DESARROLLO DE UN APLICATIVO EN ROS PARA LA SEGMENTACIÓN DE OBJETOS DE INTERÉS EN INTERIORES USANDO UN ROBOT MÓVIL

Juan Camilo Chavez Castro

Universidad del Valle

Facultad de Ingeniería

Escuela de Ingeniería Eléctrica y Electrónica

Santiago de Cali, Valle del Cauca, Colombia

2025

DESARROLLO DE UN APLICATIVO EN ROS PARA LA SEGMENTACIÓN DE OBJETOS DE INTERÉS EN INTERIORES USANDO UN ROBOT MÓVIL

Juan Camilo Chavez Castro

Trabajo de grado para optar por el título de Ingeniero Electrónico

Director

Eval Bladimir Bacca Cortes, Ph.D.

Universidad del Valle

Facultad de Ingeniería

Escuela de Ingeniería Eléctrica y Electrónica

Santiago de Cali, Valle del Cauca, Colombia

2025

DEDICATORIA

Se lo dedico a Dios, por su amor y apoyo que me permitió caminar por cada parte de este recorrido.

Este trabajo se lo dedico especialmente a mis padres, de no ser por ese acompañamiento tan increíble que recibí de ellos, su amor, comprensión y motivación, este documento no existiría.

A mi Lupe, mi perrita, que te me fuiste muy pronto, tu amor siempre se sentirá y siempre permanecerá con nosotros.

Te lo dedico a ti Juan Camilo Chavez Castro, que sepas que puedes lograr grandes cosas siempre que te lo propongas.

AGRADECIMIENTOS

Quiero empezar agradeciendo a Dios, porque muchas veces en que la frustración me cegaba y me impedía avanzar, siempre sentí su apoyo al buscar en Él.

A mi madre querida por su comprensión, apoyo y amor que recibí en todo momento en este proceso.

A mi padre por sus consejos, sus palabras de motivación, su acompañamiento y apoyo.

A mis amigos y compañeros que me ayudaron en este proceso, los compañeros del laboratorio de robótica, los compañeros de la carrera, los compañeros de natación, a cada uno que aporto su granito de arena en este proyecto.

A el profesor Bladimir Bacca, por su paciencia, su guía y apoyo en este trabajo de grado.

A los profesores que tuve la oportunidad de conocer y aprender, que han aportado a mi crecimiento y conocimiento de manera satisfactoria, y por lo cual estoy sumamente agradecido.

A todas las personas que a lo largo de mi carrera compartieron, apoyaron, los amigos que conocí a lo largo de todo este camino y los familiares que siempre estuvieron apoyándome.

Y quiero agradecerme a mi, mi persistencia, por no renunciar hasta lograrlo, por todas las veces que lograste superarte y levantarte, y porque finalmente lograste terminar.

Muchas muchas gracias!

Tabla de contenido

Índice de figuras

Índice de cuadros

RESUMEN

En el contexto del avance de la robótica móvil autónoma, la percepción artificial en interiores ha adquirido gran relevancia para aplicaciones como logística, salud y asistencia personalizada. Esta percepción se apoya en técnicas de clasificación, detección y segmentación, que permiten a los robots identificar objetos de interés en su entorno. A pesar del progreso en esta área, la segmentación en linea, adaptable a diversos escenarios y fácil de integrar en plataformas robóticas, sigue siendo un área de mejora.

El objetivo general de este trabajo fue "Desarrollar un aplicativo en ROS para la segmentación de objetos de interés en interiores usando un Robot Móvil Pioneer 3DX" y se materializó con el aplicativo LU-SEG, una plataforma modular que integra sensores RGB-D, algoritmos de visión y nodos ROS para ofrecer segmentación en linea de al menos cinco clases ejecutado sobre el robot móvil Pioneer 3DX. La metodología de desarrollo utilizada fue Scrum, permitiendo avanzar de forma iterativa e incremental en la construcción del sistema, estructurado en cuatro módulos principales: Home, Configuración, Ejecución y Replay.

Se evaluaron distintos enfoques de segmentación, tanto instancias como semántica, seleccionando los modelos YOLOv11m-seg, Mask R-CNN, ESANet-RGB y L-RASSP MobileNetV3 para entrenamiento y evaluación. Este último fue el modelo integrado al sistema final por su equilibrio entre precisión y bajo consumo computacional (mIoU de 79.9 % con solo 3.2 M de parámetros). Las pruebas de validación: integración, cuantitativas, por distancia y de campo, demostraron la eficacia del sistema con una precisión promedio del 91.66 % y una sensibilidad del 79.55 % en ambientes reales.

LU-SEG demostró ser una herramienta versátil para la experimentación con segmentación en entornos interiores. Su arquitectura modular, compatibilidad con ROS y capacidad de operación en procesamiento continuo lo convierten en una plataforma útil para la validación de técnicas de visión artificial en escenarios reales. Más que una solución definitiva, representa un punto de partida valioso para el aprendizaje y la exploración de nuevas estrategias de percepción visual aplicadas a la robótica móvil y su aplicación en sistemas autónomos.

Palabras claves– percepción artificial, visión artificial, redes neuronales, aprendizaje profundo, robótica móvil, segmentación semántica, segmentación de instancias, SCRUM.

ABSTRACT

In the context of the advancement of autonomous mobile robotics, indoor artificial perception has gained significant importance in applications such as logistics, healthcare, and personalized assistance. This perception relies on techniques such as classification, detection, and segmentation, which allow robots to identify objects of interest in their environment. Despite progress in this field, online segmentation—adaptable to various scenarios and easily integrable into robotic platforms—remains an area with room for improvement.

The general objective of this work was "to develop a ROS-based application for the segmentation of objects of interest in indoor environments using a Pioneer 3DX mobile robot," which materialized in the LU-SEG application. LU-SEG is a modular platform that integrates RGB-D sensors, vision algorithms, and ROS nodes to deliver online segmentation of at least five classes, running on the Pioneer 3DX mobile robot. The development methodology employed was Scrum, enabling iterative and incremental progress in building the system, which is structured into four main modules: "Home", "Configuración", "Ejecución", and "Replay".

Various segmentation approaches, both instance-based and semantic, were evaluated, selecting the YOLOv11m-seg, Mask R-CNN, ESANet-RGB, and L-RASSP MobileNetV3 models for training and performance assessment. The latter was integrated into the final system due to its balance between accuracy and low computational cost (achieving 79.9% mIoU with only 3.2M parameters). The validation process, which included integration, quantitative, distance-based, and field tests, confirmed the system's effectiveness, reaching an average accuracy of 91.66% and a sensitivity of 79.55% in real-world environments.

LU-SEG proved to be a versatile tool for experimenting with segmentation in indoor settings. Its modular architecture, compatibility with ROS, and capability for continuous processing make it a valuable platform for validating computer vision techniques in real scenarios. Rather than being a final solution, it serves as a solid starting point for learning and the development of new visual perception strategies in mobile robotics and autonomous systems.

Keywords– artificial vision, neural networks, deep learning, mobile robotics, semantic segmentation, instance segmentation, SCRUM.

Capítulo 1

INTRODUCCIÓN

En la sociedad actual, la robótica móvil autónoma y los vehículos autónomos desempeñan un papel cada vez más importante. Estas tecnologías están revolucionando diversos sectores, como el transporte, la logística [1], la agricultura [2] (figura 1.1) , la medicina [3], y muchos otros. La capacidad de los robots y vehículos autónomos para moverse de manera independiente y realizar tareas específicas sin la intervención humana directa tiene un impacto significativo en la eficiencia, la seguridad y la calidad de vida.

Figura 1.1: Los robots pronto ayudarán a los productores a elegir semillas.
Fuente: [2]

En particular, la percepción artificial, que incluye técnicas como la segmentación en interiores, desempeña un papel importante en la capacidad de los robots y vehículos autónomos para interactuar de manera efectiva con su entorno. La segmentación en interiores permite identificar y distinguir objetos de interés en ambientes cerrados, lo que es esencial para el funcionamiento seguro y eficiente de los sistemas autónomos en entornos como hogares, oficinas, almacenes, hospitales, entre otros.

Al utilizar técnicas de segmentación en interiores, los robots y vehículos autónomos pueden tomar decisiones informadas basadas en la comprensión y el análisis de su entorno. Esto les permite evitar obstáculos, navegar de manera segura, interactuar con objetos y realizar tareas

específicas. Por ejemplo, en el ámbito de la logística [4], los vehículos autónomos pueden utilizar la segmentación en interiores para identificar y clasificar objetos en un almacén, facilitando así la gestión y el transporte de mercancías (figura 1.2).

Figura 1.2: Robots implementados en logística.
Fuente: [4]

Además, la segmentación en interiores también juega un papel fundamental en aplicaciones de asistencia personalizada y cuidado de la salud (figura 1.3). Los robots autónomos pueden utilizar la segmentación para identificar personas y proporcionar asistencia en tareas cotidianas o monitorear la salud de las personas. [5], [6].

Figura 1.3: Robots en medicina.
Fuente: [6]

La segmentación de escenas en interiores sigue siendo una tarea compleja que requiere un análisis preciso de múltiples factores, incluida la detección de objetos, el análisis de la forma, la identificación de características y la comprensión de la profundidad. Además, es necesario contar con soluciones que puedan segmentar de manera continua y en línea, en especial para aplicaciones como los vehículos autónomos y la robótica donde es necesario la interacción con el entorno para la realización de tareas, ya sean rudimentarias, complejas o repetitivas.

También la segmentación desempeña un papel crucial en la identificación y delimitación de objetos específicos de interés en mapas semánticos [7] (figura 1.4). En entornos exteriores, se pueden distinguir elementos como carreteras, aceras, edificios, árboles, vehículos, peatones, señales de tráfico, entre otros. En entornos interiores, la segmentación permite identificar y delimitar objetos como mesas, ventanas, paredes, pisos, computadoras, armarios, entre otros. La versatilidad de la segmentación radica en su capacidad para adaptarse a diferentes objetivos.

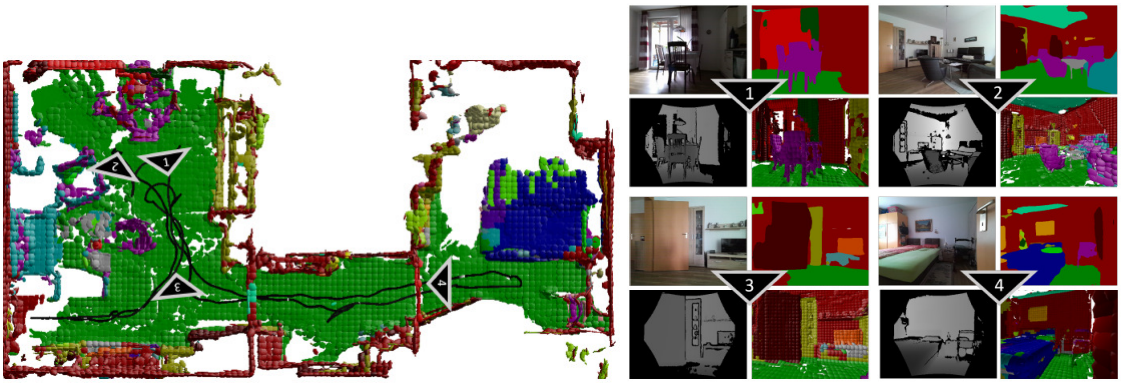


Figura 1.4: Izquierda: vista aérea del mapa semántico, incluida la trayectoria del robot (negro). Derecha: Poses (1-4), se observa imagen en profundidad (blanco y negro), imagen a color, segmentación semántica de ESANet-R34-NBt1D y representación 2D que visualiza el mapa a partir de la pose del robot.

Fuente: [7]

Lo anterior, lleva a plantearse formas de poder experimentar con estos algoritmos de segmentación, para probar los métodos existentes en entornos reales de forma rápida y sencilla. La motivación de este trabajo surge del deseo de aprender sobre la robótica, la inteligencia artificial, la visión artificial, las nuevas tecnologías, y de entender cómo estas tecnologías están transformando nuestra sociedad y como esta evolucionando el uso de vehículos y robots autónomos.

La robótica móvil autónoma, los vehículos autónomos y las estrategias de percepción artificial, como la segmentación en interiores, son tecnologías que están cambiando nuestra sociedad poco a poco. En este orden de ideas, se presenta este trabajo, donde se documentará e implementará una técnica de segmentación mediante un aplicativo diseñado para experimentar con un robot móvil y darle percepción en entornos interiores. Posteriormente, se evaluará la solución propuesta mediante la presentación de resultados.

1.1. Planteamiento del problema

En los últimos años, la expansión de la investigación y comercialización de vehículos autónomos [1] ha aumentado la utilidad de la visión en robótica. Según el informe de Modor

Intelligence, se espera que el mercado crezca a una tasa compuesta anual del 9,86 % durante el período de pronóstico (2021-2026), debido al creciente uso de robots cognitivos en diferentes campos [8]. Desde la detección de presencia hasta la inspección en tiempo real, la capacidad de detectar e identificar objetos de interés se ha vuelto fundamental en el desarrollo de vehículos autónomos capaces de interactuar con su entorno, que a su vez permite que el robot realice funciones rudimentarias, repetitivas o complejas.

La aplicación de los vehículos autónomos se ha expandido a distintos entornos, incluyendo ambientes en interiores. Empresas como Amazon y Walmart han visto el potencial de esta tecnología para mejorar la eficiencia en sus operaciones, por ejemplo, Amazon está utilizando sus robots móviles para mejorar el flujo de inventarios en centros logísticos [9], están encargados de trasladar estanterías de productos a destinos específicos; mientras que en Walmart utilizan Alphabot, robot móvil de la compañía Alert Innovation, para automatizar el sistema de almacenamiento, recuperación y selección de productos [10], y desde el 2020 se implementa oficialmente en Walmart de Sallent, New Hampshire [11].

La automatización de tareas en ambientes cerrados plantea desafíos en el reconocimiento de objetos o tópicos, que se abordan mediante diversos métodos de segmentación. Por ejemplo, en [12] plantean un método de segmentación basado en mapas de disparidad y el modelo estadístico CRF para la reconstrucción de entornos urbanos. Otro enfoque se presenta en [13], que utiliza un algoritmo de crecimiento de regiones para segmentar planos en interiores. Además, se han aplicado métodos basados en aprendizaje profundo y extracción de características en nubes de puntos, como se describe en [14]. Sin embargo, a pesar de que existen numerosos métodos de segmentación, sigue siendo objeto de estudio.

La ausencia de una solución automatizada de segmentación y análisis interior conlleva ineficiencias operativas medibles. Estudios comparativos con robots de campo mostraron que la automatización puede reducir el trabajo repetitivo entre 25 % y 90 %, acortar los plazos de ejecución en promedio 2.3 veces y disminuir costos en torno al 13 % en los casos analizados [15]. En paralelo, en operaciones de almacén y logística, persisten barreras de adopción ligadas al costo y al retorno de inversión: se reporta que proyectos con robots móviles requieren típicamente entre dos y tres años para alcanzar el ROI y que la disponibilidad presupuestal es el principal obstáculo, lo que prolonga dependencias de procedimientos manuales menos eficientes [16].

En este orden de ideas, se pretende que automáticamente un robot móvil sea capaz de analizar escenas en interiores al desplazarse a lo largo de estos ambientes, segmente objetos de interés y almacene los resultados. Usando ROS (Robotic Operative System)[17] permitirá que el aplicativo desarrollado pueda ser reproducible en otros contextos. Lo anterior lleva a plantear la pregunta problema: ¿Cómo desarrollar un aplicativo en ROS para la segmentación de objetos de interés en interiores usando un robot móvil?

1.2. Objetivos

A continuación, se presentan los objetivos planteados para este trabajo de grado.

1.2.1. Objetivo General

Desarrollar un aplicativo en ROS para la segmentación de objetos de interés en interiores usando un Robot Móvil Pioneer 3DX.

1.2.2. Objetivos Específicos

1. Sintetizar información bibliográfica relacionada con los métodos de segmentación usando diversos sensores con el propósito de definir requerimientos funcionales y no funcionales del aplicativo.

2. Diseñar un aplicativo en ROS que permita la segmentación de al menos 5 objetos de interés en interiores, cumpliendo con los requerimientos propuestos.

3. Implementar el aplicativo en ROS para realizar la segmentación de objetos de interés en interiores utilizando un Robot Móvil Pioneer 3DX como plataforma base.

4. Realizar pruebas de integración, cuantitativas y de campo para validar el aplicativo en ROS destinada a la segmentación de objetos de interés en interiores usando un Robot Móvil Pioneer 3DX.

1.3. Solución propuesta

Como respuesta a la problemática planteada, se decidió implementar sistemas interconectados: un sistema robótico, un sistema de comunicación y un sistema de software (figura 1.5). El sistema robótico cuenta con un robot Pioneer-3DX que a su vez tiene actuadores, que son los motores y sensores propioceptivos para sensar odometría. El sistema robótico también tiene un sensor exteroceptivo que es la camara estereo ZED2 para capturar imágenes RGB-D. El cerebro de esta máquina es un mini PC marca KINGDEL con el sistema operativo ubuntu 20.04, que a su vez cuenta con ROS Noetic.

El sistema de comunicación entre el PC maestro y el PC del robot se establecerá en una red local. Vale mencionar que ROS cuenta con un método de comunicación llamado Ros Multi-Máquina [18], que permite la conexión entre nodos de diferentes dispositivos, los protocolos de comunicación que utiliza este método son: TCPROS [19] y UDPROS [20].

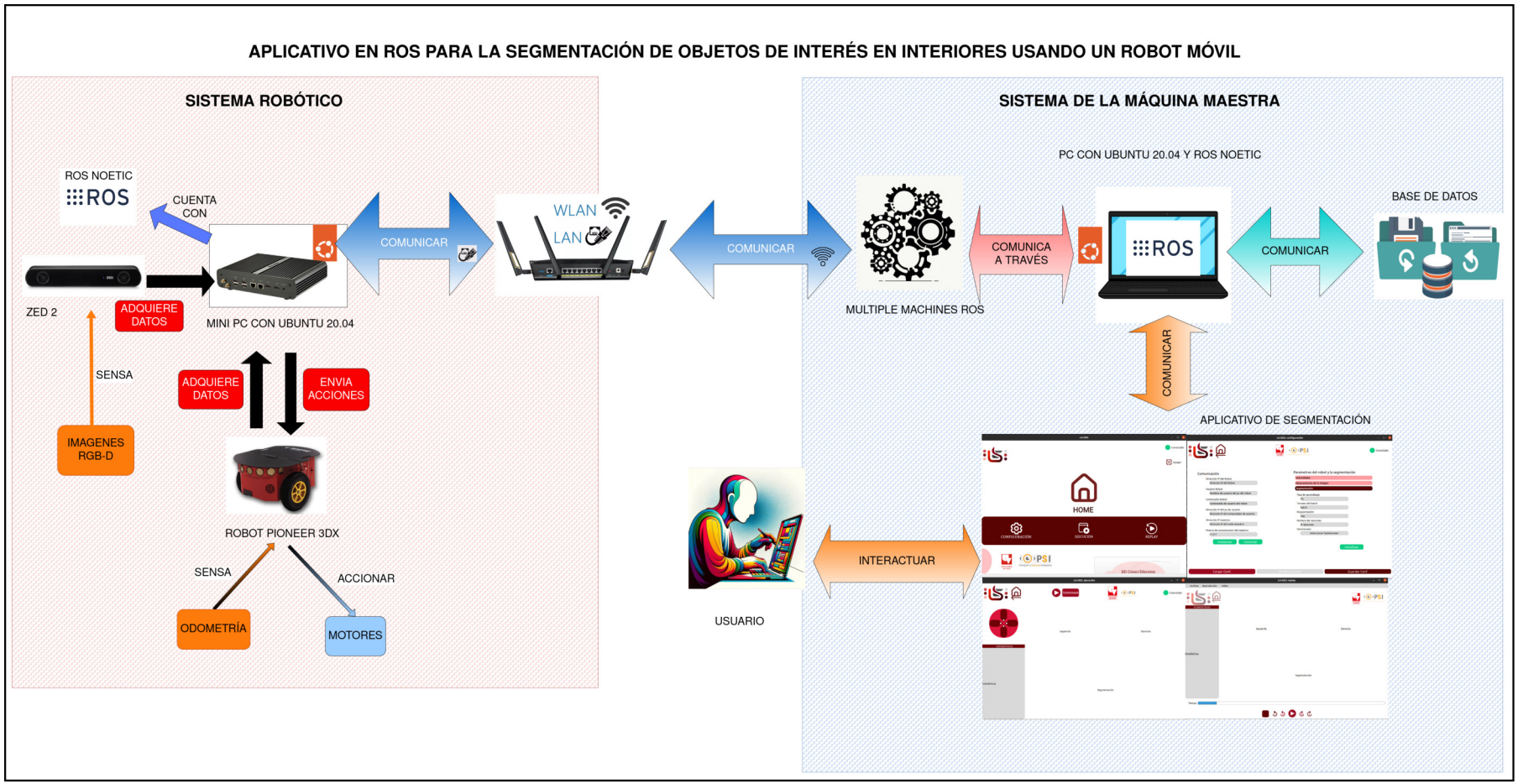


Figura 1.5: Aplicativo de Segmentación usando un robot móvil.

El sistema de software cuenta con una una aplicación que permite al usuario experimentar con un algoritmo de segmentación basado en redes neuronales. Esta aplicación es capaz de comunicarse con el robot, permitiendo al usuario observar cómo el robot reconoce, identifica y ubica los objetos de interés a su alrededor. La aplicación presenta ventanas específicas para configuración, ejecución y replay, permitiendo a los usuarios ajustar parámetros de comunicación y parámetros del algoritmo como la confianza, radio de cálculo de distancia y opciones del modelo, como también ajustar la velocidad del robot, con el fin de realizar una sesión de experimentación y su respectiva grabación. Posteriormente permite la revisión y análisis de sesiones pasadas para observar el comportamiento del algoritmo en la sesión.

1.4. Estructura del documento

Este documento se estructura en varios capítulos clave, comenzando con la Introducción, donde se presenta el problema que se aborda, se definen los objetivos del proyecto y la solución propuesta para el aplicativo de segmentación de objetos de interés en interiores usando un robot móvil. El siguiente capitulo es el marco teórico y los antecedentes, que proporcionan el sustento académico y técnico para el desarrollo del proyecto. En esta sección se revisan las teorías y estudios previos que fundamentan el diseño e implementación del aplicativo, brindando una comprensión clara de los conceptos esenciales para su desarrollo.

El capítulo de Desarrollo del aplicativo se divide en dos subcapítulos principales. El primero aborda el diseño, que incluye los requerimientos del aplicativo, el diagrama conceptual, las tablas relacionales de la base de datos y el diagrama de clases y de nodos. El segundo subcapítulo describe la implementación del aplicativo, con detalles sobre los módulos de configuración, ejecución y reportes.

Posteriormente, el capítulo de Validación y Pruebas detalla las pruebas funcionales, de integración, y los análisis de los resultados obtenidos, señalando los alcances del sistema. Finalmente, el documento resume los hallazgos principales en las Conclusiones, donde se evalúan los logros y se sugieren posibles mejoras y futuros trabajos.

Capítulo 2

ANTECEDENTES Y MARCO TEÓRICO

2.1. Introducción

La ingeniería ha experimentado un gran avance en el campo de la robótica y los vehículos autónomos, lo que ha permitido su aplicación en una amplia variedad de entornos y áreas. Por otro lado, lo que permite a los robots móviles autónomos interactuar con su entorno es el reconocimiento de este. Así mismo, la identificación y segmentación de objetos de interés se vuelve crítico para el desarrollo de esta área.

Este capítulo expone un resumen de las investigaciones consultadas referente a las diferentes temáticas sobre la clasificación, reconocimiento y métodos de segmentación en imágenes. También, se presentan los conceptos que permitieron abordar el problema planteado. Este apartado, se divide principalmente en dos secciones: la primera corresponde al resumen y análisis de los trabajos consultados respecto a las diferentes algoritmos para la segmentación y la segunda, contiene los conceptos relacionados con la visión artificial, la robótica, el *Deep Learning* y la metodología usada para el desarrollo del aplicativo móvil.

2.2. Antecedentes

Se realizó una búsqueda relacionada a los métodos de segmentación, especialmente los que se pueden aplicar para vehículos autónomos, en la base de datos IEEE Xplore, la base de datos Arxiv de la universidad Cornell, el repositorio de la Universidad de los Andes, la Universidad Autónoma de Occidente y la biblioteca de la Universidad del Valle. Para clasificar la búsqueda se compararon los siguientes criterios: tipo de referencia, sensor o tipo de dato de entrada, dataset en caso de usarse, método de segmentación, el entorno donde lo aplican, los objetos de interés y el rendimiento tanto en precisión como velocidad de procesamiento. Los criterios se exponen en la Tabla 2.1.

Ref	Sensor	Dataset	Método de segmentación	Ambiente	Objetos de interés	Rendimiento en precisión	Rendimiento en velocidad de procesamiento
[21]	Cámara de vídeo	*N/A*	K-means, CNN, aprendizaje supervisado	Simulado	Personas y obstáculos	Exactitud: 92%	0.45 Hz
[22]	Cámara monocular	*N/A*	Segmentación basada en el modelo cilíndrico y segmentación basado en la distancia euclidiana y la densidad	Exteriores	Troncos y copa de árboles Fuste	*N/A*	*N/A*
[13]	RGB-D	Kinect Dataset, Segcomp ABW	Crecimiento de regiones	Interiores	Planos	IoU: 1 – 0.6	100 Hz – 33 Hz
[23]	LiDAR de 16 canales, 6 cámaras Mako G319	Mapillary Vistas	Aprendizaje profundo y extracción de características	Urbano	Tabla 2.2	mIoU: 68.23%	*N/A*
[14]	LiDAR	SemanticKITTI	Aprendizaje profundo y extracción de características en nubes de puntos	Urbano	Estimación de elevación sobre el suelo	mIoU: 83.6%	55.61 Hz
[24]	LiDAR de 32 canales	*N/A*	Segmentación basada en la distancia euclidiana y segmentación basada en el rango cilíndrico.	Urbano	Personas y vehículos	*N/A*	32 Hz y 4 Hz
[25]	RGB-D y RGB	NYUv2, SUNRGB-D CityScapes	ESA Net RGB-D, Aprendizaje profundo	Interiores y Exteriores	Mesas, asientos, puertas, pisos, paredes, personas, etc. Vehículos, elementos de tráfico, cielo, vegetación, etc	NYUv2: mIoU: 51.58% CityScapes: mIoU: 80.09%	NYUv2: 29.7 Hz CityScapes:6.2 Hz
[26]	RGB-D	NYUv2, SUNRGB-D	GAD, Aprendizaje profundo	Interiores	Paredes, pisos, armarios, camas, asientos, mesas, puertas, personas, etc.	mIoU: 59.6%	*N/A*
[27]	LiDAR	KITTI 3D	PointPillars, Aprendizaje profundo	Exteriores	Carros, ciclistas, peatones.	AP: 66.19%	62 Hz
[28]	LiDAR	KITTI 3D	VoxelNet, Aprendizaje profundo	Exteriores	Carros, ciclistas, peatones.	AP: 58.28%	*N/A*
[29]	LiDAR	KITTI 3D	SECOND, Aprendizaje profundo	Exteriores	Carros, ciclistas, peatones.	AP: 60.56%	20 Hz
[30]	LiDAR	SemanticKITTI	CGGC-Net, Aprendizaje profundo	Exteriores	Personas, bicicletas, motos, edificios, carros, etc	mIoU: 58.4%	*N/A*
[30]	RGB-D	S3DIS	CGGC-Net, Aprendizaje profundo	Interiores	Piso, paredes, puertas, mesas, asientos, etc.	mIoU: 70.2%	*N/A*
[31]	RGB-D	MS COCO, Synthetic ToyCar3	Aprendizaje profundo Mask R-CNN y segmentación geométrica a través de crecimiento de regiones	Simulado	Figuras sintéticas	*N/A*	1 Hz
[32]	RGB-D	NYUv2, SUNRGB-D	Aprendizaje profundo, convolución 2.5D	Interiores	*N/A*	mIoU: 47.3%	*N/A*
[33]	RGB-D	S3DIS	EG-PointNet, Aprendizaje profundo	Interiores	Columnas, paredes, mesas, asientos, puertas, ventanas, etc	mIoU: 59%	*N/A*
[34]	RGB	MS COCO	YOLO11m-seg, Aprendizaje profundo	Urbanos, Exteriores, Interiores, Animales, etc.	Personas, Vehículos, Animales, Objetos en Exteriores, Accesorios, Objetos de deportes, Alimentos, Muebles, Electrónica, etc.	mAP[50-95]mask: 41.5%	*N/A*
[35]	RGB	MS COCO	Maksrcnn-ResNeXt-101-FPN , Aprendizaje profundo	Urbanos, Exteriores, Interiores, Animales, etc.	Personas, Vehículos, Animales, Objetos en Exteriores, Accesorios, Objetos de deportes, Alimentos, Muebles, Electrónica, etc.	APmask: 37.1%, AP50mask: 60.0%	*N/A*
[36]	RGB	CityScapes, Subconjunto de 20 clases de COCO	LRASPP_MobileNetV3_large, Aprendizaje profundo	Urbanos	Personas, Vehículos Objetos en Exteriores e Interiores, elementos de tráfico, cielo, vegetación, tv, sillas, sofás, macetas, etc	mIoU: 72.64%	*N/A*

Tabla 2.1: Tabla comparativa de los métodos de segmentación.

N/A: No aplica, no especifica, no se utilizó

2.2.1. Antecedentes nacionales

En el repositorio de la Universidad de los Andes y la Universidad Autónoma de Occidente se encontraron los siguientes trabajos.

En [21] presenta una solución para la detección y seguimiento de personas y la detección de obstáculos, en un ambiente simulado en ROS. Se comparan dos algoritmos de detección de personas: YOLO [37], que utiliza redes neuronales convolucionales y logró una velocidad de procesamiento de 0.45 FPS con una precisión promedio AP del 92 %; y SVM de la librería OpenCV, un algoritmo de aprendizaje supervisado que logró una velocidad de procesamiento de 0.66 FPS, sin embargo, no detectó a las personas en todos los casos.

Para la detección de obstáculos utilizó el método de segmentación K-means, que divide la imagen en píxeles y asigna cada píxel al grupo más cercano basándose en la similitud de sus características. Aunque logró detectar obstáculos, no identifica qué tipo de objetos son.

En el trabajo [22] se presentó un sistema de percepción robótica que utiliza técnicas de visión artificial para estimar características dendrométricas en árboles de fuste único. El sistema empleó una cámara monocular para capturar imágenes 2D y generar un mapa de nube de puntos de los árboles escaneados. Luego, se aplicó un proceso de segmentación para clasificar los puntos 3D asociados al tronco y la copa del árbol.

La segmentación del tronco se realizó con el algoritmo Çylinder model segmentation", el cual detecta formas cilíndricas y estima un modelo cilíndrico para cada objeto en la escena. La segmentación de la copa se llevó a cabo con el algoritmo "Density based spatial clustering of applications with noise"(DBSCAN), que utiliza la distancia euclidiana para agrupar puntos similares y clasificar los puntos restantes como ruido. Para validar el sistema, se compararon las características dendrométricas estimadas con otros métodos de medición manuales.

2.2.2. Antecedentes internacionales

Internacionalmente, se han realizado muchos aportes científicos al manejo de redes neuronales y procesamiento de imágenes para la segmentación, clasificación y reconocimiento de objetos, especialmente los destinados a los vehículos y robots autónomos. Dichos avances han abarcado una amplia variedad de contextos, incluyendo tanto entornos internos como externos, y se han aplicado en diversas funcionalidades, desde el seguimiento y localización de objetos hasta la generación de mapas semánticos detallados.

En el articulo [13] presentan un método eficiente de segmentación de planos utilizando sensores RGB-D. Aprovechan la estructura de la nube de puntos de estos sensores para implementar un algoritmo de crecimiento de región que sea computacionalmente eficiente. Su algoritmo se basa en la propagación de una región inicial a partir de píxeles adyacentes que cumplen con

ciertos criterios de similitud. Los umbrales y parámetros utilizados se adaptan a la profundidad medida, lo que permite un avance más rápido en las coordenadas de píxeles. El método fue evaluado en dos datasets, los criterios de evaluación fueron el cálculo de los valores IoU (Intersection over Union), la cual es una métrica en problemas de segmentación que mide la superposición entre dos regiones en una imagen, y la velocidad de procesamiento por imagen. Obtuvieron valores IoU entre 0.8 y 0.6 con una velocidad entre 30 ms y 20 ms en el Kinect Dataset de Oehler y entre 1 y 0.8 IoU con una velocidad entre 30 ms y 10 ms en el SEGCOMP ABW Dataset.

Por otra parte, en [23] abordan un método para la creación de un mapa semántico para la conducción autónoma en entornos urbanos. Para ello, fusionan el mapa de puntos obtenidos mediante LiDAR con imágenes semánticas capturadas por 6 cámaras especializadas para visión artificial (Mako G319). La fusión se realiza a través de transformaciones geométricas, extrayendo pequeñas regiones del mapa de nube de puntos densos y proyectándolas en la imagen segmentada semánticamente para recuperar información de profundidad y extraer características viales aplicables en el mapeo HD.

Aunque la segmentación no es el enfoque principal, es uno de los componentes clave para la creación del mapa semántico. Para lograr una segmentación semántica, se utilizó un enfoque basado en aprendizaje profundo. En primer lugar, se empleó una arquitectura de red DeepLabV3+ [38] para extraer características globales y locales de la escena de imágenes 2D. Posteriormente, estas características se clasificaron en objetos de interés (Tabla 2.2) mediante una arquitectura pre-entrenada ResNeXt50 [39].

Objetos de interés			
Cordón de cera	Cebra peatonal	Vegetación	Camino/carril
Edificios	Personas	Camiones	Tapas de alcantarilla
Señalización de carril	Señales de tráfico	Motociclistas	Bicicletas
Postes	Motocicletas	Buses	Cielo
Carros	Ciclistas	Acera	

Tabla 2.2: Objetos de interés para la creación de un mapa semántico probabilístico.

Para entrenar toda la red utilizan el dataset Mapillari Vistas [40] que contiene escenas e imágenes de calles destinas a escenarios para vehículos autónomos en entornos urbanos. La métrica que utilizaron para observar el rendimiento de segmentación en su método fue mIoU *(mean Intersection over Union)*, obteniendo resultados de 68.23 %.

Mientras que en [23] se emplea el aprendizaje profundo y la extracción de características en imágenes 2D, en el artículo [14] evalúan directamente las características extraídas de las nubes de puntos generados por el LiDAR mediante el método GndNet. Este método utiliza la red PointNet [41] junto con una red de codificación de características de pilar/voxel, Pillar Feature Encoding, para clasificar la nube de puntos en dos clases: terrestres y no terrestres. El objetivo

de esta clasificación es diferenciar entre el suelo y los objetos en el entorno. Los resultados obtenidos mostraron un alto rendimiento, con un valor de 83.6 % en la métrica mIoU.

En un estudio comparativo [24], se evaluaron dos métodos de segmentación de nubes de puntos obtenidas con un LiDAR de 32 canales: segmentación basada en la distancia euclidiana y la segmentación basada en el rango cilíndrico.

