesamiento y no hay un único punto de fusión de datos. La fusión se lleva a cabo en todos y cada uno de los nodos y cada uno de ellos produce una estimación del objeto utilizando la información global[31] [9].

iii. *Fusión distribuida:* cada sensor realiza un preprocesado de forma independiente de los datos y posteriormente la fusión se realiza en los nodos específicos de fusión, que pueden ser uno o varios [31] [34] [9].

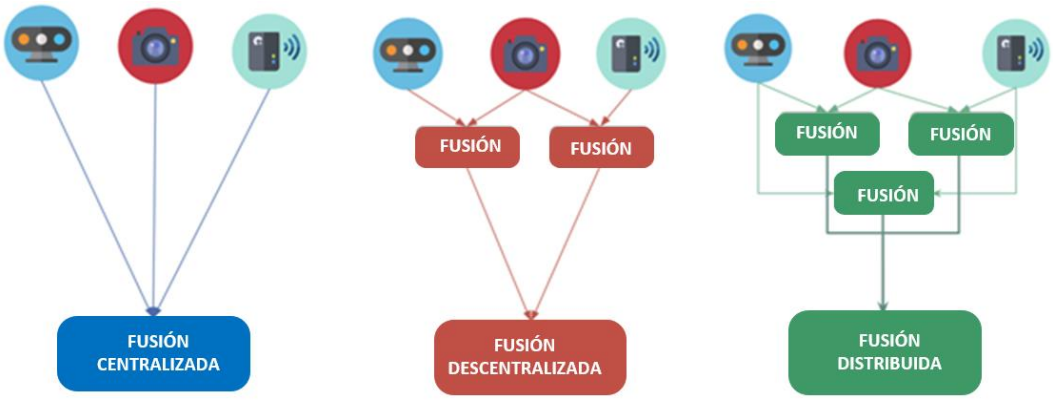


Figura 3.9 Clasificación basada en el nivel de centralización (Fuente: Traducción de [8]).

5. Clasificación basada en los niveles de fusión del JDL

Este modelo propuesto en 1997 por el *Joint Directors of Laboratories* (JDL) y el *Departamento de Defensa Americano (DoD)* es uno de los más recurridos a la hora de hacer una clasificación de la fusión de datos. Se distinguen 5 niveles en el proceso de fusión de datos y se realiza una distinción de fusión de bajo nivel o de alto nivel, siendo esta segunda la correspondiente a los niveles a partir del 2, en los que se contribuye al proceso de toma de decisiones [31] [9].

i. *Nivel 0: Procesamiento previo de la fuente*. Consiste en la evaluación de datos para reducir la cantidad de datos y desechar la información innecesaria.

ii. *Nivel 1: Evaluación de objetos.* Esta fase se encarga de transformar la información de entrada en estructuras de datos consistentes. Clasificando e identificando los objetos, así como y realizando un seguimiento de su estado y orientación.

iii. *Nivel 2: Evaluación de la situación***.** Se identifican en esta fase las situaciones probables partiendo de los datos dados y eventos observados y estableciendo relaciones entre los objetos. Proporciona una descripción de las relaciones entre los eventos observados y los objetos detectados. Generalmente se requiere información del entorno y conocimiento a priori para identificar situaciones.

iv. *Nivel 3: Evaluación de impacto (o refinamiento de amenazas)*. Se va a realizar la evaluación de la situación actual para valorar el impacto de las actividades detectadas en el nivel 2 y para obtener una perspectiva propia e identificar posibles riesgos, vulnerabilidades y oportunidades operacionales en el futuro.

v. *Nivel 4: Refinamiento de procesos (o gestión de recursos).* Consiste en gestionar los recursos de los sensores, para alcanzar su máxima eficiencia. Para ello llevan a cabo una planificación, prioridad de tareas, reserva y control de recursos.

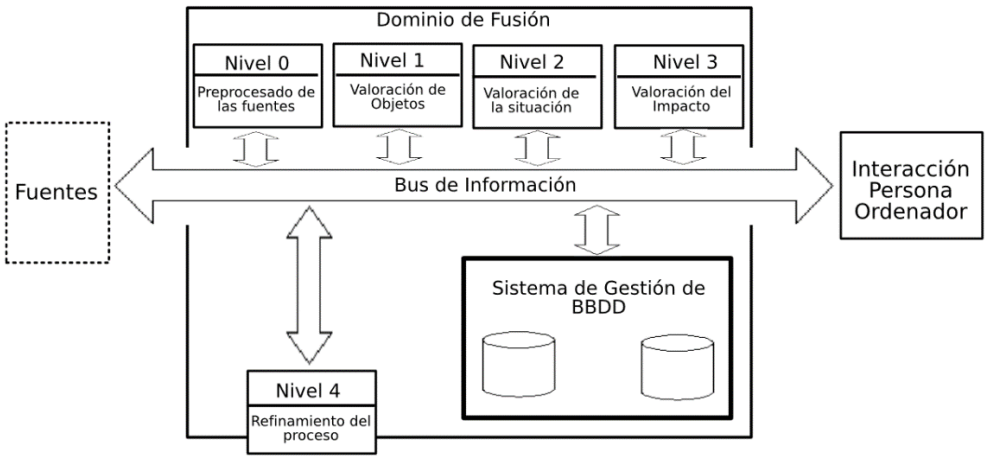


Figura 3.10 Clasificación del JDL (Fuente: [9]).

3.2.2 MÉTODOS Y ALGORITMOS EN LA FUSIÓN DE SENSORES

En esta sección se recogen los diferentes métodos y algoritmos más recurridos a la hora de llevar a cabo la fusión de sensores, considerando tres categorías diferentes: la asociación de datos, la estimación del estado y la fusión de decisión [31].

3.2.2.1 Técnicas de asociación de datos

A la hora de definir las técnicas de asociación de datos hay que tener en cuenta una serie de problemas a afrontar: la recepción de los datos en el nodo de la observación de cada sensor se da en intervalos de tiempo discretos y, muy frecuentemente, no concretos; las observaciones pueden provenir de los objetivos y del ruido; existe una falta de conocimiento a cerca de las observaciones que se generarán a cerca de un objetivo, etc. Por lo tanto, el fin de la asociación de datos es determinar el conjunto de observaciones o mediciones que se generan del mismo objetivo a lo largo del tiempo.

Este proceso de asociación de datos es un paso previo de la estimación del estado de los objetivos detectados y resulta clave porque, en caso de incoherencia en este proceso de asociación, la estimación se comportará incorrectamente. El proceso de asociación de datos también podría aparecer en todos los niveles de fusión, pero la granularidad varía según el objetivo de cada nivel.

Las técnicas más conocidas son: “vecinos cercanos” (*nearest-neighbour*), consiste en agrupar o seleccionar los datos que están más cercanos unos con otros [9]; “asociación de datos probabilística” (PDA), asigna una probabilidad a cada hipótesis asociada a una medida valida de un objetivo, estas medidas hacen referencia a las medidas que caen en la ventana de validación de un objecto en el instante de la medición; “asociación de datos probabilística (JPDA), se diferencia de las PDA en que las probabilidades de asociación se calculan considerando y combinando simultáneamente todas las observaciones e hipótesis; “modelos gráficos”, representan una descomposición condicional de la probabilidad conjunta, los nodos del gráfico denotan variables aleatorias, los bordes denotan la posible dependencia entre las variables aleatorias, y las placas denotan la replicación de una subestructura, con la indexación apropiada de las variables relevantes [31].

3.2.2.2 Técnicas de estimación del estado

Las técnicas de estimación del estado, también conocidas como técnicas de seguimiento, pretenden determinar el valor del estado (típicamente la posición de un objetivo) dadas las observaciones o medidas del propio objeto en movimiento.

La fase de estimación del estado es una componente de cualquier algoritmo de fusión desde el momento en que las observaciones pueden venir de sensores o fuentes distintas y el objetivo final es obtener un estado global del objetivo a partir de las observaciones. El problema de estimación implica, desde una perspectiva matemática, buscar los valores del vector de estado (posición, velocidad, tamaño, etc.) que mejor encajen con los datos observados. Estas observaciones se ven corrompidas por errores de medida, y por la propagación del ruido durante el proceso de medición.

Si consideramos la jerarquía del JDL previamente explicada en el apartado anterior, los métodos de estimación del estado se encuadran dentro del nivel 1 (valoración de objetos). Se puede hacer una distinción entre dos grupos dentro de estas técnicas, las dinámicas lineales y las no lineales. En las primeras, el problema de estimación tiene una solución estándar, el ruido sigue la distribución gaussiana y no nos referimos a él como un entorno desordenado; en este caso, la solución teórica óptima se basa en el filtro de Kalman [9].

- **Filtro de Kalman**

Propuesto por Kalman en 1960, este filtro de es la técnica de estimación más popular, combina datos medidos en intervalos de tiempo sucesivos entre sí para proporcionar una estimación de un parámetro con la máxima fiabilidad. El filtro utiliza un algoritmo en tiempo discreto para eliminar el ruido de las señales recibidas de los sensores para fusionar los datos y estimar el estado de cualquier sistema dinámico; puede estimar datos en el pasado (suavizado), presente (filtrado) y futuro (predicción). En el suavizado se reconstruye el cambio del proceso después de haberlo realizado, basado en los datos medidos. En el filtrado, el estado real del proceso es estimado utilizando una medición real y la información obtenida por las anteriores. La predicción, por su parte, requiere

una cantidad mucho mayor de información anterior y un modelo adecuado del sistema para crear una estimación fiable del estado real del proceso [33].

El filtro de Kalman consiste en asociar una medida de incertidumbre "P" a cada estimación calculada y se describe mediante dos entradas representadas en ecuaciones lineales. El uso de la incertidumbre es ventajoso ya que uno de los sensores tendrá mediciones mejores y más precisas que el otro.

Si el sistema se puede describir como un modelo lineal y el error se puede modelar como ruido Gaussiano, el filtro de Kalman obtiene estimaciones estadísticas óptimas, pero esto supone que a veces no sea práctico en las aplicaciones actuales en tiempo real. Es por esto por lo que, para tratar con modelos dinámicos no lineales y medidas no lineales, sean necesarios otros métodos.

La modificación del filtro de Kalman, conocida como *"Extended Kalman Filter*" (EKF) es un enfoque conocido para implementar filtros no lineales. En la última década, el filtro de Kalman extendido de Kalman (EKF), el filtro de Kalman extendido multiplicativo (MEKF) y el filtro de Kalman sin aroma (UKF) se han desarrollado para tratar otros problemas como la inestabilidad a la posición del sensor o actitud del sensor [33].

3.2.2.3 Técnicas de fusión de decisiones

Generalmente, en el dominio de fusión de datos, una decisión se toma en función del conocimiento de la situación percibida y proporcionada por varias fuentes. Estas técnicas de fusión de decisiones tienen como objetivo hacer una inferencia de alto nivel sobre los eventos y actividades que se producen a partir de los objetivos detectados [31]. Estos métodos se encuentran en el nivel 2 (evaluación de la situación) y el nivel 3 (evaluación del impacto) del modelo de fusión de datos JDL.

- **Método de inferencia Bayesiana**

La fusión de información basada en la inferencia Bayesiana proporciona un formalismo para combinar evidencia de acuerdo con las reglas de la teoría de probabilidades. La incertidumbre, representada en términos de probabilidades puede tener valores en el

intervalo [0,1] donde 0 indica falta total de creencia y 1 creencia absoluta. Se basa en la regla de Bayes:

$$P(Y|X) = \frac{P(Y|X)P(Y)}{P(X)} \tag{3.1}$$

- P(Y|X) representa la creencia en la hipótesis Y dada la información X.
- P(X|Y) probabilidad de obtener X, siendo la hipótesis Y cierta.
- P(Y) probabilidad a priori de la hipótesis.
- P(X) es una constante de normalización.

La principal desventaja de la inferencia bayesiana es que las probabilidades P(X) y P(X|Y) deben ser conocidas.

- **Métodos de inteligencia artificial**

Cuando se pretenden alcanzar inferencias de alto nivel, se requiere razonamiento humano como puede ser el reconocimiento de patrones, planificación, deducción y aprendizaje. Estos procesos de inferencia utilizados por sistemas se basan en un grupo de datos iniciales y unas reglas básicas. El aprendizaje profundo *("Deep Learning")* es ejemplo de este tipo de métodos.

El aprendizaje profundo es un nuevo campo del aprendizaje automático, un método de aprendizaje multinivel que utiliza componentes no lineales para transformar los datos de cada capa en características jerárquicas más abstractas. Las DNN y las CNN son los principales modelos de aprendizaje profundo.

