POSGRADO INTERINSTITUCIONAL EN CIENCIA Y TECNOLOGÍA

CENTRO DE INVESTIGACIONES EN ÓPTICA, A. C.

VISUAL SLAM PARA VEHÍCULOS AUTÓNOMOS EN ENTORNOS DINÁMICOS

TESIS

QUE PARA OBTENER EL GRADO ACADEMICO DE:

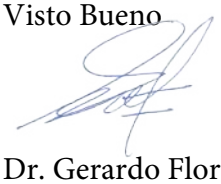
MAESTRO EN CIENCIA Y TECNOLOGÍA EN LA ESPECIALIDAD DE MECATRÓNICA

PRESENTA
ING. LUIS ANDRÉS MORENO JIMÉNEZ

DIRECTOR DE TESIS
DR. GERARDO RAMÓN FLORES COLUNGA

LEÓN DE LOS ALDAMA, GUANAJUATO,
OCTUBRE, 2024

Visto Bueno



Dr. Gerardo Flores

Visual SLAM para vehículos autónomos en entornos dinámicos

Agradecimientos

En este momento, cuando las ideas toman forma, me siento muy agradecido. Esta tesis es un reflejo del apoyo, el amor y la orientación que he recibido a lo largo de este viaje.

En primer lugar, estoy profundamente agradecido a Dios por darme sabiduría y paciencia, por estar conmigo en cada paso del camino y por hacer posible que hoy esté aquí.

Agradezco sinceramente a mi familia, que ha sido mi pilar, fuente de amor y refugio. Especialmente, a mi hermano, el Dr. Hugo, por su invaluable apoyo y amistad. A Celeste y a mis amigos de laboratorio, Hugo, Chimi, Verdín y Antonio, con quienes he compartido muchas aventuras y anécdotas que han marcado mi vida profesional y personal.

Mi profundo agradecimiento al Dr. Gerardo, por sus valiosos consejos y experta orientación, y al Consejo Nacional de Humanidades, Ciencias y Tecnologías (CONAHCYT). Su apoyo al otorgarme una beca de maestría facilitó mi trayectoria académica y confirmo mi pasión por la ciencia. Gracias a su confianza, pude perseguir mis sueños con determinación y recursos.

Gracias de todo corazón.

Publicaciones

Journal papers

- Robot Autónomo para la detección y clasificación de fresas. **L.A. Moreno**, G. Mejía, A. Montes de Oca, Gerardo Flores. TIES, Revista de Tecnología e Innovación en Educación Superior, n.o. 8 (PUBLICADO).
- EVS-SLAM: Embedded Visual SLAM for Autonomous Vehicles in Dynamic Environments. **L.A. Moreno**, Gerardo Flores. IEEE, Robotics and Automation Letters (SOMETIDO).

Resumen

Esta tesis presenta un sistema Visual SLAM dinámico de bajo coste computacional diseñado para vehículos y robots móviles en entornos exteriores y dinámicos. Este sistema se ha mejorado mediante el uso de una arquitectura DeepLabv3, una red neuronal de segmentación semántica, que facilita la detección precisa de objetos y mejora la eficiencia computacional. Esta capacidad de segmentación es clave para recopilar información sobre objetos dinámicos, lo que resulta esencial para crear mapas de datos dinámicos. El sistema propuesto tambien implementa un método para ignorar esta información en su proceso de autolocalización, minimizando así los posibles errores. Paralelamente, se ha desarrollado un método para excluir la información dinámica del sistema VSLAM, lo que se traduce en una mejora de la localización y la precisión de los mapas.

Índice general

Índice de cuadros

Índice de figuras

CAPÍTULO 1

Introducción

El desarrollo de Simultaneous Localization and Mapping (SLAM), especialmente una de sus ramas Visual SLAM (VSLAM), se ha convertido en una importante herramienta en robótica y vehículos autónomos. En el contexto actual, se buscan métodos eficientes para dar autonomía a los vehículos en entornos dinámicos y exteriores, el VSLAM ofrece una solución. Este sistema se diferencia del SLAM tradicional, que suele basarse en sensores como el GPS [6] o LIDAR [7]. El VSLAM utiliza cámaras que sustituyen o complementan a estos sensores para realizar la autolocalización y el mapeo [8]. Esto le da ventajas únicas en entornos con mucha información visual.