Los resultados obtenidos mediante la implementación en ROS indicaron que el método basado en el rango cilíndrico es más rápido y efectivo en la segmentación, incluso a mayores distancias donde la densidad de puntos es menor. El método basado en el rango cilíndrico obtuvo una frecuencia de segmentación de 36 Hz, mientras que el método de agrupamiento euclidiano solo alcanzó 4 Hz. Los autores sugieren que el método basado en el rango cilíndrico tiene el potencial de ser utilizado en vehículos autónomos donde la latencia es crucial.

En los artículos de revisión y comparación [42] y [43], se mencionan métodos importantes de segmentación basados en aprendizaje profundo y algunos datasets populares en la actualidad. Mientras que [42] se centra en los métodos para imágenes RGB-D, [43] se enfoca en los métodos para la nube de puntos capturadas por LiDAR.

Entre los datasets más conocidos para los métodos de segmentación RGB-D se encuentran NYUv2, SUN-RGBD, SceneNet RGB-D, entre otros. Por otro lado, algunos dataset más utilizados para la segmentación de mapas de nube de puntos obtenidos por LiDAR son KITTI 3D Objec Detection, SemantiKITTI, modelNet, nuScene. En cuanto a los modelos de segmentación RGB-D, se presenta una descripción general poniendo en relevancia la funcionalidad de la profundidad, ofreciendo ejemplos como ACNet, ESANet, 3DN-Conv, 2.5-Conv, GAD y otros modelos. Por otra parte, para los modelos basados en LiDAR se exponen modelos basados en voxels, puntos, en gráficos o vista múltiple, como VoxelNet, SECOND, PointPillar.

Las métricas de evaluación que utilizan son IoU, mIoU, Average Precision (AP), Accuracy, el tiempo inferencia y procesamiento. Por ejemplo, algunas menciones importantes en métodos de segmentación en RGB-D como la ESA Net* [25], alcanza un rendimiento de hasta 29.7 FPS, y precisiones de 51.58 % mIoU evaluado en NYUv2 dataset y 48.04 % mIoU evaluado en SUN-RGBD dataset. Además, el método GAD [26], tiene precisiones de 59.6 % mIoU en NYUv2 y 54.5 % mIoU en SUN-RGBD.

Por otro lado, en términos de métodos de segmentación para LiDAR, se destaca PointPillars [27], el cual logró un AP del 66.19 % en la identificación de objetos 3D, evaluado en el dataset KITTI 3D. Asimismo, se mencionan VoxelNet [28] y SECOND [29] que obtuvieron un AP del 58.25 % y 60.56 %, respectivamente.

El artículo [30], introduce la CGGC-Net, una avanzada red neuronal convolucional diseñada para la segmentación semántica de nubes de puntos LiDAR. Ante el avance de los escáneres 3D y la creciente necesidad de percepción espacial detallada, la CGGC-Net se distingue por su habilidad de integrar información geométrica local detallada desde múltiples campos receptivos. Un aspecto innovador es la incorporación de una pérdida contrastiva, que optimiza las características semánticas para ser más discriminativas, mejorando la clasificación punto a punto.

La CGGC-Net ha demostrado un alto rendimiento, particularmente en el conjunto de datos SemanticKITTI, que se basa en nubes de puntos obtenidas de LiDAR. A pesar de su enfoque inicial en nubes de puntos de LiDAR, la CGGC-Net ha demostrado ser versátil al ser evaluada en nubes de puntos derivadas de sensores RGB-D, en particular con el dataset S3DIS. Esta adaptabilidad es respaldada por resultados cuantitativos: en SemanticKITTI, la red logró un mIoU del 58.4 %. En S3DIS, alcanzó una precisión general (OA) del 88 %, una precisión media (mAcc) del 80.3 % y un mIoU del 70.2 %.

Por otra parte, el trabajo [31] se centra en la tarea de rastrear y reconstruir simultáneamente múltiples objetos en movimiento en una escena, particularmente relevante para aplicaciones robóticas. Aunque la reconstrucción y el rastreo dinámico son los objetivos principales, un módulo vital en este proyecto es la segmentación geométrica-semántica de las secuencias RGB-D de entrada. El sistema implementa una estrategia combinada de segmentación, utilizando método paralelo entre RGB y profundidad. Vale aclarar que este trabajo se realizó con el framework ROS.

Las imágenes RGB entrantes se procesan con Mask R-CNN [35] para detectar y reconocer instancias de objetos, prediciendo una máscara de segmentación para cada uno de ellos. Paralelamente, los cuadros de profundidad correspondientes se procesan mediante un enfoque de segmentación geométrica que agrupa puntos 3D a través de una estrategia de crecimiento regiones. Estos dos resultados de segmentación se combinan emparejando los clusteres extraídos con las máscaras de segmentación predichas utilizando una medida de superposición 2D, utilizando un umbral de superposición de 80 %.

El estudio [32] introduce la çonvolución 2.5D", una innovadora operación de convolución que busca aprovechar la relación geométrica tridimensional inherente a las imágenes RGB-D, pero sin el alto costo computacional de las convoluciones 3D. Esta propuesta emula una convolución 3D utilizando núcleos de convolución 2D, permitiendo así un procesamiento eficiente que captura relaciones espaciales entre píxeles de manera similar a la convolución 3D. La singularidad de este enfoque radica en su economía de recursos computacionales, dado que al fusionar información RGB con datos de profundidad, se minimiza la carga computacional vinculada a la segmentación semántica.

Esta combinación resulta en un modelo que no solo es capaz de manejar información geométrica de manera efectiva, sino que también puede integrarse sin problemas en CNNs preentrenadas. Por ejemplo, los autores exponen que adaptaron la red DeepLabv3+[38], con la que, en el dataset NYUDv2, se obtuvo un mIoU del 48.1 % y una precisión del 75.3 %. Mientras que en el dataset SUN-RGBD, se alcanzó un mIoU del 47.3 % y una precisión del 81.9 %. Estos resultados, junto con la eficiencia computacional obtenida al combinar características de profundidad y RGB, validan la eficacia y potencial del enfoque de convolución 2.5D en la segmentación semántica.

La investigación [33] aborda el desafío de identificar con precisión objetos en entornos interiores complejos utilizando brazos robóticos y utilizando la segmentación semántica de nubes de puntos, los autores presentan EG-PointNet, una red neuronal que incorpora el módulo EG-conv. En pruebas realizadas en el conjunto de datos S3DIS, demostró efectivo, logrando una precisión media (mAcc) del 69.2 %, una precisión general (OA) del 85.9 % y un mIoU del 59 %. Los autores mencionan que la implementación de esta investigación tiene el potencial de mejorar las interacciones entre robots y entornos interiores desafiantes, con el objetivo de integrarlo en módulos de visión de brazo robótico para ayudar a pacientes con trastornos del movimiento.

Modelos de segmentación de instacias como [34] y [35] fueron evaluados en el conjunto de datos MS COCO para la deteccion y segmentación de 80 diferentes clases. En sus configuraciones de mayor tamaño (YOLOv11-X y Mask R-CNN con backbone ResNeXt-101-FPN), obtuvieron un mAP[50-95]mask de 43.5 % y 37.1 %, respectivamente.

YOLOv11 tambien cuenta con modelos más pequeños, por ejemplo, YOLO11m que es la modelo medio que existe, teniendo casi un tercio de parametros que el modelo YOLO11x cuenta con resultados similares con un mAP[50-95]mask de 41.3 %. De forma análoga, Mask R-CNN ofrece distintos tamaños de red: la versión de tamaño medio con backbone ResNet-50-FPN registra un mAP[0.50-0.95]mask de 34.2 %.

Un modelo que tiene buen rendimiento con tamaño reducido es MobileNet. En [36] se presenta la variante LRASPP_MobileNetV3_Large, diseñada para segmentación semántica y evaluada en dos datasets. En Cityscapes[44] alcanzó un mIoU de 72.64 %; mientras que en el subconjunto de COCO limitado a las mismas 20 clases de Pascal VOC[45] obtuvo un mIoU de 57.9 % y un pixel accuracy de 91.2 % [46]. Es un modelo relativamente compacto que pesa 12,5 MB y cuenta con 3’221,538 parámetros.

En los antecedentes se puede evidenciar numerosos de métodos de segmentación que abarcan diferentes enfoques, como el aprendizaje profundo, la utilización de características similares en regiones de la imagen, el uso de las características de forma, entre otros. Estos métodos se utilizan en una gran variedad de aplicaciones y entornos, incluyendo el uso en vehículos autónomos en ambientes interiores.

Los objetos de interés comunes en dichos entornos incluyen mesas, asientos, pisos, paredes, camas, puertas, pinturas, contenedores de basura, personas, ventanas, repisas, armarios, televisores, techos, computadores, plantas, entre otros. Estos objetos son frecuentemente utilizados para evaluar y validar la efectividad de los métodos de segmentación en escenas interiores.

2.2.3. Datasets

En los últimos años, la comprensión de escenas interiores se ha visto impulsada tanto por la llegada de sensores RGB-D asequibles como por la consolidación, fiabilidad y uso extendido de los grandes conjuntos de datos RGB tradicionales. Los primeros combinan la información de color (RGB) con mapas de profundidad (D), lo que enriquece la descripción geométrica del entorno y facilita la segmentación de objetos. Sin embargo, los datasets puramente RGB (ejemplo ADE20K [47], COCO [48] o ImageNet[49]) siguen desempeñando un papel clave.

El preentrenamiento de los backbones en conjuntos RGB, gracias a su gran diversidad de escenas y objetos, aporta características visuales robustas que facilitan la transferencia a tareas en interiores. Mantener las métricas sobre el dominio RGB permite, además, garantizar la comparación con trabajos previos mediante validación cruzada. Por otro lado, en escenarios desafiantes, donde la información de profundidad puede resultar ruidosa, por ejemplo, en presencia de vidrios o superficies reflectantes o cuando se emplean sensores de bajo costo, los modelos basados exclusivamente en RGB pueden ofrecer un rendimiento superior frente a aquellos que incorporan datos de profundidad.

Por ello, esta sección revisa datasets RGB-D representativos de interiores (NYU Depth v2, SUN RGB-D y S3DIS) y conjuntos RGB de uso general o específicos de interiores (el subconjunto de interiores de ADE20K y MS COCO), destacando sus clases y nivel de anotación.

2.2.3.1. RGB-D Datasets

A continuación, se presentan tres de los *datasets* RGB-D representativos empleados en la investigación de visión por computadora para entornos interiores, los cuales han sido utilizados en tareas de segmentación semántica, detección de objetos y reconstrucción tridimensional. Estos conjuntos de datos destacan por combinar información de color y profundidad, permitiendo a los modelos aprender relaciones espaciales más ricas y diversas. Las características principales de cada *dataset* se resumen en la Tabla 2.3.

NYUv2 Dataset [50] (2012): Secuencias RGB-D capturadas con Microsoft Kinect en interiores domésticos desordenados, con anotaciones detalladas de superficies, objetos y relaciones de soporte, diseñadas para evaluar algoritmos de segmentación y percepción en escenarios de vida real.

SUNRGB-D Dataset [51] (2015): Conjunto de 10,335 imágenes RGB-D con etiquetas semánticas, capturadas con distintos sensores (Intel RealSense, Asus Xtion, Kinect v1/v2) para impulsar las principales tareas de comprensión de escenas.

Stanford 2D-3D-S Dataset (S3DIS) [52] (2016): Con cobertura de más de 6 000 m^2, este dataset incluye 70 000 imágenes RGB-D y modalidades 2.5D/3D (normales, nubes XYZ) con anotaciones semánticas y geométricas a nivel de instancia. Capturado con la cámara Matterport, tres sensores de luz estructurada en rotación 360°, está pensado para tareas de comprensión y reconstrucción detallada de espacios interiores.

Dataset	# Imágenes	Tipos de datos	# De clases (objetos de interés)	Sensor
NYUv2	1449 imágenes etiquetadas	RGB-D	13	Microsoft Kinect
SUN RGB-D	10,335 imágenes con anotaciones	RGB-D	40	Intel RealSense Asus Xtion, Kinect v1 y v2
S3DIS	70,000+ imágenes RGB; 1,413 imágenes equirectangulares	RGB-D, normales superficiales, imágenes XYZ globales	13	Cámara Matterport

Tabla 2.3: Características de los datasets RGB-D de entornos interiores.

2.2.3.2. RGB Datasets

Por otro lado, se presentan dos de los datasets RGB con subconjuntos con objetos presentes en interiores y las características principales se muestran en la Tabla 2.4.

ADE20K Dataset [47] (2016-2018): Reúne más de 25,000 imágenes densamente anotadas, procedentes de los conjunto de datos LabelMe[53], SUN[54] y Places[55]. En promedio se segmentan 19.5 instancias de 10.5 categorías por imagen. Los diferentes objetos ascienden a 2,693 clases (si se incluyen partes son 3,169 clases), para facilitar la comparación entre métodos, el benchmark oficial define el subconjunto SceneParse150 con las 150 clases más frecuentes, que cubre el 92.8 % de los píxeles.

MS COCO 2017 Dataset [48] (ECCV 2014, edición 2017): Consta de cerca de 118,000 imágenes de entrenamiento y 5,000 de validación. Incluye 80 categorías de objetos, cada fotografía contiene en promedio 3.5 categorías y 7.7 instancias, brindando un contexto rico para el aprendizaje de modelos que deben razonar sobre múltiples objetos simultáneamente. La métrica oficial es la precisión media (mAP) en IoU 0,50–0,95 para detección y segmentación de instancias, lo que permite comparaciones directas entre diferentes modelos y métodos.

Dataset	# Imágenes etiquetadas	Tipos de datos	# de clases (objetos de interés)
ADE20K	25 574 + 2 000 (*train+val*)	RGB	2 693
MS COCO	118 000 + 5 000 (*train2017+val2017*)	RGB	80

Tabla 2.4: Características de los datasets RGB ampliamente usados para segmentación de escenas interiores y generales.

2.3. Marco Teórico

En este aparatado se definen conceptos importantes empleados en este trabajo de grado, que permiten brindar al lector una contextualización de la investigación y los conocimientos necesarios para abordar esta temática. Además, también se presenta información del algoritmo que se va a implementar.

2.3.1. ROS (Robotic Operating System)

ROS es un framework de código abierto para el desarrollo robótico que proporciona herramientas, bibliotecas y documentación para facilitar la comunicación modular entre sensores, actuadores, algoritmos de percepción, planificación y controladores, permitiendo combinar componentes según las necesidades de cada aplicación [17]. Una de las características más importante de ROS, es su arquitectura basada en nodos independientes que realizan tareas específicas y se comunican mediante mensajes que transportan datos de sensores, comandos y otra información. Estas interacciones se gestionan a través de tópicos, acciones y servicios, mientras que el ROS Master coordina las publicaciones y suscripciones de cada nodo. (ver Figura 2.1).

Tópicos: Canales unidireccionales en los que un nodo publica mensajes y otros nodos se suscriben para recibirlos, ideal para la transmisión continua de información (datos de sensores y/ó actuadores)

Acciones: Permiten ejecutar tareas complejas y prolongadas con interacción continua entre nodos; cada acción incluye meta, retroalimentación y resultado, y los nodos envían solicitudes y reciben actualizaciones hasta su finalización.

Servicios: Comunicación cliente-servidor de solicitud-respuesta entre nodos, ideal para tareas puntuales sin interacción continua, como ajustar parámetros o ejecutar cálculos específicos.

ROS también incluye herramientas de lanzamiento, introspección, depuración, visualización, trazado, registro y reproducción, que facilitan el desarrollo de aplicaciones robóticas.

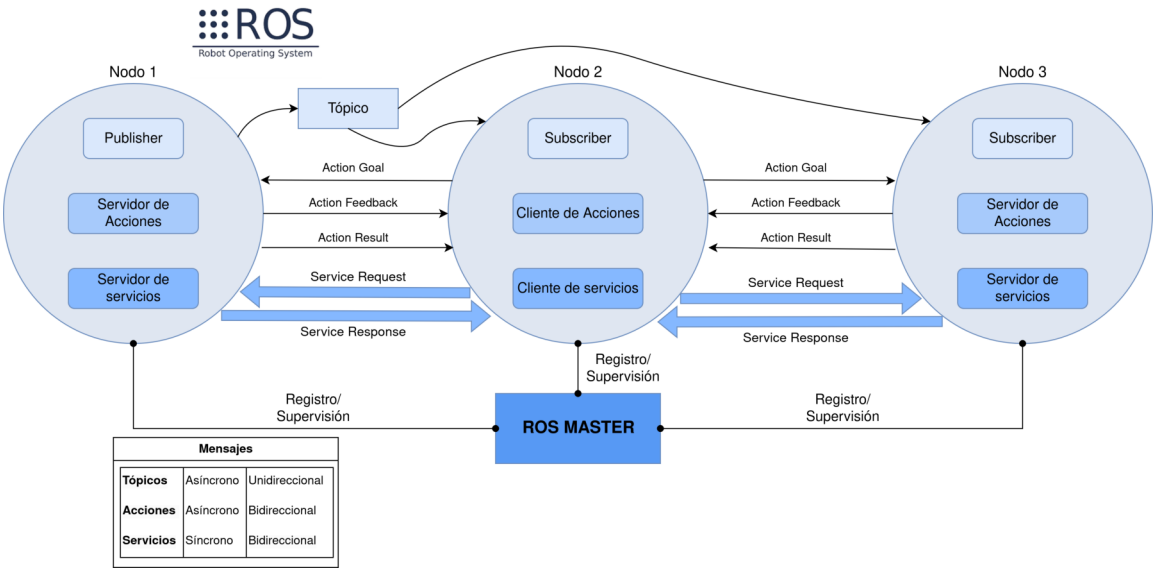


Figura 2.1: Mensajes y diagrama de Nodos en ROS.

2.3.2. Robot Móvil Pioneer 3DX

Un robot móvil es, según [56], una máquina capaz de desplazarse autónomamente, y de acuerdo con [57], un dispositivo controlado por software que, mediante sensores, identifica y recorre su entorno. Se clasifican por medio (aéreo, terrestre o submarino) y, en el caso terrestre, por locomoción (tracción diferencial, orugas, patas, ruedas omnidireccionales, etc.) [58]. Un ejemplo es el Pioneer 3DX (Fig. 2.2), plataforma diferencial de dos ruedas, esto implica que cada rueda o actuador motorizado del vehículo puede girar a velocidades y direcciones independientes. Este robot esta diseñado por MobileRobots Inc., y esta equipado con sonar frontal, codificadores, batería, firmware ARCOS y compatible con el Pioneer SDK advanced de desarrollo de software de robótica móvil [59].

Figura 2.2: Robot Móvil Pioneer 3DX

2.3.3. Metodología SCRUM

Scrum es un marco de trabajo ágil para la gestión y desarrollo de proyectos. Se basa en principios de transparencia, inspección y adaptación para promover la colaboración y la entrega

incremental de valor [60]. Los proyectos se dividen en ciclos de trabajo llamados "sprints", que son períodos de tiempo fijos y cortos (generalmente de 1 a 4 semanas). Durante cada sprint, se realiza una planificación, se lleva a cabo el desarrollo y se realiza una revisión para obtener retroalimentación.

Scrum es una metodología ágil que promueve equipos autoorganizados y multidisciplinarios, con roles definidos como el *Scrum Master* (facilita el proceso), el *Product Owner* (define los requerimientos y prioridades del producto) y el *Equipo de Desarrollo* (implementa las funcionalidades). Se apoya en herramientas visuales como tableros y en reuniones periódicas —*Daily Scrums y Sprint Reviews*— para facilitar la coordinación. Su objetivo es entregar valor de forma temprana y continua, adaptándose mediante la retroalimentación del cliente (figura 2.3).

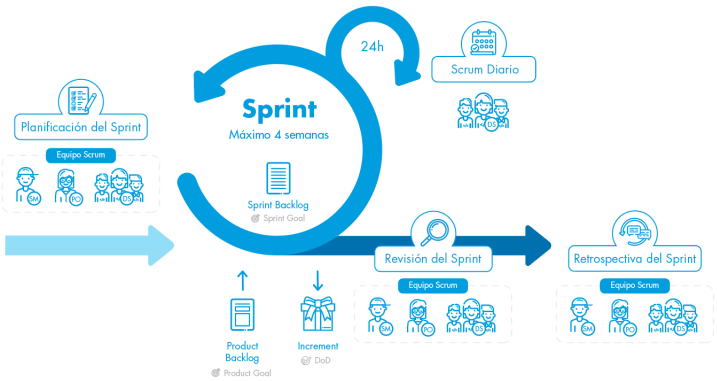


Figura 2.3: Proceso de Scrum
Fuente: https://netmind.net/scrum-el-pasado-y-el-futuro/

2.3.4. Segmentación

La segmentación en visión artificial es el proceso de dividir una imagen en diferentes regiones u objetos significativos y distinguibles, basándose en características inherentes, como el color, la textura, la forma o el brillo. Tiene diversas aplicaciones, como el reconocimiento, seguimiento o detección de objetos, y se utiliza en áreas como la robótica o la medicina [61]. En segmentación todos los pixeles que pertenecen a una misma categoría tienen una etiqueta común, las principales maneras de realizar este proceso es por similitud o discontinuidad.

Los métodos de segmentación basados en similitud se enfocan en detectar características similares entre los píxeles de una imagen para formar segmentos. Estos métodos utilizan parámetros y umbrales para facilitar la segmentación de la imagen. Algunos de estos métodos se basan en técnicas de aprendizaje automático. Por otro lado, los métodos basados en discontinuidad se centran en detectar cambios en los valores de intensidad de los píxeles dentro de una imagen para generar segmentos. Estas técnicas son utilizadas en la detección de líneas, puntos y bordes.

En cuanto a los modelos de segmentación tradicionales, se utilizan diferentes enfoques como la segmentación por regiones, bordes, clústeres y umbrales. Sin embargo, en los enfoques más recientes, se ha adoptado el aprendizaje profundo como base para la segmentación, como U-Net[62], SegNet[63], DeepLab[38], entre otros.

2.3.4.1. Tradicionales

Basada en regiones: Esta técnica divide la imagen en regiones homogéneas según características como color, textura o intensidad. Entre sus enfoques destacan la segmentación dividida y combinada, que fragmenta y luego fusiona regiones según su similitud, y la segmentación basada en gráficos, que utiliza la estructura gráfica de la imagen para delimitar regiones mediante bordes [61]. En [64], se emplea esta última estrategia integrando tanto la intensidad como la profundidad como criterios clave (figura 2.4).

Figura 2.4: De izquierda a derecha: imagen original, límites extraídos de la intensidad de la imagen, límites extraídos de la profundidad, límites resultantes de la fusión de la información RGB y de profundidad

Basada en borde: Esta técnica detecta los contornos que separan objetos del fondo en una imagen. Entre los métodos más comunes se encuentran el detector de Canny, que emplea un algoritmo de varias etapas secuenciales; el operador de Sobel, basado en gradientes horizontales y verticales; y el Laplaciano de Gauss (LoG), que combina suavizado gaussiano con detección de bordes mediante el operador laplaciano [61] (figura 2.5).

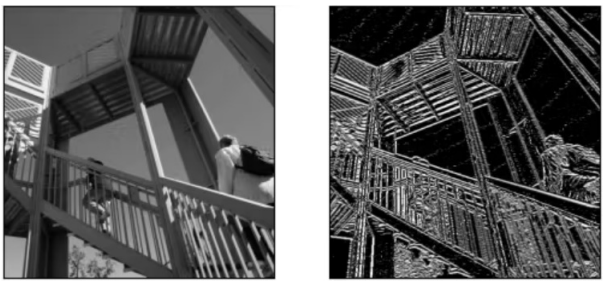


Figura 2.5: Ejemplo de detección de bordes con el operador Laplaciano-Gaussiano

Basada en clústeres o agrupamiento: Esta técnica agrupa píxeles con características similares en segmentos, siendo K-means y Mean Shift unos de los métodos más comunes. K-means se dividen los píxeles en K clústeres según una métrica de distancia, mientras que Mean Shift

agrupa desplazando los puntos hacia regiones de mayor densidad. Aunque son eficientes, su precisión puede verse limitada en escenas complejas [61]. En [65] y [66], K-means es una pieza central en los algoritmos de segmentación propuestos (figura 2.6).

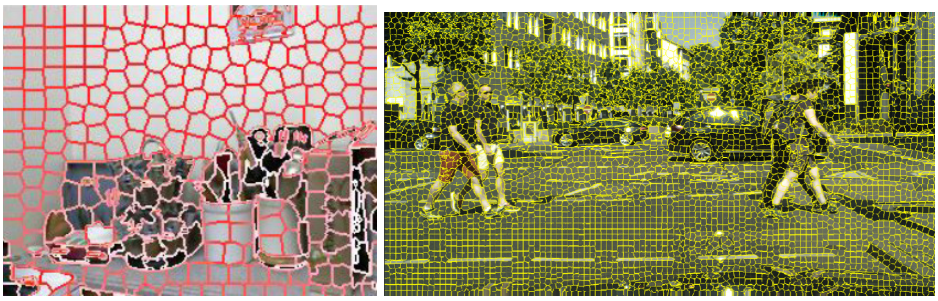


Figura 2.6: Ejemplo de imágenes segmentadas basado en Superpixel

Basada en umbrales o Thresholding: Esta técnica segmenta la imagen dividiendo los píxeles según su intensidad con respecto a un valor umbral. Puede aplicarse de forma global, asignando píxeles por encima del umbral al primer plano y los demás al fondo (figura 2.7), o mediante umbral adaptativo, que ajusta el valor localmente según la variación de iluminación o contraste. El umbral adaptativo es útil en imágenes con iluminación no uniforme o contraste variable [61].

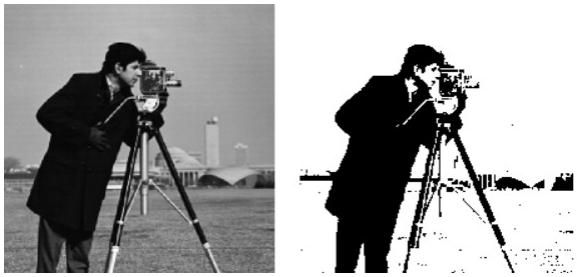


Figura 2.7: Ejemplo de segmentación basado en Umbrales

2.3.4.2. Segmentación basada en aprendizaje profundo

Las redes neuronales aplicadas a la segmentación de imágenes aprenden a identificar características relevantes a partir de un proceso de entrenamiento supervisado. Generalmente adoptan una arquitectura de tipo codificador–decodificador, en la que el codificador extrae representaciones jerárquicas de las características visuales mediante el uso de filtros convolucionales, mientras que el decodificador reconstruye la máscara de segmentación con la resolución original de la imagen (ver Figura 2.8).

Los enfoques de segmentación basados en aprendizaje profundo difieren en múltiples aspectos, como la arquitectura de la red, el uso de filtros y esquemas de muestreo, así como la incorporación de módulos adicionales. Estas variaciones influyen tanto en la estructura del

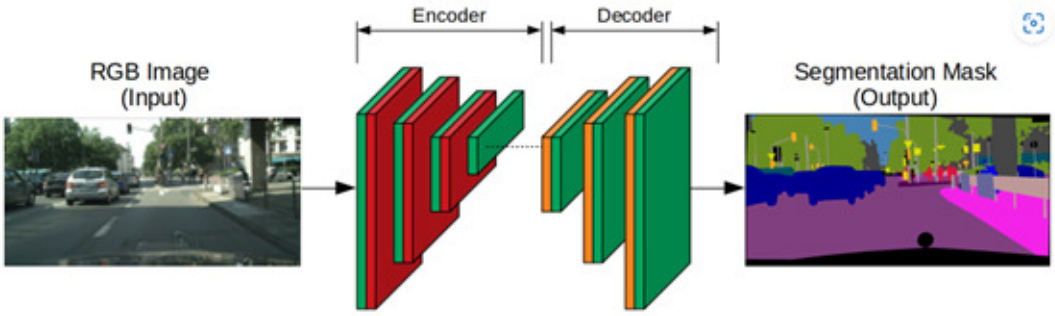


Figura 2.8: Arquitectura básica de un modelo de red neuronal para la segmentación de imágenes RGB. (Los cuadros verdes en el codificador y el decodificador representan capas convolucionales, mientras que los cuadros rojos y naranjas representan capas de muestreo descendente y muestreo ascendente respectivamente). *Fuente: [61].*

modelo como en la estrategia de entrenamiento, permitiendo su adaptación a diferentes requisitos y contextos. Dentro de este campo, se destacan dos tareas fundamentales: la segmentación semántica y la segmentación de instancias.

En la **segmentación semántica**, cada píxel se asigna a una categoría general (como silla, pared o suelo), sin distinguir entre objetos individuales; el modelo U-Net [62], con arquitectura codificador-decodificador, es un referente en esta tarea, evaluada típicamente mediante mean Intersection-over-Union (mIoU). En cambio, la **segmentación de instancias** combina detección y segmentación, identificando cada objeto por separado y generando una máscara única para él, incluso si comparte clase con otros; Mask R-CNN [35] es un modelo representativo de este enfoque, que añade una rama de predicción de máscaras sobre Faster R-CNN y se evalúa con Average Precision (AP) en múltiples umbrales de IoU.

Ambos enfoques necesitan **máscaras de referencia** cuya representación, ya sea mediante **polígonos** (listas de vértices, nativo en MS COCO [48] y Cityscapes [44]) o **máscaras binarias** $H \times W$ (1 = objeto, 0 = fondo), impacta directamente la estructura y el tamaño del conjunto de datos.

2.3.4.3. Métodos de mejoramiento de imagen

Ecualización de histograma: Técnica de procesamiento que mejora el contraste de imágenes con rangos de intensidad limitados [67], redistribuyendo los valores de intensidad para obtener un histograma más uniforme. Esto permite resaltar detalles relevantes de la imagen (ver figura 2.9).

Figura 2.9: De izquierda a derecha: Imagen Original, Imagen aplicando un ecualización de histograma.

Transformación exponencial: Esta técnica amplía el rango dinámico de intensidades aplicando una función exponencial a los píxeles [68], lo que resalta detalles en zonas oscuras. Con un coeficiente gamma entre 0 y 1 se obtiene una imagen más brillante, mientras que valores mayores a 1 la oscurecen. Es útil cuando los detalles relevantes se concentran en los niveles bajos de intensidad (ver figura 2.10).

Figura 2.10: De izquierda a derecha: Imagen Original, Imagen aplicando una transformada exponencial con coeficiente gamma de 0,3.

Filtro gaussiano: Utilizado para suavizar imágenes y reducir el ruido [67], este filtro aplica una convolución con una máscara basada en la función gaussiana. Es eficaz para eliminar ruido de alta frecuencia y produce un efecto de suavizado visual (ver Figura 2.11).

Figura 2.11: Aplicación del filtro gaussiano con un kernel de [5,5] en los dos tipos de transformaciones de imagen.

En este contexto, estos métodos de mejoramiento de imagen podrían ser útiles cuando el robot opera en entornos con condiciones de iluminación extremas. La ecualización de histograma

se adapta bien tanto a escenarios oscuros como brillantes, mientras que la transformación exponencial resulta más efectiva en ambientes predominantemente oscuros o muy iluminados. No obstante, ambas técnicas pueden intensificar el ruido, por lo que podría ser beneficioso aplicar un filtro gaussiano para suavizar artefactos y preservar la calidad visual.

2.3.4.4. Algunas métricas para evaluar métodos de segmentación.

La evaluación de algoritmos de segmentación requiere métricas que midan su precisión, eficacia y capacidad de adaptación a diversos entornos. En especial en el contexto de vehículos autónomos destinado a entornos interiores, donde las condiciones pueden ser cambiantes, variadas y dinámicas. Se hace necesario contar con un criterio objetivo que permita escoger mejores parámetros de un modelo o el algoritmo más adecuado.

1. Intersección sobre Unión (IoU) [61, 69]: Mide el grado de superposición entre la máscara generada por el modelo y la de referencia (ground truth). Su valor oscila entre 0 y 1, siendo mayor cuanto más precisa es la segmentación.

$$IoU = \frac{A \cap B}{A \cup B} \tag{2.1}$$

Donde A representa el conjunto de píxeles clasificados correctamente por el algoritmo de segmentación y B representa el conjunto de píxeles de referencia en la segmentación de interés.

2. Precisión [61][69]: Calcula el porcentaje de predicción positiva realizada por los clasificadores que son correctos.

$$\text{Precisión} = \frac{TP}{TP + FP} \tag{2.2}$$

Donde TP es el número de clases positivas correctamente clasificadas como positivas y FP es el número de clases negativas clasificadas incorrectamente como positivas.

3. Sensibilidad (Recall) [61][69]: Calcula el porcentaje de patrones positivos que el clasificador detecta correctamente. Mide la proporción de píxeles de la clase de interés que son correctamente identificados por el algoritmo de segmentación.

$$\text{Sensibilidad} = \frac{TP}{TP + FN} \tag{2.3}$$

Donde FN es el número de clases positivas clasificadas incorrectamente como negativas.

4. Average Precision (AP) a un umbral fijo de IoU[70]: Para cada categoría c (o promediando sobre todas ellas), la **AP** a un umbral de solapamiento τ se define como el área bajo la curva Precisión–Recall:

$$\text{AP}_\tau = \int_0^1 \text{Precisión}(r)\, dr \tag{2.4}$$

Donde la función Precisión(r) se interpola en cada nivel de *recall* r.

5. mAP en múltiples umbrales IoU[70]: Con el fin de evaluar la calidad de la segmentación bajo diferentes tolerancias de solapamiento, COCO computa la AP en un conjunto de diez umbrales $\{\tau_k\} = \{0{,}50, 0{,}55, \ldots, 0{,}95\}$ y promedia los resultados:

$$\mathrm{mAP}_{\mathrm{mask}} = \frac{1}{|\{\tau_k\}|} \sum_{\tau_k} \mathrm{AP}_{\tau_k}. \tag{2.5}$$

Esta métrica refleja la capacidad de localización como la delineación de las máscaras y es la principal métrica para trabajos de segmentación de instancias.

Vale mencionar que la AP descrita en el apartado anterior corresponde a la métrica oficial del reto PASCAL VOC [45]. En el benchmark MS-COCO, el nombre "AP" se reserva para la media sobre diez umbrales de IoU (0.50:0.95); por eso en la literatura aparece como mAP[0.50:0.95] [48]. Aunque comparten etiqueta, ambas cifras se calculan de forma distinta y no son directamente comparables.

Para cuantificar la calidad de las segmentaciones, se utilizará la métrica *IoU* junto con las métricas de *precisión* y *sensibilidad (recall)*, que aportan información tanto del acierto global como de la capacidad del modelo para identificar correctamente cada clase. En los experimentos de segmentación de instancias se recurrirá a *mAP50* y *mAP[0.50:0.95]*, proporcionando una valoraración de la calidad de los modelos entrenados y su comportamiento frente a diferentes tipos de objetos presentes en entornos interiores.

2.3.5. Descripción de los modelos elegidos

En este apartado se presentan los modelos seleccionados a nivel conceptual para los experimentos, cubriendo tanto segmentación de instancias como semántica. Se eligieron dos arquitecturas para cada tipo de segmentación, con diferentes enfoques y tamaños de red. Los modelos seleccionados se muestran en la Tabla 2.5.