A) Modelos DNN

La red neuronal profunda ("*Deep Neural Network*", DNN) se divide en tres categorías: capa de entrada, capas ocultas y capa de salida. A través de múltiples capas ocultas, una DNN puede aprender relaciones más complejas partiendo de datos de entrada. Además, cuantas más capas ocultas, más profundo es el aprendizaje de la red neuronal artificial y mayor es la precisión de las salidas con respecto a los resultados reales. Las neuronas, o las características de los datos de muestra, se encuentran en la capa de entrada, que es la primera en la estructura de una DNN, como se muestra en la Figura 3.11 En cada capa oculta, una función de activación transforma las salidas de los datos

para que sean la entrada de la siguiente capa oculta, repitiendo el proceso para cada capa oculta que haya en la DNN. Por último, la cantidad de neuronas en la capa de salida está determinada por el número de etiquetas de muestra, esta capa y la última capa oculta de la red se conectan mediante regresión logística [35].

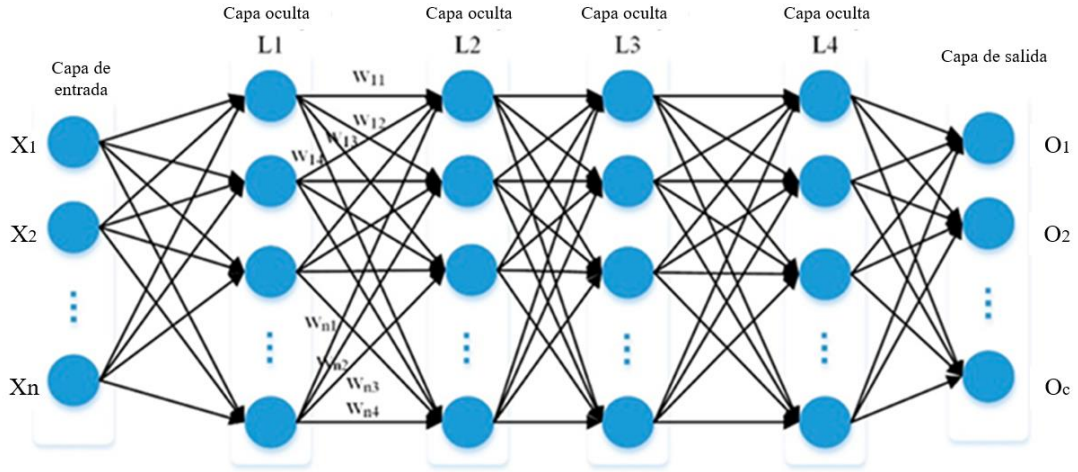


Figura 3.11 Estructura de la DNN (Fuente: Traducción de [10]).

B) Modelos CNN

Las redes neuronales convolucionales ("*Convolutional Neural Networks*", CNN) pueden tomar una entrada de la capa de entrada, asignar importancia a varios aspectos de esta (con pesos y sesgos) y ser capaces de diferenciarlos entre sí.

Las CNN tienen dos capas de red: las convolucionales y las de agrupación. Las primeras, operan los datos con convoluciones en lugar de con la multiplicación de matrices. Éstas se convierten en mapas de características, representados en planos.

Por otra parte, la capa de agrupación se encarga de reducir el tamaño de la característica o aspecto ya convolucionado, extrayendo las características dominantes y manteniendo el proceso de entrenamiento efectivo del modelo. Las capas de convolución y agrupación se conectan de forma alterna, donde reciben los datos de la capa anterior. Las capas de convolución y agrupación van seguidas de una capa totalmente conectada que aprende combinaciones no lineales de las características obtenidas de la salida convolucionada a través de un perceptrón multicapa con una

función de activación. La salida de la capa totalmente conectada se almacena en la capa de salida, como vectores unidimensionales [35].

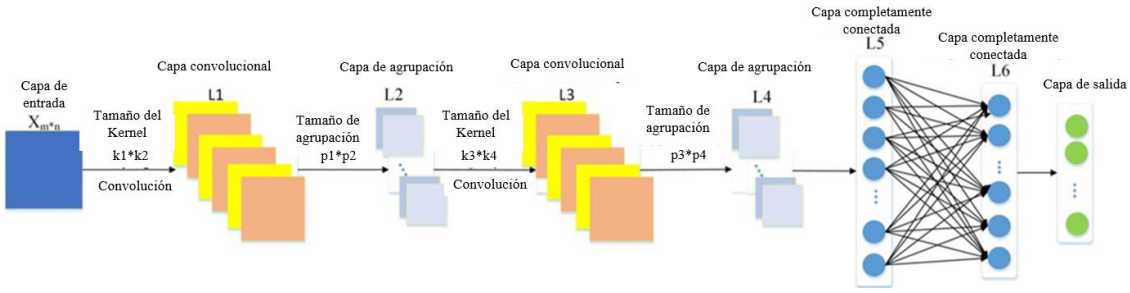


Figura 3.12 Estructura del CNN (Fuente: Traducción de [10]).

- **Método de mapeo de la cuadrícula de ocupación**

La cartografía del entorno es la base de la navegación en robótica, ya que los sistemas robóticos de éxito se basan en mapas para la localización, la planificación de la trayectoria y otros procesos de toma de decisiones de los robots. La necesidad de abordar la cartografía de un entorno desconocido, de modo que los resultados del mapa sean los mismos que el entorno, exige la creación de mapas de cuadrícula de ocupación.

Un mapa de cuadrícula de ocupación es un algoritmo en robótica probabilística diseñado para resolver problemas produciendo mapas 2D/3D que representan el espacio de trabajo del robot, partiendo de la recopilación de información proporcionada por los datos de los sensores.

Estos mapas consisten en dividir el espacio en celdas, donde cada celda tiene un valor de probabilidad de su estado (si está libre u ocupado por un obstáculo). A la hora de representar los mapas de cuadrícula de ocupación, existen 2 opciones, cuadrículas de ocupación binaria y cuadrículas de ocupación probabilística. Por un lado, en la ocupación de cuadrícula binaria se representan los espacios de trabajo ocupados con valores verdaderos y los espacios de trabajo libres con valores falsos, mostrando así dónde están los obstáculos y si el robot puede moverse por ese espacio. Por otro lado, la cuadrícula de ocupación probabilística utiliza valores de probabilidad para hacer representaciones más detalladas. Un valor cercano a 1 representa una alta certeza de que las celdas contengan obstáculos, mientras que la no ocupación se representa con

valores cercanos al 0. Una vez que se conoce la posición del robot, los datos de los sensores y el estado de la celda (ocupada o no), el robot comienza a trabajar en la rejilla de probabilidad. A medida que la posición cambia y los sensores siguen leyendo y las celdas se actualizarán [36].

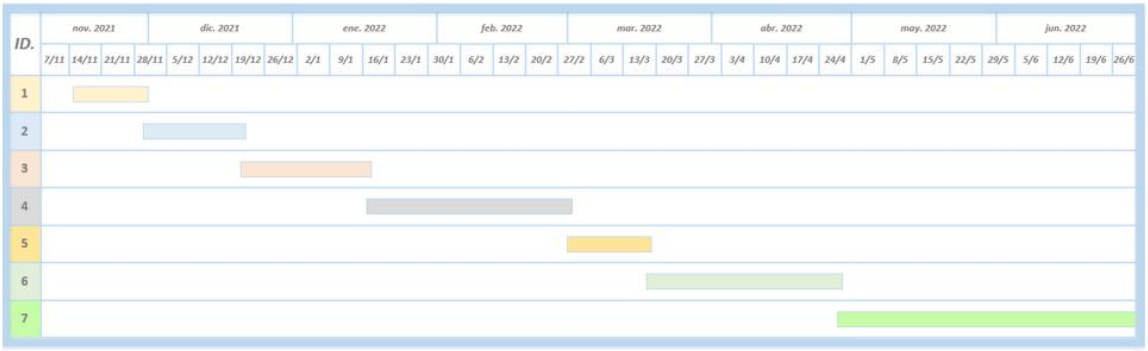
4. Desarrollo

4.1 FASES DE DESARROLLO

El desarrollo del trabajo se ha realizado de acuerdo con la planificación que queda reflejada en el siguiente diagrama de Gantt de la Tabla 4.1:

Tabla 4.1 Diagrama de Gannt del proyecto y tabla de correspondencias (Fuente: propia).

TAREA	ID.
Estudio del LiDAR y cámara	1
Estudio métodos de fusión de sensores	2
Aprendizaje del entorno de ROS	3
Desarrollo y creación del paquete de ROS de detección de objetos	4
Prueba y evaluación en un escenario del mundo real	5
Implementación identificación objetos	6
Redacción del proyecto escrito	7



Inicialmente se realizó un estudio de los diferentes equipos a utilizar, tanto del LiDAR como de la cámara, comprendiendo la tecnología y el funcionamiento general de ambos Tabla 4.1: [Tarea 1]. Dentro de este estudio previo al desarrollo práctico del proyecto, se indagó en las últimas investigaciones y publicaciones a cerca de los diferentes métodos de fusión de sensores y más exhaustivamente en el campo de la detección e identificación de objetos Tabla 4.1: [Tarea 2].

Una vez analizado el panorama, se inició el proceso de aprendizaje autodidacta del entorno de ROS. Para ello se hizo uso de la propia plataforma de aprendizaje de ROS, Wiki ROS [37], de otra serie de plataformas como "*The Construct*" [38] y de diversos foros Tabla 4.1: [Tarea 3].

Posteriormente se inició el desarrollo del propio paquete de ROS para la detección de objetos que se adaptase a los sensores disponibles. Para ello, se modificaron diversos paquetes ya creados y normalizados tanto por ROS como por otros creadores de renombre para ajustarse a las necesidades propias Tabla 4.1: [Tarea 4]. Paralelamente se fue evaluando el paquete en escenarios reales, concluyendo en una evaluación final en la que se corrigieron todos los errores finales Tabla 4.1: [Tarea 5].

Tras finalizar el desarrollo de la parte de detección de objetos, se comprendió que se podía llevar más allá de la detección y se implementaron una serie de herramientas que permitieron la identificación independiente de los objetos detectados. Así mismo, se realizaron las pruebas necesarias en escenarios reales y grabaciones del funcionamiento en vivo Tabla 4.1: [Tarea 6].

Finalmente se llevó a cabo la redacción de este documento en el que quedan reflejados todos los conocimientos adquiridos y los ensayos realizados Tabla 4.1: [Tarea 7].

4.2 TECNOLOGÍAS EMPLEADAS

Para el desarrollo del proyecto se ha hecho uso de distintos dispositivos físicos y de una arquitectura software que aporta los mecanismos de comunicación y razonamiento para integrar la información proporcionada por estos dispositivos.

4.2.1 DISPOSITIVOS FÍSICOS

Si bien ya se ha explicado en el "Apartado 3"de manera general el sensor lídar y la cámara de profundidad, conviene analizar más detalladamente los dispositivos utilizados para este proyecto, el "RPLIDAR A1M8" y la "Intel RealSense Depth Camera D435".

4.2.1.1 Análisis del RPLIDAR A1M8

Figura 4.1 RPLIDAR A1M8 (Fuente: [39]).

El escáner láser "RPLIDAR A1M8" es un sistema de escáner 2D giratorio de 360° desarrollado por SLAMTEC, capaz de detectar obstáculos en un rango de 0.2 a 6 metros, obteniendo datos precisos sobre la posición de estos [40]. Puede trabajar de forma excelente en todo tipo de entornos interiores y exteriores sin luz solar.

El láser gira en el sentido de las agujas del reloj y realiza tres lecturas. Su precisión y velocidad de muestreo también son convincentes, la frecuencia de escaneo de RPLIDAR A1 alcanza los 5,5 Hz al muestrear 360 puntos en cada ronda y puede configurarse hasta un máximo de 10 Hz. Además, el escáner "RPLIDAR A1M8" hace su trabajo usando menos de 5 mW, con un valor típico de 3 mW.

A parte de sus características técnicas, este lídar resulta interesante debido a su bajo coste y su compacto diseño, que permite instalarlo en robots móviles. Puede determinar la forma general de una habitación y su contenido, y resulta muy eficaz en todas las aplicaciones de vigilancia, cartografía o navegación autónoma.

A) Conexión del sistema

Figura 4.2 Conexión del sistema (Fuente: propia).

El RPLIDAR A1 contiene un sistema de escáner de alcance y un sistema de motor. Tras encender cada subsistema, el "RPLIDAR A1" empieza a girar y a escanear en el sentido de las agujas del reloj. El usuario puede obtener datos del escáner de alcance a través de la interfaz de comunicación (puerto serie /USB) [40].

El lídar cuenta con un sistema de detección y adaptación de la velocidad, ajustando la frecuencia del escáner láser automáticamente según la velocidad del motor. Así mismo, el sistema anfitrión puede obtener la velocidad real del RPLIDAR A1 a través de la interfaz de comunicación.