Modelo	Backbone	Tarea	Parámetros	Descripción clave
YOLOv11-m-seg [34]	YOLOv11-M	Inst. Seg.	22'363,842	Versión ligera de una sola etapa con rama de máscara integrada.
Mask R-CNN [35]	ResNeXt-101-FPN	Inst. Seg.	107'427,760	Detector R-CNN de dos etapas con decodificador de máscaras de alta resolución.
ESANet-RGB [25]	ResNet-34	Sem. Seg.	32'059,408	Codificador multi-escala diseñado para segmentación semántica en imágenes RGB.
LRASPP MobileNet V3-Large [36]	MobileNet V3-Large	Sem. Seg.	3'218,988	Arquitectura ultraligera para óptimo compromiso precisión-cómputo.

Tabla 2.5: Resumen de los modelos seleccionados.

2.3.5.1. Arquitectura de YOLOv11-m-seg

YOLOv11-m-seg es la variante "medium" de la undécima generación de la familia YOLO, diseñada para segmentación de instancias en una sola etapa. Según lo descrito en [34], el modelo realiza detección de objetos y segmentación de forma concurrente. Para ello, extiende el **head** clásico de detección con una rama para máscaras que aprende coeficientes para combinar prototipos y recortarlos mediante las cajas predichas. Este diseño mantiene un equilibrio óptimo entre precisión y eficiencia computacional en comparación con sus predecesores. Fue desarrollado para cubrir necesidades en dispositivos periféricos y entornos de alto rendimiento, y su arquitectura se ilustra en la Figura 2.12.

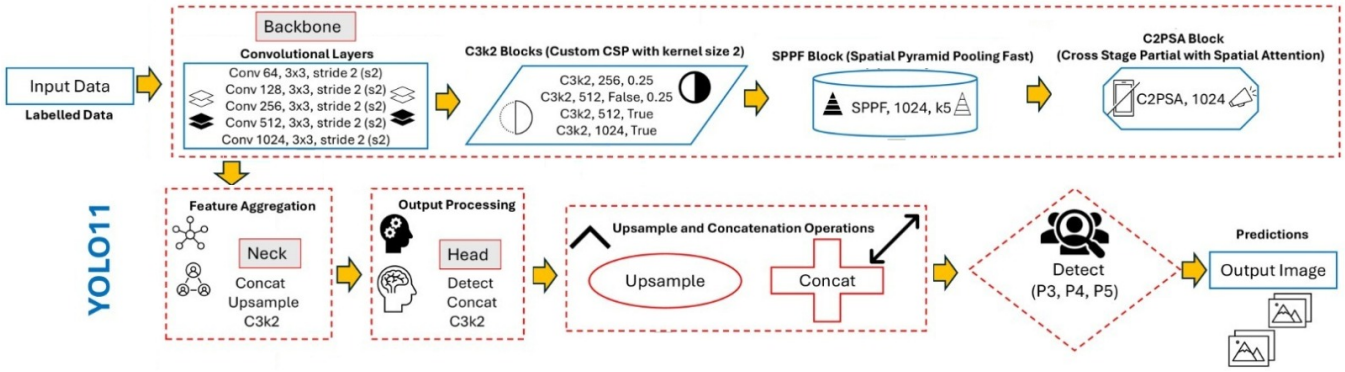


Figura 2.12: Arquitectura simplificada de YOLO 11
Fuente: Figura tomada de [71].

El modelo se estructura en tres bloques principales: el *Backbone*, que extrae mapas de características a distintas resoluciones mediante capas convolucionales; el *Neck*, que fusiona dichas características multiescala empleando técnicas de sobremuestreo y concatenación para recuperar detalles finos y contextuales; y la *Head*, que a partir de los mapas resultantes genera las predicciones finales (cuadros delimitadores, máscaras y etiquetas de clase). Esta arquitectura de "una sola etapa" característica de YOLO optimiza la eficiencia computacional y la atención espacial sin sacrificar precisión, lo que la hace idónea para segmentación de instancias en línea.

2.3.5.2. Arquitectura de Mask R-CNN (ResNeXt-101-FPN)

Mask R-CNN [35] combina detección de objetos y segmentación de instancias en un esquema de dos etapas. Primero, un backbone **ResNeXt-101** extrae mapas de características robustos; a continuación, un **Feature Pyramid Network (FPN)** fusiona esos mapas a múltiples escalas para detectar objetos de distintos tamaños. La Red de Propuesta de Regiones (RPN) genera propuestas que, tras un alineamiento espacial preciso mediante RoIAlign, se procesan en dos cabezales paralelos: uno para clasificación y regresión de cajas, y otro para predicción de máscaras binarias de alta resolución (por defecto 28×28) para cada instancia. Su arquitectura se puede observar en la Figura 2.13.

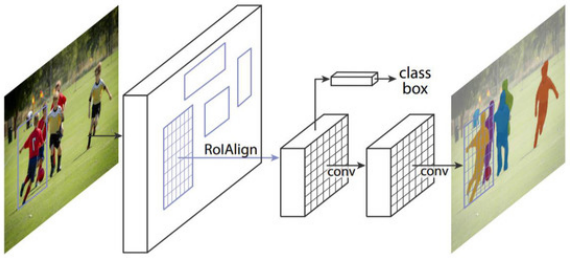


Figura 2.13: Arquitectura simplificada de Mask R-CNN
Fuente: Figura tomada de [35].

Con aproximadamente 107 millones de parámetros, Mask R-CNN ofrece una arquitectura robusta orientada a entregar máscaras de alta calidad y a manejar objetos de tamaños variados, aunque con mayor latencia en comparación con métodos de una sola etapa. Su arquitectura robusta lo hace ideal cuando la prioridad es la precisión en segmentación, especialmente en aplicaciones que toleran tiempos de procesamiento más largos.

2.3.5.3. Arquitectura de ESANet Modalidad RGB

La variante monocular de ESANet adapta el Efficient Scene Analysis Network [25] prescindiendo del encoder de profundidad y de la fusión RGB-D. Inspirada en SwiftNet [72], emplea un encoder ligero basado en ResNet-34 con bloques factorized Non-Bottleneck-1D (NBt1D), un módulo de contexto similar al módulo de agrupación piramidal de PSPNet [73] y un decoder de tres etapas que combina refinamiento convolucional, upsampling ligero y conexiones laterales para recuperar detalle. Su diseño completo se ilustra en la Figura 2.14.

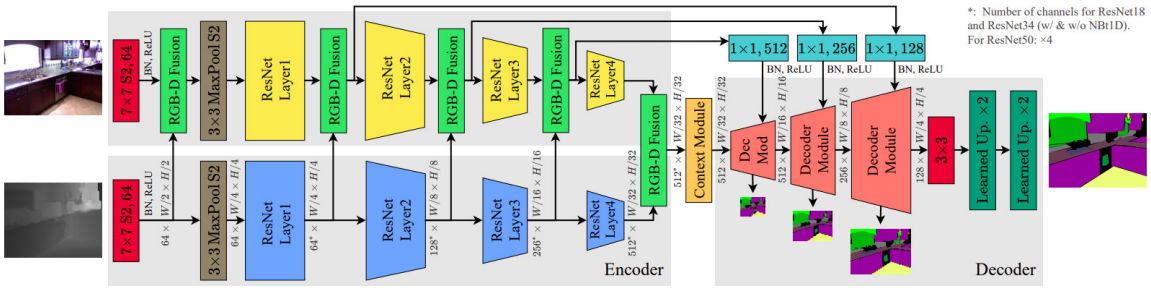


Figura 2.14: Arquitectura completa de ESANet RGB-D
Fuente: Figura tomada de [25].

ESANet–RGB ha sido concebido con un enfoque de eficiencia computacional. Sus autores diseñaron la arquitectura para: procesar distintas modalidades de entrada (en la versión original RGB-D), generar representaciones multiescala que capturan tanto detalles de bajo nivel como contexto global y producir una predicción final que conserve la misma resolución que la imagen de entrada. De este modo, lograr un equilibrio óptimo entre velocidad y precisión, lo que lo hace idóneo para aplicaciones de inferencia en línea que exigen baja latencia.

2.3.5.4. Arquitectura de LR-ASPP MobileNetV3-Large

MobileNetV3-Large se utiliza como backbone ligero, al que se añade el decoder Lite R-ASPP (Lite Reduced Atrous Spatial Pyramid Pooling) propuesto en [36]. Su arquitectura se observa en la Figura 2.15.

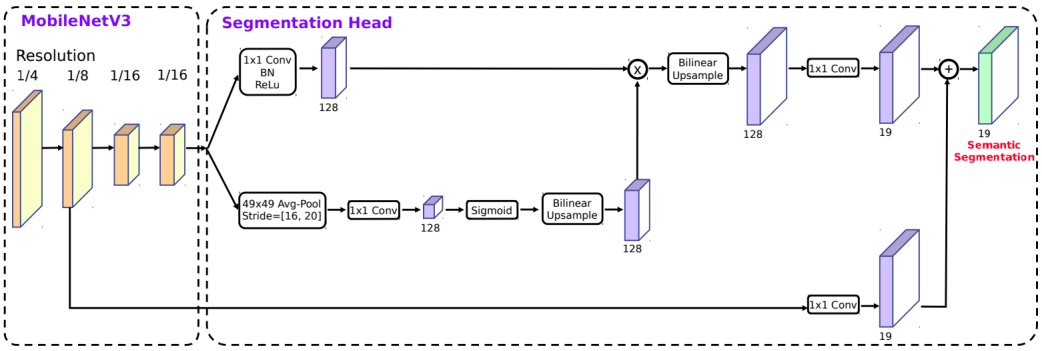


Figura 2.15: Arquitectura de MobileNetV3 con Head LR-ASPP
Fuente: Figura tomada de [36].

El backbone MobileNetV3-Large emplea bloques invertidos con convoluciones depthwise separables, expansiones 1×1, activación *hard-swish* y módulos *squeeze-and-excitation*, optimizados mediante NAS y NetAdapt para equilibrar precisión y latencia en CPU; el decoder Lite R-ASPP procesa de forma eficiente esas características, realzando regiones importantes y ampliando el campo visual a través de conexiones especializadas, generando mapas de segmentación a una resolución de 1/8. Con apenas 3.2 millones de parámetros lo hace idóneo para segmentación semántica en dispositivos con recursos limitados.

2.3.6. Observaciones finales

La presente búsqueda y síntesis de antecedentes y datasets ha ayudado a identificar ciertos aspectos importantes para el desarrollo e implementación de los diferentes métodos de segmentación. En primer lugar, se observa la presencia de diversos sensores que sirven para capturar los datos necesarios para realizar esta tarea, como el LiDAR, el Kinect, las cámaras monoculares, las cámaras estéreo, las cámaras de video, etc. Pero en general estos sensores brindan 3 tipos de datos: nube de puntos, imágenes RGB e imágenes RGB-D.

De lo anterior se deriva el tipo de entorno donde generalmente se utilizan. En entornos exteriores, las nubes de puntos y las combinaciones de diferentes sensores y tipos de datos son predominantes. Se observan combinaciones como las nubes de puntos generadas por LiDAR junto con imágenes RGB obtenidas de cámaras de video [23], o segmentación a través de la generación de nube de puntos a través de secuencias de imágenes RGB de una cámara monocular [22], como también existen implementaciones con solo la generación de nube de puntos

del LiDAR [27, 28, 29, 30]. Cabe resaltar que el uso de combinaciones de tipos de datos puede enriquecer la extracción de características del entorno, sin embargo, aumenta la exigencia computacional.

Por otro lado, en entornos interiores prevalece el uso de imágenes RGB-D y RGB. De ahí que, en esta revisión de datasets para segmentación en espacios interiores predominen aquellos basados en imágenes RGB-D y RGB [50, 51, 52, 47, 48]. Además de los datasets utilizados en interiores como NYUv2 y SUNRGB-D, también están los destinados a exteriores como KITTI 3D y SemanticKITTI. Esto indica la amplitud de aplicaciones y la diversidad de escenarios en los que se aplican los métodos de segmentación. Datasets como NYUv2, SUNRGB-D, MS COCO y KITTI 3D son recurrentes en las investigaciones, lo que indica su relevancia y utilidad en la validación de métodos de segmentación.

Es importante destacar que, al contar con una gama amplia de características, posibilita la detección y segmentación de un conjunto más extenso de clases. Se observó que los métodos tradicionales son efectivos en esta labor, sin embargo, suelen identificar un número menor de clases en comparación con los métodos basados en aprendizaje profundo o aquellos que combinan técnicas tradicionales de segmentación con aprendizaje profundo. La razón radica en la naturaleza del aprendizaje profundo, ya que están diseñado para gestionar grandes volúmenes de datos, por lo tanto, un buen entrenamiento de estas redes basadas en el aprendizaje profundo y buen conjunto de datos puede producir resultados óptimos y de calidad.

En el contexto de vehículos autónomos, se ha reconocido que el uso de la segmentación puede variar sus aplicaciones, desde extraer características dasométricas de árboles, el seguimiento y reconstrucción de múltiples objetos o hasta la creación de mapas semánticos. Aunque algunas investigaciones que trabajan interiores a menudo se centran en objetos estáticos como paredes, puertas, sillas, pisos, mesas, etc. Es evidente que estos métodos de segmentación pueden cambiar o mejorar dependiendo de la aplicación que se vaya a aplicar y extenderse a objetos en movimiento, como personas, es el caso de algunos artículos [25, 26, 31].

Para justificar aún más la elección de los objetos de interés, es crucial considerar el propósito final de segmentación. Si el objetivo es la navegación robótica, entonces los elementos de acceso y los objetos móviles son críticos. Si el propósito es el diseño interior o la interacción en un espacio cerrado en concreto, entonces los planos estructurales, el mobiliario y objetos específicos del espacio son importantes. La elección y justificación de los objetos de interés deben alinearse con el objetivo final de la segmentación y análisis.

Por último, se observó que no todos los trabajos reportan métricas de precisión y velocidad. Entre aquellos que sí lo hacen, existe una amplia variabilidad tanto en el tipo como en el rango de las métricas empleadas. Las medidas de rendimiento más comunes corresponden al rendimiento de precisión, destacándose el uso de *mIoU* y *mAP (AP)* como indicadores principales. Cabe resaltar que algunos estudios priorizan la velocidad de procesamiento sobre la precisión,

lo que evidencia que, en el contexto de los vehículos autónomos, ambos aspectos resultan igualmente relevantes para garantizar un desempeño eficiente y confiable.

Capítulo 3

DESARROLLO DEL APLICATIVO LU-SEG

3.1. Introducción

En este capítulo, se presenta el desarrollo de Lu-Seg, un aplicativo para la experimentación de un algoritmo de segmentación implementado en un robot móvil diferencial, y se abordan tanto su diseño como su implementación, organizados en dos secciones: “Diseño”, donde se definen los requerimientos, la interfaz de usuario, la base de datos y su API, así como los diagramas conceptuales, relacional y de nodos ROS; e “Implementación”, que abarca la preparación de datos, la puesta en marcha de las arquitecturas de red y la creación de los módulos de configuración, ejecución y reportes.

Para asegurar un desarrollo coherente y eficaz, se presentan los requerimientos funcionales y no funcionales, un diagrama conceptual, uno relacional y uno de nodos, con el objetivo de visualizar la interacción de todos los elementos del sistema, las tablas de la base de datos, y la interacción entre los nodos (paquetes de software) involucrados en el aplicativo.

A continuación, se expone el preprocesamiento y filtrado aplicados a MS COCO y ADE20K. Posteriormente, se implementa y se entrenan cuatro arquitecturas candidatas (YOLOv11, Mask R-CNN, ESANet–RGB y LR-ASPP MobileNetV3), evaluándolas según su precisión, recall y el promedio de IoU o mAP segun la tarea de segmentación. Este análisis permite identificar el modelo con buen desempeño según los recursos computacionales que se disponen y el objetivo planteado de procesamiento en linea, facilitando una integración óptima en el aplicativo Lu-Seg.

Finalmente, se describe la integración del modelo seleccionado en Lu-Seg y los módulos de configuración, ejecución, así como el modulo replay con estadísticas de desempeño. Esta sección cierra el ciclo de desarrollo, asegurando que la aplicación sea robusta, reproducible y capaz de segmentar en línea los objetos de interés en interiores.

3.2. Diseño del Aplicativo LU-SEG

En esta sección se expone el diseño del aplicativo Lu-Seg por medio de requerimientos funcionales y no funcionales, el diagrama conceptual, el diagrama relacional de la base de datos, y el diagrama de nodos del aplicativo en ROS. Por otro lado, las historias de usuario de la metodología Scrum se presenta como la documentación adicional en anexos.

3.2.1. Requerimientos Funcionales y No Funcionales

Los requerimientos funcionales establecen las características operativas que el sistema debe cumplir para asegurar su correcto funcionamiento, presentados en las tablas 3.1, 3.2, 3.6. Estos se dividen en dos categorías: los requerimientos del frontend (tabla 3.2), que describen las funciones y la interfaz de usuario, y los requerimientos del backend (tabla 3.6), que especifican las funcionalidades del lado del robot. Por otro lado, los requerimientos no funcionales, presentados en la tabla 3.7, detallan las características del software en términos de bibliotecas, entornos de desarrollo y versiones. Además, incluyen las especificaciones de hardware necesarias para la compatibilidad de la aplicación.

Tabla 3.1: Especificación de Requerimientos Funcionales

Universidad del Valle	**Universidad del Valle – DESARROLLO DE UN APLICATIVO EN ROS PARA LA SEGMENTACIÓN DE OBJETOS DE INTERÉS EN INTERIORES USANDO UN ROBOT MÓVIL –**	Rev.: 000
Título: Especificación de Requerimientos Funcionales		**Documento:** ERF-001

REVISIÓN HISTÓRICA			
Rev.	**Descripción del Cambio**	**Autor**	**Fecha**
001	Construcción del documento	Juan Camilo Chavez Castro	19/10/2023
002	Correcciones	Juan Camilo Chavez Castro	23/10/2023
003	Correcciones	Juan Camilo Chavez Castro	1/10/2023
004	Correcciones	Juan Camilo Chavez Castro	14/11/2024
005	Revisión		

Tabla 3.2: Requerimientos Funcionales para el Frontend

Ref #	Frontend	Categoría	Paquete
	Home		
1	El home tendrá 3 botones para 3 apartados. Un botón llevará a una ventana de configuración, otro a ejecución y el último a sesión previas.	(E)	GUI

	Configuración		
2	La interfaz contará con un campo de texto para que el usuario digite la dirección IP del computador del robot.	(E)	Comunicación
3	La interfaz contará con un campo de texto para el nombre de usuario del computador del robot.	(E)	Comunicación
4	La interfaz contará con un campo de texto para la contraseña de usuario del computador del robot.	(E)	Comunicación
5	La interfaz contará con un campo de texto para la dirección IP de la máquina donde se ejecuta el nodo maestro.	(E)	Comunicación
6	La interfaz contará con un campo de texto para el puerto de comunicación (11311 por defecto).	(E)	Comunicación
7	Una vez el usuario haya llenado los campos de texto, el botón "conectar" estará disponible en la interfaz gráfica.	(E)	Comunicación
8	Si el usuario no se logra conectar, el sistema mostrará una ventana de diálogo con el posible problema.	(E)	Comunicación
9	La interfaz mostrará al usuario si está conectado al robot móvil.	(E)	Comunicación
10	La interfaz permitirá al usuario modificar los valores de velocidad lineal y angular del robot a través de campos de texto y un botón.	(E)	Movimiento del robot
11	La interfaz debe proporcionar la capacidad de guardar, cargar y modificar configuraciones mediante tres botones dedicados.	(E)	GUI, Base de datos
12	La interfaz gráfica permitirá al usuario nombrar la sesión a través de una ventana de diálogo.	(E)	GUI, Base de datos
	Ejecución de la sesión		
13	La interfaz gráfica debe proporcionar un botón para iniciar la ejecución.	(E)	GUI
14	Una vez el usuario oprime el botón iniciar la ejecución, el sistema empezará a grabar la sesión actual.	(E)	GUI, Base de datos
15	La interfaz gráfica debe permitir al usuario enviar comandos de teleoperación al robot móvil con botones dedicados al movimiento: dos botones para movimientos lineales (adelante/atrás) y dos para girar (izquierda/derecha).	(E)	Movimiento del robot
16	El sistema permitirá la segmentación y visualización de al menos cinco objetos de interés (silla, mesa, planta, laptop/computador, sofá) usando la cámara RGB-D.	(E)	Segmentación Online
17	La interfaz debe permitir la visualización de estadísticas, como la tasa de detección.	(E)	Segmentación Online
18	La interfaz debe proporcionar una lista de objetos detectados junto con su etiqueta y ubicación en pantalla.	(E)	Segmentación Online
19	La interfaz debe permitir la visualización de la distancia relativa a los objetos de interés detectados.	(E)	Segmentación Online
20	En la interfaz existirá un botón para guardar y terminar la sesión; si se sale sin guardar, el sistema mostrará un diálogo para confirmar la acción.	(E)	Base de datos
	Replay-Sesiones Previas		
21	La ventana sesiones previas tendrá una barra de herramientas con cuatro botones: cargar sesiones, iniciar/reanudar reproducción, pausar y eliminar sesiones previas.	(E)	GUI
22	Al seleccionar "cargar sesiones previas", la interfaz mostrará una lista con nombre, fecha y hora de cada sesión disponible.	(E)	GUI, Base de datos
23	El usuario podrá elegir la sesión deseada mediante un botón dedicado; al hacerlo, se mostrará su nombre en la ventana de sesiones previas.	(E)	GUI
24	Tras la selección, el sistema habilitará los botones de reproducción y pausa.	(E)	GUI
25	Al optar por eliminar sesiones previas, la interfaz mostrará nuevamente la lista de sesiones disponibles.	(E)	GUI
26	La ventana dispondrá de un botón dedicado para eliminar la sesión seleccionada.	(E)	GUI

Continúa en la siguiente página

Replay-Sesiones Previas (continuación)			
27	Si se elige una sesión para eliminar, se mostrará un diálogo de confirmación antes de proceder.	(E)	GUI, Base de datos

Tabla 3.6: Requerimientos Funcionales para el Backend

	Backend		
1	El sistema debe ser capaz de adquirir datos de un sensor RGB-D y enviarlos al nodo maestro.	(E)	Hardware
2	El robot móvil debe ser capaz de moverse de acuerdo con los comandos de teleoperación enviados por el sistema.	(E)	Hardware
3	El sistema debe ser capaz de preprocesar los datos de entrada del sensor RGB-D, para ajustar brillo, contraste y filtrar ruido, esto es: ecualización del histograma, realce exponencial para cambiar el rango dinámico, y filtrado de ruido usando filtro gaussiano.	(E)	Captura de datos, sensores

Tabla 3.7: Especificación de Requerimientos No Funcionales

Universidad del Valle	**Universidad del Valle – DESARROLLO DE UN APLICATIVO EN ROS PARA LA SEGMENTACIÓN DE OBJETOS DE INTERÉS EN INTERIORES USANDO UN ROBOT MÓVIL –**	Rev.: 000
Título: Especificación de Requerimientos No Funcionales		**Documento:** ERF-001

REVISIÓN HISTÓRICA			
Rev.	**Descripción del Cambio**	**Autor**	**Fecha**
001	Construcción del documento	Juan Camilo Chavez Castro	19/10/2023
002	Correcciones	Juan Camilo Chavez Castro	23/10/2023
003	Correcciones	Juan Camilo Chavez Castro	1/11/2023
004	Correcciones	Juan Camilo Chavez Castro	15/11/2024
005	Revisión		

Ref #	**Descripción**	**Categoría**
1.0	Para el desarrollo de este proyecto se usará Ubuntu 20.04 tanto en el computador del robot móvil, como en el computador maestro.	E
2.0	Los 2 computadores contarán con la distribución de ROS Noetic.	E
3.0	Los 2 computadores contarán con Python 3.8.	E
4.0	Se utilizará Xampp 8.2.12 para Linux para la configuración y administración de la base de datos.	E
5.0	Se utilizará PyQt5 y Qt Designer para la interfaz de usuario.	E
6.0	El robot móvil será el Pioneer 3DX.	E
8.0	Se utilizará la Cámara estéreo ZED 2.	E
9.0	Se utilizará un enrutador Tp-Link para la red local.	E
10.0	Se utilizará el framework Pytorch para redes neuronales, v. 2.4.1.	E
11.0	Se utilizará el framework TensorFlow para redes neuronales, v. 2.13.1.	E
12.0	Se utilizará la herramienta CUDA, v. 12.4.	E
13.0	Un computador contará con una tarjeta gráfica Nvidia.	O

3.2.2. Diagrama Conceptual

El diagrama conceptual (Fig. 3.1) ilustra la arquitectura del sistema desarrollado para la segmentación de objetos de interés en interiores utilizando un robot móvil y un aplicativo basado en ROS. Este sistema se divide en dos componentes principales: el Sistema Robótico y el Sistema de la Máquina Maestra, que trabajan de manera conjunta para realizar la adquisición, procesamiento y presentación de datos.

- **Sistema Robótico:** Este subsistema incluye el robot móvil Pioneer 3DX, equipado con sensores como la cámara estéreo ZED 2 y odometría integrada, se gestionan mediante un mini PC con Ubuntu 20.04 y ROS Noetic. El robot es responsable de la adquisición de datos y la ejecución de acciones basadas en comandos de teleoperación enviados desde el sistema maestro.

- **Sistema de la Máquina Maestra:** Este subsistema se compone de un computador maestro también equipado con Ubuntu 20.04 y ROS Noetic, encargado de recibir los datos sensoriales enviados por el robot, procesarlos y almacenarlos en una base de datos. Además, incluye un aplicativo de segmentación que permite al usuario interactuar con el sistema, visualizar los resultados de segmentación y emitir comandos al robot móvil.

Figura 3.1: Aplicativo de Segmentación usando un robot móvil.

La comunicación entre ambos subsistemas se realiza mediante una red local (WLAN/LAN), lo que garantiza la transferencia de datos en entornos interiores. La integración en ROS y el diseño modular asegura el buen manejo de la información y los datos que se transmiten entre

sí. El usuario final puede interactuar con el sistema a través de una interfaz gráfica desarrollada con PyQt5 y Qt Designer. El propósito de este diagrama es proporcionar una visión general clara de la relación entre los diferentes componentes, sus roles dentro del sistema y cómo trabajan juntos.

3.2.3. Diagrama Relacional

El diagrama relacional de la base de datos (Figura 3.2) describe la estructura lógica de las tablas, los atributos y las relaciones, proporciona una visión organizada de los datos almacenados en 3 diferentes tablas:

- **Sesiones:** Registra la información general de cada sesión y del usuario que realiza la sesión, incluyendo datos como fecha, hora, ubicación y parámetros configurados por el usuario. Los parámetros de configuración corresponden al apartado de comunicación, configuración de robot, configuración del algoritmo de segmentación y configuración de la cámara.
- **int_datos:** Contiene estadísticas globales del algoritmo y datos temporales durante las sesiones.
- **Etiquetas:** Almacena las anotaciones y predicciones realizadas sobre los datos procesados, incluyendo coordenadas y distancias asociadas a los objetos detectados.

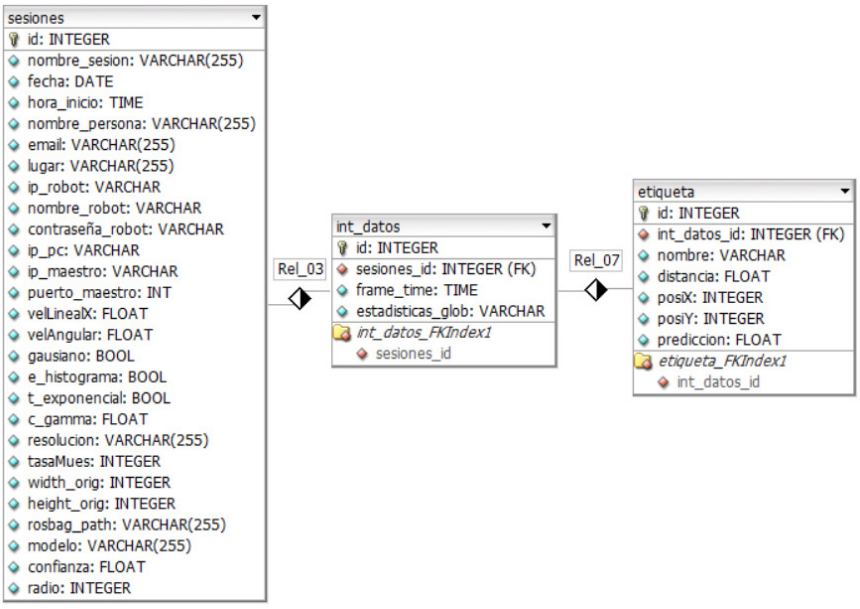


Figura 3.2: Diagrama relacional de la base de datos.

3.2.4. Diagrama de Nodos

El diagrama de nodos expone el flujo de los mensajes, en este caso tópicos y servicios, entre los diferentes nodos que componen el sistema. Este aplicativo cuenta con dos diagramas de nodos principales.

El diagrama del módulo de *Ejecución* (Fig. 3.3) muestra la interacción entre los nodos a implementar. En primer lugar, la cámara ZED2 y su nodo asociado capturan la información visual del entorno, seguido por el nodo `algoritmo_segmentacion`, encargado de procesar las imágenes y datos. La interfaz gráfica, gestionada por el nodo `gui_segmentacion`, permite al usuario visualizar los resultados del procesamiento, además de ofrecer el servicio *SessionConfig*, que transmite los parámetros de configuración al nodo de segmentación. Finalmente, el control y teleoperación del robot se realizan mediante el nodo `Rosaria`. Todos los tópicos visualizados por el usuario se registran en un archivo *rosbag*, que pertenece a un nodo propio encargado de su manejo.

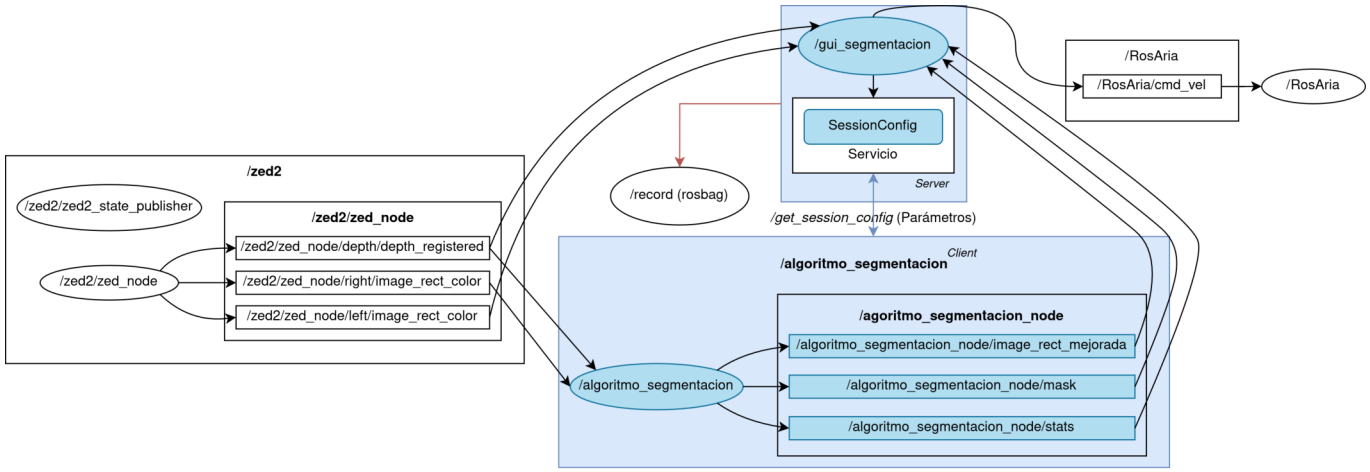


Figura 3.3: Diagrama de Nodos para el módulo Ejecución.

Por otro lado, el diagrama del módulo *Replay* (Fig. 3.4) representa exclusivamente el sistema de reproducción, compuesto por los nodos `rosbag_manager`, la interfaz gráfica gestionada por el nodo `gui_segmentacion` y `play`, este último correspondiente a la herramienta `rosbag` de ROS. En este esquema interviene el servicio *RosbagControl*, que recibe los comandos *play*, *pause* y *stop* desde el nodo `gui_segmentacion` y los transmite al nodo `rosbag_manager`, encargado de controlar la reproducción del nodo `play`.

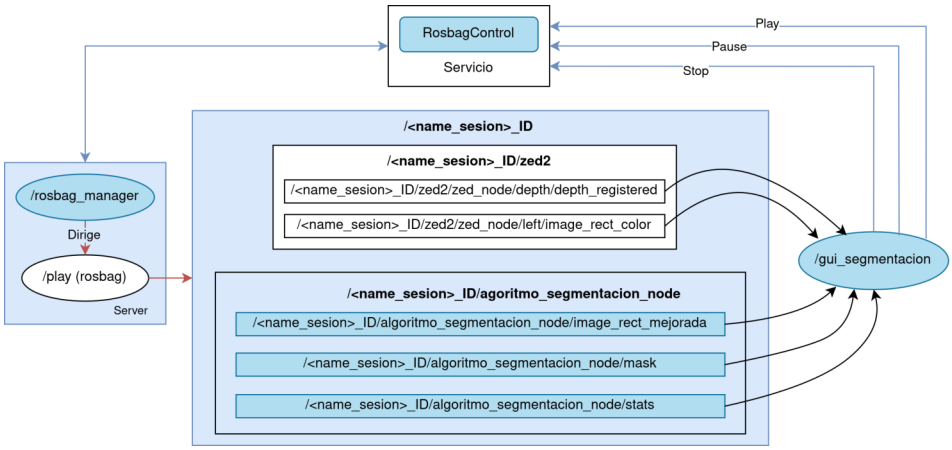


Figura 3.4: Diagrama de Nodos para el módulo Replay.

3.3. Preparación y filtrado de los Datasets utilizados

Para este trabajo, se adaptó MS COCO y ADE20K mediante preprocesamiento y filtrado: se selecciona las clases de interiores de interés, se limpia y convierte sus anotaciones, se ajusta resoluciones y se realiza un proceso de preprocesamiento, y finalmente se define los conjuntos de entrenamiento y validación. Estos pasos garantizan datos representativos y adecuados para los experimentos de segmentación.

La selección de clases empleadas para el entrenamiento difiere según el tipo de modelo y el conjunto de datos utilizado. Para la segmentación de instancias se consideraron seis categorías del conjunto COCO: *chair*, *couch*, *tv*, *laptop*, *potted plant* y *dining table*. En el caso de la segmentación semántica basada en ADE20K, se seleccionaron cinco clases: *chair*, *sofa*, *plant*, *table* y una clase agrupada que integra *laptop*, *monitor* y *computer*, con el fin de unificar dispositivos de apariencia y función similares. Esta elección se basó tanto en la disponibilidad de categorías en los conjuntos de datos como en su relevancia para escenarios interiores.

En particular, el sistema LU-SEG fue diseñado para operar en espacios universitarios, específicamente en el edificio de Ingeniería Eléctrica y Electrónica de la Universidad del Valle, caracterizado por oficinas, laboratorios y zonas comunes con mobiliario de oficina, mesas, sillas, dispositivos electrónicos y elementos decorativos como plantas. Estas clases representan adecuadamente los objetos estructurales y funcionales más frecuentes en dichos entornos, permitiendo evaluar el desempeño del sistema en condiciones reales.

3.3.1. MS COCO

El conjunto de datos MS COCO (2017)[35] fue seleccionado por su gran número de instancias y su variedad de escenas de interior y exterior. Para adaptarlo a las necesidades de este estudio,

segmentación de objetos de interiores, se siguieron dos flujos de trabajo paralelos, uno para YOLOv11 y otro para Mask R-CNN, tal como se detalla a continuación.

3.3.1.1. Filtrado de datos

Pipeline para YOLOv11 (Ultralycs): Primero, se descargaron los archivos de anotaciones "`instances_train2017.json`" e "`instances_val2017.json`" y las carpetas de imágenes `train2017` y `val2017`. A continuación, se utilizó la utilidad `convert_coco` de Ultralytics, activando el parámetro `cls91to80 True` para transformar las 91 clases originales en las 80 clases estándar de COCO en formato YOLO (`.txt`). Tras esta conversión, se filtraron las imágenes y sus correspondientes archivos `.txt` para conservar únicamente las seis categorías de interés: silla (`chair`), sofá (`couch`), maceta con planta (`potted plant`), mesa de comedor (`dining table`), monitor (`tv`), portátil (`laptop`). Todas las demás anotaciones se eliminaron y los identificadores remanentes se remapearon a un rango compacto $\{0,\ldots,5\}$ usando el mapa:

$$\{56 \to 0,\ 57 \to 1,\ 58 \to 2,\ 60 \to 3,\ 62 \to 4,\ 63 \to 5\}.$$

Pipeline para Mask R-CNN (Detectron2): En paralelo, con Detectron2 se descargaron las mismas anotaciones e imágenes, pero el filtrado se realizó directamente sobre el JSON original. Se generó un nuevo archivo `instances_filtered.json` que sólo contiene seis categorías de interés, remapeadas mediante:

$$\{62 \to 0,\ 63 \to 1,\ 64 \to 2,\ 67 \to 3,\ 72 \to 4,\ 73 \to 5\}.$$

Al registrar este JSON con `register_coco_instances`, Detectron2 automáticamente ignora las imágenes sin anotaciones relevantes, por lo que no fue necesario eliminar manualmente ningún archivo de imagen.

Tras aplicar el filtrado, se obtuvo 26,844 pares de imagen-etiqueta para el **conjunto de entrenamiento** y 1,173 para el **conjunto de validación**. Gracias a estos dos procesos de filtrado, se obtuvo particiones de entrenamiento y validación centradas exclusivamente en los objetos de interiores de interés, consiguiendo un total de **78,939** instancias para el entrenamiento y un total de **3,588** instancias para la validación, lo que deja listos los datos para los experimentos de segmentación con ambos frameworks. A continuación se presenta un resumen de las instancias por clase del dataset MS COCO filtrado (Tabla 3.9), seguido de las gráficas correspondientes de distribución (Fig 3.5) y el conteo de imágenes y etiquetas por partición.

Tabla 3.9: Instancias filtradas de MS COCO por clase y conjunto.

Clase	Train	Val
silla	38,073	1,771
sofá	5,779	261
macetas con plantas	8,631	342
mesa	15,694	695

Clase	Train	Val
tv	5,803	288
laptop	4,959	231
Total	78,939	3,588

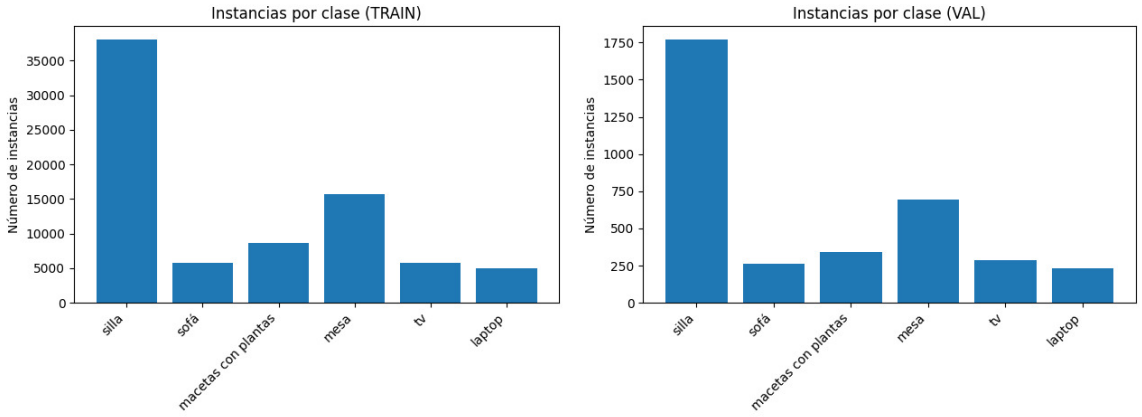


Figura 3.5: Distribución de instancias por clase para los conjuntos Train y Val en MS COCO.

3.3.1.2. Preprocesamiento

En YOLOv11 se emplea el pipeline completo de Ultralytics (Mosaic, randaugment, configuracón de color, brillo y saturación, etc.), redimensionando todas las entradas a 640×640 pixeles.

En Mask R-CNN (Detectron2), el preprocesamiento por defecto consiste en escalar aleatoriamente el lado más corto de la imagen a uno de los valores {640, 672, 704, 736, 768, 800} px y limitar el lado más largo a un máximo de 1,333 px, preservando la relación de aspecto original. Posteriormente, se aplica una rotación horizontal con una probabilidad de 0.5. No se incorporaron recortes aleatorios ni ajustes de color adicionales, por lo que el preprocesamiento en Mask R-CNN puede considerarse más moderado en comparación con el empleado en YOLOv11.

La naturaleza de la tarea de segmentación de instancias muchas veces implica el desbalance de clases dentro de los conjuntos de datos de entornos reales. Esto se debe a la inevitable presencia de objetos que aparecen con menos frecuencia o de gran tamaño, frente a otros mayor representados dentro de la escena u objetos pequeños que pueden aparecer con frecuencia en una escena. Por lo que es necesario mencionar que, aunque no se configuro manualmente "pesos de clase", ambos métodos aplican mecanismos de reequilibrio que actúan directamente sobre la función de pérdida y la estrategia de muestreo (Focal Loss en YOLOv11 y muestreo balanceado en Mask R-CNN) que evitan que la clase mayoritaria (como el fondo) domine el entrenamiento:

- **YOLOv11 (Ultralytics):** utiliza **Focal Loss** para las tareas de *clasificación* y *objectness*, lo cual atenúa el peso de las instancias muy frecuentes y refuerza el aprendizaje sobre aque-

llas más difíciles o poco representadas. Este mecanismo actúa directamente sobre la función de costo durante el entrenamiento, mejorando la sensibilidad del modelo frente a clases minoritarias y equilibrando su rendimiento global [74, 75].

- **Mask R-CNN (Detectron2):** aplica un muestreo balanceado de ejemplos positivos y negativos (aproximadamente 25 %–75 %) en la RPN y la RoI Head, lo que reduce la influencia del fondo durante la retropropagación [35, 76]. Sin embargo, este muestreo equilibra únicamente la proporción entre fondo y objeto, sin corregir el desbalance entre clases de objetos. Para ello, Detectron2 ofrece mecanismos adicionales, como el *Repeat Factor Training Sampler*, que incrementa la frecuencia de clases minoritarias, aunque dicho procedimiento no fue implementado en este trabajo.

3.3.2. ADE20K

El conjunto de datos ADE20K[47] destaca por tener más de 2,600 categorías semánticas que abarcan escenas interiores y exteriores. Para adaptarlo a los modelos ESANet–RGB y LR-ASPP MobileNetV3 se implementó un proceso de extracción y limpieza de anotaciones, filtrado de las clases de interés, remapeo de identificadores, ajuste de resolución y aplicación de técnicas de preprocesamiento. En las siguientes subsecciones se describen en detalle los pasos empleados.

3.3.2.1. Filtrado de datos

El proceso de filtrado, ilustrado en la Figura 3.6, consistió en reorganizar y depurar el conjunto ADE20K para adaptarlo a las clases de interés del estudio.

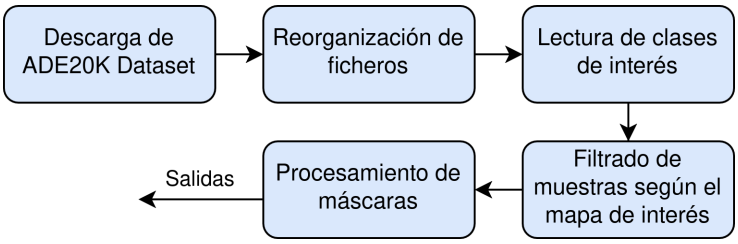


Figura 3.6: Diagrama de flujo para el filtrado del Dataset ADE20K

Primero, se descargó el dataset desde su sitio oficial[1], el cual incluye imágenes, máscaras, archivos de índices y metadatos. Luego, se reorganizaron las imágenes (`.jpg`) y sus correspondientes máscaras de segmentación (`_seg.png`) en una estructura limpia con carpetas separadas para imágenes y máscaras. A continuación, se procesó el archivo `objects.txt` para construir un diccionario de clases válidas, filtrando solo aquellas muestras que contenían al menos una de las categorías de interés. Finalmente, las máscaras fueron procesadas y los píxeles que no pertenecían a las clases de interés fueron reasignados al fondo (valor 0), se remapea las clases al rango [0...N] (donde 0 es background y N el número de clases) y se genera versiones filtradas y unificadas, almacenadas en carpetas separadas.

[1] `https://ade20k.csail.mit.edu/index.html#Download`

A diferencia del proceso aplicado al dataset MS COCO, en el caso de ADE20K se trabajó con un subconjunto específico de **cinco clases**, en el cual las categorías de monitor, computador y laptop fueron unificadas en una sola clase. Como resultado del filtrado, se obtuvieron 10,885 imágenes y sus respectivas máscaras para el **conjunto de entrenamiento**, y 935 para el **conjunto de validación**.

La Tabla 3.10 detalla el mapeo entre las clases originales de ADE20K y el esquema de categorías unificadas utilizado en este trabajo, mientras que la Figura 3.7 muestra la distribución de imágenes que contienen cada una de estas cinco categorías seleccionadas

Clase	**ADE20K**			**Unificada**		
	ID	**#Img. Train**	**#Img. Val**	**ID**	**#Img. Train**	**#Img. Val**
Table	799	4	0	1	6109	488
	3090	5	0			
	2448	7	0			
	406	7	0			
	724	745	61			
	594	122	8			
	2684	5116	414			
	2692	29	7			
	1948	208	18			
Chair	1561	3	0	2	4178	344
	471	3939	320			
	472	2	1			
	1164	6	1			
	3104	56	2			
	2679	327	33			
Laptop/Computer/Monitor	1407	187	18	3	1366	104
	1583	234	22			
	591	318	29			
	2733	783	53			
Sofa	2473	1572	106	4	1572	106
Plants	1910	5120	445	5	6189	543
	1981	1525	141			
	978	1575	138			
	1911	27	0			
	1908	16	2			
	1907	1	0			

Tabla 3.10: Agrupación de subcategorías de ADE20K en clases unificadas.

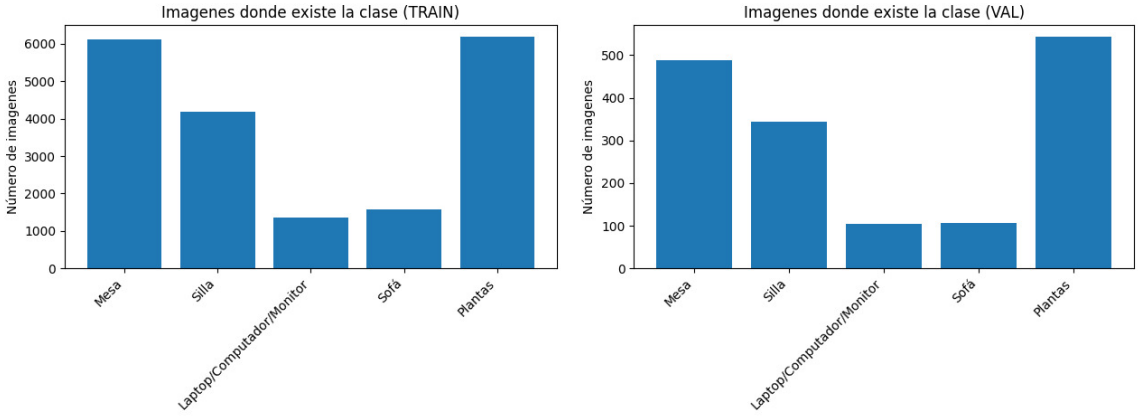


Figura 3.7: Distribución de imágenes en ADE20K filtrado y unificado.

3.3.2.2. Preprocesamiento

El preprocesamiento en los datos sigue el flujo de la Figura 3.8.

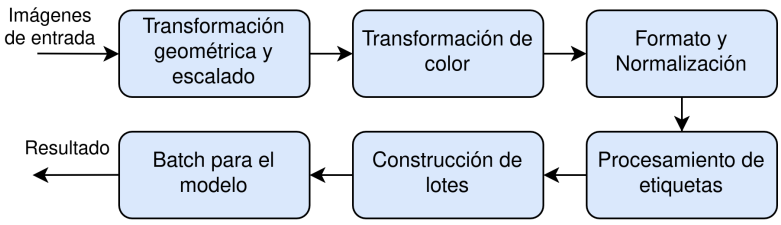


Figura 3.8: Diagrama de flujo del preprocesamiento.

A continuación, se describen los pasos:

1. Transformación geométrica y escalado: Cada par imagen/máscara fue sometido a un reescalado aleatorio con un factor entre 1.0 y 1.4, con el objetivo de simular variaciones en la distancia. Posteriormente, las imágenes se redimensionaron a una resolución fija y se aplicaron recortes aleatorios, lo que contribuye a mejorar la robustez del modelo frente a distintas composiciones espaciales y posiciones de los objetos en escena.

2. Transformación de color y espacial: Se aplicaron variaciones aleatorias en matiz, saturación y brillo (espacio HSV) para simular distintas condiciones de iluminación, junto con giros horizontales probabilísticos para aumentar la diversidad espacial.

3. Formato y normalización: Las imágenes se transformaron a tensores con el reordenamiento de canales (HWC $\rightarrow$ CHW) y se normalizaron utilizando las medias y desviaciones estándar de ImageNet [49], para estabilizar el entrenamiento.

4. Procesamiento de etiquetas: Se generaron versiones de las máscaras a múltiples resoluciones (1/8, 1/16 y 1/32) para aplicar pérdidas multiescala durante el entrenamiento. La clase "void" (fondo) se añadió explícitamente como una categoría adicional. Además, se calcularon **pesos de clase ponderados** con el fin de compensar el desequilibrio entre clases, dado que en la mayoría de las imágenes el fondo representa una proporción sustancial de los píxeles. Estos pesos se obtuvieron mediante el esquema de *frecuencia mediana* (*median frequency balancing*), donde cada clase recibe un factor inversamente proporcional a su frecuencia de aparición en las imágenes. Así, las clases menos representadas adquieren un peso mayor en la función de costo, evitando que el modelo se sesgue hacia las categorías más frecuentes.

5. Construcción de lotes: Las muestras se agruparon en lotes (batch) homogéneos y de tamaño consistente, aplicando un procedimiento *collate* encargado de verificar dimensiones y organizar conjuntamente las imágenes, sus máscaras y las versiones reducidas correspondientes.

6. Salida al modelo: Cada batch sale listo para alimentar la red, con sus datos de entrada, sus etiquetas (y sus versiones multi-escala), sus máscara con sus dimensiones originales para evaluación, sus imágenes originales para visualización.

Es importante señalar que para ESANet–RGB se empleó una resolución de 480 × 640 px y para LR-ASPP MobileNetV3 se utilizó 520 × 520 px, con el fin de mantener la compatibilidad con sus respectivos pesos preentrenados. Para abordar el **balance de clases**, se aplicaron los pesos ponderados calculados mediante la frecuencia mediana a nivel de píxel. Estos valores se incorporaron directamente en la función de pérdida, afectando el cálculo del error en cada iteración de entrenamiento. De este modo, los errores asociados a clases minoritarias, como objetos pequeños o poco frecuentes, contribuyen proporcionalmente más al gradiente, mejorando la capacidad del modelo para reconocerlas sin degradar el rendimiento general.

3.4. Implementación de las arquitecturas de red neuronal

En esta sección se presentan los detalles de implementación y los resultados de entrenamiento de las redes de segmentación consideradas. En el apartado 3.4.1 se describen los protocolos de entrenamiento, los hiperparámetros y las métricas obtenidas para YOLOv11-m-seg y Mask R-CNN. A continuación, en el apartado 3.4.2, se exponen los ajustes y resultados de ESANet–RGB y LR-ASPP MobileNetV3. Finalmente, en el apartado 3.4.3 se comparan cuantitativamente todas las arquitecturas y se selecciona el modelo más adecuado para su integración en el aplicativo LU-SEG.

3.4.1. Entrenamiento y resultados de YOLOv11 y MaskR-CNN

En este apartado se detallan los protocolos de entrenamiento y los resultados obtenidos con dos arquitecturas de segmentación de instancias: YOLOv11-m-seg y Mask R-CNN (ResNeXt-101-FPN). Ambos modelos arrancaron de pesos pre-entrenados proporcionados por sus res-

pectivos frameworks (Ultralytics y Detectron2) y se entrenaron en Google Colab haciendo uso de una GPU NVIDIA L4. A continuación se presentan los hiperparámetros clave, los tiempos de cómputo y las métricas de desempeño sobre los conjuntos filtrados de MS COCO.

3.4.1.1. Entrenamiento y resultados de YOLOv11m-seg

Para YOLOv11 se empleó el script estándar de Ultralytics partiendo de pesos preentrenados. Se entrenó durante 50 épocas con early stopping (paciencia 8), lote de 16 imágenes de 640×640, optimizador "auto" (SGD/Adam), y congelando 9 capas del backbone (ver Tabla 3.11). Durante el entrenamiento se pudo observar que el gasto promedio en GPU nunca alcanzo más de 11 GB de ram, por lo que para un posterior entrenamiento se podría optar por una GPU de menor capacidad como lo es NVIDIA T4 que ofrece hasta 15 GB de ram en GPU de Google Colab o descongelar algunas capas más.

Parámetro	**Valor**
Épocas	50 (35 completadas)
Número de clases	6
Batch size	16
Image size	640×640
Optimizer	auto (SGD/Adam)
LR inicial (lr0)	0.01
LR final (lrf)	0.01×lr0
Patience	8
Freeze layers	9
GPU	NVIDIA L4 con 22.5 GB de ram
Parámetros	22'363,842
Tiempo de entrenamiento	≈ 5,94 h

Tabla 3.11: Configuración de entrenamiento para YOLOv11m-seg.

A continuación se presentan los **resultados cuantitativos** en el conjunto de validación, distinguiendo la detección de cajas y la segmentación de máscaras por clase, así como el promedio global (ver Tabla 3.12), también se presenta la distribución de las métricas de segmentación en la Figura 3.9. Donde el modelo alcanzó un $\text{mAP}_{50}(\textit{mask})$ de 63.2% y un $\text{mAP}_{50:95}(\textit{mask})$ de 41.4% en el conjunto de validación.

Tabla 3.12: Desempeño por clase y promedio general: métricas de detección (Box) y segmentación (Mask) para YOLOv11.

Clase	**Tarea**	**Precisión**	**Recall**	mAP_{50}	$\text{mAP}_{50:95}$
chair	Box	0.691	0.543	0.613	0.412
	Mask	0.657	0.513	0.547	0.267
couch	Box	0.674	0.670	0.707	0.564
	Mask	0.655	0.648	0.640	0.434

Continúa en la siguiente página

Continúa la tabla 3.12

Clase	Tarea	Precisión	Recall	mAP_{50}	$mAP_{50:95}$
potted plant	Box	0.658	0.563	0.594	0.372
	Mask	0.654	0.553	0.580	0.299
dining table	Box	0.651	0.529	0.586	0.433
	Mask	0.546	0.442	0.391	0.220
tv	Box	0.841	0.778	0.852	0.667
	Mask	0.834	0.771	0.847	0.645
laptop	Box	0.822	0.797	0.847	0.723
	Mask	0.788	0.762	0.788	0.620
Promedio global	Box	0.723	0.646	0.700	0.529
	Mask	0.689	0.615	0.632	0.414

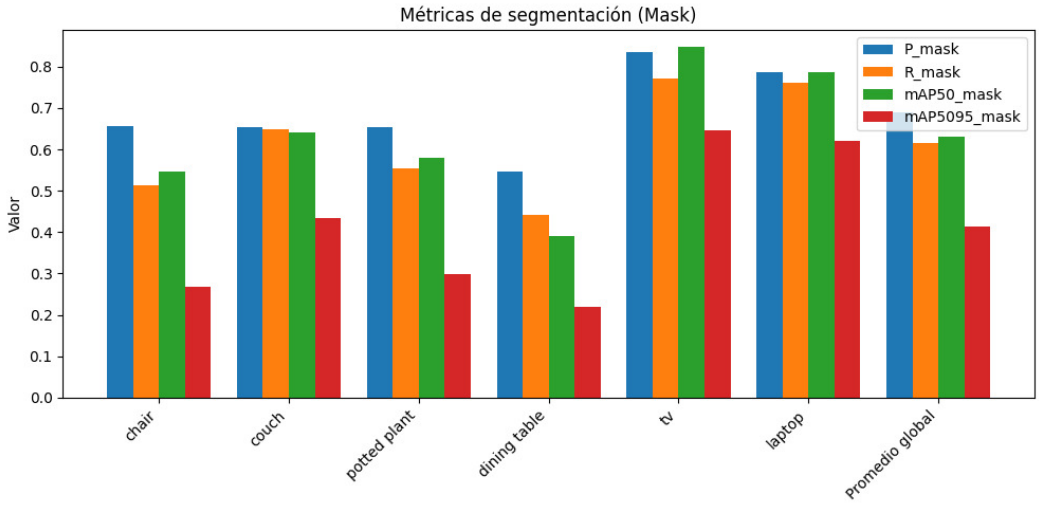


Figura 3.9: Resultados de métricas para segmentación con YOLOv11

Se observa que las clases con formas bien definidas, como *TV* y *laptop*, alcanzan los valores de desempeño más altos, mientras que los objetos de contornos más complejos o con mayor variabilidad de apariencia, como *dining table* y *potted plant*, presentan resultados inferiores, especialmente en la rama de segmentación por máscara. En particular, la categoría *dining table* registra el valor más bajo de $mAP_{50:95}$, constituyéndose como la de menor rendimiento. El promedio global confirma que, en general, la detección obtiene un desempeño ligeramente superior al de la segmentación en términos de mAP.

Adicionalmente, se evaluó el tiempo de inferencia del modelo sobre 500 imágenes válidas, obteniendo un promedio por imagen de $\overline{x} = 0{,}0351$ s y una desviación estándar de $\sigma = 0{,}0065$ s. Este resultado es ideal para aplicaciones de procesamiento continuo y en linea, siempre que se disponga de una GPU de gama media o superior.

En la Figura 3.10.a se muestran las curvas de precisión-recall: las de *TV* y *laptop* son muy verticales, indicando alta precisión incluso en un recall elevado, mientras que *dining table* decae

antes, indicando menor estabilidad. La curva global se acerca a la esquina superior derecha, lo que revela un buen desempeño en segmentación de instancias. En la Figura 3.10.b se ilustran predicciones sobre el conjunto de validación: en general las máscaras ajustan con precisión y solo aparecen leves falsos positivos (por ejemplo, una planta en maceta con baja confianza).

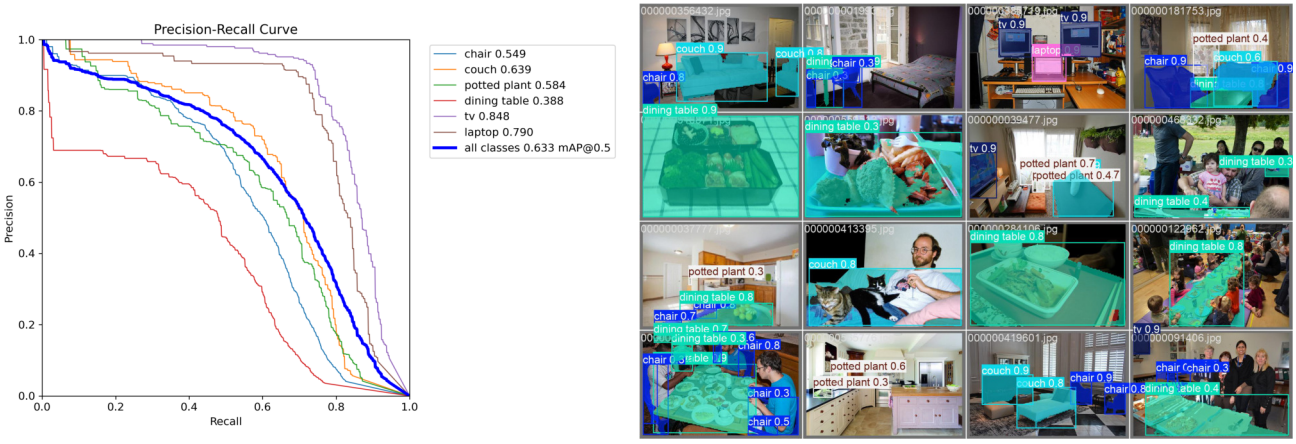


Figura 3.10: a) Curvas de Precisión-Recall para segmentación de instancias, b) Predicciones en el conjunto de validación usando YOLOv11.

3.4.1.2. Entrenamiento y resultados de Mask R-CNN (ResNeXt-101-FPN)

Mask R-CNN se configuró en Detectron2 con los pesos prentrenados para COCO con la plantilla: `COCO-InstanceSegmentationmask_rcnn_X_101_32x8d_FPN_3x.yaml`. Se entrenó con batch size de 8 imágenes de resolución variable (lado corto en $[640, 800]$ píxeles y lado largo ≤ 1333), tasa de aprendizaje LR igual a 0.001, 33,560 iteraciones, y ROI heads con `batch_size_per_image` igual a 128 (ver Tabla 3.13). Durante el entrenamiento se pudo observar que el gasto promedio en GPU nunca alcanzo más de 13 GB de ram, también se podría optar por una GPU de menor capacidad como NVIDIA T4.

Parámetro	**Valor**
Configuración base	X_101_32x8d_FPN_3×
Número de clases	6
Batch size	8
LR	0.001
Iteraciones	33 560
ROI batch/image	128
Etapas del backbone congeladas	4
GPU	NVIDIA L4 con 22.5 GB de ram
Parámetros	107’427,760
Tiempo de entrenamiento	≈ 12 h

Tabla 3.13: Configuración de entrenamiento para Mask R-CNN.

A continuación, se presentan los **resultados cuantitativos** en el conjunto de validación, distinguiendo la detección de cajas (Box) y la segmentación de máscaras (Mask) por clase, así como el promedio global (ver Tabla 3.14). Donde los resultado globales fueron un AP_{50} de 59.845 % y un AP de 38.197 %.

Clase	**Detección (Box)**		**Segmentación (Mask)**	
	AP	**AP_{50}**	**AP**	**AP_{50}**
chair	31.582	—	21.871	—
couch	44.642	—	34.879	—
potted plant	32.317	—	26.862	—
dining table	38.046	—	20.257	—
tv	57.721	—	61.279	—
laptop	64.933	—	64.032	—
Promedio global	44.874	66.115	38.197	59.845

Tabla 3.14: AP por categoría y promedio global para Mask R-CNN

En cuanto al desempeño de inferencia, Mask R-CNN presentó un tiempo promedio por imagen de $\overline{x} = 0{,}2830$ s, con una desviación estándar de $\sigma = 0{,}0370$ s (500 muestras). Este valor es significativamente mayor que el observado en YOLOv11, reflejando el tamaño y la complejidad de la arquitectura y su consecuente demanda de recursos computacionales, lo que limita su uso en este tipo de aplicaciones de procesamiento continuo.

En la Figura 3.11 se puede observar la distribución de resultados por clase y global, al igual que la implementación de YOLO, los mejores resultados fueron para las clases ”laptop” y ”tv(monitor)”, y la clase con más dificultad para segmentar y detectar fue ”dining table”. Por otra parte, en la imagen (Fig. 3.12) se muestra una serie de predicciones, donde se puede detectar algunos falsos positivos de la clase ”chair”, pero en general los resultados se ven en su mayoría correctamente.

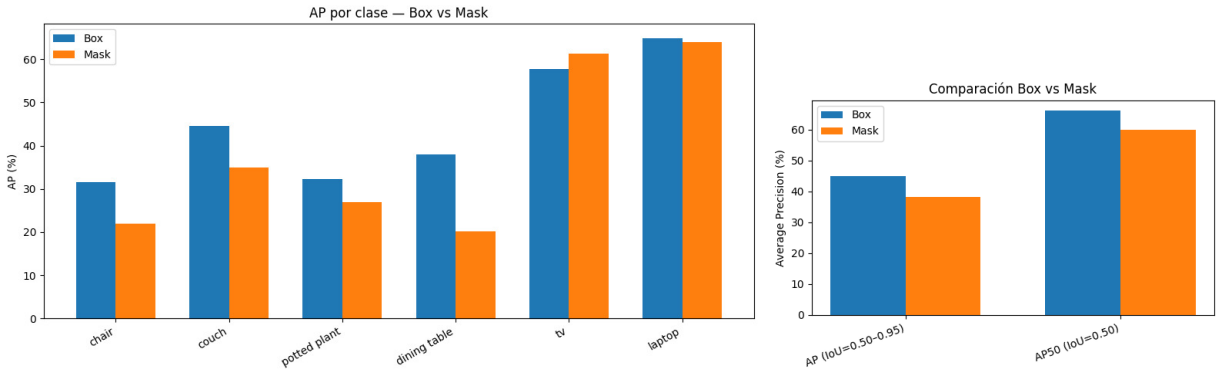


Figura 3.11: Métricas de evaluación para Mask R-CNN.

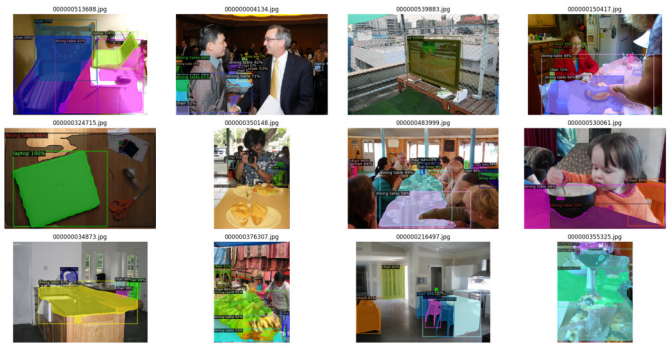


Figura 3.12: Predicciones en el conjunto de evaluación para Mask R-CNN.

3.4.1.3. Comparación de las arquitecturas de segmentación de instancias

En la Figura 3.13 se presenta una comparación entre las métricas más relevantes de segmentación de instancias obtenidas con YOLOv11 y Mask R-CNN, considerando $\text{mAP}_{50:95}$ y AP_{mask}, respectivamente. En términos generales, YOLOv11 mostró un desempeño ligeramente superior en la mayoría de las clases, así como en el promedio global ($\text{mAP}_{50:95} = 41{,}4\,\%$) frente a Mask R-CNN ($AP_{\text{mask}} = 38{,}20\,\%$). Con excepción en la categoría *laptop*, donde Mask R-CNN alcanzó un mejor resultado ($AP_{\text{mask}} = 64{,}03\,\%$) en comparación con YOLOv11 ($\text{mAP}_{50:95} = 62{,}0\,\%$). La diferencia más notable se presenta en la clase *couch*, donde YOLOv11 obtuvo una ventaja considerable ($\text{mAP}_{50:95} = 43{,}4\,\%$) frente al valor de $AP = 34{,}89\,\%$ registrado por Mask R-CNN.

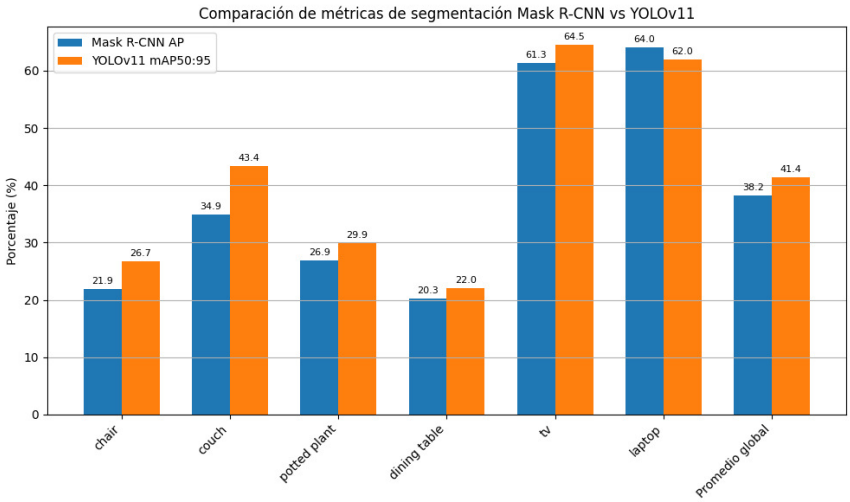


Figura 3.13: Comparación de métricas de segmentación de instancias entre YOLOv11 y Mask R-CNN.

Considerando no solo los resultados, sino también factores como el tiempo de entrenamiento, la cantidad de parámetros y el tipo de aplicación objetivo, YOLO se posiciona como la mejor opción para esta tarea. Su entrenamiento requirió aproximadamente la mitad del tiempo que

Mask R-CNN (6 vs. 12 horas), además de contar con un modelo mucho más compacto (22 millones frente a 107 millones de parámetros). Por su diseño, YOLO está orientado al procesamiento en línea y de baja latencia, alineándose mejor con los objetivos de este trabajo.

3.4.2. Entrenamiento y resultados de ESANet RGB y L-RASPP MobileNetV3

En este apartado se detallan el proceso de entrenamiento y los resultados obtenidos con los modelos ESANet RGB y L-RASPP MobileNetV3. Para el modelo ESANet, se utilizaron los pesos pre-entrenados con ImagenNet en el backbone ResNet34, mientras que para el modelo L-RASPP MobileNetV3, se utilizaron los pesos pre-entrenados de un subconjunto de 20 clases del dataset COCO, equivalente a PASCAL VOC. Ambos modelos fueron entrenados en Google Colab utilizando una GPU NVIDIA T4. Las métricas de desempeño evaluadas fueron el mIoU, la precisión y recall. A continuación, se presentan un diagrama de flujo que ilustra el proceso de entrenamiento (Figura 3.14),seguido de detalles adicionales.

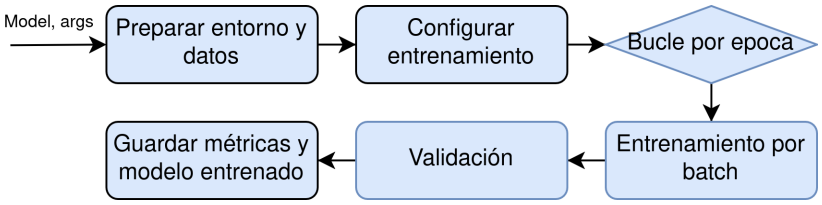


Figura 3.14: Etapas de la función de entrenamiento.