B) Funcionamiento

Este lídar basa su funcionamiento en el principio de la triangulación láser y utiliza un hardware de adquisición y procesamiento de visión de alta velocidad desarrollado por SLAMTEC.

El sistema, montado en un rotador giratorio con un sistema de codificación angular incorporado, mide los datos de distancia en más de 2000 veces por segundo y con una salida de alta resolución de la distancia (<1% de la distancia) en una exploración de 360 grados del entorno.

El sistema RPLIDAR A1 utiliza el láser infrarrojo de baja potencia para emitir la señal, accionándolo mediante un pulso modulado y esta señal es reflejada por el objeto a detectar, véase la Figura 4.3. La señal devuelta es muestreada por el sistema de adquisición de visión en el RPLIDAR A1 y el procesador de señales digitales integrado en el RPLIDAR A1 comienza a procesar los datos de la muestra, el valor de la distancia de salida y el valor del ángulo entre objeto y el RPLIDAR A1, a través de la interfaz de comunicación.

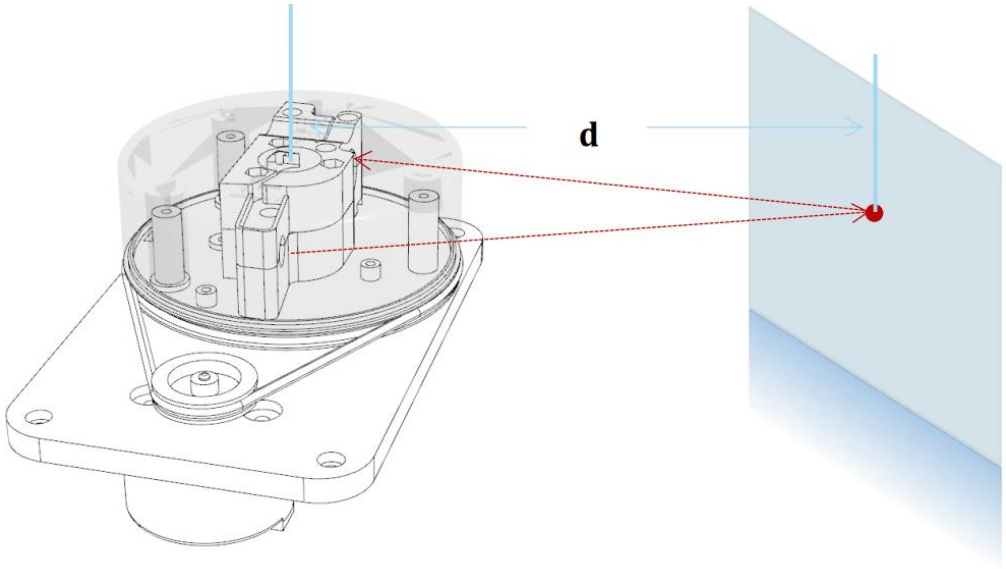


Figura 4.3 Principio de funcionamiento del RPLIDAR A1(Fuente: [40]).

El láser emite en un tiempo muy corto, garantizando la seguridad de las personas y animales domésticos, y alcanzando la norma de seguridad láser de Clase I, por la que un láser se considera seguro en todas las condiciones de utilización razonablemente previsibles según la UNE EN 60825-1 /A2-2002 [41]. Cuando el RPLIDAR A1 está funcionando, los datos de muestreo saldrán a la interfaz de comunicación. Cada punto de muestreo contiene la siguiente información [40]:

- Distancia (mm), entre el núcleo giratorio del lídar y el punto de muestreo.
- Grado de orientación (grados), de la medición.
- Calidad (nivel) de la medición.
- Bandera de inicio (booleano), marcando el inicio de un nuevo escaneo.

4.2.1.2 Análisis “Intel® RealSense™ Depth Camera D435”

Figura 4.4 Cámara “Intel® RealSense™ D435” (Fuente: [42]).

La cámara de profundidad “Intel® RealSense™ D435” es una solución estéreo que ofrece profundidad de calidad para una gran variedad de aplicaciones [42].

Esta cámara forma parte de la serie de cámaras “Intel® RealSense™ D400”. Este grupo de cámaras toma el hardware y software de detección de profundidad más reciente de Intel y los empaqueta en productos fáciles de integrar, ofreciendo una integración sencilla lista para usar y permitiendo una generación completamente nueva de dispositivos equipados con visión inteligente.

El campo de visión de la “Intel® RealSense™ D435” tiene un alcance de hasta 10 m, con un comportamiento ideal en el rango de 0,3m a 3m y está diseñada para utilizarla en un entorno tanto de interiores como de exteriores.

La D435, alimentada por USB, consta de un par de sensores de profundidad, un sensor RGB y un proyector de infrarrojos. Por lo que respecta a su tecnología de profundidad, se sirve de la técnica de profundidad estéreo ya comentada en el apartado anterior. Cuenta con un campo de visión de profundidad de 87° × 58°, una profundidad mínima de 28 cm a resolución máxima (de hasta 1280 x720) y una velocidad de hasta 90 fotogramas por segundo en profundidad.

En cuanto a su imagen RGB, tiene un campo de visión de 69° × 42°, una resolución de 1920x1080 a una velocidad de 30 fotogramas por segundo y una resolución del sensor RGB de 2MP [43]. Así mismo, el módulo de la cámara es el correspondiente a la línea de Intel RealSense D430 junto con la cámara RGB y la placa de procesador de visión es la propia D4 de Intel RealSense.

A su vez, es compatible con el Intel RealSense SDK 2.0. Este kit de desarrollo de software admite un procesamiento rápido de datos 3D de código abierto, que incluye la reconstrucción de escenas, visualización y aprendizaje automático en 3D, autocalibración de la cámara en menos de 15 segundos, etc. [44]. Al ser altamente personalizable, proporciona una solución de bajo coste que es liviana y poderosa, lo que permite el desarrollo de soluciones de detección que pueden comprender e interactuar con su entorno.

Gracias a sus dimensiones 90mm x 25 mm x 25 mm (longitud, profundidad y altura respectivamente) se puede integrar con facilidad en cualquier parte. Esto junto con sus características técnicas relacionadas con el amplio campo de visión, la convierten en la solución preferida para aplicaciones como la navegación robótica, la realidad virtual y aumentada o el reconocimiento de objetos, donde ver la mayor parte posible de la escena es de vital importancia.

4.2.2 SOFTWARE: ROBOT OPERATING SYSTEM (ROS)

Robot Operating System (ROS), traducido al castellano como "sistema operativo robótico", es un meta sistema operativo de código abierto para el desarrollo de software para robots [45]. ROS proporciona los servicios que se esperan de un sistema operativo, como el control de dispositivos de bajo nivel, la abstracción de hardware, la implementación de funcionalidades de uso común, el paso de mensajes entre procesos y la gestión de paquetes mediante el uso de las herramientas y las bibliotecas adecuadas para construir, escribir y ejecutar código en múltiples ordenadores.

A diferencia de otras plataformas robóticas de software, el objetivo principal que define a ROS es "soportar el reúso de código en la investigación y desarrollo dentro de la robótica" [45].

El diseño de ROS va desde el nivel del sistema de archivos hasta el nivel de la comunidad ROS. Este diseño, junto con las herramientas de infraestructura propias que brinda ROS, va a permitir llevar a cabo decisiones independientes de desarrollo e implementación; pero perfectamente acoplables entre ellas.

Conviene entender la ejecución de procesos en ROS como una red de procesos acoplados entre sí que se sirven de la propia comunicación de ROS. Estos procesos se representan en una arquitectura de grafos donde el procesamiento tiene lugar en nodos que pueden publicar, recibir o multiplexar información procesada de sensores, control, actuadores, etc. Esta estructura distribuida basada en nodos es la que permite el diseño individualizado, pero fácilmente acoplable a los demás procesos. Estos procesos se pueden agrupar en paquetes y pilas, que, a su vez, pueden ser intercambiados, compartidos y distribuidos.

Así mismo, ROS soporta un sistema federado de repositorios de código, que permite distribuir dicha colaboración y que ha permitido crecer a esta comunidad en los últimos años.

En apoyo a este objetivo principal de colaboración, se reúnen una serie de características dentro del marco ROS [45]:

1. ROS está diseñado para ser lo más liviano posible, de tal manera que, el código escrito para ROS se pueda usar con otras estructuras de software de robots.

2. El modelo de desarrollo preferido consiste en escribir bibliotecas agnósticas en ROS con interfaces funcionales limpias.
3. ROS es fácil de implementar en cualquier lenguaje de programación moderno, ya sea Python, C ++ y Lisp, y cuenta con bibliotecas experimentales en Java y Lua.
4. Para facilitar la activación y desactivación de los dispositivos de prueba, ROS cuenta con un marco de prueba de integración incorporado llamado "*rostest*".
5. ROS es apropiado para grandes sistemas de rutinas de ejecución y para grandes procesos de desarrollo.

Aunque no es un entorno de trabajo en tiempo real, existe la posibilidad de integrar ROS con código en tiempo real. Recientemente, en un intento de implementar ROS como un entorno de trabajo en tiempo real, se ha desarrollado ROS 2, que aprovechará las bibliotecas y tecnologías modernas para añadir soporte para código en tiempo real y hardware embebido.

A) Conceptos fundamentales

Conviene distinguir los 3 niveles de conceptos dentro de ROS: el nivel del sistema de archivos, el nivel del gráfico de computación y el nivel de la comunidad.

- *Nivel del sistema de archivos:*

  Este nivel hace referencia a los recursos de ROS que se encuentran en el disco. Entre ellos, los paquetes son la principal unidad de organización del software. Estos paquetes pueden contener los procesos de tiempo de ejecución de ROS, las bibliotecas dependientes de ROS, los conjuntos de datos, los archivos de configuración, o cualquier información que convenga organizarla en conjunto.

  Así mismo, se pueden encontrar en este nivel: los metapaquetes, paquetes especializados que representan un grupo de otros paquetes relacionados; los manifiestos de paquetes , archivos que proporcionan metadatos sobre un paquete; los repositorios, paquetes que comparten la misma versión y se pueden lanzar juntos; los tipos de mensajes, definen las estructuras de datos de los mensajes enviados por ROS ; los tipos de servicios, definen las estructuras de datos de solicitud y respuesta para servicios en ROS.

- *Nivel de gráficos de computación:*

  Considerando una estructura gráfica, los procesos de ROS se representan como nodos conectados por aristas denominadas tópicos. Cada nodo tendrá la posibilidad de intercambiar mensajes con otros nodos o de establecer y recuperar datos compartidos de una base de datos común llamada servidor de parámetros. Esto es posible gracias al ROS Master (maestro), aquí se registran todos los nodos y sus comunicaciones a través de los tópicos, garantizando la actualización constante del servidor de parámetros. El maestro establece una comunicación entre iguales (peer-to-peer) entre los procesos que tienen lugar entre los nodos, sin necesidad de que los mensajes y las llamadas de servicio pasen por él.

  Resulta imprescindible analizar más detenidamente los elementos en los que se basa ROS para comprender todo su proceso de comunicación: nodos, mensajes, tópicos, servicios y el servidor de parámetros.

  El **nodo**, considerado el centro de la programación de ROS, representa un proceso que ejecuta el gráfico de ROS y se encarga de realizar cierta tarea. Por ejemplo, un nodo puede encargarse de tareas como la localización, la planificación de rutas, el análisis de datos de un sensor, etc.

  Una vez que el maestro ha registrado el nombre de cada nodo, éste podrá realizar acciones basadas en la información recibida de otros nodos, enviar información a otros nodos o enviar y recibir peticiones de acciones hacia y desde otros nodos a través de mensajes. Estos **mensajes** entre nodos son una estructura de datos simple, que comprenden campos escritos, incluyendo enteros, booleanos, puntos flotantes, matrices, estructuras anidadas, etc.

  Los mensajes se enrutan a través de un sistema de transporte del tipo publicación/suscripción sirviéndose de los tópicos. Estos **tópicos** son los diferentes nombres que se utilizan para identificar el contenido del mensaje. Para enviar mensajes a un tópico, el nodo los publicará en tópico determinado y los nodos interesados en esa información estarán suscritos a ese tópico. Puede haber múltiples editores y suscriptores para un mismo tópico, y un mismo nodo puede publicar y/o suscribirse a múltiples tópicos de forma anónima.

La solicitud/respuesta que anuncian los nodos se realiza a través de **servicios**, que se definen mediante un mensaje de solicitud y otro de respuesta. Un nodo proveedor ofrece un servicio bajo un nombre y un cliente utiliza el servicio enviando el mensaje de solicitud y esperando la respuesta.