El proceso de entrenamiento inicia con la preparación del entorno, que incluye la creación de directorios para almacenar *checkpoints*, la configuración de los *dataloaders* de entrenamiento y validación, y la inicialización del *logger* para el registro de métricas. Se calculan los pesos de clase a partir de la frecuencia de las categorías del conjunto de datos, con el propósito de equilibrar la función de pérdida y mitigar el efecto del desbalance entre clases. Finalmente, se definen el optimizador y el *scheduler*, responsables de ajustar la tasa de aprendizaje a lo largo del proceso de entrenamiento.

Cada época de entrenamiento se compone de iteraciones por lotes, en las que el modelo realiza un *forward pass* para generar las máscaras de segmentación y calcula una pérdida multiescala ponderada con los pesos de clase previamente determinados. Esta función de costo combina las pérdidas por escala normalizadas por el número de píxeles ponderados por clase, asegurando que las categorías menos representadas mantengan una influencia significativa en la optimización. Posteriormente, se ejecuta la retropropagación con recorte de gradientes para garantizar estabilidad numérica y se actualiza el *learning rate scheduler* tras cada lote. Al finalizar cada época, el modelo se valida utilizando las etiquetas a resolución completa, ignorando los píxeles marcados como *void*, y se actualiza la matriz de confusión para calcular las métricas globales y por clase. Finalmente, se registran los valores de pérdida, métricas y se guardan los *checkpoints* correspondientes.

3.4.2.1. Entrenamiento y resultados de ESANet-RGB

El modelo fue entrenado durante 50 épocas, con un batch size de 16 y optimizador SGD, las imágenes se redimensionaron a 480×640 píxeles. Se congelaron las primeras 5 épocas excepto la capa de salida, con descongelamiento progresivo hasta el final de la tercera capa del encoder. El mejor modelo se obtuvo en la época 49 y el tiempo total de entrenamiento fue aproximadamente 10 horas (ver Tabla 3.15).

Tabla 3.15: Configuración para el entrenamiento de ESANet

Parámetro	**Valor**
Épocas	50
Tamaño de Imagen	480×640 píxeles
Batch size	16
Número de clases	5 (sin void)
Optimización	SGD
Lr inicial	0.08
Freeze epochs	5
Freeze hasta capa	encoder.layer3.5.bn2.bias
Total de parámetros	32'059,408
Parámetros entrenados	26'460,624
GPU	NVIDIA T4 con 15 GB de ram
Tiempo total	$\approx$ 10 h

El protocolo incluyó congelar inicialmente el backbone para estabilizar el aprendizaje y evitar sobreajuste temprano, seguido de un descongelamiento progresivo para afinar toda la red. Se utilizó un esquema de pérdida multiescala para optimizar el modelo en distintas resoluciones, y la validación consideró las etiquetas en su resolución original para calcular métricas.

A continuación, se presentan los **resultados cuantitativos** obtenidos en el conjunto de validación para segmentación semántica, distinguiendo las métricas por clase así como el promedio global (ver Tabla 3.16). Los resultados globales muestran un mIoU de 85.222 %, una precisión píxel de 91.661 % y una sensibilidad píxel de 92.139 %.

Clase	**IoU**	**Precisión**	**Recall**
TABLE	0.843252	0.903200	0.927032
CHAIR	0.800289	0.912051	0.867214
LAPTOP/COMPUTER/MONITOR	0.780879	0.852636	0.902711
SOFA	0.854096	0.926694	0.915983
PLANTS	0.982600	0.988461	0.994002
Promedio	**0.852223**	**0.916609**	**0.921388**

Tabla 3.16: Métricas por clase y promedio general para ESANet-RGB.

Se evaluó el rendimiento temporal de ESANet-RGB sobre 500 imágenes del conjunto de validación, obteniendo un tiempo promedio de inferencia de $\overline{x} = 0{,}0464$ s y una desviación estándar

de σ = 0,0178 s por imagen. Estos valores evidencian un comportamiento consistente y una capacidad de procesamiento rápida y adecuada para aplicaciones que requieren la generación continua de máscaras de segmentación semántica.

La Figura 3.15.a presenta la distribución de los resultados del entrenamiento, destacando que "Plantas" fue la clase con mejor rendimiento y "Laptop/Computador/Monitor" la más desafiante. Aunque la precisión y sensibilidad son buenas en esta última y comparables a las otras clases, su menor IoU sugiere dificultades para una superposición precisa.

La Figura 3.15.b presenta las curvas Precisión-Recall para cada clase y el promedio general. Se observa que las clases "Plantas", "Sofá", "Mesa" y "Silla" mantienen una alta precisión incluso a valores elevados de recall, lo que indica un buen desempeño en la identificación de la mayoría de los píxeles. En contraste, la clase "Laptop/Computador/Monitor" muestra una caída en su curva antes de alcanzar un recall de 0.8, sugiriendo que el modelo tiene mayor dificultad para mantener precisión al detectar todos los píxeles de esta categoría. Sin embargo, el resultado sigue siendo bueno, ya que la precisión se mantiene estable casi hasta un recall de 0.7. La curva promedio (línea punteada) refleja un rendimiento robusto general.

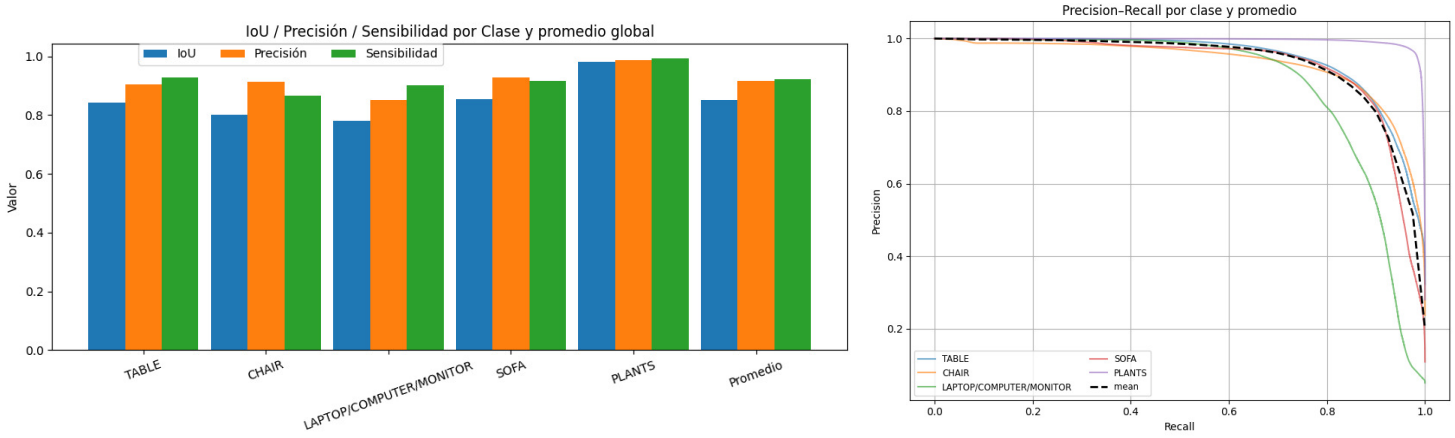


Figura 3.15: a) Resultados de métricas para segmentación, b) Curvas de Precisión-Recall a nivel de píxel para ESANet-RGB

La Figura 3.16 muestra ejemplos visuales del desempeño del modelo de segmentación en diferentes escenas del conjunto de validación. En cada fila se presenta una imagen original, su máscara Ground True (GT) y la predicción generada por el modelo. Se puede observar que, en general, el modelo logra segmentar correctamente objetos principales como sillas, mesas y otros muebles, capturando sus posiciones con buena precisión. Sin embargo, hay áreas donde la predicción no coincide exactamente con el GT, con ciertas imprecisiones especialmente visible en objetos complejos, muy cercanos entre sí o posiciones difíciles.

Figura 3.16: Muestras de predicciones usando ESANet-RGB

3.4.2.2. Entrenamiento y resultados de L-RASPP MobileNetV3

Se entrenó durante 100 épocas, con batch size de 32, optimizador SGD y tasa de aprendizaje inicial de 0.01. Las imágenes fueron redimensionadas a 520×520 píxeles durante el entrenamiento. No se aplicó congelamiento de capas. El entrenamiento se realizó en Google Colab con GPU NVIDIA T4, obteniendo el mejor modelo en la época 91 (ver Tabla 3.17).

Parámetro	**Valor**
Épocas	100
Tamaño de Imagen	520×520 píxeles
Batch size	32
Número de clases	5 (sin void)
Optimizador	SGD
LR inicial	0.01
Freeze epochs	0
Freeze hasta capa	Ninguna
Total parámetros	3,218,988
Parámetros entrenados	3,218,988
GPU	NVIDIA T4 con 15 GB de ram
Tiempo total	≈ 9 h

Tabla 3.17: Configuración para el entrenamiento de L-RASPP MobileNetV3.

Se exponen los **resultados obtenidos** en el conjunto de validación (ver Tabla 3.18). Los resultados globales muestran un mIoU de 79.871 %, una precisión píxel de 88.359 % y una sensibilidad píxel de 88.841 %.

El análisis temporal sobre 500 imágenes del conjunto de validación mostró un tiempo promedio de inferencia de $\overline{x} = 0{,}0301$ s y una desviación estándar de $\sigma = 0{,}0187$ s por imagen. Este

Clase	IoU	Precisión	Recall
TABLE	0.763111	0.856439	0.875044
CHAIR	0.727499	0.869219	0.816917
LAPTOP/COMPUTER/MONITOR	0.714390	0.801535	0.867913
SOFA	0.831782	0.924042	0.892828
PLANTS	0.956768	0.966721	0.989354
Promedio	**0.798710**	**0.883591**	**0.888411**

Tabla 3.18: Métricas por clase y promedio general para MobileNetV3.

resultado confirma que MobileNetV3 posee el procesamiento más rápido entre los modelos evaluados, mostrando su alta eficiencia computacional y requerimientos de ejecución continua.

La Figura 3.17.a presenta gráficamente los resultados obtenidos, se observa que, al igual que en ESANet, la clase “Planta” presenta el mejor desempeño, mientras que “Laptop / Computador / Monitor” sigue siendo la más desafiante, con el IoU más bajo, sin embargo, sus resultados en precisión y sensibilidad son sobresalientes estando por encima del 80 %. De forma análoga, la gráfica 3.17.b muestra la buena precisión en la clase ”Plantas”, y la caida temprana en la curva de la clase ”Laptop / Computador / Monitor”. Aunque esta última muestra un desempeño algo inferior respecto al resto, sus resultados no son malos. Por otro lado, la curva promedio se mantiene estable hasta valores altos de recall, lo que refleja un buen comportamiento en la tarea de segmentación semántica.

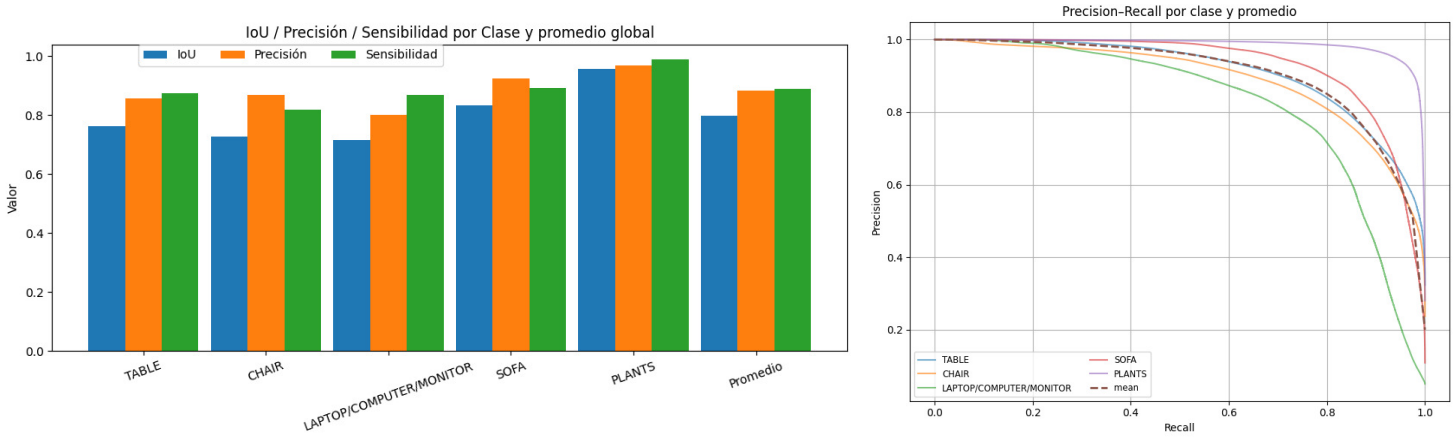


Figura 3.17: a) Resultados de métricas para segmentación, b) Curvas de Precisión-Recall a nivel de píxel para MobileNetV3

La Figura 3.18 presenta muestras de predicciones hechas con este modelo, donde se observan algunos falsos negativos y falsos positivos, pero en general se presentan máscaras consistentes con el ground true.

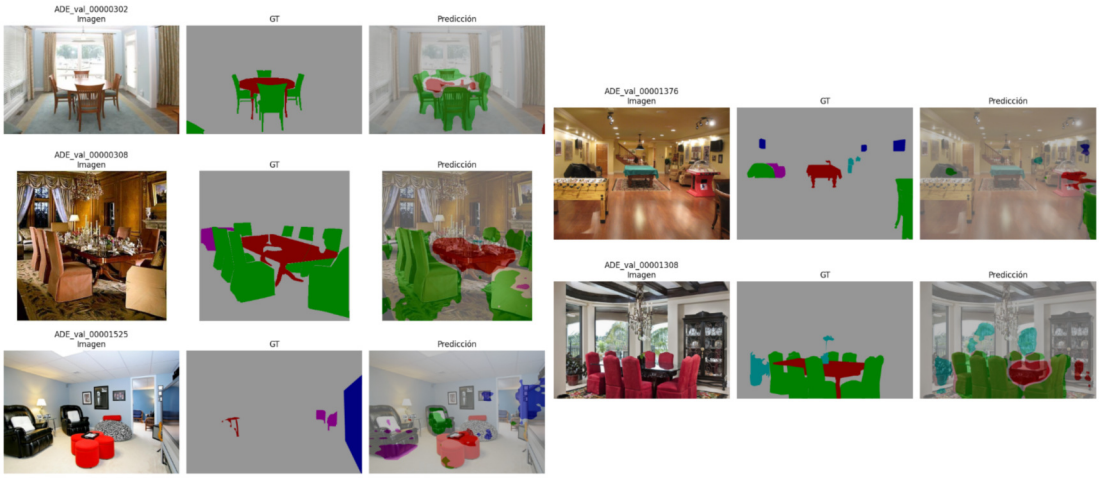


Figura 3.18: Muestras de predicciones usando L-RASPP MobileNetV3

3.4.2.3. Comparación de las arquitecturas de segmentación semántica

La Figura 3.19 muestra el IoU por clase y el promedio global obtenido por ESANet–RGB y L-RASPP MobileNetV3 en el conjunto de validación. ESANet–RGB alcanza un mIoU de 85,2 %, frente al 79,9% de MobileNetV3, lo que supone una mejora de 5,3 puntos porcentuales en capacidad de superposición. Esta diferencia, aunque moderada, puede ser relevante cuando la precisión de segmentación es crítica.

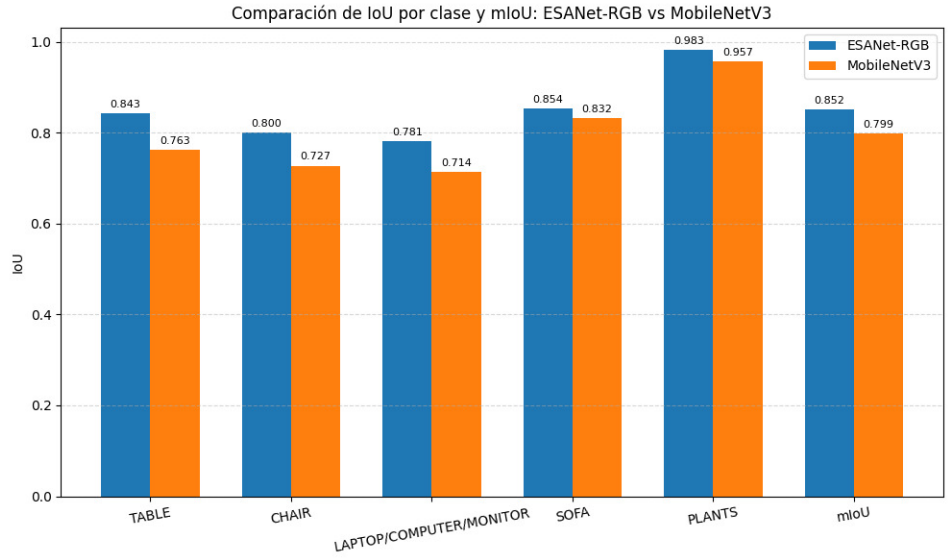


Figura 3.19: Comparación de la métrica IoU para modelos de segmentación semántica entre ESANet-RGB y L-RASSP MobileNetV3.

No obstante, MobileNetV3 completa su entrenamiento en una razón menor de tiempo/epoca (aprox. 9 h/100 epoch vs. 10 h/50 epoch), y utiliza solo 3,2 M de parámetros, un 90% me-

nos que los 32 M de ESANet–RGB, manteniendo un rendimiento muy cercano. Por ello, para aplicaciones en línea y entornos con recursos limitados, MobileNetV3 representa una opción más equilibrada; mientras que si se requiere la máxima calidad de máscara y se dispone de capacidad de cómputo, ESANet–RGB ofrece la mejor precisión.

3.4.3. Selección del modelo

Se evaluaron dos líneas de trabajo: segmentación de instancias (YOLOv11-m-seg y Mask R-CNN) y segmentación semántica (ESANet–RGB y L-RASPP MobileNetV3). En cada grupo, YOLOv11 y ESANet-RGB obtuvieron las métricas más altas, mientras que MobileNetV3 quedó muy cerca del rendimiento de ESANet–RGB pero con un modelo mucho más ligero a comparación de los otros 3 modelos. La Tabla 3.19 presenta un resumen de los resultados globales obtenidos para cada modelo, incluyendo las métricas de desempeño, el número de parámetros y los tiempos promedio de inferencia.

Criterio/Modelo	**YOLOv11m-seg**	**Mask RCNN**	**ESANet RGB**	**L-RASPP MobileNetV3**
Métrica Representativa Global	mAP[50:95] (Mask): 41.4%	AP (Mask): 38.19%	mIoU: 85.22%	mIoU: 79.87%
# de Parámetros	22'363,842	107'427,760	32'059,408	3'218,988
Tiempo de Inferencia (500 imágenes) $\overline{x} \pm \sigma$	0.0351 ± 0.0065 s	0.2830 ± 0.0370 s	0.0464 ± 0.0178 s	0.0301 ± 0.0187 s
Tarea	Seg. Inst.	Seg Inst.	Seg Sem.	Seg Sem.

Tabla 3.19: Resumen de los resultados de los modelos entrenados.

Por tanto, considerando el tamaño del modelo, la simplicidad de integración en el sistema LU-SEG, el tiempo de inferencia y las métricas globales obtenidas, se opta finalmente por L-RASPP MobileNetV3. Esta elección asegura una buena calidad de segmentación acompañada de un despliegue sostenible en recursos computacionales.

3.5. Implementación del Aplicativo LU-SEG

El aplicativo LU-SEG está diseñado para gestionar de forma integrada todo el flujo de trabajo de segmentación semántica: desde la configuración de parámetros, el cálculo de la distancia relativa entre el robot y el objeto, hasta la visualización de resultados y la reproducción de grabaciones de sesión. Su arquitectura modular facilita la separación de responsabilidades en cuatro componentes principales: Home, Configuración, Ejecución y Replay.

3.5.1. Módulo Home

El *Módulo Home* actúa como una unidad de entrada que orquesta la navegación entre módulos y el estado de conexión a los servicios de ROS y a la base de datos (Fig. 3.20). Este módulo

incorpora una serie de validaciones que aseguran el flujo correcto de la aplicación: para acceder al modo de Ejecución se requiere una sesión activa (creada y parametrizada previamente en Configuración) y el nodo de segmentación desplegado. Para entrar a Replay es necesario que los nodos de cámara y segmentación estén detenidos y que el servicio de Rosbag esté activo, de modo que las grabaciones se reproduzcan sincronizadas con el reloj de ROS; y en todos los casos debe mantenerse la conexión a la base de datos para almacenar y recuperar la información de las sesiones.

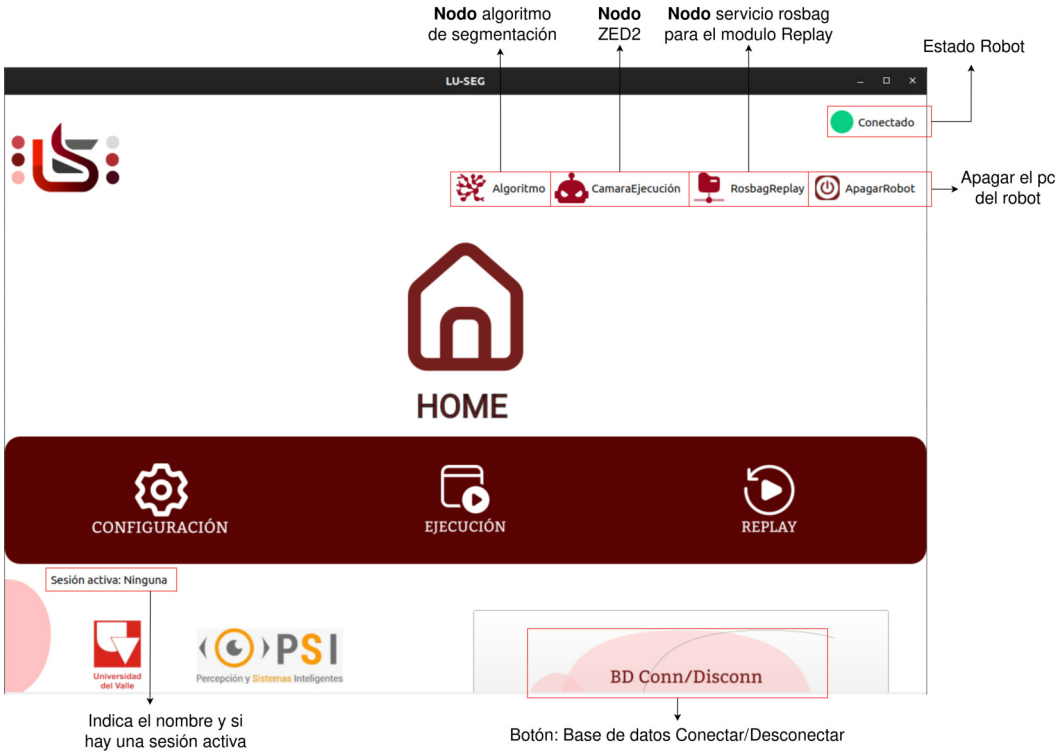


Figura 3.20: Ventana Home

La interfaz de este módulo incluye ocho botones y dos indicadores de estado. Tres de los botones permiten acceder a los módulos de Configuración, Ejecución y Replay; otros tres controlan el inicio o la detención de los nodos `algoritmo_segmentacion`, `zed2` y `rosbag_manager`; un botón apaga el robot cuando está conectado; y el último gestiona la conexión a la base de datos. Los indicadores muestran el estado de la conexión al robot y el nombre de la sesión actualmente activa.

3.5.2. Módulo Configuración

El módulo de Configuración permite al usuario definir tres tipos de parámetros (ver Fig. 3.21):

1. **Comunicación con el robot:** dirección IP del robot, credenciales de usuario del robot (nombre y contraseña), dirección IP del PC local, la IP donde se ejecuta el nodo maestro (Master URI) y su respectivo puerto de comunicación.

2. **Parámetros de ejecución:** incluyen las velocidades lineal y rotativa del robot; opciones de mejoramiento de imagen (transformada exponencial, ecualización de histograma, filtro gaussiano); selección de modelo para la segmentación semántica, umbral de confianza, radio de precisión para el cálculo de distancias; así como la resolución y la tasa de muestreo de la cámara.

3. **Datos de sesión y usuario:** nombre de la sesión, nombre de la persona, correo electrónico y ubicación donde se realiza la sesión.

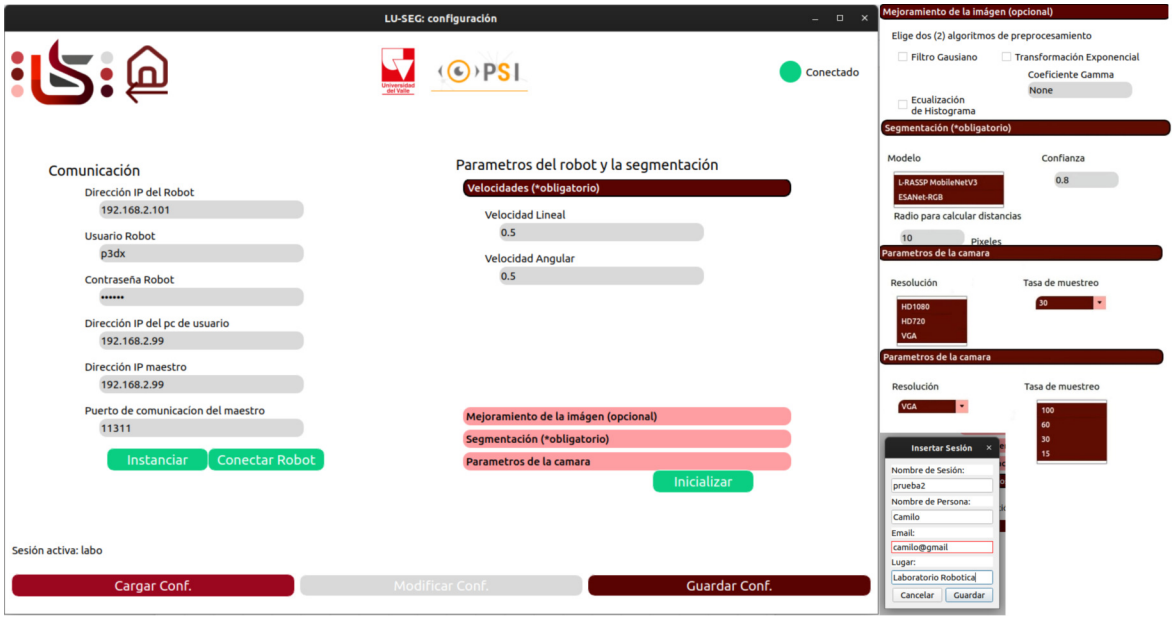


Figura 3.21: Ventana Configuración

Además, este módulo facilita la carga de configuraciones preexistentes, la actualización de las mismas o el almacenamiento de nuevas, estableciendo la configuración guardada o actualizada como sesión activa, esto usando los 3 botones en la parte inferior de la interfaz: *Cargar Conf*, *Modificar Conf* y *Guardar Conf*. Vale mencionar que no solo se incluyo el modelo L-RASSP MobileNetV3, también se integro el modelo ESA-Net RGB, aunque este es mucho más exigente con los recursos computacionales pero con una mayor precisión. Todos estos parámetros de configuración se pueden guardar en la base de datos usando el botón *Guardar Conf.*

3.5.3. Módulo Ejecución

El Módulo Ejecución se encarga de suscribirse a los tópicos de la cámara y del nodo de segmentación, que ejecuta el modelo seleccionado, superpone máscaras y estadísticas sobre la

imagen, además este módulo permite enviar comandos de teleoperación al robot conectado y grabar los tópicos de la sesión (Fig. 3.22).

La interfaz está compuesta por **ocho botones**:

- **Dos botones de grabación de sesión:** permite iniciar la captura de datos por medio de la herramienta `rosbag record` al presionar el botón *Play*, y se puede detener al pulsar *Terminar*.
- **Un botón de navegación de ventanas:** Retornar al módulo Home.
- **Un botón para activar Rosaria:** Arranca el nodo `Rosaria` para la comunicación y control remoto del robot P3DX.
- **Cuatro botones para el control de movimiento del robot (D-Pad):** el pad direccional permite envíar mensajes `cmd_vel` para avanzar, retroceder y girar a izquierda o derecha. Además, permite realizar estas mismas funcionalidades mediante las teclas de flecha del teclado.

Además, cuenta con un panel de estadísticas globales, por cada fotograma, la probabilidad media por clase, el número de regiones detectadas, el área promedio en píxeles y el porcentaje de cobertura de cada clase.

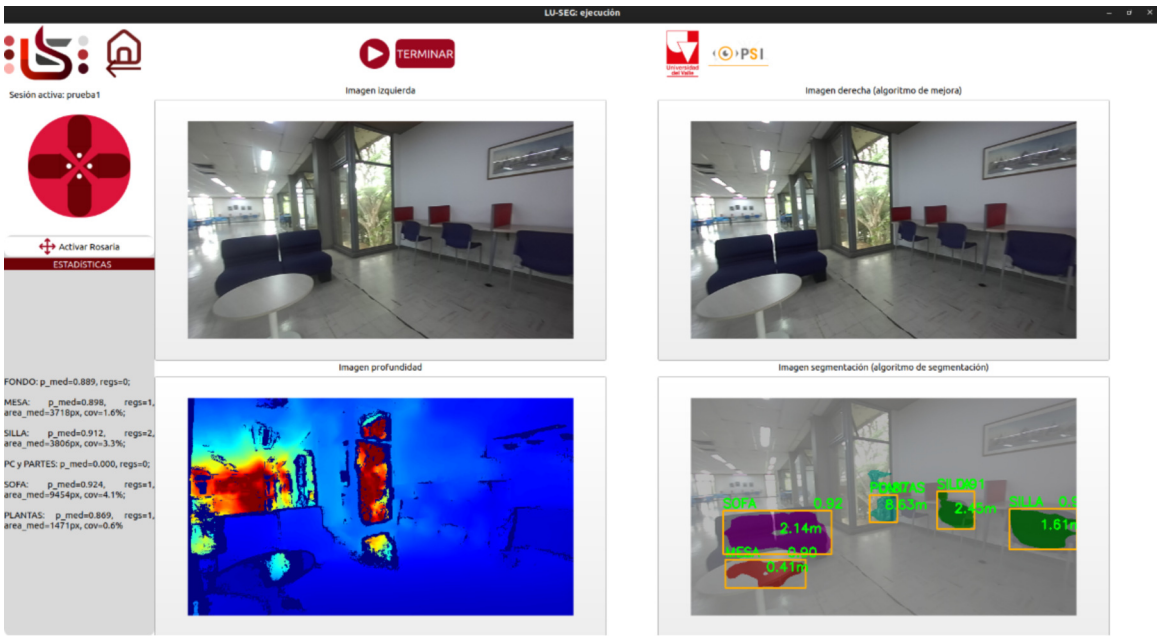


Figura 3.22: Ventana Ejecución

El nodo ROS `algoritmo_segmentacion` inicia consultando un servicio de sesión para cargar modelo, resolución y umbral de confianza. A continuación, sincroniza las imágenes de color y profundidad de la ZED2 con `message_filters`, aplica filtros de mejora de imagen (corrección gamma, ecualización y suavizado gaussiano) en GPU o CPU según disponibilidad y configuración, y normaliza los píxeles con medias y desviaciones de ImageNet.

Posteriormente, ejecuta la red (LR-ASPP MobileNetV3 o ESANet-RGB) con los mejores pesos obtenidos anteriormente en el entrenamiento, estos pesos están en formato `.pth`, para obtener la máscara semántica y el mapa de confianza. A continuación se realiza un post-procesado por componentes conectados de la librería OpenCV por cada clase para extraer regiones válidas y a partir de centroides y la imagen profundidad, calcula la distancia promedio en cada región dentro de un radio dado. Finalmente, genera la imagen anotada (máscara coloreada, bounding boxes, etiquetas de clase y métricas) y publica tres tópicos ROS: imagen mejorada, máscara segmentada y estadísticas, cerrando así el bucle de procesamiento en línea y habilitando la visualización frame a frame en la GUI.

El flujo del funcionamiento del Módulo Ejecución se detalla en los siguientes procedimientos (1-5).

Pseudocódigo 1: Arranque y configuración para el Módulo de Ejecución

Entrada : —
Salida : Parámetros de sesión cargados; nodos `/zed2` y `/algoritmo_segmentacion` listos
Procedimiento: Arranque y configuración

```
// Arranca el nodo /gui_segmentacion y muestra la ventana Home
iniciarNodo("gui_segmentacion")
VentanaHome()
// Muestra botones: Configuración, Ejecución, Algoritmo, CamaraLaunch
if usuario pulsa Configuración then
    openConfiguracion() → VentanaConfiguracion()
    Acción
        Usuario configura parámetros y activa una sesión
        → Inicia el Servicio SessionConfig
    if usuario pulsa símbolo de regresar then
        openHome()→VentanaHome()
    end
end
// Activación de Nodos y Abrir Ventana Ejecución
if usuario pulsa CamaraLaunch then
    toggle_camara_launch() → iniciar_camara_launch()
end
if usuario pulsa Algoritmo then
    toggle_algoritmo() → iniciar_algoritmo()
end
if usuario pulsa Ejecución then
    openEjecucion()→VentanaEjecucion()
end
```

Pseudocódigo 2: Nodo Captura de datos de la cámara ZED2

Entrada : Archivo de configuración (`.yaml`), imágenes izquierda y derecha
Salida : Imágenes publicadas en tópicos */zed2/zed_node/depth/depth_registered*, */zed2/zed_node/right/image_rect_color*, */zed2/zed_node/left/image_rect_color*

Procedimiento: Nodo `/zed2`, captura de datos RGB-D

```
// Inicializa configuración y Arranca el nodo /zed2/zed_node
cargarConfiguración(zed2-conf.yaml)
iniciarNodo("zed2/zed_node")

// Adquisición de frames
left, right ← capturarFrames()
disparidad ← mapaDisparidad(left, right)
depthReg ← registrarProfundidad(disparidad)

// Publicación en tópicos
Publicar→(/zed2/zed_node/depth/depth_registered, depthReg)
Publicar→(/zed2/zed_node/right/image_rect_color, right)
Publicar→(/zed2/zed_node/left/image_rect_color, left)
```

Pseudocódigo 3: Nodo Segmentación

Entrada : Imagen derecha de color e imagen de profundidad de ZED2
Salida : Publicación de tópicos: */algoritmo_segmentacion_node/image_rect_mejorada*, */algoritmo_segmentacion_node/mask*, */algoritmo_segmentacion_node/stats*

Procedimiento: Nodo `algoritmo_segmentacion`

```
// Inicializar nodo, Espera y carga los parámetros de configuración de sesión con el
   servicio SessionConfig
iniciarNodo("algoritmo_segmentacion")
wait_for_service('/get_session_config')
config ← llamarServicio(/get_session_config, SessionConfig)

// Preparar GPU e Instanciar Clases
verificarDisponibilidadGPU()
mejorador, inferencia ← MejoramientoImagen(config), ModelInference(config)

// Suscripción y sincronización de imágenes
color_sub ←suscribir(/zed2/right/image_rect_color)
depth_sub ←suscribir(/zed2/depth/depth_registered)
ts_sync ← ApproximateTimeSynchronizer([color_sub, depth_sub], queue_size = 3)
registerCallback(_on_image_and_depth)
Funcion _on_image_and_depth(imgColorMsg, imgDepthMsg):
    // Conversión y mejoramiento
    imgColor, imgDepth ← toOpenCV(imgColorMsg, imgDepthMsg)
    min_depth, max_depth = 0.3, 15 (metros)
    imgDepth ← convertionToReal(nan = min_depth, posinf = max_depth, neginf = min_depth)

    // Filtros (opcional según configuración)
    if config.applyFilters then
        // exponencial, histograma ó gaussiano
        processed_img ← mejorador.applyFilters(imgColor, config.filterSet)
    end

    // Aplica el proceso de inferencia y postprocesamiento
    overlay_mask, stats = inferencia.overlay_with_stats(processed_img, imgDepth)
```

Procedimiento: Continuación Nodo `algoritmo_segmentacion`

Funcion (continuación) `_on_image_and_depth(`*imgColorMsg*, *imgDepthMsg*`)`**:**

`// Publicación`

Publicar→(*/algoritmo_segmentacion_node/image_rect_mejorada*, *processed_img*)

Publicar→(*/algoritmo_segmentacion_node/mask*, *overlay_mask*)

Publicar→(*/algoritmo_segmentacion_node/stats*, *stats*)

Pseudocódigo 4: Función de Inferencia y Postprocesamiento *overlay_with_stats()*

Entrada : Imagen procesada de color e imagen de profundidad

Salida : Imagen anotada y estadísticas globales

Procedimiento: Función *overlay_with_stats()*

Preprocesamiento

`// 1) Redimensión y conversión a tensor`

imgResized ← resize(*imgColor*, *args.size*) *tensorColor* ← normalizeAndToTensor(*imgResized*, ImageNetMeans, ImageNetStds)

Inferencia

`// 2) LR-ASPP MobileNetV3 o ESANet-RGB`

rawMask ← forward(*tensorColor*)

Postprocesamiento

`// 3) Máscara aplicando threshold y mapa de confianza`

probMap ← softmax(*rawMask*)

conf_map ← max(*probMap*)

mask ← threshold(*probMap*, *args.confidence*)

`// 4) Regiones por clase usando OpenCV`

for 1, *args.num_classes* + 1 **do**

regions, *centroids* ← connectedComponentsWithStats(*mask*)

region_valid, *centroids_valid* ← filterRegions(*regions*, *centroids*, *minArea* = *total_pixels* ∗ 0.003, *maxArea* = *total_pixels* ∗ 0.65)

end

`// 5) Estadísticas globales`

stats ← computeRegionStats(*regions_valid*, *conf_map*)`// probabilidad media, área, # de regiones, cobertura por clase`

`// 6) Distancia con filtro por clase`

distances ← computeRegionDistances(*mask*, *region_valid*, *centroids_valids*, *imgDepth*, *radius* = *args.radius*)

`// 7) Anotación`

maskColored ← colorMask(*mask*) *overlay_mask* ← draw_boxes_labels_distances(*maskColored*, *conf_map*, *regions_valid*, *centroids_valid*, *distances*)

`// 8) Retorno de de la imagen anotada y las estadísticas globales`

return *overlay_mask*, *stats*

Pseudocódigo 5: Nodo GUI Segmentación: Ventana Ejecución

Entrada : Imagen izquierda de color, imagen de profundidad de ZED2, imagen mejorada, imagen anotada y estadisticas del Nodo de segmentación

Salida : Visualización de tópicos: */zed2/zed_node/depth/depth_registered*, */zed2/zed_node/left/image_rect_color*, */algoritmo_segmentacion_node/image_rect_mejorada*, */algoritmo_segmentacion_node/mask*, */algoritmo_segmentacion_node/stats*

Procedimiento: Nodo `gui_segmentacion`: VentanaEjecucion()

```
// Inicialización y Suscripción de tópicos
inicializaciónBotones()
left_sub ←suscribir(/zed2/left/image_rect_color)
depth_sub ←suscribir(/zed2/depth/depth_registered)
right_sub ←suscribir(/algoritmo_segmentacion_node/image_rect_mejorada)
mask_sub ←suscribir(/algoritmo_segmentacion_node/mask)
stats_sub ←suscribir(/algoritmo_segmentacion_node/stats)

// Callbacks de imágenes y estadisticas
// Llama el calback, convierte msg::Image ROS a cv::Mat y actualiza la interfaz
handle_images()→ update_image()

// Refresta el panel de estadísticas
refresh_stats_label()

// Inicio del Nodo RosAria y telecontrol del robot
if Usuario pulsa Activar Rosaria then
    toggle_rosaria() → start_rosaria()
end
while Rosaria esta activado do
    Acción
        Usuario pulsa algún botón del D-Pad
        → El robot se mueve de acuerdo al telecomando dado por el usuario
end

// Gestión de grabación rosbag
if Usuario pulsa "Play" then
    start_rosbag_record()
end
if Usuario pulsa "Terminar" then
    stop_rosbag_record()
end
```

3.5.4. Módulo Replay

Este módulo permite reproducir y visualizar grabaciones de rosbag realizadas en el módulo ejecución (3.5.3) de forma sincronizada con el reloj de ROS, los mensajes visualizados son: las métricas históricas y las cuatro señales de imagen: izquierda, derecha, profundidad y máscara anotada. El usuario puede seleccionar la sesión que desea a través del *combo box Archivo* para visualizar el rosbag file asociado y dispone de controles interactivos de reproducir, pausar y terminar la reproducción (Fig 3.23).
La interfaz dispone de **tres botones** y **tres combo box**:

- **Un botón de navegación de ventanas:** Retornar al módulo Home.
- **Dos botones y un combo box para la reproducción interactiva:** un *combo box* que ofrece los comandos *Play*, *Pausar* y *Terminar*, junto a dos botones que replican estas mismas acciones. Además, al presionar la barra espaciadora se puede alternar entre pausa y reanudación.

- **Dos combo box para la gestión de sesiones:** el *combo box Archivo* incluye las opciones *Abrir Sesión* y *Eliminar Sesión*, y al seleccionar cualquiera de ellas aparece un segundo combo box con el listado de sesiones disponibles.

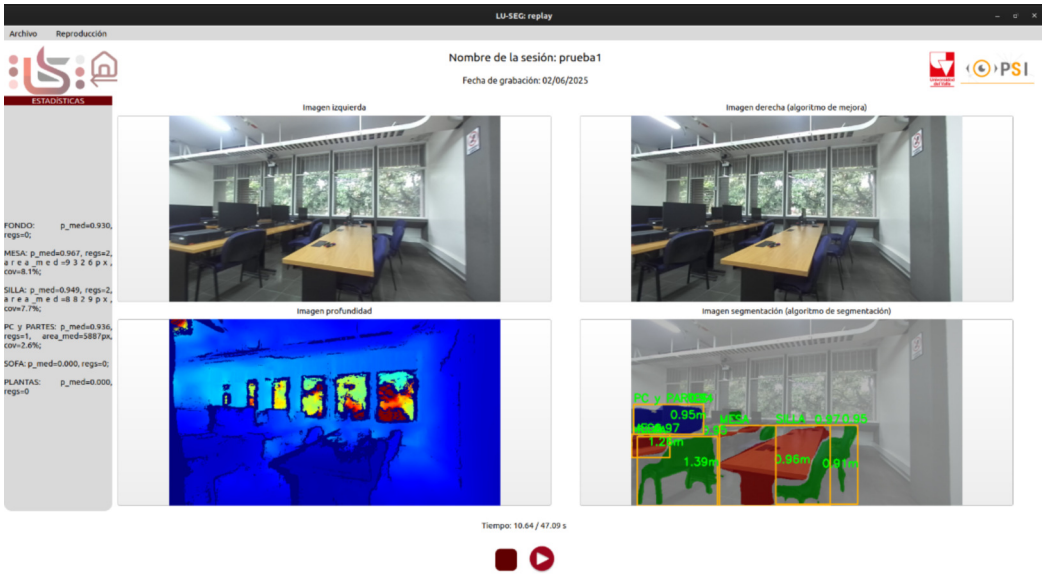


Figura 3.23: Ventana Replay

El funcionamiento de este módulo se detalla en los siguientes pseudocódigos (6-8)

Pseudocódigo 6: Arranque para el Módulo de Replay

Entrada : —
Salida : nodo `rosbag_manager` listo
Procedimiento: Inicio de rosbag manager y la Ventana Replay

```
// Muestra la ventana Home
VentanaHome()
// Muestra botones: Replay, RosbagReplay
if usuario pulsa RosbagReplay then
    toggle_rosbag_manager() → iniciar_rosbag_manager()
end
if usuario pulsa Replay then
    openReplay() → VentanaReplay()
end
```

Pseudocódigo 7: Nodo Rosbag Manager y nodo "play" de rosbag

Entrada : Comandos de reproducción *play*, *pause* y *stop*
Salida : Servicio RosbagControl, inicio del servicio play de rosbag y gestión de comandos de reproducción
Procedimiento: Nodo `rosbag_manager` y "play"

```
// Inicia el servicio de control y gestión de reproducción
iniciarNodo(rosbag_manager")
Inicia el Servicio RosbagControl
```

Procedimiento: Continuación Nodo `rosbag_manager` y "play"

```
// Se prepara para recibir comandos para la gestión de reproducción de rosbag
if command == "play" then
    play_rosbag(bag_path, < name_session > _ID)
    if No esta corriendo el proceso de rosbag play then
        → Arranca rosbag play -clock -prefix <name_session>_ID bag_path
    end
    else
        → Inyecta un comando de -espacio- en el procesos de rosbag play para reanudar la reproducción
    end
end
if command == "pause" then
    pause_rosbag()
    → Inyecta un comando de -espacio- en el procesos de rosbag play para pausar la reproducción
end
if command == "stop" then
    stop_rosbag()
    → Mata el proceso de rosbag play
end
// Publicación en tópicos
Nodo "play" de rosbag
    while Existe el proceso de rosbag play do
        Publicar → (<name_session>_ID/zed2/zed_node/depth/depth_registered)
        Publicar → (<name_session>_ID/zed2/zed_node/left/image_rect_color)
        Publicar → (<name_session>_ID/algoritmo_segmentacion_node/image_rect_mejorada)
        Publicar → (<name_session>_ID/algoritmo_segmentacion_node/mask)
        Publicar → (<name_session>_ID/algoritmo_segmentacion_node/stats)
    end
```

Pseudocódigo 8: Nodo GUI Segmentación: Ventana Replay

Entrada : Imagen izquierda de color, imagen de profundidad de ZED2, imagen mejorada, imagen anotada y estadisticas del Nodo de "play" de rosbag

Salida : Visualización de tópicos: *<name_session>_ID/zed2/zed_node/depth/depth_registered*, *<name_session>_ID/zed2/zed_node/left/image_rect_color*, *<name_session>_ID/algoritmo_segmentacion_node/image_rect_mejorada*, *<name_session>_ID/algoritmo_segmentacion_node/mask*, *<name_session>_ID/algoritmo_segmentacion_node/stats*

Procedimiento: Nodo `gui_segmentacion`: VentanaReplay()

```
// Inicialización de la interfaz y sus botones
inicializaciónBotones()
// Espera el servicio RobagControl
wait_for_service('rosbag_control')
// Instancia un timer para verificar el estado del proceso de reproducción
process_check_timer.start(1000)
```

Procedimiento: Continuación Nodo `gui_segmentacion`: VentanaReplay()

```
// Elección de la sesión a reproducir ó Eliminación de una sesión
Acción: Elegir una sesión
    if Usuario pulsa "Archivo" then
        → Se despliega un combo box con las opciones "Abrir Sesión" o "Eliminar Sesión"
    end
    if Usuario pulsa "Abrir Sesión" then
        → Se despliega un combo box con la lista de todas las sesiones.
        Usuario elige una sesión → Se actualiza la interfaz con la sesión elegida, se muestra el nombre y la
          fecha de grabación
        name_session, ID ← sesion.nombre_sesion, sesion
        // Suscripción de tópicos
        left_sub ←suscribir(<name_session>_ID/zed2/left/image_rect_color)
        depth_sub ←suscribir(<name_session>_ID/zed2/depth/depth_registered)
        right_sub ←suscribir(<name_session>_ID/algoritmo_segmentacion_node/image_rect_mejorada)
        mask_sub ←suscribir(<name_session>_ID/algoritmo_segmentacion_node/mask)
        stats_sub ←suscribir(<name_session>_ID/algoritmo_segmentacion_node/stats)
    end
    if Usuario pulsa "Eliminar Sesión" then
        → Se despliega un combo box con la lista de todas las sesiones
        Usuario elige una sesión → Se elimina la sesión elegida de la lista y de la base de datos si el usuario
          confirma la acción
    end
// Gestión de reproducción rosbag
if Usuario pulsa "Play" then
    play_reproduccion() → Invoca el servicio rosbag_control(command ="play", bag_path,
      name_session_ID)
end
if Usuario pulsa "Pausar" then
    pausar_reproduccion() → Envía el comando rosbag_control(command="pause")
end
if Usuario pulsa "Detener" then
    stop_reproduccion()→ Envía el comando rosbag_control(command="stop")
end
// Callbacks de imágenes y estadisticas
// Llama el calback, convierte msg::Image ROS a cv::Mat y actualiza la interfaz
handle_images()→ update_image()
// Refresta el panel de estadísticas msg::String a String
update_statistics()
```

3.6. Conclusiones

En este capítulo se presentó el desarrollo completo de LU-SEG, desde los fundamentos de diseño hasta la implementación de cada módulo. El diseño modular de LU-SEG, basado en cuatro componentes claros (Home, Configuración, Ejecución y Replay), permitió una implementación ordenada y escalable, facilitando tanto la navegación de usuario como la integración de servicios ROS y la gestión de sesiones en base de datos.

Los requerimientos funcionales del sistema permitieron implementar tanto la interacción con la interfaz como el procesamiento y control del robot: el sistema debe permitir navegar entre los cuatro módulos principales mediante botones claramente identificados (3.5.1); configurar la conexión al robot (dirección IP, credenciales, URI del maestro y puerto) y los parámetros de ejecución (velocidades, mejoras de imagen, selección de modelo, umbral de confianza, radio de precisión y ajustes de cámara), así como guardar y cargar sesiones con sus metadatos (3.5.2).

El sistema debe disponer de controles para iniciar y detener la ejecución, grabar las sesiones con rosbag, enviar comandos de tele-operación (avance, retroceso y giros) y visualizar en linea las cuatro señales de imagen (color, mejora, profundidad y máscara) junto a estadísticas de detección y distancia, capturar y publicar en ROS datos RGB-D del robot móvil; ejecutar el preprocesamiento de imagen (ecualización de histograma, realce exponencial y filtrado gaussiano); y procesar los comandos de tele-operación para controlar el movimiento del robot de forma coordinada con la interfaz (3.5.3).

También debe permitir gestionar las sesiones previas mediante menús desplegables con opciones para abrir, reproducir, pausar, detener y eliminar archivos rosbag, siempre con confirmación del usuario (3.5.4). Estos requerimientos, detallados en las Tablas 3.2 y 3.6, garantizan que LU-SEG satisfaga tanto las necesidades de usabilidad como los objetivos de segmentación en línea y gestión de sesiones.

Por otra parte, la preparación de datos incluyó el filtrado y remapeo de clases en MS COCO (78,939 instancias de entrenamiento, 3,588 de validación) y ADE20K (10,885 / 935 imágenes). Con ello se obtuvo un conjunto de datos representativo que permitió entrenar cuatro arquitecturas.

En la tarea de **segmentación de instancias**, YOLOv11m-seg alcanzó un mAP50 de 63.2 % y un mAP50 : 95 de 41.4 % en máscaras, cuenta con 22 M de parámetros. Mask R-CNN obtuvo un AP_{50} de 59.8 % y un AP global de 38.2 %, 107 M de parámetros. YOLO ofreció así un mejor rendimiento en esta tarea y en estas 6 clases: silla, sofá, mecetas con plantas, mesas, tv/monitor, laptop.

Para **segmentación semántica**, ESANet–RGB logró un mIoU de 85.2 %, precisión de 91.7 % y sensibilidad de 92.1 % (32 M parámetros). L-RASPP MobileNetV3 alcanzó un mIoU de 79.9 %, precisión de 88.4 % y sensibilidad de 88.8 % con sólo 3.2 M de parámetros. La diferencia de 5.3 puntos en IoU, frente a un 90 % menos de parámetros, hizo de MobileNetV3 la opción más equilibrada.

En los experimentos de segmentación de instancias, las clases con mejor desempeño fueron “tv” y “laptop”, mientras que en segmentación semántica la clase con mejor resultado fue “plantas”. Esto se explica porque los métodos de instancias primero localizan la región completa del objeto, luego refinan su contorno y por ultimo clasifican, lo que favorece a objetos con

contornos bien definidos (como televisores o portátiles). En cambio, los modelos semánticos clasifican directamente píxel a píxel, basándose más en características de color y textura que en la forma, lo cual beneficia a la clase "plantas", cuyas características cromáticas prevalecen sobre la forma y las variaciones morfológicas.

Es importante destacar que la clase "mesa" obtuvo resultados relativamente bajos en segmentación de instancias, a pesar de su forma genéricamente regular. Esto puede atribuirse al hecho de que el entrenamiento se realizó principalmente con imágenes de mesas de comedor, donde la presencia de alimentos, cubiertos y otros objetos alteraba significativamente su contorno, dificultando la generación de máscaras coherentes.

El Módulo de Ejecución integró los dos modelos de segmentación semántica en un nodo ROS que sincroniza imágenes RGB-D, aplica mejoras (exponencial, ecualización, filtro gaussiano) en GPU/CPU, ejecuta la inferencia, extrae regiones conectadas, calcula distancias promedio y estadísticas por clase (probabilidad media, número de regiones, área y cobertura) y publica resultados (imagen mejorada, máscara y estadísticas) a 15 Hz aproximados.

Finalmente, el Módulo de Replay permite al usuario reproducir las sesiones grabadas (rosbags) sincronizando el reloj de ROS con la GUI para revisar métricas históricas. Así, LU-SEG no solo ofrece segmentación en vivo con métricas detalladas, sino también herramientas de evaluación offline.

En conjunto, LU-SEG evidencia la integración de modelos de segmentación para su uso en la robótica, ofreciendo un balance entre precisión y facilidad de uso. Su arquitectura modular y la elección de L-RASPP MobileNetV3 permiten un despliegue ligero, mientras que el módulo de Replay permite reproducir y analizar grabaciones de sesión. De este modo, LU-SEG es una herramienta versátil para la experimentación de tareas de visión por computador en robótica, especialmente en entornos interiores.

Capítulo 4

VALIDACIÓN DE LA APLICACIÓN LU-SEG

4.1. Introducción

En este capítulo se describe la estrategia de validación del aplicativo LU-SEG, diseñada para comprobar tanto la conformidad del sistema con sus requerimientos funcionales como su rendimiento en condiciones reales. En primer lugar, se llevan a cabo pruebas de integración por módulos (Home, Configuración, Ejecución y Replay) para asegurar la correcta interacción con ROS y la gestión de sesiones. A continuación, se realizan pruebas cuantitativas sobre el modelo de segmentación, incluyendo matrices de confusión con muestras en movimiento, análisis de precisión a distintas distancias y verificación de estimaciones de profundidad.

Asimismo, se documenta un experimento de campo en el que el robot recorre diferentes espacios interiores, registrando fecha, hora, condiciones de iluminación y duración de la grabación. Finalmente, se presentan las limitaciones observadas y se extraen conclusiones sobre la viabilidad de LU-SEG y posibles mejoras futuras.

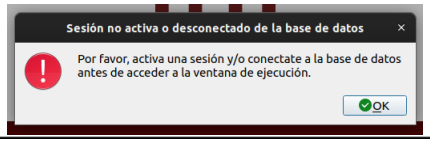
4.2. Pruebas de integración

En esta sección se describen las pruebas de integración diseñadas para verificar que cada módulo de LU-SEG cumpla correctamente con los requerimientos funcionales y se integre de forma coherente con el ecosistema ROS. Para ello, los casos de prueba se organizan en torno a los cuatro módulos principales (Home, Configuración, Ejecución y Replay) y el subsistema de backend, especificando para cada requerimiento tanto el hardware como el software requerido, el objetivo, los datos de entrada, el procedimiento y los resultados esperados y obtenidos. De este modo, se garantiza la trazabilidad entre las especificaciones y la implementación, comprobando tanto la navegación y la captura de parámetros como el registro de datos, la teleoperación del robot, el procesamiento de imágenes RGB-D, la visualización del algoritmo de segmentación y la reproducción de sesiones.

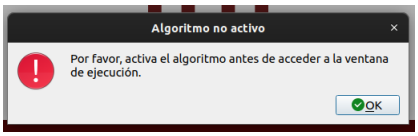
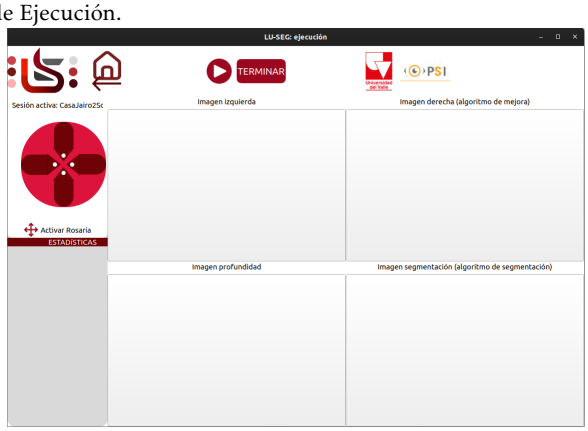
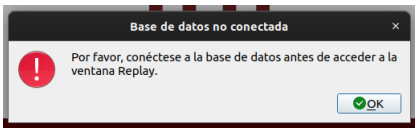
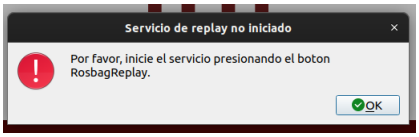
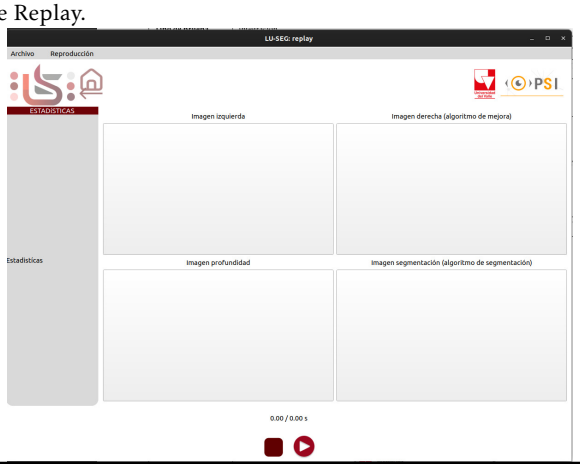
4.2.1. Prueba de Integración - Módulo Home

Esta subsección presenta la prueba de integración destinada a validar la navegación entre los distintos módulos de la aplicación a partir del módulo *Home*. Se puede observar en la Tabla 4.1 el procedimiento y los resultados obtenidos.

Tabla 4.1: Prueba de Integración – Navegación entre módulos.

Requerimientos funcionales de la prueba: -R1- Permitir la navegación entre los diferentes módulos del aplicativo	
Tipo de prueba	Integración
Hardware requerido	Computador
Software requerido	Aplicativo LUSEG, Ubuntu 20.04, ROS Noetic
Objetivo	Verificar que la navegación entre los tres botones (Configuración, Ejecución y Replay) respete las dependencias de sesión, base de datos y nodos ROS activos, mostrando mensajes de error cuando corresponda.
Precondiciones	– La aplicación está arrancada en la ventana Home. – Estado de la base de datos: *conectada* o *desconectada*. – Sesión activa: *configurada* o *no configurada*. – Nodos `algoritmo_segmentacion` y `rosbag_manager`: *activos* o *inactivos*.
Procedimiento	1. Pulsar el botón *Configuración*. 2. Pulsar el botón *Ejecución* bajo cada combinación de precondiciones: a) Sesión no activa o BD desconectada. b) BD conectada & sesión activa & nodo `algoritmo_segmentacion` inactivo. c) BD conectada & sesión activa & nodo `algoritmo_segmentacion` activo al oprimir el botón *Algoritmo* previamente. 3. Pulsar el botón *Replay* bajo cada combinación de precondiciones: a) BD desconectada. b) BD conectada & `rosbag_manager` inactivo. c) BD conectada & nodo `rosbag_manager` activo al oprimir el botón *RosbagReplay* previamente.
Resultado Esperado	La aplicación debe abrir la ventana correspondiente o presentar un mensaje de error claro que explique la dependencia incumplida en cada uno de los tres pasos.
Resultado Obtenido	1. Se abre siempre la ventana de Configuración, sin requisitos previos.  2.a) Ventana de diálogo: 

Continúa en la siguiente página

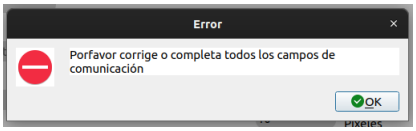
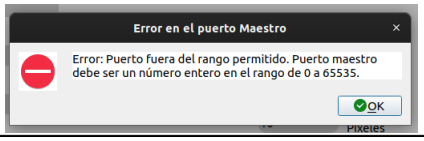
Requerimientos funcionales de la prueba: -R1- (continuación)	
Resultado Obtenido	2.b) Ventana de diálogo:  2.c) Se carga la ventana de Ejecución.  3.a) Ventana de diálogo:  3.b) Ventana de diálogo:  3.c) Se carga la ventana de Replay. 

4.2.2. Prueba de Integración - Módulo Configuración

La validación del módulo *Configuración* se divide en tres pruebas principales: (i) verificación de los campos de comunicación entre el PC y el robot (ver Tabla 4.2, requerimiento R2-R9); (ii) validación de los parámetros del algoritmo, la cámara y el robot (ver Tabla 4.3, requerimiento

R10); y (iii) comprobación del guardado y la carga de estos parámetros configuración desde y hacia la base de datos (ver Tabla 4.4, requerimientos R11 y R12). Al igual que el módulo anterior, se especifican los requisitos y condiciones previas, el procedimiento y los resultados esperados y obtenidos.

Tabla 4.2: Prueba de Integración – Campos de comunicación e indicador de conexión

Requerimientos funcionales de la prueba:-R2–R9- Validar los campos de comunicación con el robot (IP robot, usuario, contraseña, IP PC, IP maestro, puerto), el comportamiento del botón "Conectar" y el indicador de estado de conexión al robot.	
Tipo de prueba	Integración
Hardware requerido	Computador
Software requerido	Aplicativo LUSEG, Ubuntu 20.04, ROS Noetic
Objetivo	Asegurar que cada campo acepta solo los datos esperados, que el sistema muestra los mensajes de error adecuados y el estado de conexión al robot.
Precondiciones	– La ventana Configuración está abierta. – El estado de conexión es: desconectado
Procedimiento	1. Intentar escribir letras en los campos numéricos (IP y puerto). 2. Rellenar cada campo con datos válidos: – IP robot: "192.168.2.101" , Usuario robot: "p3dx" , Contraseña robot: "....." (oculto) , IP PC: "192.168.2.100", IP maestro: "192.168.2.100", Puerto: "11311". 3. Pulsar *Instanciar* sin rellenar algún campo. 4. Ingresar un puerto fuera de rango ("65555") y oprimir el botón *Instanciar*. 5. Con todos los campos instanciados correctamente, pulsar "Conectar" con la IP del robot: a) IP no válida o robot apagado. b) Robot accesible.
Resultado esperado	El sistema debe: Rechazar caracteres inválidos en cada campo; mostrar el mensaje de error preciso según el caso (campos incompletos, puerto fuera de rango, robot inaccesible); establecer la conexión cuando los parámetros sean válidos y el robot responda, y visualizar el estado de conexión.
Resultado Obtenido	1. No se permiten caracteres no numéricos ni puntos fuera de lugar. 2. Todos los campos aceptan el formato correcto, ventana de diálogo:  3. Ventana de diálogo:  4. Ventana de diálogo 

Continúa en la siguiente página

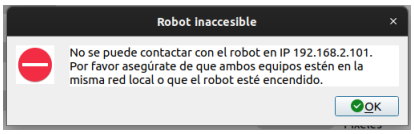
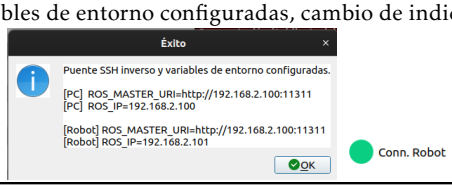
Requerimientos funcionales de la prueba:-R2–R9- (continuación)	
Resultado Obtenido	5. Con todos los campos instanciados correctamente, pulsar “Conectar” con la IP del robot: a) Ventana de diálogo:  b) Robot accesible: Éxito y variables de entorno configuradas, cambio de indicador de estado a “Conectado”. 

Tabla 4.3: Prueba de Integración – Parámetros del robot y del Algoritmo de segmentación

Requerimientos funcionales de la prueba:-R10- Ajuste de velocidades lineal y angular, así como parámetros de segmentación y cámara.	
Tipo de prueba	Integración
Hardware requerido	Computador
Software requerido	Aplicativo LUSEG, Ubuntu 20.04, ROS Noetic
Objetivo	Verificar que los campos de velocidad y parámetros de ejecución acepten únicamente valores dentro de su rango, y que al pulsar “Inicializar” se inicializan correctamente todas las opciones.
Precondiciones	– La ventana Configuración está abierta.
Procedimiento	1. Intentar escribir caracteres no numéricos en los campos de velocidad y confianza. 2. Introducir valores fuera de rango y pulsar “Inicializar”: a) Velocidad lineal = 1.2 (rango 0.0 – 0.6). b) Velocidad angular = 1.0 (rango 0.0 – 0.9). c) Confianza = 1.5 (rango 0.0 – 0.1). d) Seleccionar ambos métodos de mejora (histograma y exponencial). e) Radio de distancia = valor no entero, p.ej. 5.7. 3. Dejar vacío alguno de los campos obligatorios (velocidades, confianza, radio). 4. Rellenar todos los campos con valores válidos: – Velocidad lineal = 0.5, velocidad angular = 0.8. – Seleccionar un único método de mejora. – Modelo: MobileNetV3 (por defecto), confianza = 0.75, radio = 5. – Resolución y tasa de muestreo (opciones estáticas). Pulsar “Inicializar”.
Resultado esperado	El sistema debe: Rechazar caracteres no numéricos en cada campo; mostrar el mensaje de error específico según el parámetro fuera de rango o la selección inválida; bloquear la inicialización si faltan campos obligatorios; instanciar correctamente todas las opciones al pulsar “Inicializar” con valores válidos.
Resultado Obtenido	1. No se permiten letras ni símbolos fuera de punto decimal. 2. Muestra ventanas de diálogo con los mensajes: a) “Error: La velocidad lineal está fuera del rango permitido (0.0 a 0.6)”. b) “Error: La velocidad angular está fuera del rango permitido (0.0 a 0.9)”. c) “Error: La confianza está fuera del rango permitido $0.0 < c \leq 1.0$”. d) “Error: No puedes seleccionar ambos métodos (histograma y exponencial)”. e) “Error: El radio de precisión debe ser un número entero”.

Continúa en la siguiente página

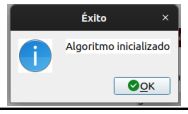
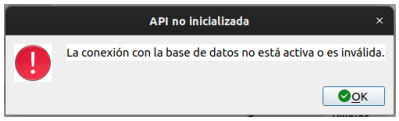
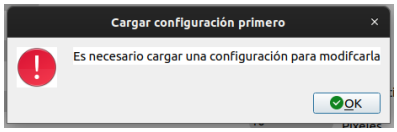
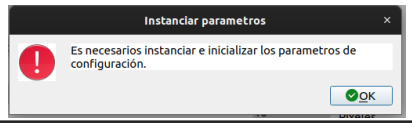
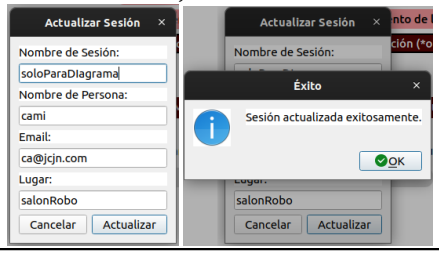
Requerimiento R10: (continuación)	
Resultado Obtenido	3. Ventana de diálogo:  4. Todos los parámetros se inicializan sin error. 

Tabla 4.4: Prueba de Integración – Guardar, cargar parámetros de configuración, nombrar e ingresar metadatos de sesiones en base de datos

Requerimientos funcionales de la prueba:-R11–R12- Guardado, carga y nombrado de sesiones en base de datos.	
Tipo de prueba	Integración
Hardware requerido	Computador
Software requerido	Aplicativo LUSEG, Ubuntu 20.04, ROS Noetic
Objetivo	Verificar que el sistema gestione correctamente las sesiones: muestre errores cuando no haya conexión, liste configuraciones guardadas, y abra el diálogo de metadatos (nombre, usuario, email, lugar) para guardar o actualizar.
Precondiciones	– Ventana Configuración abierta. – Campos de conexión y parámetros de ejecución ya instanciados.
Procedimiento	1. Pulsar “Guardar Conf” o “Cargar Conf” sin conexión a BD. 2. Conexión activa a BD, pulsar “Cargar Conf”. 3. Pulsar “Modificar Conf” sin haber cargado ninguna. 4. Pulsar “Guardar Conf” o “Modificar Conf” sin instanciar/inicializar parámetros. 5. Rellenar todos los parámetros y pulsar “Guardar Conf”. 6. Con Ventana de metadatos de sesión abierto y botón “Actualizar” (en modificación), cambiar valores y confirmar.
Resultado esperado	Mensajes de error claros cuando no hay conexión a la BD o faltan pasos previos; listado de sesiones disponibles y carga automática de sus parámetros; ventana de diálogo de metadatos (guardar/actualizar) sólo tras instanciar parámetros; persistencia correcta en la base de datos tras guardar o actualizar.
Resultado Obtenido	1. Ventana de diálogo  2. Se despliega lista de sesiones guardadas. Seleccionar una ⇒ se cargan todos los parámetros en sus campos correspondientes. 3. Ventana de diálogo:  4. Ventana de diálogo: 

Continúa en la siguiente página

Requerimientos R11–R12: (continuación)	
Resultado Obtenido	5. Ventana para ingresar metadatos: ⇒ al confirmar, los datos se registran en la BD:  6. Ventana para actualizar metadatos⇒ la BD refleja los cambios correctamente. 

4.2.3. Prueba de Integración - Módulo Ejecución

La validación del módulo *Ejecución* se llevó a cabo mediante tres pruebas específicas: (i) verificación de la grabación de sesiones, correspondiente a los requerimientos R13, R14 y R20 (ver Tabla 4.5); (ii) validación del envío y recepción de comandos de teleoperación entre la aplicación y el robot, asociada a los requerimientos R15 y BR2 del backend (ver Tabla 4.6); y (iii) validación del objetivo principal del sistema: la segmentación en linea de objetos de interés en interiores, cubriendo al menos cinco clases distintas, con sus respectivas anotaciones y estadísticas (ver Tabla 4.7), lo cual cumple con los requerimientos R16–R19, BR1 y BR3 del backend.