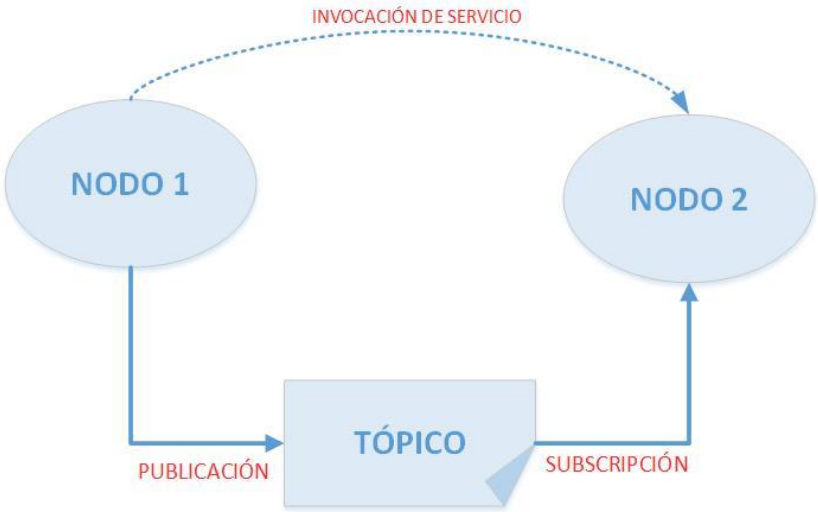


Figura 4.5 Funcionamiento protocolo de comunicación ROS (Fuente: propia).

Por último, el **servidor de parámetros** es la base de datos compartida que permitirá el acceso de la comunidad a la información y forma parte del maestro.

- *Nivel de la comunidad de ROS*

  Son los recursos que ROS ofrece a su comunidad para permitir el intercambio de conocimientos y software, es decir, sus distribuciones, repositorios, su propia wiki, un sistema de seguimiento de incidentes, las listas de correo, ROS respuestas y su blog.

B) Sistemas operativos

El software para ROS se prueba principalmente en sistemas Ubuntu y Mac OS X, aunque la comunidad ROS ha estado contribuyendo con soporte para Fedora, Gentoo, Arch Linux y otras plataformas Linux. ROS actualmente solo se ejecuta en plataformas basadas en Unix y en W10 mediante virtualización [45].

4.2.2.1 Paquetes y herramientas prediseñadas de ros

La comunidad de ROS ofrece una serie de paquetes y herramientas prediseñados que van a resultar clave para el desarrollo del propio paquete final. Se pretende en este apartado explicar únicamente los más importante y necesarios de comprender.

- ***'darknet_ros', YOLO: Real-Time Object Detection***

Este paquete está desarrollado para la detección de objetos en imágenes de cámara integrando YOLO [46]. El acrónimo YOLO significa "*You Only Look Once*" (solo se mira una vez) y se refiere a un algoritmo de detección de objetos en tiempo real que emplea redes neuronales convolucionales (CNN) y que se diferencia de otros por su alta velocidad de detección, su capacidad de aprendizaje y su gran precisión. Mientras que otros algoritmos, que realizan la detección en varias propuestas de regiones y acaban realizando la predicción varias veces para las diversas regiones de una imagen, YOLO pasa la imagen una vez por la red neuronal y emite su predicción. Se basa en una decisión final de detección realizada mediante la combinación de tres técnicas: bloques residuales, regresión de cajas delimitadoras e intersección sobre unión ("*Intersection Over Union*", IOU).

La primera consiste en dividir la imagen en varias cuadrículas de la misma dimensión, si el centro de un objeto cae en una celda de la cuadrícula, la detección de ese objeto será responsabilidad de esa celda de la cuadrícula.

La caja delimitadora, conocido en inglés como "*bounding box*", es un contorno que resalta un objeto en una imagen. Cada caja delimitadora en la imagen consta de los siguientes atributos: el centro de la caja delimitadora (x,y), la anchura (w), la altura (h) y la clase. Cada celda de la cuadrícula predice las cajas delimitadoras y los valores de confianza de esas cajas.

La intersección sobre la unión (IOU) permite saber cómo se solapan las cajas. YOLO utiliza IOU para proporcionar una caja de salida que rodea perfectamente los objetos. Cada celda de la cuadrícula predice las cajas delimitadoras y los valores de confianza de esas cajas, reflejando la confianza que tiene el modelo en que la caja contenga un objeto y también la precisión que tiene la caja predicha. La predicción de confianza representa el IOU entre la caja predicha y las cajas reales. Al utilizar esta técnica, se garantiza que las cajas

delimitadoras predichas sean exactamente iguales a las cajas reales y las cajas delimitadoras innecesarias se eliminan cuando las características (anchura, altura clase, etc.) del objeto no se cumplen.

El paquete de *'darknet_ros'* va a detectar clases preentrenadas incluidas en los conjuntos de datos de PASCAL VOC y COCO o incluso puede crear una red con sus propios objetos de detección.

COCO es un conjunto de datos de detección de objetos y segmentación a gran escala. Se caracteriza por lo siguiente: segmentación de objetos, reconocimiento en contexto, segmentación de material superpixelado, contiene 330K imágenes (>200K etiquetadas), dispone de 1,5 millones de instancias de objetos, diferencia 80 categorías de objetos y 5 leyendas por imagen [25].

Por otra parte, PASCAL VOC, acrónimo de *"Pattern Analysis, Statistical Modelling, and Computational Learning Visual Object Classes"* proporciona conjuntos de datos de imágenes estandarizados para más de 20 clases diferentes para tareas de detección de objetos, segmentación semántica u otras tareas de clasificación.

Basándose en el conjunto de datos Pascal VOC 2012, YOLO puede detectar 20 clases de objetos y basándose en el conjunto de datos COCO, puede detectar hasta 80 clases de objetos, véase la Figura 4.6.

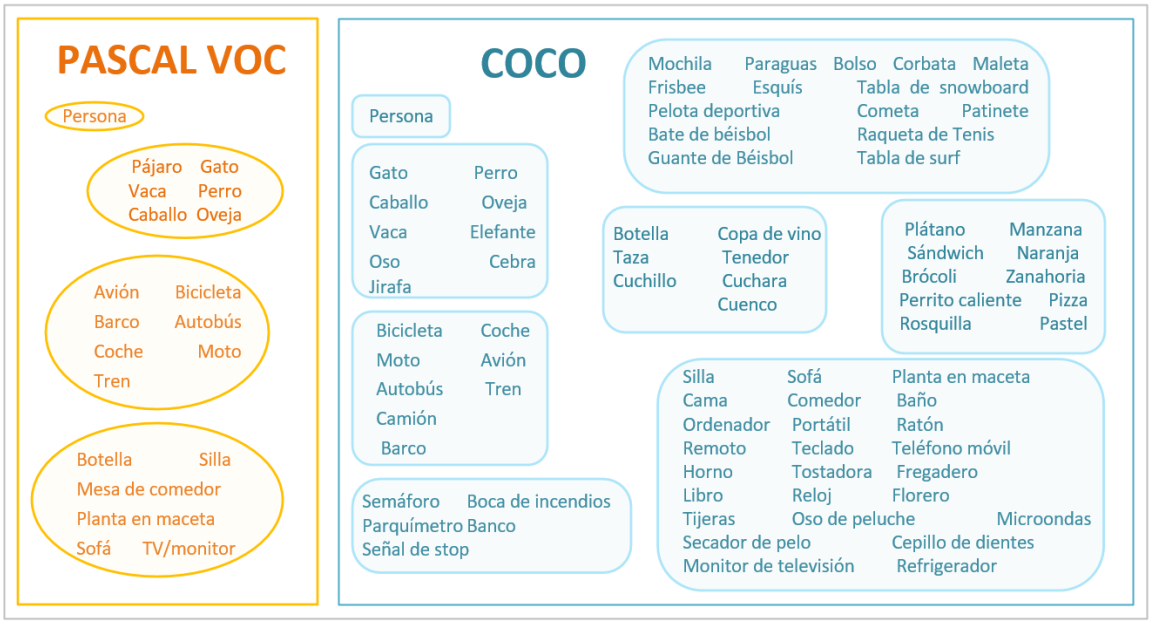


Figura 4.6 Clases de objetos de los Datasets Pascal VOC y COCO (Fuente: propia).

- ***'obstacle-detector-fusion'***

Este paquete se utiliza para detectar y rastrear objetos a partir de datos proporcionados por un escáner lídar 2D, el "RPLiDAR A1", y una cámara de profundidad, la "Intel RealSense D435" [47]. El principio de funcionamiento de la detección se basa en gran medida en el propio paquete *'obstacle_detector'* [48].

El paquete *'obstacle_detector'* proporciona utilidades para detectar y rastrear obstáculos a partir de los datos proporcionados por los escáneres láser 2D. Los obstáculos detectados vienen en forma de segmentos de línea o círculos. Utiliza un método de agrupamiento basado en la densidad para agrupar nubes de puntos, *PointCloud,* y crear una representación geométrica de objetos dentro de la vecindad del sensor. Primero, la cámara detecta los objetos y luego la medición se corrige con el resultado de detección del lídar, de ahí el nombre de "fusión". El estado dinámico de los objetos detectados (posición, velocidad, aceleración) se estima utilizando un filtro de partículas [48].

- ***'rplidar'***

Este paquete simplemente proporciona el manejo básico en ROS de los dispositivos para el escáner láser del 2D RPLIDAR A1/A2 y A3 y está desarrollado por los propios creadores Slamtec [49]. Va a permitir configurar el lídar y establecer la comunicación ROS entre el dispositivo y el ordenador. El RPLIDAR A1 realiza mediciones de distancia a alta velocidad con más de 2 000/4 000 muestras por segundo. Para un escaneo que requiere 360 muestras por rotación, se puede lograr la frecuencia de escaneo de 10 Hz. Los usuarios pueden personalizar la frecuencia de escaneo de 2 Hz a 10 Hz libremente controlando la velocidad del motor de escaneo. El controlador publica datos *sensor_msgs/LaserScan* dependientes del dispositivo.

- ***'rviz'***

Esta herramienta 'rviz', abreviatura de "visualización ROS", es una herramienta de software de visualización 3D para robots, sensores y algoritmos. Permite ver la percepción del robot de su mundo (real o simulado). El propósito de rviz es permitir la representación precisa de lo que ocurre en el entorno de un robot, utilizando los datos de los sensores.

4.3 ENSAYOS REALIZADOS

En este apartado queda reflejada la manera de proceder en cuanto a la parte práctica del proyecto, es decir, el desarrollo del paquete final de ROS. El objetivo final es demostrar cómo se ha llegado a la opción óptima para el equipo disponible, teniendo en cuenta que la meta final era poder llevar a cabo la identificación y detección de objetos a través del lídar y la cámara.

4.3.1 LASERSCAN + YOLO

En este primer intento se pretendía utilizar los datos de tipo *LaserScan* que proporciona el lídar, encontrar la distancia al objeto escaneado y asociar el ángulo con el ángulo en YOLO. Para implementar el sistema de detección de objetos en imágenes YOLO, se utilizó el paquete prediseñado de ROS, *'darknet_ros'* [46]. A partir de esta detección se generaban las "cajas delimitadoras", que identificaban los objetos que la cámara que se utilizó en un principio, la Logitech c310 HD, captaba en su campo de visión.

La obtención de datos de la cámara se realizó a partir de otro paquete prestablecido de ROS, *'usb_cam'* [50]. De tal forma que, se publicaban los mensajes del sensor en el tópico *'image_raw'*, al que estaba suscrito *'darknet_ros'* y finalmente se visualizaba todo a partir de la herramienta ROS *'rviz'.*

Una vez conseguidos los datos publicados por el sistema de detección YOLO, se trató de combinarlos con los datos de lídar. Estos datos se obtuvieron utilizando el paquete prediseñado de ROS *'rplidar'* [49], esto creaba el nodo *'rplidarNode'* que leía el escaneo en bruto del lídar utilizando el SDK de RPlidar y lo convertía en un mensaje del tipo *LaserScan*. Sin embargo, era muy complejo comparar los datos de profundidad del lídar con una cámara 2D que no proporcionaba ese tipo de datos y no daba la posición espacial del objeto desde YOLO.

4.3.2 IMPLEMENTACIÓN DEL RECONOCIMIENTO DE OBJETOS LIDAR

El segundo intento consistió en convertir los datos *LaserScan* del lídar en datos del tipo *PointCloud* y de esta manera implementar un método de clustering, que, combinado con filtros de Kalman, pudiera detectar objetos.

En primer lugar, la conversión del *LaserScan* a *PointCloud* se realizó creando un nuevo paquete propio *'laser2pc'* a partir del paquete prediseñado de ROS *'laser_geometry'* [51]. De esta manera, los datos *PointCloud* se suscribían al nodo correspondiente para poder realizar posteriormente la detección.

Para implementar el detector de objetos, se utilizó el paquete *'multiple-object-tracking-lidar'* [52]. El detector de objetos podía entonces publicar los datos de posición de un máximo de 6 objetos a partir de la entrada de datos lídar. Con estos datos, se pretendía fusionarlos con los datos de posición del software de detección YOLO.

Nuevamente, no fue posible debido a la falta de datos de profundidad de la cámara 2D.

Se comprendió entonces que para poder llevar a cabo una correcta fusión de los datos y poder solucionar estos problemas, se iba a requerir una cámara de profundidad. Se configuró entonces la “Intel RealSense Depth Camera D435”.

4.3.3 LIDAR Y ESCANEO DE LA IMAGEN DE PROFUNDIDAD

Convirtiendo la imagen de profundidad de la nueva cámara en datos del tipo *LaserScan* y utilizando los datos *LaserScan* del lídar, es posible integrar los datos de distancia y de reconocimiento de objetos de ambos y fusionarlos.

Se implementó el paquete *‘object-detector-fusion’* [47]. Este configura la cámara RealSense utilizando el ROS Wrapper para dispositivos Intel RealSense *'realsense-ros'* [53] desactivando las características que no son necesarias y configura el RPLidar para publicar los datos en *LaserScan*, haciendo uso del paquete *'rplidar'*. Así mismo, convierte los datos de profundidad de la cámara en *LaserScan*. Finalmente, calcula la posición media de los objetos detectados utilizando los datos de posición de los sensores lídar y de la cámara de profundidad.

Una vez implementada la parte de detección de objetos a partir de la fusión de datos de la cámara y el lídar, se complementó el paquete con la idea original de identificación de

objetos a través de YOLO. Así pues, el paquete final sería capaz de realizar la detección de objetos, visualizándose en la herramienta de ROS *'rviz'* y proporcionando los datos de velocidad relativa, la aceleración y la dirección del objeto; y, de manera independiente, la identificación de objetos de YOLO.

4.3.3.1 Análisis gráfico de ros

Para una mayor comprensión del paquete de ROS, conviene repasar y analizar el gráfico de cálculo de ROS que la propia herramienta de ROS, *'rqt_graph'* genera, véase la Figura 4.7 :

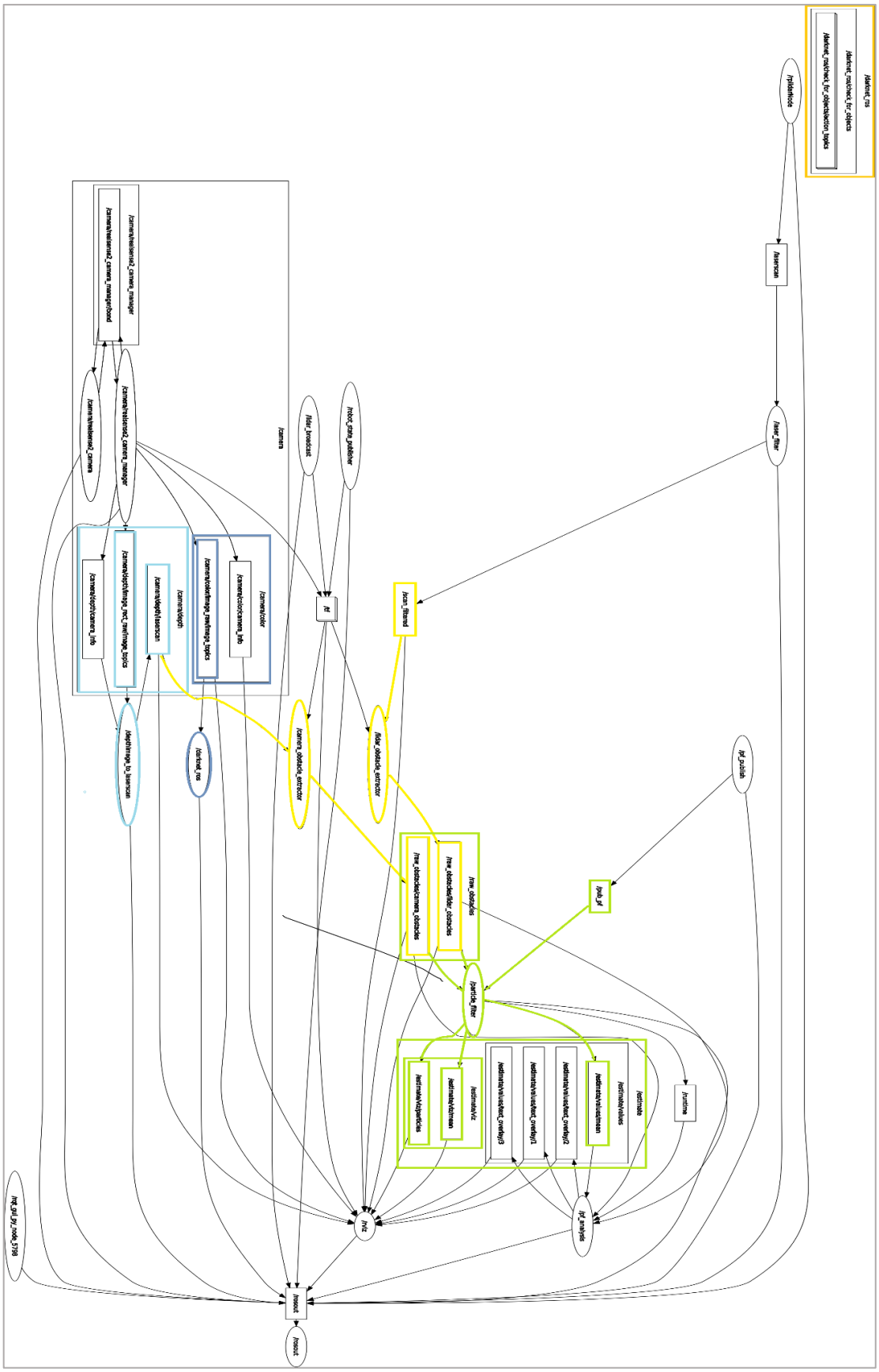


Figura 4.7 Gráfico de cálculo general de 'rqt_graph' (Fuente: propia).

Para convertir los datos de profundidad 2D en datos de profundidad 3D, este paquete hace uso de un paquete ROS prediseñado *'depthimage-to-laserscan'* [54]. Este tiene la función de tomar una imagen de profundidad y genera un escaneo láser 2D basado en los parámetros proporcionados, para lograr esto, este paquete suscribe los mensajes de tipo *'sensor_msgs/Image'* en el tópico *'camera/depth/image_rect_raw/image_topics'* y publica *'sensor_msgs/LaserScan'*, como se observa en el gráfico ampliado de la Figura 4.8:

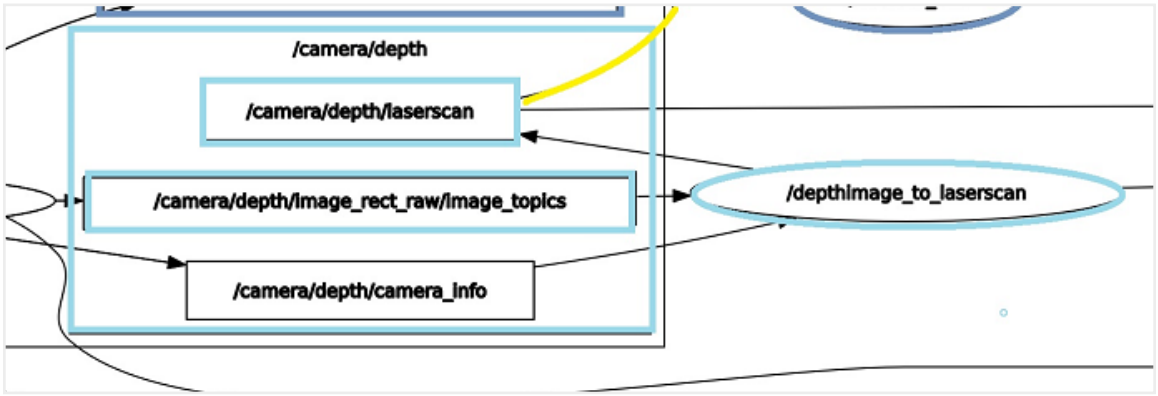


Figura 4.8 Gráfico ampliado, sección de 'depthimage-to-laserscan' (Fuente: propia).

La parte de detección de obstáculos se basa en el paquete prediseñado, *'obstacle_detector'* [48]. Este paquete en un principio está diseñado para procesar los datos en el siguiente orden: dos escaneos láser, fusión de escaneos, escaneo fusionado o pcl, extractor de obstáculos, obstáculos, rastreador de obstáculos y obstáculos refinados. Sin embargo, en el escenario de este proyecto, es suficiente con la extracción pura de obstáculos directamente de un escaneo láser, sin necesidad seguimiento.

Se implementa dos veces este paquete, una para el extraer los obstáculos detectados por el lídar y otro para la cámara:

La primera, correspondiente al lídar, se subscribe al tópico *'/scan_filtered'* y estaría publicando datos en un nuevo tópico *'/raw_obstacles/lidar_obstacles'*, resaltado en amarillo en la Figura 4.9 . Lo mismo se aplica a la cámara, pero en este caso, su nodo

extractor está suscrito a su propio tópico *'/camera/depth/laserscan'*, resaltado en azul clarito en la Figura 4.9 y publicará sus datos en el tópico *'raw_obstacles/camera_obstacles'*:

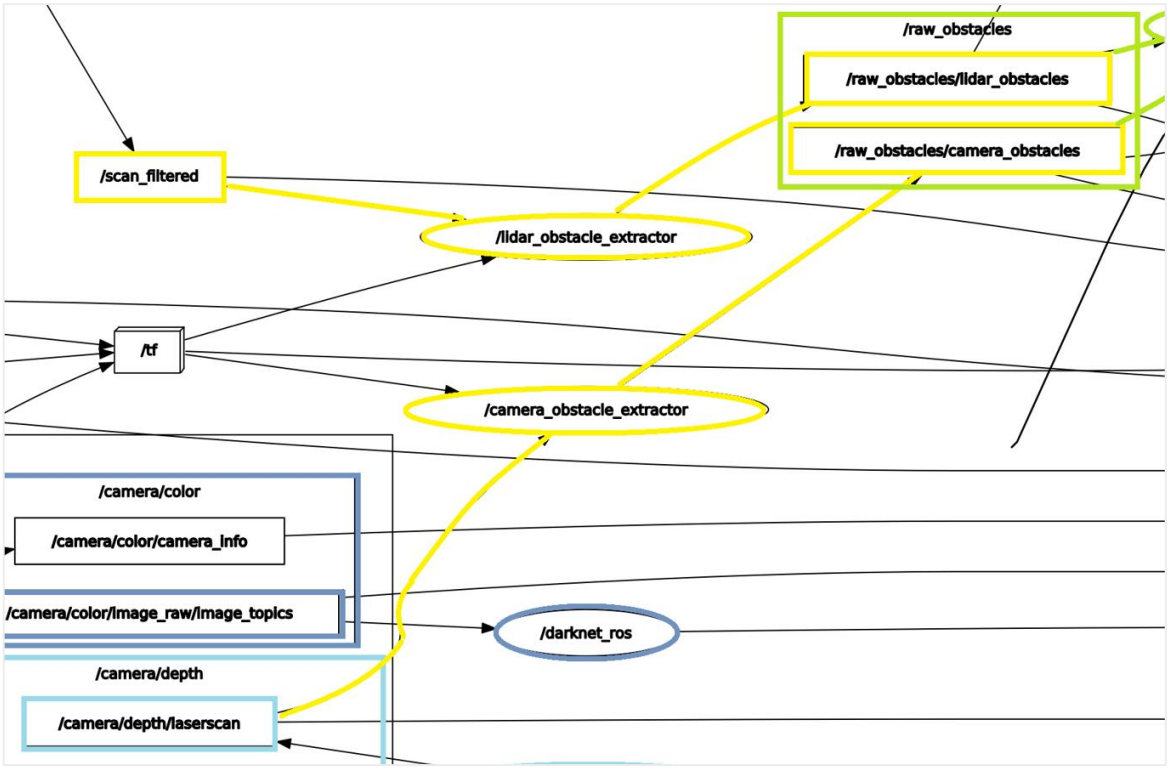


Figura 4.9 Gráfico ampliado, sección 'obstacle_detector' (Fuente: propia).