Tabla 4.5: Prueba de Integración – Grabación de la sesión en ejecución.

Requerimientos funcionales de la prueba:-R13–R14, R20- Iniciar y terminar la grabación de sesión (rosbag)	
Tipo de prueba	Integración
Hardware requerido	Computador, Cámara estéreo ZED2
Software requerido	Aplicativo LUSEG, Ubuntu 20.04, ROS Noetic, paquete de *zed_wrapper* de ROS
Objetivo	Verificar que los botones *Play* y *Terminar* permitan controlar correctamente la grabación de los tópicos de imagen y estadísticas, y que muestren los diálogos de éxito o error adecuados.
Precondiciones	– La ventana "Ejecución" está abierta. – Botón *Play*: habilitado; Botón *Terminar*: deshabilitado. – Nodo `algoritmo_segmentacion` activo y un publicador de ZED2 (o reproducción de un rosbag) emitiendo en: `/zed2/zed_node/left/image_rect_color`, `/zed2/zed_node/right/image_rect_color`, `/zed2/zed_node/depth/depth_registered`.
Procedimiento	1. Pulsar *Play* 2. Pulsar *Terminar* 3. Cerrar la ventana mientras graba.
Resultado esperado	– Al pulsar *Play*, comienza la grabación sin restricciones adicionales. – Al pulsar *Terminar* o cerrar la ventana, se detiene la grabación y se muestra "La grabación de rosbag se ha detenido correctamente."

Continúa en la siguiente página

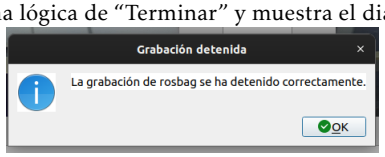
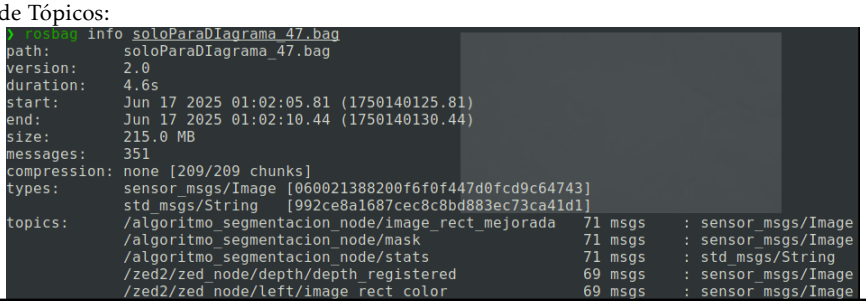
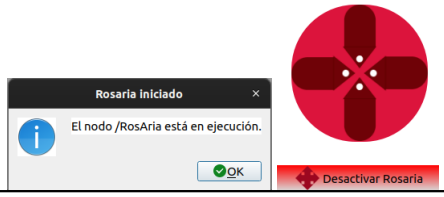
Requerimientos R13–R14, R20: (continuación)	
Resultado esperado	– Los archivos `.bag` contienen los tópicos de imagen y estadísticas grabados.
Resultado Obtenido	1. Botón *Play* se deshabilita; *Terminar* se habilita; comienza `rosbag record` en segundo plano de los tópicos de imagen, el tópico `/algoritmo_segmentacion_node/image_rect_mejorada` y `/algoritmo_segmentacion_node/stats`. 2. Botón *Terminar* se deshabilita; *Play* se habilita; aparece diálogo "La grabación de rosbag se ha detenido correctamente." 3. Invoca automáticamente la misma lógica de "Terminar" y muestra el diálogo de éxito.  Grabación de Tópicos: 

Tabla 4.6: Prueba de Integración – Envío de comandos de tele-operación

Requerimientos funcionales de la prueba:-R15 y Backend R2- Envío de comandos de tele-operación (avance, retroceso, giros) mediante botones y teclas.	
Tipo de prueba	Integración
Hardware requerido	Computador, Robot P3DX , Router
Software requerido	Aplicativo LUSEG, Ubuntu 20.04, ROS Noetic, paquete de */rosaria* de ROS
Objetivo	Comprobar que, tras activar Rosaria, la interfaz envía mensajes `/cmd_vel` correctos al nodo `/RosAria` al pulsar o mantener los botones y teclas direccionales.
Precondiciones	– Ventana "Ejecución" abierta.
Procedimiento	1. Pulsar *Activar Rosaria*. 2. Mantener pulsado "Arriba" (o tecla Up). 3. Mantener pulsado "Atrás" (o tecla Down). 4. Mantener pulsado "Izquierda" (o tecla Left). 5. Mantener pulsado "Derecha" (o tecla Right). 6. Pulsar *Desactivar Rosaria*.
Resultado esperado	– Rosaria se activa o informa del error de conexión. – Cada dirección envía un `Twist` apropiado en `/RosAria/cmd_vel`. – El robot se mueve mientras el botón o la tecla esté pulsada y se detiene al soltar. – Al desactivar, el nodo se cierra y la interfaz regresa al estado inicial.
Resultado Obtenido	1.a) Caso éxito: ventana de diálogo "Rosaria iniciado — El nodo /RosAria está en ejecución.", el botón muestra *Desactivar Rosaria* y el robot pita. 

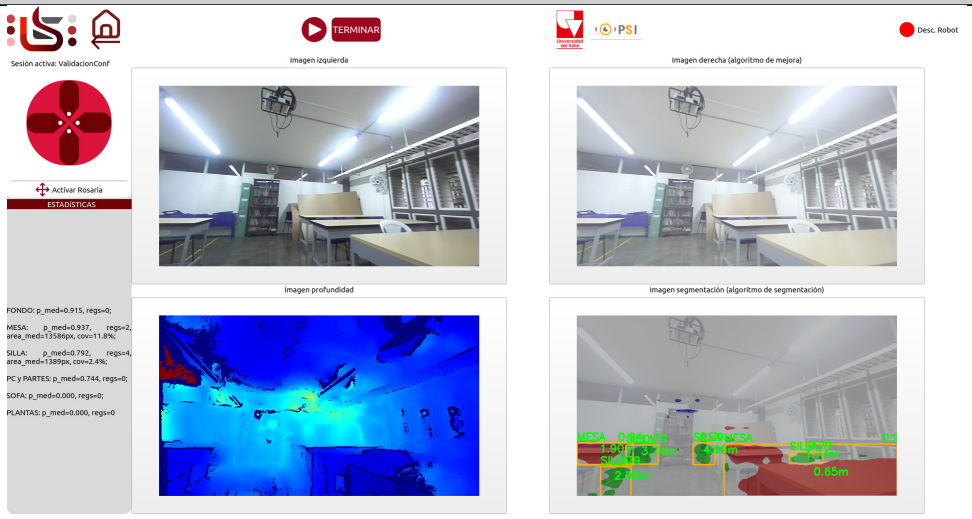
Continúa en la siguiente página

Requerimientos R15 y BR2: Envío de comandos de tele-operación (continuación)	
Resultado Obtenido	1.b) Caso fallo (robot inaccesible): Ventana de diálogo:  2. El robot avanza mientras se mantiene pulsado y se detiene al soltar el botón o tecla. 3. El robot retrocede mientras se mantiene pulsado y se detiene al soltar el botón o tecla.. 4. El robot gira a la izquierda mientras se mantiene pulsado y se detiene al soltar el botón o tecla. 5. El robot gira a la derecha mientras se mantiene pulsado y se detiene al soltar el botón o tecla. 6. El nodo `/RosAria` es detenido, el botón vuelve a nombrarse *Activar Rosaria* y el robot pita.

Tabla 4.7: Prueba de Integración – Segmentación de objetos, visualización de estadísticas y distancias relativas

Requerimientos funcionales de la prueba:-R16–R19 y Backend R1 y R3- Segmentación de al menos cinco objetos de interés (plantas, sillas, PCs, sofás y mesas) visualización de estadísticas (precisión media, regiones, área media en pixeles, cobertura en porcentaje), detecciones y distancias relativas.	
Tipo de prueba	Integración
Hardware requerido	Computador, Cámara estéreo ZED2, GPU Nvidia
Software requerido	Aplicativo LUSEG, Ubuntu 20.04, ROS Noetic, paquete *zed_wrapper* de ROS, CUDA, Pytorch 2.4.1, OpenCV para GPU
Objetivo	Comprobar que el nodo `/algoritmo_segmentacion` procesa y publica correctamente las imágenes y estadísticas, y que la GUI muestra máscaras, bounding–boxes, etiquetas, probabilidades y distancias.
Precondiciones	– Ventana "Ejecución" abierta. – Nodo `/algoritmo_segmentacion` en ejecución. – Fuente de imágenes activa (ZED2 o rosbag) publicando en: `/zed2/zed_node/left/image_rect_color`, `/zed2/zed_node/right/image_rect_color`, `/zed2/zed_node/depth/depth_registered`.
Procedimiento	1. Observar que, tras iniciar la ventana, aparece en panel izquierdo la sección de estadísticas globales y en panel derecho las cuatro imágenes (color, profundidad, mejora y anotación). 2. Para cada una de las cinco clases esperadas (*silla*, *mesa*, *planta*, *PC*, *sofá*), presentar un objeto real en escena. 3. Verificar en el panel de estadísticas globales. 4. Visualizar los diferentes métodos de mejora de imagen disponibles en la imagen superior derecha, según configuración previa.
Resultado esperado	– Se segmentan y distinguen al menos cinco clases diferentes. – La GUI muestra máscaras, cajas, etiquetas, probabilidades y distancias relativas. – El panel de estadísticas refleja valores coherentes para cada clase.
Resultado Obtenido	1. Cumple. 2.– La máscara se colorea correctamente sobre la imagen anotada. – Se dibuja un *bounding–box* alrededor de la región. – Se muestra el nombre de la clase y su probabilidad junto a la caja. – Muestra la distancia relativa al objeto. 3.Para cada clase detectada, aparecen: – *P_medio* (probabilidad media), – *regs* (número de regiones), – *area_media* (píxeles), – *cov%* (cobertura porcentual). 4. Se pueden visualizar cambios de luminosidad (ecualización de histograma y transformada exponencial) y la imagen se ve borrosa (filtro gaussiano).

Continúa en la siguiente página

Requerimientos	R16–R19 y BR1, BR3: (continuación)
Resultado Obtenido	

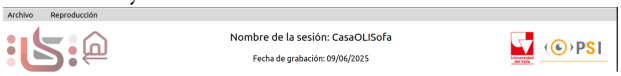
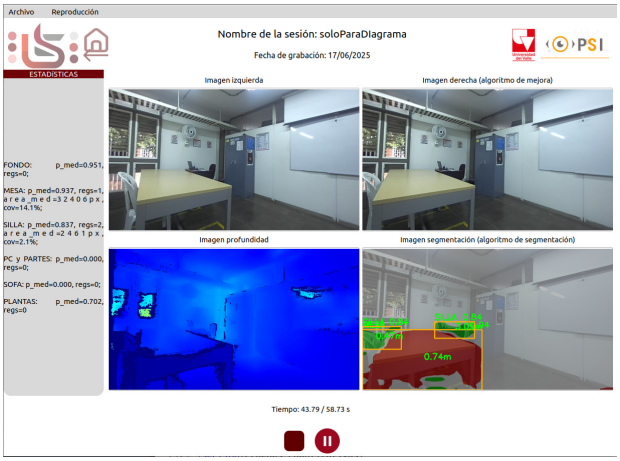
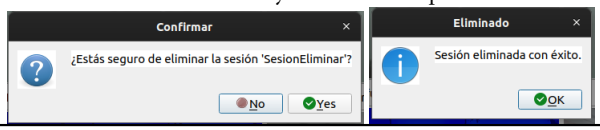
4.2.4. Prueba de Integración - Módulo Replay

Esta prueba cubre los requerimientos R21–R27 y valida la funcionalidad del módulo *Replay*, específicamente en cuanto a la gestión de sesiones previas, lo que incluye su apertura, visualización y eliminación según la acción del usuario, así como su correcta reproducción desde la interfaz (ver Tabla 4.8).

Tabla 4.8: Prueba de Integración – Gestión y reproducción de sesiones previas

Requerimientos funcionales de la prueba:-R21–R27- Gestión y reproducción de sesiones previas.	
Requerimientos	R21–R22: Carga y listado de sesiones previas. R23–R24: Play, pausa y detención de la reproducción con sincronización de reloj. R25–R27: Eliminación de sesiones con confirmación.
Tipo de prueba	Integración
Hardware requerido	Computador
Software requerido	Aplicativo LUSEG, Ubuntu 20.04, ROS Noetic.
Objetivo	Comprobar que la interfaz de Replay permite cargar, listar, reproducir, pausar y eliminar sesiones guardadas, mostrando mensajes y sincronizando el tiempo correctamente.
Precondiciones	– Ventana “Replay” abierta. – Base de datos conectada. – Nodo `rosbag_manager` activo.
Procedimiento	1. Pulsar el combo “Archivo” y seleccionar “Abrir Sesión”. 2. Seleccionar una sesión de la lista. 3. Pulsar “Play” sin haber grabado la sesión en la Ventana Ejecución. 4. Seleccionar una sesión válida (con grabación previamente) y pulsar “Play”. 5. Presionar “Pausa” o barra espaciadora durante la reproducción: 6. Pulsar el botón “Detener” (cuadrado). 7. Pulsar combo “Archivo” → “Eliminar Sesión”: 8. Confirmar eliminación.
Resultado esperado	– Menús muestran y cargan correctamente las sesiones disponibles. – “Play” sin `rosbag_path` muestra error; con `rosbag_path` inicia reproducción.

Continúa en la siguiente página

Requerimientos funcionales de la prueba:-R21–R27- (continuación)	
Resultado esperado	– Pausa y detención funcionan tanto con botones como con barra espaciadora, y tiempo “actual/final” se sincroniza. – Eliminación pide confirmación y refleja el cambio en la base de datos.
Resultado Obtenido	1. Se despliega lista de sesiones (nombre de sesión). 2. Se muestra en pantalla el nombre y fecha de la sesión activa.  3. Ventana de diálogo “La sesión no tiene un rosbag_path definido.” 4. Comienza reproducción, panel izquierdo muestra estadísticas, panel derecho las imágenes; el botón cambia a “Pausa”.  5. Pausa la reproducción, el botón vuelve a “Reanudar” (Play).  6. Se detiene la reproducción y, al volver a pulsar Play, se reinicia desde el comienzo. 7. Lista de sesiones; al seleccionar una, aparece diálogo “¿Estás seguro de eliminar la sesión ‘X’?”. 8. Ventana de diálogo “Sesión eliminada con éxito.” y la sesión desaparece de la lista (BD actualizada). 

4.3. Pruebas Cuantitativas

Con el fin de evaluar objetivamente el desempeño del aplicativo LU-SEG y su sistema de segmentación en linea, se llevaron a cabo dos tipos de pruebas cuantitativas: una centrada en la precisión del modelo y otra orientada al análisis de la distancia relativa entre el sensor y los objetos detectados. Ambas pruebas se realizaron en entornos representativos de uso real, incluyendo oficinas, pasillos y salas de estar, utilizando el modelo L-RASSP basado en MobileNet V3 operando en procesamiento en línea.

4.3.1. Pruebas de precisión

Para determinar la precisión del sistema, se recolectaron 140 imágenes distribuidas en 7 sesiones de grabación, en las cuales la cámara se desplazó por distintos escenarios interiores. Cada sesión fue analizada a partir de la grabación original, pausando la secuencia en momentos clave para capturar fotogramas representativos. Posteriormente, se seleccionaron las imágenes de modo que las cinco clases de interés estuvieran adecuadamente distribuidas a lo largo del conjunto de datos, el proceso de anotación del *ground truth* se realizó de forma completamente manual. A partir de estas muestras, se construyó una matriz de confusión que permitió calcular métricas clave como la precisión y la sensibilidad del modelo. Estas pruebas permiten cuantificar la capacidad del sistema para detectar correctamente objetos de interés bajo condiciones reales durante la ejecución continua.

La Figura 4.1 muestra la matriz de confusión obtenida a partir de todas las predicciones realizadas por el sistema durante la validación. En esta matriz se presentan los valores enteros sin normalizar, lo que permite apreciar la distribución real de los errores por clase. Se destaca un buen desempeño general en la mayoría de clases, aunque se observan errores frecuentes entre clases visualmente similares, como *Silla* y *Sofá*, y una tendencia del modelo a dejar de detectar algunos objetos, clasificándolos erróneamente como fondo.

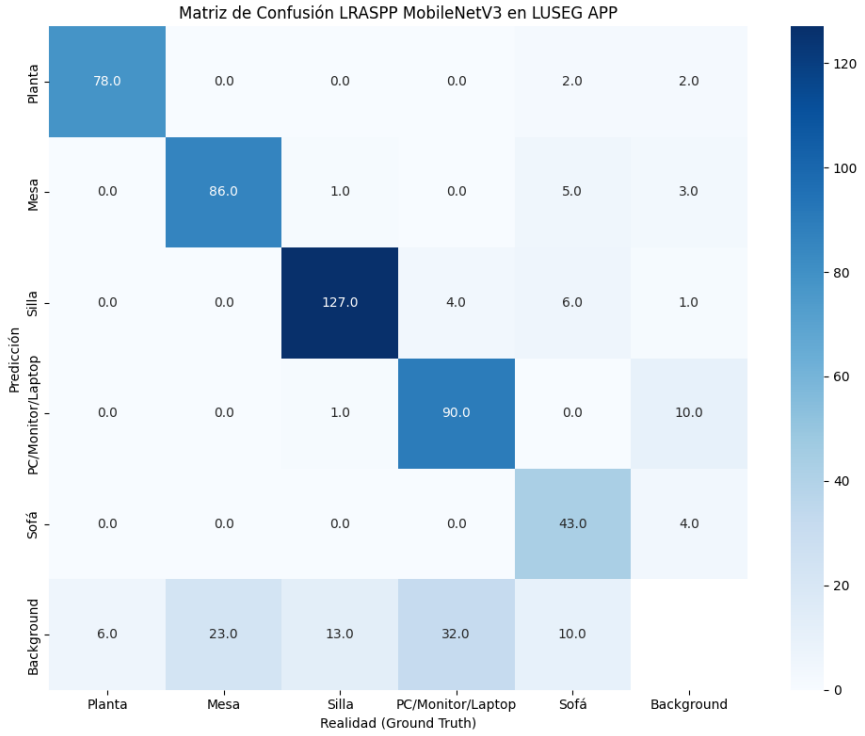


Figura 4.1: Matriz de Confusión para la Validación de MobileNetV3 en LU-SEG APP

La Tabla 4.9 presenta los valores obtenidos para cada clase, incluyendo el número total de regiones válidas (*GT*) anotadas manualmente, utilizadas como referencia para la validación.

La columna *True Positives (TP)* representa las detecciones correctas del modelo, mientras que los *False Positives (FP)* corresponden a detecciones erróneas en las que se identificó un objeto inexistente o perteneciente a otra clase. Los *False Negatives (FN)* representan los casos en los que un objeto existente en la imagen no fue detectado por el sistema, y los *True Negatives (TN)* agrupan los casos en los que el modelo identificó correctamente la ausencia de la clase correspondiente. Se observa que las clases con mayor frecuencia de aparición, como *Silla*, *Mesa* y *PC/Monitor/Laptop*, presentan también un número elevado de aciertos, lo cual resulta coherente con su mayor presencia en los entornos evaluados.

Clase	Regiones Validas (GT)	TP (Aciertos)	FP	FN	TN
Planta	84	78	4	6	461
Mesa	109	86	9	23	435
Silla	142	127	11	15	404
PC/Monitor/Laptop	126	90	11	36	411
Sofá	66	43	4	23	477

Tabla 4.9: Valores de TP, FP, FN y TN para cada clase, a partir de la matriz de confusión final. El total corresponde a 527 objetos reales más 20 casos de fondo predichos como alguna clase (FP_{fondo}), para un total de 547 regiones evaluadas.

La diferencia en el número de anotaciones por clases es una consecuencia de los entornos presentes en las sesiones realizadas: por ejemplo, un laboratorio u oficina suele contener numerosas sillas, mientras que una sala de estar cuenta con 1 o máximo 3 sofás. A pesar de esta variabilidad, los resultados evidencian la capacidad general del sistema para identificar correctamente los objetos en condiciones reales de operación.

A partir de la matriz de confusión se calcularon las métricas de precisión y sensibilidad por clase, cuyos valores se presentan en la Tabla 4.15. En términos generales, se observa un buen desempeño en la mayoría de las clases, con un **promedio general de precisión de 0.9166** (ver Métrica 2.2) y una **sensibilidad promedio de 0.7955** (ver Métrica 2.3). Estos valores indican que el sistema realiza predicciones mayormente correctas cuando identifica una clase, aunque presenta cierta limitación para detectar la totalidad de los objetos presentes, como lo refleja la diferencia entre ambas métricas.

Clase	Precisión	Recall (sensibilidad)
Planta	0.9512	0.9286
Mesa	0.9053	0.7890
Silla	0.9203	0.8944
PC/Monitor/Laptop	0.8911	0.7143
Sofá	0.9149	0.6515
Promedio Total	**0.9166**	**0.7955**

Tabla 4.10: Precisión y sensibilidad por clase a partir de la matriz de confusión.

4.3.2. Pruebas de distancia relativa

En esta prueba se analizó cómo varía el desempeño del sistema en función de la distancia entre la cámara y los objetos detectados, realizando las pruebas para cada una de las clases de interes: plantas, mesas, sillas, equipos de cómputo (PC, monitores, laptops) y sofás; y se capturaron imágenes a diferentes distancias. Esta evaluación permite conocer el rango efectivo de segmentación y las limitaciones del modelo en términos de profundidad, especialmente en escenarios donde los objetos se encuentran alejados o parcialmente visibles.

Es importante señalar que las pruebas no pudieron realizarse en un mismo entorno para todas las clases debido a limitaciones espaciales y la disposición de objetos. La clase *Sofá* se evaluó en una sala de estar; *Monitor (PC)*, *Mesa* y *Silla* en un espacio de estudio que ofrecía un rango de distancia adecuado; y *Planta* en un pasillo del edificio E53 de Ingeniería Electrónica de la Universidad del Valle, donde se encontraba disponible el objeto.

La Tabla 4.11 muestra que la clase *Planta* fue detectada correctamente hasta 8.4 m, con errores que aumentan gradualmente con la distancia. A partir de 9.6 m la segmentación se vuelve inestable y falla por completo a 10.8 m. El error promedio fue de 0.24 m, evidenciando una buena precisión en rangos cercanos e intermedios.

Distancia Nominal (m)	**Resultado**	**Valor Observado (m)**	**Error (m)**
1.2	✓	1.26	0.06
2.4	✓	2.36	0.04
3.6	✓	3.61	0.01
4.8	✓	4.68	0.12
6.0	✓	5.74	0.26
7.2	✓	6.88	0.32
8.4	✓	8.02	0.38
9.6	Inestable	8.86	0.74
10.8	✗	–	–
		Error Promedio	**0.24**

Tabla 4.11: Desempeño del sistema para segmentación de *Planta* según distancia. Altura: 118 cm, Base: 33 cm × 33 cm.

La Tabla 4.12 compara el desempeño para las clases *Silla*, *Mesa* y *Monitor*. *Silla* mostró mayor alcance, con detecciones estables hasta 4.5 m, mientras que *Mesa* y *Monitor* fueron confiables solo hasta 2.7 m. A partir de 3.6 m, ambas presentan fallos consistentes. Los errores promedio en los rangos válidos fueron de 0.26 m, 0.25 m y 0.11 m, respectivamente.

Distancia (m)	**Silla**			**Mesa**			**Monitor**		
	Res.	Obs.	Err.	Res.	Obs.	Err.	Res.	Obs.	Err.
0.9	✓	1.36	0.46	✓	1.49	0.59	✓	1.03	0.13
1.8	✓	2.27	0.47	✓	1.70	0.10	✓	1.85	0.05

Continúa en la siguiente página

Continúa la tabla 4.12

Distancia (m)	Silla			Mesa			Monitor		
	Res.	Obs.	Err.	Res.	Obs.	Err.	Res.	Obs.	Err.
2.7	✓	2.63	0.07	✓	2.65	0.05	✓	2.84	0.14
3.6	✓	3.83	0.23	✗	–	–	✗	–	–
4.5	✓	4.18	0.32	✗	–	–	✗	–	–
5.4	Inestable	5.41	0.01	✗	–	–	✗	–	–
Error Promedio			**0.26**			**0.25**			**0.11**

Tabla 4.12: Comparación del desempeño según distancia para las clases *Silla* (Altura: 82 cm, 47×52 cm), *Mesa* (Altura: 71 cm, 74×74 cm) y *Monitor* (44×29 cm).

La Tabla 4.13 muestra que la clase *Sofá* fue correctamente segmentada hasta una distancia de 2.4 m, siendo la más exigente del conjunto evaluado. Su gran tamaño, junto con su disposición en el entorno, en ocasiones dificulta que la segmentación cubra completamente el área del objeto. A partir de los 3 m, el sistema no logra realizar detecciones válidas. En el rango de 0.6 m a 2.4 m, el error promedio fue de 0.18 m.

Distancia Nominal (m)	Resultado	Valor Observado (m)	Error (m)
0.6	✓	1.18	0.58
1.2	✓	1.25	0.05
1.8	✓	1.87	0.07
2.4	✓	2.42	0.02
3.0	✗	–	–
		Error Promedio	**0.18**

Tabla 4.13: Desempeño del sistema para segmentación de *Sofá* según distancia. Altura: 68 cm, Largo: 120 cm, Ancho: 80 cm. Profundidad desde el borde: 65 cm.

La Figura 4.2 ilustra el desempeño del sistema según la distancia para cada clase. Todas muestran una estimación coherente en su rango operativo, con errores crecientes a mayor distancia, especialmente en clases pequeñas o de bajo contraste como el *Monitor*. Las barras de error evidencian diferencias entre clases, destacando que *Plantas* y *Sillas* alcanzaron mayores distancias; en el primer caso, posiblemente por la riqueza visual de sus texturas y formas, y en el segundo, por su geometría estructurada y fácilmente reconocible.

4.4. Prueba de Campo

En esta sección se describe las prueba de campo diseñada para validar el desempeño de LU-SEG en un entorno real de interiores, bajo condiciones de iluminación variables y distintos tipos de mobiliario. El objetivo principal es observar el comportamiento del sistema de segmentación y tele-operación fuera de un entorno controlado, tanto la detección de objetos a diferentes distancias como la capacidad de grabación y transmisión de datos RGB-D.

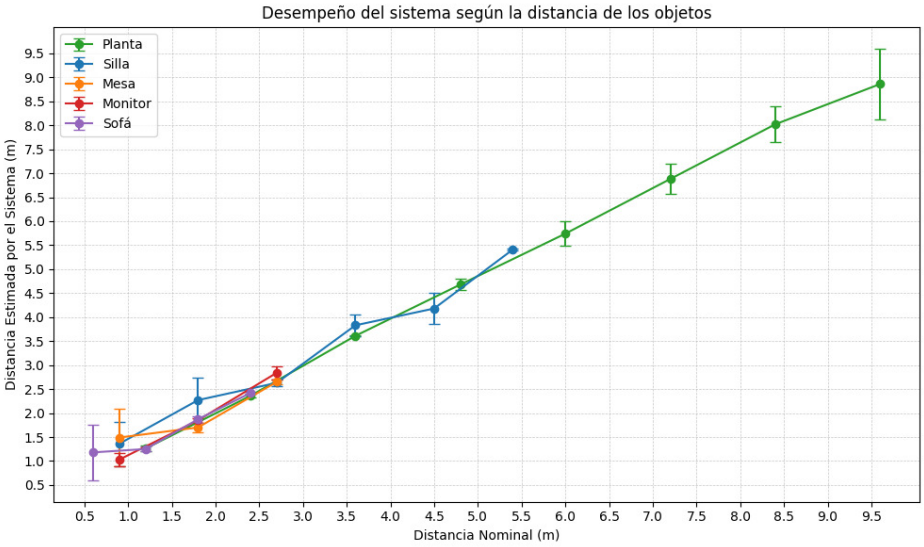


Figura 4.2: Desempeño por clase según distancia relativa.

La prueba de campo se realizó el 5 de junio de 2025 entre las 6:15 pm y 6:45 pm, con el estudiante Harold Riascos como operador. El recorrido inició al atardecer, con iluminación natural en descenso, y posteriormente bajo luz artificial proveniente de las lamparas. El trayecto abarcó varios puntos de la Escuela de Ingeniería Eléctrica y Electrónica de la Universidad del Valle. El robot Pioneer 3DX se ubico al principio del pie de las escaleras del Edificio E53 (junto al salón 2016), la cámara estéreo ZED2 del robot se ubico a 70 cm de altura del piso; progreso el recorrido por el pasillo hacia el salón 2033 (E52) con alta presencia de macetas con plantas, las oficinas de los profesores de GICI con mobiliario típico de oficina, sillas, mesas, computadores; luego se paso a la zona de estudio del mismo edificio con mesas y sillas metálicas, y finalizando en el Laboratorio de Visión Artificial, un espacio de trabajo equipado densamente con computadores, mesas y sillas (ver Figura 4.3).

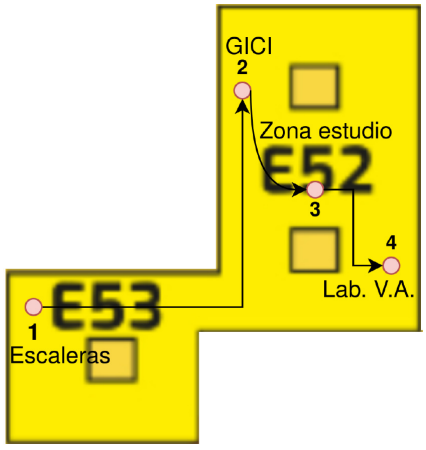


Figura 4.3: Plano del recorrido empleado

Se evaluaron manualmente **70 frames RGB-D**, seleccionados de dos grabaciones consecutivas, para analizar la coherencia de las máscaras, las métricas de segmentación y el rendimiento general del sistema. La Figura 4.4 presenta la matriz de confusión obtenida a partir de la prueba de campo, mientras que la Tabla 4.14 desglosa los valores de verdaderos positivos (TP), falsos positivos (FP), falsos negativos (FN) y verdaderos negativos (TN) por clase.

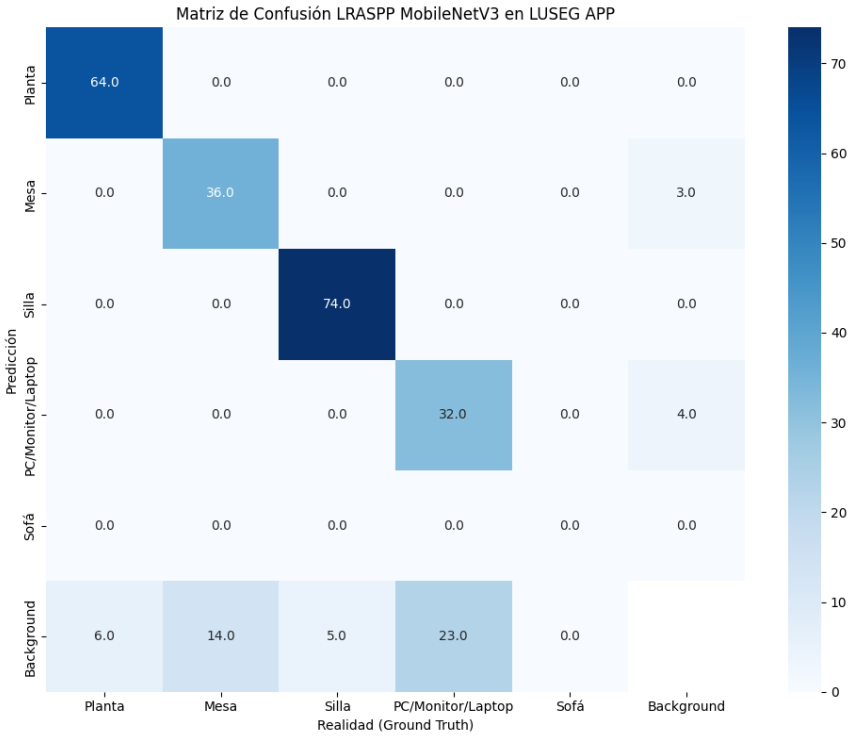


Figura 4.4: Matriz de Confusión para la prueba de campo

Clase	Regiones Validas (GT)	TP (Aciertos)	FP	FN	TN
Planta	70	64	0	6	191
Mesa	50	36	3	14	208
Silla	79	74	0	5	182
PC/Monitor/Laptop	55	32	4	23	202
Sofá	0	0	0	0	261

Tabla 4.14: Valores de TP, FP, FN y TN para cada clase, a partir de la matriz de confusión de la prueba de campo. El total corresponde a 254 objetos reales más 7 casos de fondo predichos como alguna clase (FP_{fondo}), para un total de 261 regiones evaluadas.

A partir de estos resultados se calcularon las métricas de *precisión* y *recall* (sensibilidad) mostradas en la Tabla 4.15. Los resultados muestran un desempeño alto en las clases *silla* y *planta*, con valores de precisión de 1.0 y sensibilidad superiores al 0.9, lo que evidencia una detección estable y coherente incluso en condiciones de iluminación variables. En contraste, la clase

PC/Monitor/Laptop presenta un descenso en sensibilidad (0.58), atribuible a la similitud cromática de los monitores con el entorno y a reflejos especulares en sus superficies. En general, LU-SEG alcanzó una precisión promedio del 95.3 % y una sensibilidad promedio del 78.8 %, evidenciando valores parecidos a las pruebas cuantitativas de precisión 4.3.1 donde se evaluó una mayor proporción de muestras.

Clase	**Precisión**	**Recall (sensibilidad)**
Planta	1.0000	0.9143
Mesa	0.9231	0.7200
Silla	1.0000	0.9367
PC/Monitor/Laptop	0.8889	0.5818
Sofá	*Nan*	*Nan*
Promedio Total	**0.9530**	**0.7882**

Tabla 4.15: Precisión y sensibilidad de la prueba de campo.

Duración y limitaciones de la grabación: debido a restricciones de espacio en los recursos de almacenamiento, la validación se dividió en dos sesiones. La primera grabación cubrió el trayecto desde las escaleras del edificio E53 hasta la zona de estudio del edificio E52, con una duración de 2 min 39 s y un tamaño de 12.7 GB (38 frames evaluados). Posteriormente, el archivo fue transferido para liberar espacio, y se realizó una segunda grabación en el Laboratorio de Visión Artificial, con una duración de 2 min 59 s y un tamaño de 13.9 GB (32 frames evaluados). Ambos intervalos fueron suficientes para analizar la consistencia del modelo y la estabilidad de la publicación de estadísticas y distancias, así como la integridad de las máscaras segmentadas sobre imágenes RGB-D.