Por último, como se refleja en la ampliación del gráfico en la Figura 4.10 se crea un nuevo nodo, '*particle_filter*', que utilizará los datos de medición de los resultados de estos objetos detectados tanto para estimar el estado dinámico de un objeto como para reducir el ruido de las mediciones. Para ello se suscribe a los tópicos de '*/raw_obstacles*' procedentes del proceso de detección de obstáculos y publica la información en los tópicos de *'/estimate'*:

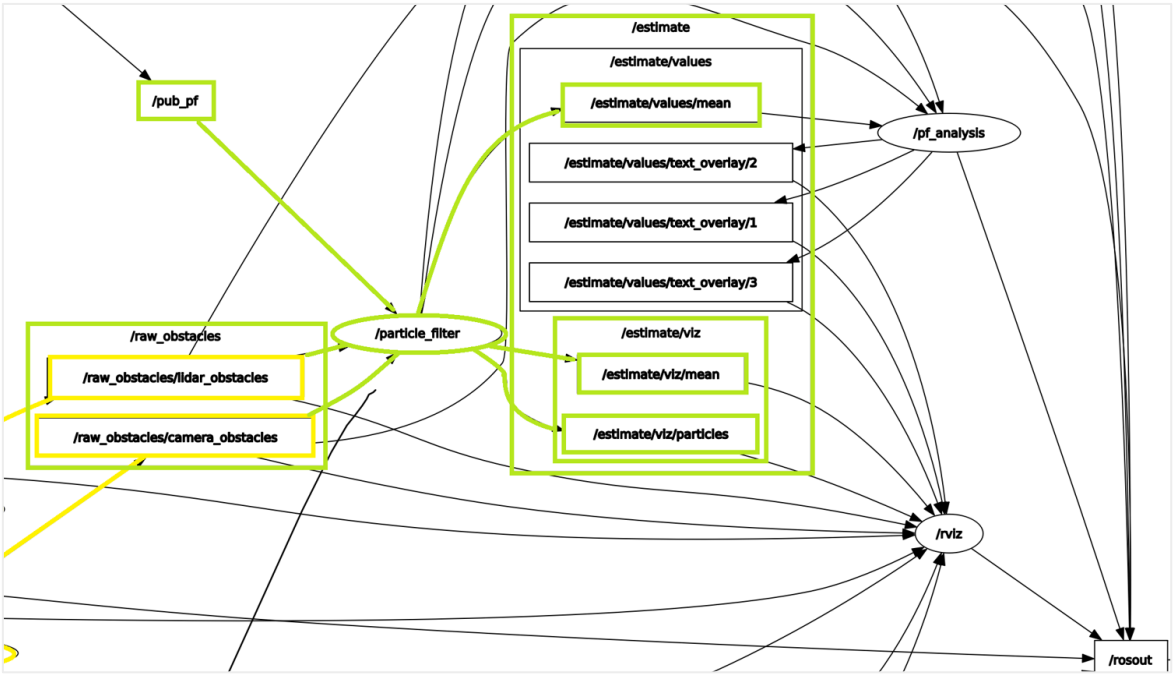
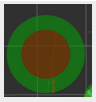


Figura 4.10 Gráfico ampliado, sección 'particle_filter' (Fuente: propia).

Como es habitual, todo esto se visualiza con la herramienta propia de ROS, *'rviz'*, acorde a la siguiente leyenda:

Tabla 4.2 Leyenda de figuras en 'rviz' (Fuente: propia).

Figura		*Equivalencia*
Círculos verdes con círculo concéntrico rojo		*Objetos detectados a partir de los datos de la cámara*
Círculo beige con círculo concéntrico azul		*Objetos detectados de los datos lídar*
Caja magenta y puntos negros que la rodean		*Media de las partículas (representadas por puntos negros) utilizadas para aproximar el estado.*

Además, cuenta con un texto superpuesto en un recuadro negro en el que se representan los datos de posición, velocidad y aceleración antes mencionados. Por defecto proporciona los datos correspondientes a tres objetos, pero es posible cambiarlo.

La visualización en conjunto sería como la siguiente, véase la Figura 4.11:

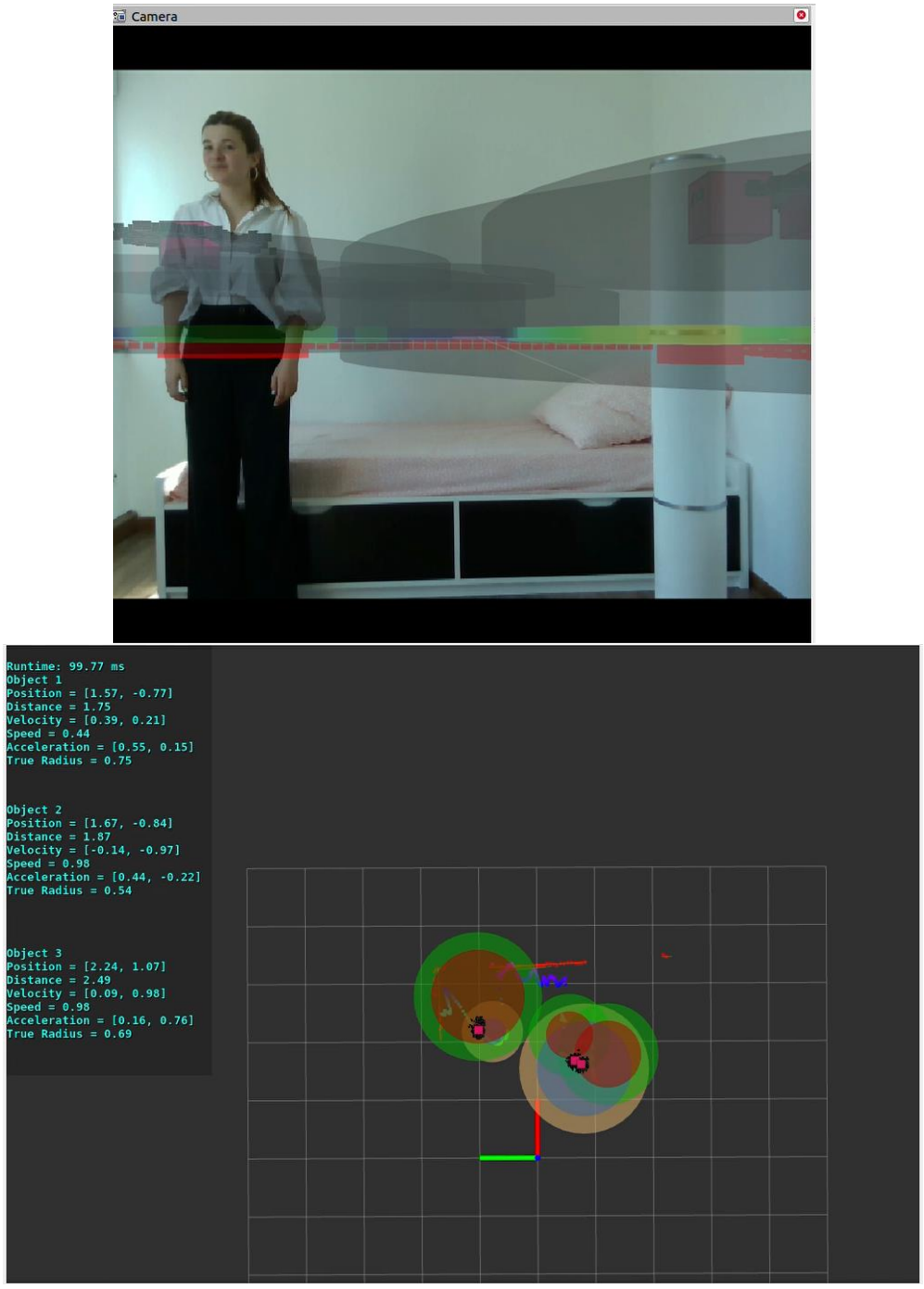


Figura 4.11 Visualización en'rviz' (Fuente: propia).

Para lograr esta visualización, ha sido necesario implementar un paquete adicional *'jsk_visualization'* [55], que es una pila para los paquetes de visualización que se utilizan en JSK lab.

Por lo que respecta a la parte de YOLO, el propio paquete de *'/darknet_ros'* va a obtener los datos de la cámara a partir del paquete de la cámara, *'/camera/realsense2_camera_manager',* como se observa en el gráfico de la Figura 4.12. De tal forma que, se publican los mensajes del sensor en el tópico *'/camera/color/image_raw/image_topics'*, al que está suscrito *'darknet_ros'*.

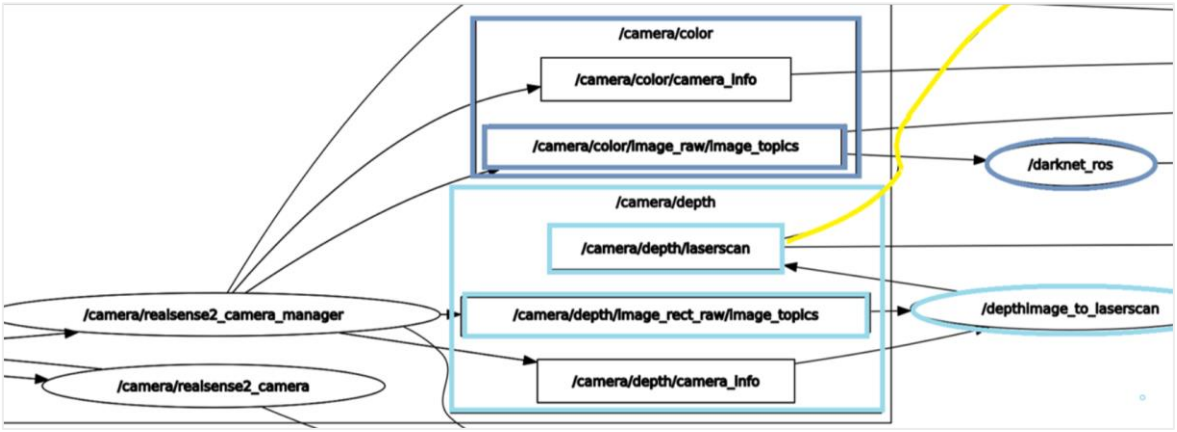


Figura 4.12 Gráfico ampliado, sección 'darknet_ros' (Fuente: propia).

Finalmente, se visualizan las cajas delimitadoras, en una nueva ventana emergente con nombre “YOLO”, como se muestra en la Figura 4.13 :

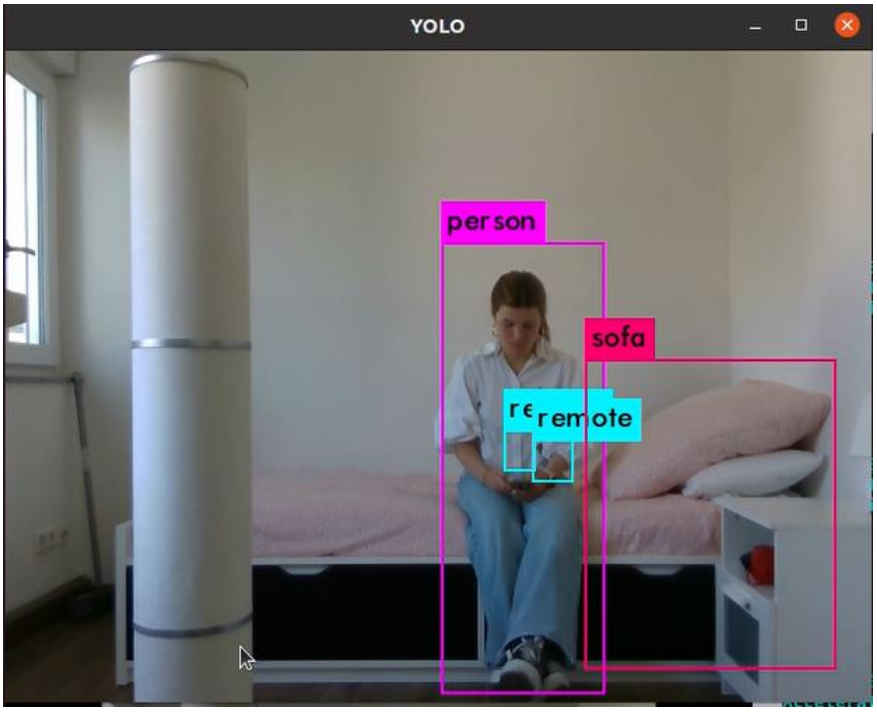


Figura 4.13 Ventana emergente YOLO (Fuente: propia).

Así mismo, en el terminal aparecen los datos obtenidos del procesamiento de YOLO, con los correspondientes porcentajes asociados a cada suposición detectada por el algoritmo. En la Figura 4.14 por ejemplo, se observa una captura de la ventana del terminal donde se encuentran reflejados estos datos.

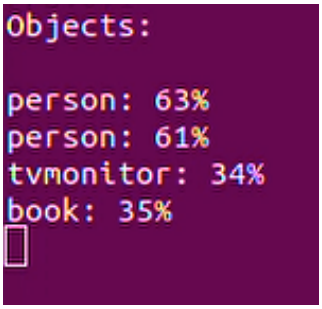


Figura 4.14 Resultado de la deducción de YOLO en la terminal (Fuente: propia).

5. Resultados obtenidos

Una vez finalizado el proyecto, se realizaron una serie de grabaciones de pantalla en vivo del funcionamiento del sistema de detección e identificación de objetos en diferentes escenarios. Así como unas pruebas finales para comparar los datos con las mediciones reales y comprobar la precisión y el error porcentual del programa.

- ***Escenario 1***

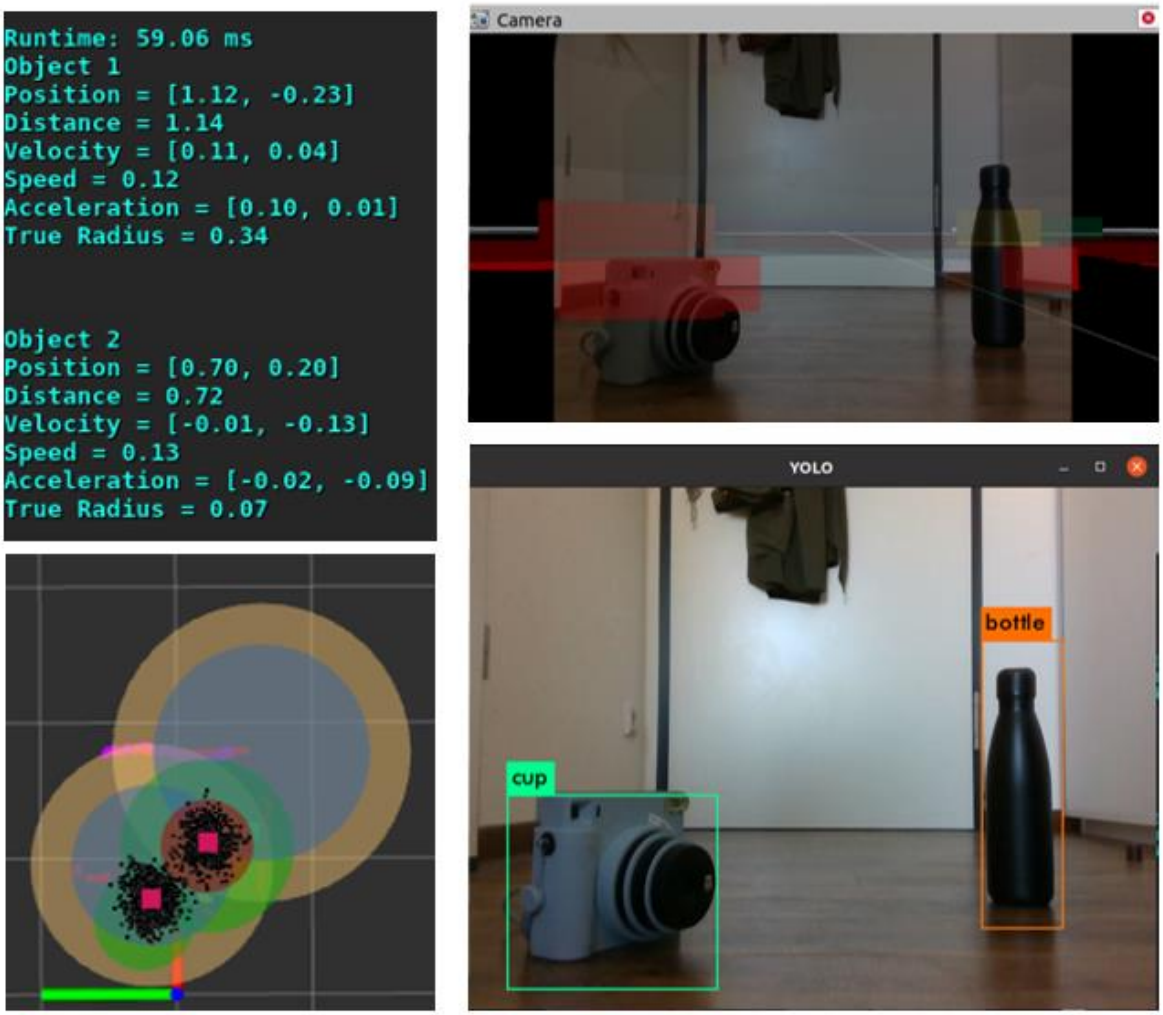


Figura 5.1 Escenario 1 de prueba (Fuente: propia).

Tabla 5.1 Resultados detección escenario 1 (Fuente: propia).

Realidad	Sistema detección + YOLO
Botella a 1.15 m	Objeto 1 a 1.14 m | Bottle
Cámara a 0.70 m	Objeto 2 a 0.72 m | Cup

Como se puede observar en la Tabla 5.1, en este primer escenario los resultados del sistema de detección son muy precisos: varia únicamente en una centésima de metro con respecto a la realidad para el primer objeto y en 2 centésimas para el segundo; en la detección por YOLO, detecta correctamente la botella y en el caso de la cámara, la identifica como una taza, como consecuencia de que en los datasets no está incluida la clase de objeto cámara.

- ***Escenario 2***

Figura 5.2 Escenario 2 de prueba (Fuente: propia).

Tabla 5.2 Resultados detección escenario 2 (Fuente: propia).

Realidad	Sistema detección + YOLO
Mesita con lámpara a 2 m	Objeto 1 a 2.02 m | x
Persona a 2,15 m	Objeto 3 a 2.21 m | Person
Silla a 1 m	Objeto 3 a 1.07 m | Chair

En esta segunda prueba, se plantea un nuevo escenario con tres objetos. Como se puede observar en la Tabla 5.2, los resultados del sistema de detección son precisos: para el objeto 1 varía en 0.02 m con respecto a la medición de la realidad, para el segundo 0.06 m y para el tercero 0.07 m; en la detección por YOLO, detecta correctamente la persona y la silla, pero en el caso de la lámpara, no la identifica porque no está incluida en los datasets.

- ***Escenario 3***

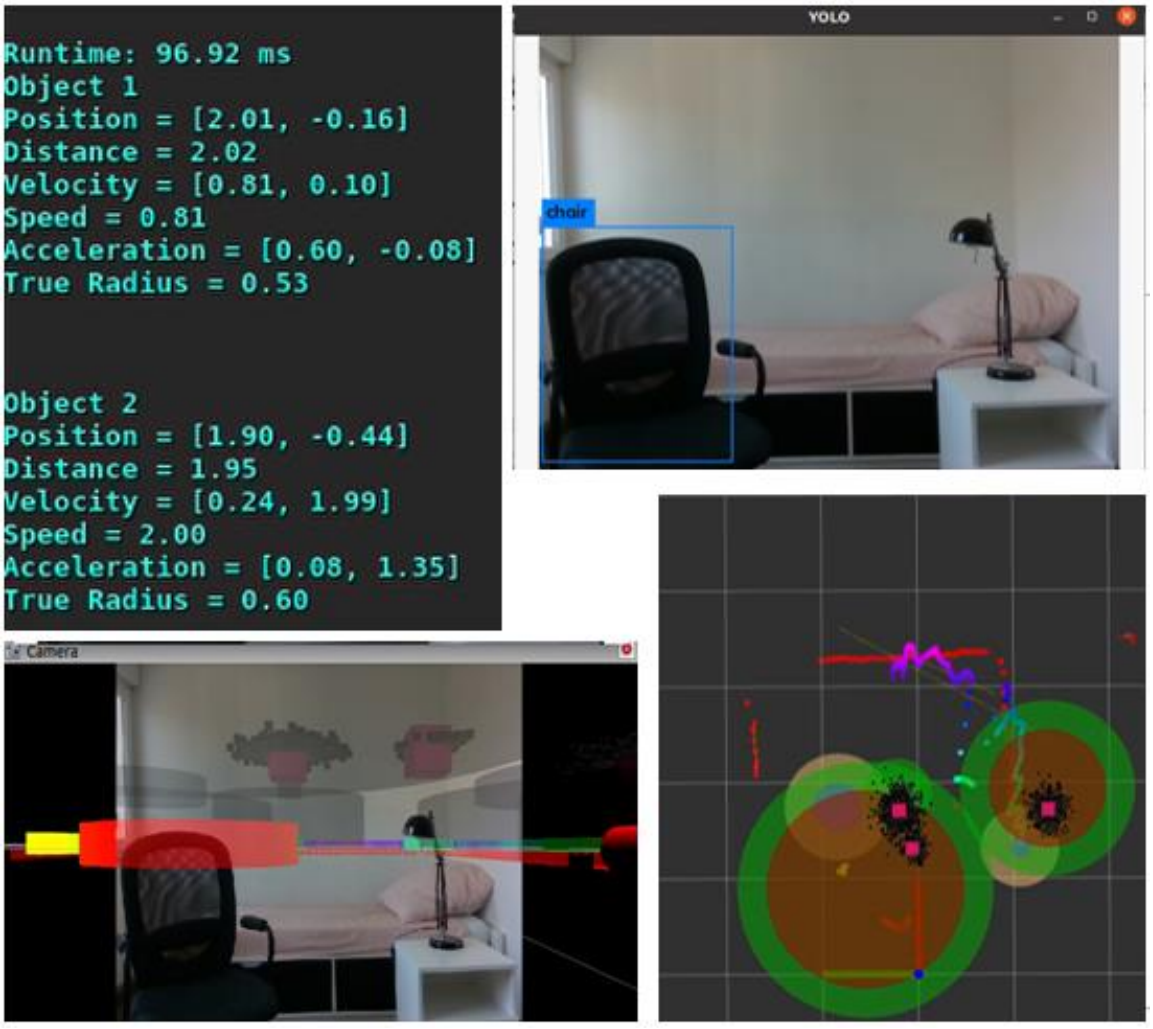


Figura 5.3 Escenario 3 de prueba (Fuente: propia).

Tabla 5.3 Resultados detección escenario 3 (Fuente: propia).

Realidad	Sistema detección + YOLO
Mesita con lámpara a 2 m	Objeto 1 a 2.02 m | x
Silla a 2 m	Objeto 2 a 1.95 m | Chair

En esta última prueba se plantea sin la persona en el escenario, manteniéndose en la misma posición la mesilla con la lámpara y moviéndose la silla a la misma distancia que la mesilla. Como se puede observar en la Tabla 5.3, los resultados del sistema de detección vuelven a ser bastante precisos: para el objeto 1 no cambia con respecto al anterior y para la silla, la variación con respecto a la realidad es de 0.05 m; en la detección por YOLO, detecta correctamente la silla y, nuevamente, no identifica la lámpara.

7. Conclusiones y recomendaciones

Retomando el objetivo principal que se propone al inicio del proyecto: la implementación de un sistema de visión artificial encargado de diferenciar e identificar objetos a través de la fusión de datos procedentes de un sensor lídar y una cámara; y tras haber expuesto el análisis exhaustivo de los distintos sensores, los métodos de fusión de sus datos en el ámbito de la visión artificial y el sistema operativo ROS, se pueden extraer una serie de conclusiones:

En la búsqueda de los sensores más apropiados para este sistema de visión artificial, la cámara, el lídar y el radar resultan los más interesantes. Sin embargo, es necesario considerar las limitaciones del equipo disponible, el lídar 2D y la cámara. Al llevarlo a la práctica y desarrollar los diferentes paquetes de fusión de datos en ROS, se comprendió que, para poder hacer una fusión óptima de los datos, la cámara debía ser específicamente una cámara de profundidad y no una 2D como se planteaba al principio. Si bien es cierto que esta combinación de sensores resultó más que suficiente para poder desarrollar el proyecto, el uso de un lídar 3D hubiese aumentado notablemente la precisión del sistema, pero también el precio.

En cuanto al entorno de programación de ROS, su manejo resulta alcanzable gracias a sus propias plataformas de aprendizaje y su comunidad de soporte. Además, al permitir intercambiar paquetes de manera pública, se facilita y aceleran mucho todos los procedimientos. Gracias a esto, en este proyecto se pudo concluir que el método de detección de objetos que más se adapta a las necesidades y los dispositivos disponibles consiste en ajustar un paquete prediseñado (‘object-detector-fusion’) y complementarlo con una parte de identificación de estos objetos a partir de la herramienta de YOLO, para visualizarlo en conjunto a través de las herramientas proporcionadas por ROS.