Análisis de sensibilidad y comportamiento ante el entorno: la prueba de campo confirmó que la diversidad de mobiliario no afectó de manera significativa el desempeño general de LU-SEG. Las clases "silla" y "planta" mantuvieron una alta estabilidad en la detección, evidenciando una buena respuesta ante variaciones de iluminación y entorno. En contraste, la clase "PC/Monitor/Laptop" presentó una menor sensibilidad, consecuencia de tener menor características cromáticas y a la variabilidad geométrica resultante de la unificación de tres objetos con formas ligeramente distintas. Estas condiciones reducen la capacidad del modelo para recuperar los contornos en vistas oblicuas o con oclusiones parciales. En conjunto, la precisión promedio alcanzó el 95.3 % y la sensibilidad el 78.8 %, reflejando en general, un desempeño robusto frente a variaciones de textura y geometría de los objetos.

Velocidad y desempeño en movimiento: la velocidad de desplazamiento del robot no afectó el rendimiento del algoritmo, ya que el sistema de tele-operación limita la velocidad lineal a un rango de 0.0 a 0.6 puntos y la rotacional de 0.0 a 0.9 puntos, para la seguridad y estabilidad del robot. Dado que LU-SEG presenta una inferencia rápida, las velocidades de movimiento y procesamiento no llegan a interferir entre sí. No obstante, no se evaluó empíricamente el punto a partir del cual una mayor velocidad podría afectar la detección.

Las imágenes de la prueba de campo (ver Figuras 4.5 y 4.6) evidencian que LU-SEG se integra correctamente con el robot Pioneer 3DX y el sensor ZED2, transmitiendo en línea tanto las vistas RGB como las máscaras de segmentación y las estimaciones de distancia hacia la interfaz gráfica. Durante el recorrido, el sistema mantuvo una segmentación coherente y estadísticas estables (número de regiones y distancia promedio), incluso ante la disminución progresiva de la iluminación. Solo se observó una leve degradación del detalle en áreas de bajo contraste. La interfaz permitió al operador monitorear simultáneamente la segmentación y la tele-operación, confirmando la usabilidad y fiabilidad del sistema en escenarios reales.

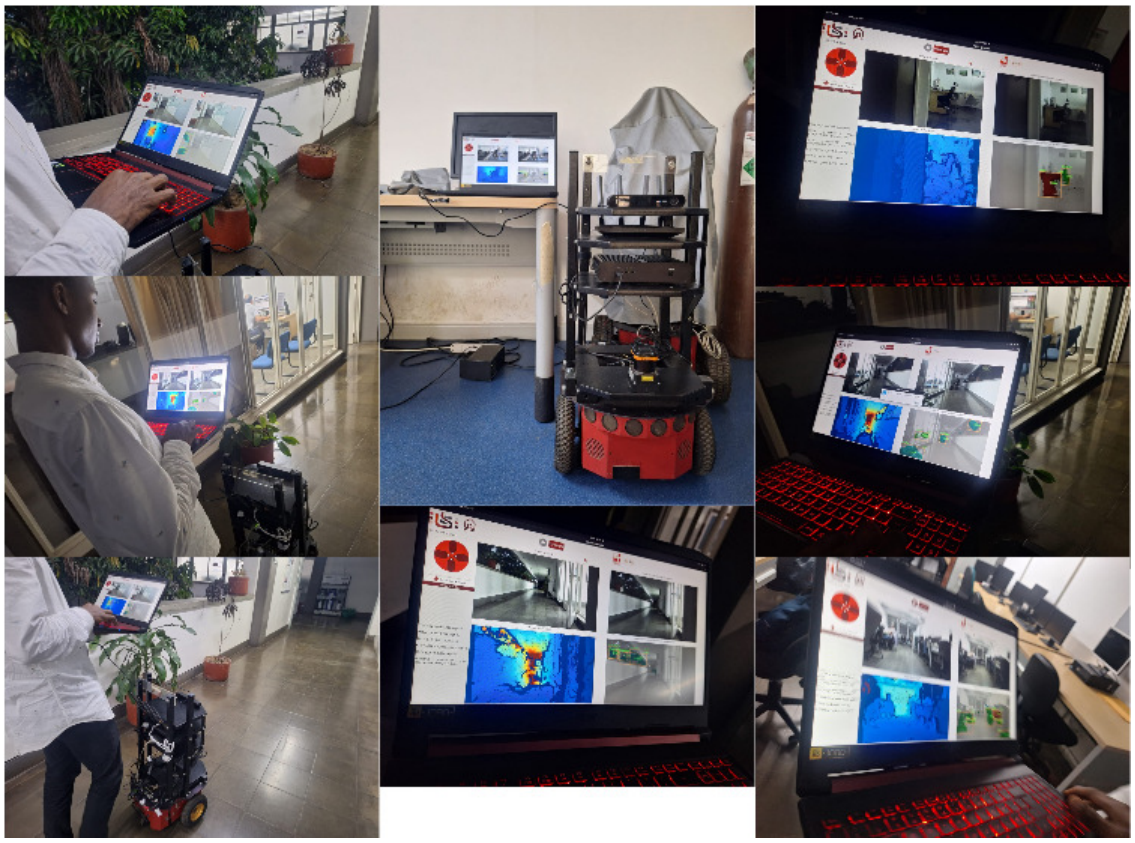


Figura 4.5: Imágenes de la Prueba de campo

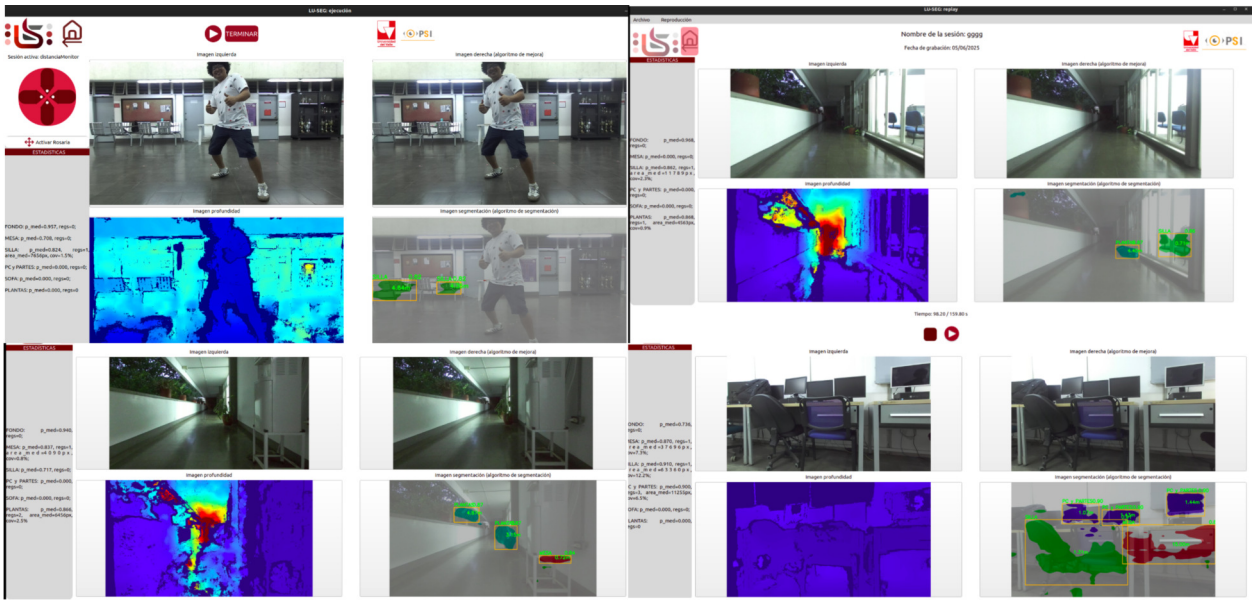


Figura 4.6: Más imágenes de la Prueba de campo

4.5. Conclusiones y limitaciones

Las pruebas de integración confirmaron que todos los módulos funcionales del sistema (configuración, ejecución, replay y backend) se comunican adecuadamente dentro del ecosistema ROS y cumplen con cada uno de los requerimientos establecidos. Esto garantiza una operación coherente y estable de la aplicación. Además, se validaron exitosamente en entornos reales tanto la segmentación en línea como la captura de datos y la teleoperación; como también se valido correctamente el guardado de parámetros de configuración, la grabación y reproducción de las sesiones realizadas.

Las pruebas cuantitativas realizadas demostraron que el sistema LU-SEG es capaz de identificar correctamente objetos de interés en entornos interiores, alcanzando una precisión promedio de 91.66 % y una sensibilidad de 79.55 %. Las clases más frecuentes como *Silla*, *Mesa* y *PC/Monitor/Laptop* presentaron altos niveles de acierto, mientras que clases menos comunes como *Sofá* evidenciaron un menor rendimiento, posiblemente debido a sus variaciones estructurales y a su forma similar a la clase *Silla*.

El análisis por distancia reveló que el sistema mantiene una segmentación efectiva dentro de rangos operativos específicos que varían según el tamaño, forma y contraste de cada clase. Por ejemplo, *Sillas* y *Plantas* fueron detectadas hasta 4.5 m y 8.4 m respectivamente, gracias a su estructura visual bien definida o riqueza de características. En cambio, objetos más pequeños, planos o demasiado grandes, como el *Monitor* ó *Sofá*, presentaron dificultades más allá de los 2.7 m y 2.4 respectivamente, y el rango de distancia que puede manejar el sistema es pequeño. Los errores promedio en estos rangos se mantuvieron por debajo de los 30 cm.

Las pruebas de campo evidenciaron la robustez general del sistema en condiciones reales, incluyendo iluminación variable, obstáculos visuales y diversidad de entornos. Aunque el sistema mantuvo un buen desempeño general, se observó que su capacidad para detectar objetos distantes disminuye en condiciones de baja luz o con oclusiones parciales, afectando la confiabilidad y sensibilidad de la segmentación y las estadísticas generadas. Los resultados obtenidos, precisión promedio del 95.3 % y la sensibilidad del 78.8 % son coherentes a los resultados de las pruebas cuantitativas de precisión. La ubicación de la cámara ZED2 a 70 cm sobre el suelo, junto con el desplazamiento estable del robot Pioneer 3DX, resultaron adecuadas para la sesión y favorecieron la adquisición de datos durante toda la prueba.

Entre las principales limitaciones identificadas se encuentran la dependencia de una buena iluminación para el rendimiento óptimo del modelo, la pérdida de precisión en objetos lejanos o poco contrastantes, y la necesidad de entornos amplios para ciertas pruebas. Asimismo, la grabación de datos en tiempo continuo estuvo restringida por el espacio de almacenamiento, lo que limitó la duración del grabado de la sesión. No obstante, los resultados obtenidos validan la viabilidad del sistema LU-SEG como herramienta para la segmentación semántica y la experimentación en robótica en interiores.

Capítulo 5

CONCLUSIONES

5.1. Conclusiones Generales

En el **Capítulo 1** se definió como pregunta problema el desarrollo de un aplicativo en ROS para segmentar objetos de interés en interiores mediante un robot móvil. Este planteamiento guió la formulación de cuatro objetivos específicos orientados a sintetizar antecedes y conocimiento teórico, diseñar e implementar un sistema funcional y validarlo tanto cuantitativa como experimentalmente en condiciones reales.

El **Capítulo 2** cumplió con el primer objetivo específico, al realizar una revisión detallada de los métodos de segmentación, tipos de sensores, métricas de evaluación, conjuntos de datos empleados en entornos interiores, metodologías de desarrollo, herramientas de trabajo, técnicas de mejoramiento de imagen para visión artificial y modelos de aprendizaje profundo para la segmentación de imágenes. Esta revisión permitió fundamentar técnicamente la elección de los modelos YOLOv11-seg, Mask R-CNN, ESANet-RGB y L-RASSP MobileNetV3, así como definir los requerimientos funcionales y no funcionales del sistema a construir.

A partir de esta base conceptual, en el **Capítulo 3** se establecieron los requerimientos más críticos para LU-SEG antes de iniciar la implementación (objetivos 2 y 3). En la sección *Home*, se definió un panel principal con tres botones que conducen a los módulos de Configuración, Ejecución y Replay, que constituye el primer requerimiento. En *Configuración*, la interfaz incluye campos de texto para los parámetros de comunicación con el robot, dirección IP, usuario, contraseña, URI del maestro y puerto, un botón "Conectar" que habilita la conexión y muestra su estado de conectividad; campos para ajustar velocidad lineal y angular y para personalizar parámetros del algoritmo de segmentación; por último, botones para guardar, cargar y nombrar sesiones en la base de datos, representando los requerimientos del 2 a 12.

En el módulo de *Ejecución* los requerimientos más importantes son el 15 al 19 y se acopla con los requerimientos del backend destacando los requerimientos BR1 y BR2. En este modulo se define incorporar la teleoperación mediante cuatro botones de movimiento (adelante, atrás,

giro izquierdo, giro derecho) y la segmentación en línea de al menos cinco clases (silla, mesa, planta, laptop/computador, sofá), junto con la visualización de estadísticas de detección, lista de objetos etiquetados y distancias relativas. El backend se encarga de adquirir y publicar datos RGB-D y de enviar comandos de movimiento al Pioneer 3DX. Finalmente, en módulo *Replay*, la GUI muestra un listado de sesiones disponibles (nombre, fecha y hora), permite seleccionar una sesión, y habilita los botones de reproducción y pausa para revisar grabaciones de sesiones previas, cubriendo así los requerimientos 21 a 24.

Bajo esta estructura, se implementaron estos cuatro módulos funcionales (Home, Configuración, Ejecución y Replay) que permiten configurar parámetros del sistema, establecer la conexión con el robot, ejecutar la segmentación en linea y gestionar sesiones de grabación. La segmentación se integró mediante nodos ROS capaces de aplicar mejoras de imagen (ecualización, realce, suavizado), ejecutar modelos semánticos sobre imágenes RGB-D (ESANet-RGB y LRASSP MobileNetV3), y publicar estadísticas por clase e imagenes anotadas, habilitando una visualización completa del entorno operativo del robot.

Durante la fase de entrenamiento y validación se evaluaron cuatro arquitecturas representativas. En segmentación de instancias, YOLOv11m-seg obtuvo un mAP_{50} de 63.2 % y $mAP_{50:95}$ de 41.4 %, mientras que Mask R-CNN alcanzó un AP_{50} de 59.8 % y $AP_{50:95}$ de 38.2 %, con un número significativamente mayor de parámetros (107 M frente a 22 M de YOLO). En segmentación semántica, ESANet-RGB logró un mIoU de 85.2 %, precisión del 91.7 % y sensibilidad del 92.1 % con 32 M de parámetros. En contraste, L-RASSP MobileNetV3 alcanzó un mIoU de 79.9 %, precisión del 88.4 % y sensibilidad del 88.8 %, con apenas 3.2 M de parámetros. Esta relación entre precisión y ligereza computacional posicionó a MobileNetV3 como la alternativa más adecuada para su despliegue en línea.

La elección de L-RASSP MobileNetV3 como modelo principal respondió a criterios de sostenibilidad computacional y precisión. Con apenas 3.2 M de parámetros logró un mIoU del 79.9 %, lo cual facilitó su ejecución fluida, alcanzando una tasa de publicación de imagen anotada de 15 Hz, también, no presentaron dificultades durante su integración con el robot Pioneer 3DX. Esta integración efectiva permitió cumplir con el objetivo de implementar el aplicativo con capacidad de operación real sobre un robot móvil.

En el **Capítulo 4**, se validó el sistema desde tres enfoques. Las pruebas de integración demostraron que LU-SEG opera correctamente dentro del ecosistema ROS, cumpliendo con todos los requerimientos funcionales establecidos. La interfaz gráfica, la grabación y reproducción de sesiones, el control del robot y la publicación de imágenes y estadísticas funcionaron de forma coordinada en cada módulo.

Las pruebas cuantitativas confirmaron la precisión del sistema, alcanzando valores del 91.66 % de precisión y 79.55 % de sensibilidad. La clase “Silla” fue segmentada con mayor estabilidad,

mientras que clases más irregulares como “Sofá” presentaron mayor variabilidad. Los resultados obtenidos durante el análisis por distancia evidenciaron que el sistema puede operar con un error medio inferior a los 30 cm dentro de los rangos efectivos por clase, con un máximo de 8.4 m para “Plantas”.

Las pruebas de campo mostraron un buen desempeño en condiciones reales, como iluminación variable, oclusiones parciales y diversidad de entornos. Se verificó la capacidad del sistema para mantener la segmentación y la publicación de estadísticas en procesamiento continuo durante un recorrido completo por distintas zonas de un entorno académico. Aunque se identificaron limitaciones asociadas a baja luz y pérdida de precisión en objetos lejanos, el desempeño general fue satisfactorio.

El cumplimiento de todos los objetivos planteados demuestra que LU-SEG es un sistema funcional, modular y capaz de realizar segmentación semántica en interiores usando un robot móvil en conjunto con ROS. La arquitectura propuesta permite futuras extensiones y ofrece una plataforma para la experimentación en visión artificial, navegación asistida y análisis del entorno en tareas robóticas.

5.2. Trabajos futuros

El *mapeo semántico* constituye una de las líneas de desarrollo más importantes derivadas del presente trabajo. A partir de los avances en segmentación aplicados a vehículos autónomos y robots móviles, la integración de modelos de segmentación en sistemas de mapeo semántico permitiría generar representaciones del entorno más ricas y comprensibles. Este tipo de mapas no solo describen la geometría del espacio, sino también el significado de los objetos y superficies que lo componen, lo que facilita que los robots comprendan su entorno y tomen decisiones informadas basadas en la semántica visual. En consecuencia, la implementación de mapeo semántico contribuiría a mejorar la navegación autónoma, optimizar la detección de obstáculos y permitir una interacción más segura e inteligente con el entorno.

Como trabajo futuro, se plantea integrar múltiples sensores (cámaras RGB y LiDAR) y extender el aplicativo a modelos de segmentación de instancias. La fusión de datos visuales y de profundidad compensaría las limitaciones de cada sensor, mejorando la precisión espacial y la robustez en condiciones de iluminación u oclusión, mientras que la segmentación de instancias permitiría distinguir objetos individuales dentro de una misma clase (p. ej., varias sillas o mesas), enriquecer la salida con conteos y distribuciones detalladas, y habilitar funcionalidades como seguimiento de objetos y estadísticas por instancia.

Finalmente, se propone integrar LU-SEG en un algoritmo de SLAM semántico centrado en objetos de interés. Las máscaras generadas actuarían como referencias en el mapa para mejorar la estimación de posición, y su fusión con los datos RGB-D optimizaría la reconstrucción 3D y

la resistencia frente a cambios del entorno, logrando una navegación y un mapeo más precisos y con mayor nivel de comprensión del escenario.

Bibliografía

[1] Mordor IntelligenceTM Industry Reports, "Autonomous mobile robot market - growth, trends, covid-19 impact, and forecasts (2023 - 2028)." https://www.mordorintelligence.com/industry-reports/autonomous-mobile-robot-market, 2023. Consultado el 7 de mayo de 2023.

[2] M. Zienkiewicz, "Rise of the machines: Robots could soon help growers make seed choices," Abril 7 2022. Consultado el 10 de mayo de 2024.

[3] MinnaLearn, "Áreas importantes de la robótica." https://courses.minnalearn.com/es/courses/emerging-technologies/robotics-and-automation/important-areas-of-robotics/. Consultado el 30 de mayo de 2023.

[4] ABB, "Abb adquiere sevensense: Ampliando su liderazgo en robótica móvil habilitada para ia de próxima generación." https://new.abb.com/news/es/detail/111398/abb-adquiere-sevensense-ampliando-su-liderazgo-en-robotica-movil-de-nueva-generacion-basada-en-ia, 11 de enero 2024. Consultado el 14 de mayo de 2024.

[5] Robotnik, "¿qué es la robótica adaptativa?." https://robotnik.eu/es/que-es-la-robotica-adaptativa/, 2023. Consultado el 30 de mayo de 2023.

[6] Robotnik, "Aplicaciones de la robótica en la medicina." https://robotnik.eu/es/aplicaciones-de-la-robotica-en-la-medicina/, 2022. Consultado el 30 de mayo de 2023.

[7] D. Seichter, P. Langer, T. Wengefeld, B. Lewandowski, D. Hochemer, and H. Gross, "Efficient and robust semantic mapping for indoor environments," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2022.

[8] Mordor IntelligenceTM Industry Reports, "mercado de visión robótica: crecimiento, tencias, impacto de covid-19 y pronósticos (2023 - 2028)." https://www.mordorintelligence.com/es/industry-reports/robotic-vision-market. Consultado el 7 de mayo de 2023.

[9] "Caso práctico de amazon robotics." https://aws.amazon.com/es/solutions/case-studies/amazon-robotics-case-study/, 2022. Consultado el 16 de abril de 2023.

[10] Alert Innovation, "Alphabot." https://www.alertinnovation.com/technologies/alphabot-asrs-system/, 2023. Consultado el 16 de abril de 2023.

[11] W. y Alert Innovation, "Walmart micro fulfillment center." https://www.alertinnovation.com/case-studies/walmart/, 2020. Consultado el 16 de abril de 2023.

[12] S. Sengupta, E. Greveson, A. Shahrokni, and P. H. S. Torr, "Urban 3d semantic modelling using stereo vision," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2013.

[13] A. Roychoudhury, M. Missura, and M. Bennewitz, "Plane segmentation using depth-dependent flood fill," in *IEEE International Conference on Intelligent Robots and Systems*, 2021.

[14] A. Paigwar, O. Erkent, D. Sierra-Gonzalez, and C. Laugier, "Gndnet: Fast ground plane estimation and point cloud segmentation for autonomous vehicles," in *IEEE International Conference on Intelligent Robots and Systems*, 2020.

[15] C. Brosque and M. Fischer, "Safety, quality, schedule, and cost impacts of ten construction robots," *Construction Robotics*, vol. 6, no. 2, pp. 163–186, 2022. Consultado el 13 de octubre de 2025.

[16] A. Alexis, "Warehouse robot momentum faces cost, roi challenges." https://www.cfodive.com/news/cost-emerges-top-barrier-adoption-warehouse-robots/723393/, 08 2024. CFO Dive. Consultado el 13 de octubre de 2025.

[17] "¿what is ros?." https://www.ros.org/. Consultado el 16 de abril de 2023.

[18] "Multiple machines." http://wiki.ros.org/ROS/Tutorials/MultipleMachines. Consultado el 3 de junio de 2024.

[19] "Tcpros." http://wiki.ros.org/ROS/TCPROS. Consultado el 3 de junio 2024.

[20] "Udpros." http://wiki.ros.org/ROS/UDPROS. Consultado el 3 de junio 2024.

[21] D. D. Dorado Castillo and J. J. Garcia Cárdenas, "Detección y seguimiento de humanos en un robot basado en procesamiento de imágenes," B.S. Thesis, Universidad de los Andes, Bogotá, Colombia, 2022.

[22] D. Tobón Collazos and V. A. Romero Cano, "Diseño y construcción de una sistema de percepción robótica para caracterización dasométrica de árboles de fuste único," B.S. Thesis, Universidad Autónoma de Occidente, Cali, Colombia, 2019.

[23] D. Paz, H. Zhang, Q. Li, H. Xiang, and H. I. Christensen, "Probabilistic semantic mapping for urban autonomous driving applications," in *IEEE International Conference on Intelligent Robots and Systems*, 2020.

[24] B. Anand, V. Barsaiyan, M. Senapati, and P. Rajalakshmi, "Quantitative comparison of lidar point cloud segmentation for autonomous vehicles," in *IEEE Vehicular Technology Conference*, 2021.

[25] D. Seichter, M. Köhler, B. Lewandowski, T. Wengefeld, and H. M. Gross, "Efficient rgb-d semantic segmentation for indoor scene analysis," in *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 13525–13531, 2021.

[26] J. Jiao, Y. Wei, Z. Jie, H. Shi, R. Lau, and T. S. Huang, "Geometry-aware distillation for indoor semantic segmentation," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019.

[27] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "Pointpillars: Fast encoders for object detection from point clouds," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 12689–12697, 2019.

[28] Y. Zhou and O. Tuzel, "Voxelnet: End-to-end learning for point cloud based 3d object detection," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2018.

[29] Y. Yan, Y. Mao, and B. Li, "Second: Sparsely embedded convolutional detection," *Sensors (Switzerland)*, vol. 18, no. 10, 2018.

[30] X. Wang, J. Yang, Z. Kang, J. Du, Z. Tao, and D. Qiao, "A category-contrastive guided-graph convolutional network approach for the semantic segmentation of point clouds," in *IEEE J Sel Top Appl Earth Obs Remote Sens*, vol. 16, 2023.

[31] M. Grinvald, F. Tombari, R. Siegwart, and J. Nieto, "Tsdf++: A multi-object formulation for dynamic object tracking and reconstruction," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2021.

[32] Y. Xing, J. Wang, X. Chen, and G. Zeng, "2.5d convolution for rgb-d semantic segmentation," in *Proceedings - International Conference on Image Processing, ICIP*, 2019.

[33] Q. Li, Y. Song, and X. Q. Jin, "Eg-pointnet: Semantic segmentation for real point cloud scenes in challenging indoor environments," in *2022 16th ICME International Conference on Complex Medical Engineering, CME 2022*, 2022.

[34] R. Khanam and M. Hussain, "Yolov11: An overview of the key architectural enhancements." https://arxiv.org/abs/2410.17725, 2024. arXiv:2410.17725 [cs.CV], version 1, October 24, 2024.

[35] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," *IEEE Transactions on Pattern Analysis and Machine Intelligence, 01 February 2020*, vol. 42, no. 2, 2018.

[36] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, Q. V. Le, and H. Adam, "Searching for mobilenetv3," 2019.

[37] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016.

[38] L. C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2018.

[39] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proceedings of the 30th IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5987–5995, IEEE, 2017.

[40] G. Neuhold, T. Ollmann, S. R. Bulo, and P. Kontschieder, "The mapillary vistas dataset for semantic understanding of street scenes," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, IEEE, 2017.

[41] R. Q. Charles, H. Su, M. Kaichun, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 77–85, 2017.

[42] S. Barchid, J. Mennesson, C. C. Djeraba, and C. Djéraba, "Review on indoor rgb-d semantic segmentation with deep convolutional neural networks chaabane chabane djeraba. review on indoor rgb-d semantic segmentation with deep convolutional neural networks. cbmi 2021-content-based multimedia indexing, review on indoor rgb-d semantic segmentation with deep convolutional neural networks," 2021.

[43] P. K. Vinodkumar, D. Karabulut, E. Avots, C. Ozcinar, and G. Anbarjafari, "A survey on deep learning based segmentation, detection and classification for 3d point clouds," *Entropy*, vol. 25, April 2023.

[44] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[45] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (VOC) challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, 2010.

[46] PyTorch Vision Team, "torchvision.models.segmentation.lraspp_mobilenet_v3_large — torchvision main documentation." `https://pytorch.org/vision/main/models/generated/torchvision.models.segmentation.lraspp_mobilenet_v3_large.html`. Consultado el 5 de mayo de 2025.

[47] B. Zhou, H. Zhao, X. Puig, T. Xiao, S. Fidler, A. Barriuso, and A. Torralba, "Semantic understanding of scenes through the ade20k dataset," *International Journal of Computer Vision*, vol. 127, no. 3, pp. 302–321, 2019.

[48] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," in *Proceedings of the 13th European Conference on Computer Vision (ECCV)*, vol. 8693 of *Lecture Notes in Computer Science*, pp. 740–755, Springer, 2014.

[49] J. Deng, W. Dong, R. Socher, L.-J. Li, and F.-F. Li, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, IEEE, 2009.

[50] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from rgbd images," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2012.

[51] S. Song, S. P. Lichtenberg, and J. Xiao, "SUN RGB-D: A RGB-D scene understanding benchmark suite," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2015.

[52] I. Armeni and et al., "3D semantic parsing of large-scale indoor spaces," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016.

[53] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman, "Labelme: A database and web-based tool for image annotation," *International Journal of Computer Vision*, vol. 77, no. 1–3, pp. 157–173, 2008.

[54] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba, "Sun database: Large-scale scene recognition from abbey to zoo," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3485–3492, 2010.

[55] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva, "Learning deep features for scene recognition using places database," in *Advances in Neural Information Processing Systems 27*, pp. 487–495, 2014.

[56] A. T. Azar and S. Vaidyanathan, *Handbook of Research on Advanced Intelligent Control Engineering and Automation*. 2015.

[57] K. Brush, "Mobile robot (mobile robotics)." `https://www.techtarget.com/iotagenda/definition/mobile-robot-mobile-robotics/`, 2019.

[58] E. Sirpa, "Robots: Métodos de locomoción terrestre." `https://medium.com/@ericksirpa/robots-m%C3%A9todos-de-locomoci%C3%B3n-terrestre-2964c68a9523/`, 2017.

[59] Mobile Robots Inc, "Pioneer3-D X." `https:www.mobilerobots.com`, Consultado el 30 de mayo de 2023.

[60] M. Ángel, "Scrum: qué es y cómo funciona este marco de trabajo." `https://www.wearemarketing.com/es/blog/metodologia-scrum-que-es-y-como-funciona.html`, 2022.

[61] A. Acharya, "Guide to Image Segmentation in Computer Vision: Best Practices." https://encord.com/blog/image-segmentation-for-computer-vision-best-practice-guide/, 2022.

[62] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, vol. 9351 of *Lecture Notes in Computer Science*, pp. 234–241, Springer, 2015.

[63] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder–decoder architecture for image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.

[64] D. Banica and C. Sminchisescu, "Second-order constrained parametric proposals and sequential search-based structured prediction for semantic segmentation in rgb-d images," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–9, IEEE, 2015.

[65] H. Hamian, M. Beikmohammadi, A. Ahmadi, and B. Nasersharif, "Semantic segmentation of autonomous driving images by the combination of deep learning and classical segmentation," in *Proceedings of the 26th International Computer Conference, Computer Society of Iran (CSICC)*, pp. 1–6, IEEE, 2021.

[66] F. Fooladgar and S. Kasaei, "Semantic segmentation of rgb-d images using 3d and local neighbouring features," in *Proceedings of the 2015 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, pp. 1–7, IEEE, 2015.

[67] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*. Pearson, 4th ed., 2018.

[68] W. Burger and M. J. Burge, *Digital Image Processing: An Algorithmic Introduction Using Java*. Springer, 2nd ed., 2016.

[69] A. L. Batallas, J. Bermeo Paucar, J. Paredes Quevedo, and H. Torres Ordoñez, "Una revisión de las métricas aplicadas en el procesamiento de imágenes," *RECIMUNDO: Revista Científica de la Investigación y el Conocimiento*, vol. 4, no. 3, pp. 267–273, 2020.

[70] J. Terven, D. Cordova-Esparza, J. Romero-González, A. Ramírez-Pedraza, and E. A. Chávez-Urbiola, "A comprehensive survey of loss functions and metrics in deep learning," *Artificial Intelligence Review*, vol. 58, 2025.

[71] R. Sapkota and M. Karkee, "Comparing yolo11 and yolov8 for instance segmentation of occluded and non-occluded immature green fruits in complex orchard environment: A technical note," *arXiv preprint arXiv:2410.19869v2*, 2024. Version v2, accessed May 2025.

[72] M. Oršić, I. Krešo, P. Bevandić, and S. Šegvić, "In defense of pre-trained imagenet architectures for real-time semantic segmentation of road-driving images," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2718–2726, 2019.

[73] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2881–2890, 2017.

[74] T. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 2999–3007, 2017.

[75] Ultralytics, "Ultralytics yolov8 documentation." `https://docs.ultralytics.com`, 2024. Última consulta: Junio 2025.

[76] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, "Detectron2." `https://github.com/facebookresearch/detectron2`, 2019.

Apéndice A

Anexos

En la figura A.1 se presenta cómo están organizados los documentos y archivos pertenecientes al desarrollo del trabajo de grado como: los documentos de la metodología SCRUM, pruebas de precisión y distancia. Se puede acceder a esta documentación a través del siguiente enlace: `https://drive.google.com/drive/folders/117sIKqWiFOCO_urFoXJWVWpZB3TfKcdc?usp=sharing`

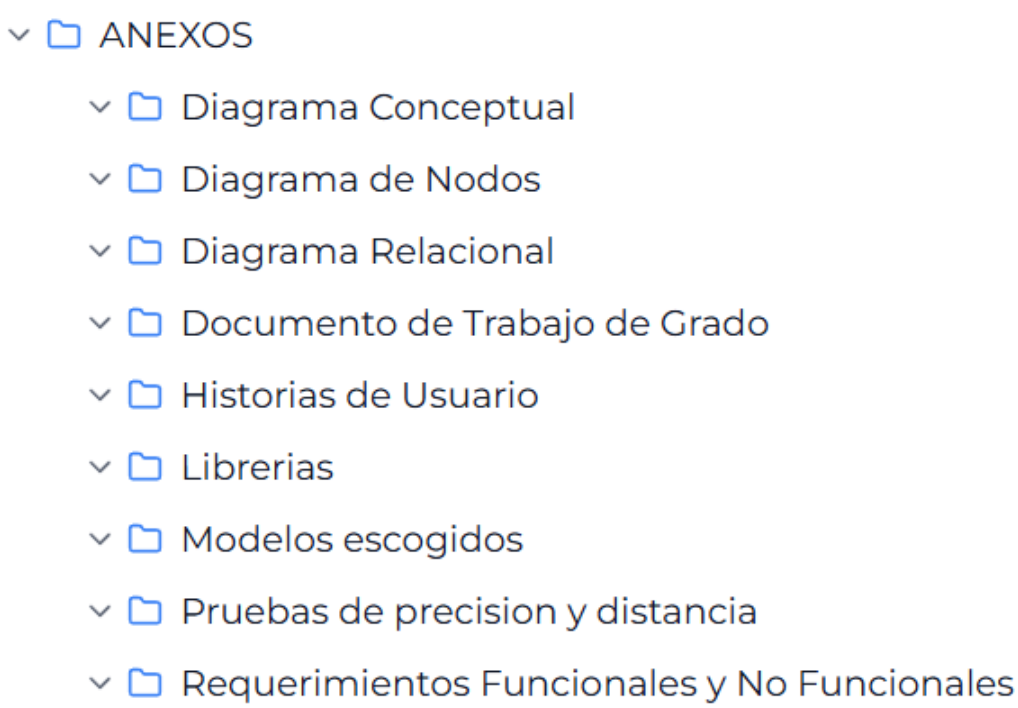


Figura A.1: Orden de los documentos en la carpeta “Anexos".

- **Diagrama Conceptual:** contiene un archivo .png con el diagrama conceptual de la aplicación.
- **Diagrama de Nodos:** contiene un archivo .png con el diagrama de nodos del sistema.

- **Diagramas Relacional:** contiene un archivo .png con el diagrama relacional de la base de datos.
- **Documento de trabajo de grado:** contiene el archivo en .pdf del trabajo de grado.
- **Historias de usuario:** contiene en un archivo de excel las historias de usuario diseñadas para el desarrollo de este aplicativo.
- **Librerias:** contiene un archivo .txt donde menciona algunas librerías complementarias que usa el aplicativo LU-SEG
- **Modelos escogidos:** contiene una carpeta con el código de entrenamiento de las redes *MobileNet v3, ESANet-RGB, YOLOv11-seg y Maskrcnn* en *Google Colab*.
- **Pruebas de precisión y distancia:** contiene un archivos excel que se usaron para calcular la matriz de confusión y los errores de distancia. Además contiene las imágenes usadas para evaluar el modelo en el aplicativo.
- **Requerimientos Funcionales y No Funcionales:** contiene 3 archivo .pdf que describe los requerimientos funcionales y no funcionales.