Los resultados muestran cómo el comportamiento de la parte de detección de objetos por YOLO podría ser mejorable. Esto se explica por la falta de una tarjeta gráfica durante el procesamiento, que hubiese acelerado mucho el entrenamiento y reconocimiento de

YOLO. Además, los datasets de los que se sirve YOLO tienen un alcance limitado. Para abordar esta problemática, las recientes investigaciones apuestan por su sustitución por la nueva red neuronal CLIP (Contrastive Language-Image Pre-training). Una comparativa entre ambas redes neuronales podría ser un punto de ampliación futura de este proyecto, analizando tanto la eficiencia computacional como la precisión en la detección de nuevos objetos.

Por lo que concierne a la aplicación real de este equipo de visión artificial de detección e identificación de objetos, sería fácil de implementar en prácticamente cualquier dispositivo gracias a las pequeñas dimensiones del conjunto: tanto en vehículos autónomos, como en sistemas de asistencia a personas de visibilidad reducida o incluso en sistemas de seguridad de control y medición de acceso de personas a ciertas áreas. A este respecto, hay que tener en cuenta que la evolución de estos sistemas está siendo más rápida que su propia regulación en términos legales, enfrentándose al conflicto ético que supone su uso, por el riesgo de invasión de la privacidad de otras personas.

8. Lista de referencias bibliográficas

[1] «big-data noun - Definition, pictures, pronunciation and usage notes | Oxford Advanced Learner's Dictionary at OxfordLearnersDictionaries.com». https://www.oxfordlearnersdictionaries.com/us/definition/english/big-data (accedido 26 de junio de 2022).

[2] «What Is Cloud Computing? A Beginner's Guide | Microsoft Azure». https://azure.microsoft.com/de-de/resources/cloud-computing-dictionary/what-is-cloud-computing/ (accedido 26 de junio de 2022).

[3] «What Is the Internet of Things (IoT)? | Oracle India». https://www.oracle.com/in/internet-of-things/what-is-iot/ (accedido 26 de junio de 2022).

[4] «artificial-intelligence noun - Definition, pictures, pronunciation and usage notes | Oxford Advanced Learner's Dictionary at OxfordLearnersDictionaries.com». https://www.oxfordlearnersdictionaries.com/definition/english/artificial-intelligence (accedido 26 de junio de 2022).

[5] «What Is Machine Learning and Why Is It Important?», *SearchEnterpriseAI*. https://www.techtarget.com/searchenterpriseai/definition/machine-learning-ML (accedido 26 de junio de 2022).

[6] «Report on The 5th Science and Technology Basic Plan». Accedido: 10 de mayo de 2022. [En línea]. Disponible en: https://www8.cao.go.jp/cstp/kihonkeikaku/5basicplan_en.pdf

[7] A. Ortega, «Sociedad 5.0: el concepto japonés para una sociedad superinteligente», p. 11.

[8] R. C. Luo, C.-C. Yih, y K. L. Su, «Multisensor fusion and integration: approaches, applications, and future research directions», *IEEE Sensors Journal*, vol. 2, n.º 2, pp. 107-119, abr. 2002, doi: 10.1109/JSEN.2002.1000251.

[9] F. C. Sotela, «Fusión de Datos Distribuida en Redes de Sensores Visuales Utilizando Sistemas Multi-Agente», p. 210.

[10] I. CORPORATIVA, «¿Qué es la visión artificial y cuáles son sus aplicaciones?», *Iberdrola*. https://www.iberdrola.com/innovacion/vision-artificial (accedido 26 de junio de 2022).

[11] «SAE Levels of Driving Automation™ Refined for Clarity and International Audience». https://www.sae.org/site/blog/sae-j3016-update (accedido 26 de junio de 2022).

[12] D. S. Kunapareddy, N. P. K. Putta, V. A. Maddala, R. K. Bethapudi, y S. R. Vanga, «Smart Vision based Assistant for Visually Impaired», en *2022 International Conference on Applied Artificial Intelligence and Computing (ICAAIC)*, may 2022, pp. 1178-1184. doi: 10.1109/ICAAIC53929.2022.9792812.

[13] «Supersense - AI for Blind», *App Store*. https://apps.apple.com/es/app/supersense-ai-for-blind/id1484547836 (accedido 26 de junio de 2022).

[14] «¿Qué es un cobot? | CADE Cobots», 13 de febrero de 2019. https://cadecobots.com/que-es-un-cobot/ (accedido 26 de junio de 2022).

[15] «Estrategias de seguridad para los cobots: aplicaciones de colaboración seguras y eficaces», *Interempresas*. https://www.interempresas.net/Robotica/Articulos/301888-Estrategias-de-seguridad-para-los-cobots-aplicaciones-de-colaboracion-seguras-y-eficaces.html (accedido 26 de junio de 2022).

[16] «Definición de sensor — Definicion.de», *Definición.de*. https://definicion.de/sensor/ (accedido 26 de junio de 2022).

[17] «IMU Funcionamiento - Diseño y evaluación de un sistema vestible para captura de movimientos or». https://1library.co/article/imu-funcionamiento-dise%C3%B1o-evaluaci%C3%B3n-sistema-vestible-captura-movimientos.rz3872dq (accedido 21 de junio de 2022).

[18] «Navegacion-Aerea.pdf». Accedido: 22 de junio de 2022. [En línea]. Disponible en: https://www.clubaereo.cl/wp-content/uploads/2019/10/Navegacion-Aerea.pdf

[19] «GPS.pdf». Accedido: 22 de junio de 2022. [En línea]. Disponible en: https://www.peoplematters.com/Archivos/Descargas/GPS.pdf

[20] «Beginner's guide to depth (Updated)», *Intel® RealSense™ Depth and Tracking Cameras*, 16 de julio de 2019. https://www.intelrealsense.com/beginners-guide-to-depth/ (accedido 1 de junio de 2022).

[21] «RGB», *Wikipedia, la enciclopedia libre*. 1 de junio de 2022. Accedido: 1 de junio de 2022. [En línea]. Disponible en: https://es.wikipedia.org/w/index.php?title=RGB&oldid=143910020

[22] S. M. Group, «A Guide to Stereovision and 3D Imaging». https://www.techbriefs.com/component/content/article/tb/pub/features/articles/14925 (accedido 2 de junio de 2022).

[23] D. J. Yeong, G. Velasco-Hernandez, J. Barry, y J. Walsh, *Sensor and Sensor Fusion Technology in Autonomous Vehicles: A Review*. 2021. doi: 10.20944/preprints202102.0459.v1.

[24] Z. Wang, Y. Wu, y Q. Niu, «Multi-Sensor Fusion in Automated Driving: A Survey», *IEEE Access*, vol. 8, pp. 2847-2868, 2020, doi: 10.1109/ACCESS.2019.2962554.

[25] Tesla, «Autopilot». https://www.tesla.com/en_IE/autopilot (accedido 21 de junio de 2022).

[26] «Waymo Driver», *Waymo*. https://waymo.com/intl/es/waymo-driver/ (accedido 24 de junio de 2022).

[27] «Waypoint - The official Waymo blog: Designed to deliver: Bringing the benefits of our 5th generation hardware to trucking», *Waymo Blog*. https://blog.waymo.com/2021/12/designed-to-deliver.html (accedido 24 de junio de 2022).

[28] A. I. Inc, «The Aurora Driver». https://aurora.tech/aurora-driver (accedido 24 de junio de 2022).

[29] «Safety_Report_2020.pdf». Accedido: 24 de junio de 2022. [En línea]. Disponible en: https://autonomous.lyft.com/wp-content/uploads/2020/06/Safety_Report_2020.pdf

[30] «Informatik 4: Definitions of Sensor Data Fusion (deutsche Version)». https://net.cs.uni-bonn.de/de/wg/sdf/what-is-it/sdf-definitions/ (accedido 14 de junio de 2022).

[31] F. Castanedo, «A Review of Data Fusion Techniques», *The Scientific World Journal*, vol. 2013, p. e704504, oct. 2013, doi: 10.1155/2013/704504.

[32] B. V. Dasarathy, «Sensor fusion potential exploitation-innovative architectures and illustrative applications», *Proceedings of the IEEE*, vol. 85, n.º 1, pp. 24-38, ene. 1997, doi: 10.1109/5.554206.

[33] W. Elmenreich, «An Introduction to Sensor Fusion», jun. 2022.

[34] «9 Types of Sensor Fusion Algorithms», *Think Autonomous*. https://www.thinkautonomous.ai/blog/?p=9-types-of-sensor-fusion-algorithms (accedido 13 de junio de 2022).

[35] H. Li, J. Huang, y S. Ji, «Bearing Fault Diagnosis with a Feature Fusion Method Based on an Ensemble Convolutional Neural Network and Deep Neural Network», *Sensors*, vol. 19, n.º 9, Art. n.º 9, ene. 2019, doi: 10.3390/s19092034.

[36] S. F. Andriawan Eka Wijaya, D. Setyo Purnomo, E. B. Utomo, y M. Akbaryan Anandito, «Research Study of Occupancy Grid map Mapping Method on Hector SLAM Technique», en *2019 International Electronics Symposium (IES)*, sep. 2019, pp. 238-241. doi: 10.1109/ELECSYM.2019.8901657.

[37] «Documentation - ROS Wiki». http://wiki.ros.org/Documentation (accedido 17 de mayo de 2022).

[38] «The Construct: A Platform to Learn ROS-based Advanced Robotics Online», *The Construct*. https://www.theconstructsim.com/ (accedido 17 de mayo de 2022).

[39] «RPLIDAR-A1 360°Laser Range Scanner _ Domestic Laser Range Scanner|SLAMTEC». https://www.slamtec.com/en/Lidar/A1 (accedido 18 de mayo de 2022).

[40] «LD108_SLAMTEC_rplidar_datasheet_A1M8_v1.0_en.pdf». Accedido: 18 de mayo de 2022. [En línea]. Disponible en: http://bucket.download.slamtec.com/e9e096e9d9f30205d665260abe2cfb0c2dd62efa/LD108_SLAMTEC_rplidar_datasheet_A1M8_v1.0_en.pdf

[41] «NTP 654. Láseres: nueva clasificación del riesgo (UNE EN 60825-1 /A2: 2002)», p. 8.

[42] «Depth Camera D435», *Intel® RealSense™ Depth and Tracking Cameras*. https://www.intelrealsense.com/depth-camera-d435/ (accedido 7 de junio de 2022).

[43] «Intel-RealSense-D400-Series-Datasheet-April-2022.pdf». Accedido: 7 de junio de 2022. [En línea]. Disponible en: https://www.intelrealsense.com/wp-content/uploads/2022/05/Intel-RealSense-D400-Series-Datasheet-April-2022.pdf

[44] «Developing depth sensing applications - collision avoidance, object detection, volumetric capture and more», *Intel® RealSense™ Depth and Tracking Cameras*. https://www.intelrealsense.com/sdk-2/ (accedido 7 de junio de 2022).

[45] «ROS/Introduction - ROS Wiki». http://wiki.ros.org/ROS/Introduction (accedido 23 de mayo de 2022).

[46] «YOLO ROS: Real-Time Object Detection for ROS». Robotic Systems Lab - Legged Robotics at ETH Zürich, 10 de junio de 2022. Accedido: 11 de junio de 2022. [En línea]. Disponible en: https://github.com/leggedrobotics/darknet_ros

[47] «GitHub - rezanatha/object-detector-fusion: 2D Object Detection and Tracking based on LiDAR and Stereo Camera Data - ROS Package». https://github.com/rezanatha/object-detector-fusion (accedido 11 de junio de 2022).

[48] M. Przybyla, «The obstacle_detector package». 1 de junio de 2022. Accedido: 12 de junio de 2022. [En línea]. Disponible en: https://github.com/tysik/obstacle_detector

[49] «rplidar - ROS Wiki». http://wiki.ros.org/rplidar (accedido 11 de junio de 2022).

[50] «usb_cam - ROS Wiki». http://wiki.ros.org/usb_cam (accedido 11 de junio de 2022).

[51] «laser_geometry - ROS Wiki». http://wiki.ros.org/laser_geometry (accedido 11 de junio de 2022).

[52] P. Palanisamy, «Multiple objects detection, tracking and classification from LIDAR scans/point-clouds». 10 de junio de 2022. Accedido: 11 de junio de 2022. [En línea]. Disponible en: https://github.com/praveen-palanisamy/multiple-object-tracking-lidar

[53] «ROS Wrapper for Intel® RealSense™ Devices». Intel® RealSense™, 10 de junio de 2022. Accedido: 11 de junio de 2022. [En línea]. Disponible en: https://github.com/IntelRealSense/realsense-ros

[54] «depthimage_to_laserscan - ROS Wiki». https://wiki.ros.org/depthimage_to_laserscan (accedido 12 de junio de 2022).

[55] «jsk_visualization». jsk-ros-pkg, 12 de junio de 2022. Accedido: 12 de junio de 2022. [En línea]. Disponible en: https://github.com/jsk-ros-pkg/jsk_visualization