Por otra parte, la gestión de elementos dinámicos en entornos dinámicos presenta desafíos que aún están siendo abordados por la comunidad científica [9]. La dificultad para diferenciar y procesar adecuadamente los objetos en movimiento supone una barrera para la precisión y la confianza de los sistemas VSLAM en aplicaciones prácticas, desde la navegación autónoma hasta la interacción robótica en espacios compartidos con humanos.

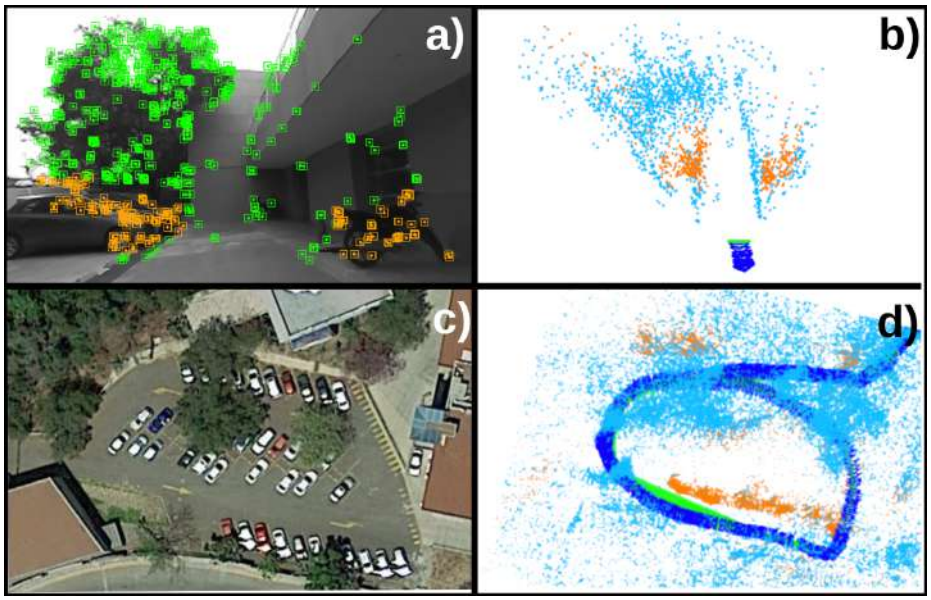


Figura 1.1: Resultados de las pruebas del VSLAM propuesto. a) Muestra la diferenciación de los puntos característicos para objetos dinámicos (naranja) y estáticos (verde). b) Muestra la nube de puntos completa de la zona vista en a). c) Muestra una vista aérea representativa de la zona mapeada, tomada en un momento distinto al de la exploración, con la trayectoria realizada. d) Muestra la trayectoria realizada y la nube de puntos correspondiente a la zona de estacionamiento.

En este marco, se propone un sistema VSLAM dinámico y eficiente que integra segmentación semantica utilizando una arquitectura DeepLabv3 modificada, que no sólo facilita la detección precisa de objetos dinámicos, sino que también optimiza la gestión de los recursos computacionales, abordando así uno de los principales retos en este campo. El objetivo de este trabajo es explorar y demostrar cómo la integración de la segmentación semántica diseñada para sistemas computacionales embebidos dentro de un marco VSLAM puede abordar los retos de los entornos dinámicos manteniendo un perfil de bajo coste computacional. La idea central es el equilibrio entre precisión y eficiencia, que es importante para aplicaciones prácticas en vehículos autónomos y robots móviles.

1.1. Antecedentes

1.1.1. Evolución del SLAM visual

En las dos últimas décadas, el campo del SLAM ha experimentado un gran desarrollo e interés, impulsado por el avance de algoritmos más eficientes que han aprovechado el desarrollo y evolución de los sensores para mejorar significativamente la estimación de la pose del robot y el mapeo en dos y tres dimensiones [10, 11]. Aunque el mapeo en dos dimensiones, a menudo realizada con sensores de rango como el LiDAR, se considera ahora resuelta por algunos en la comunidad [12], siguen existiendo retos importantes. Entre ellos incluyen cuestiones de precisión, aplicabilidad, rendimiento en términos de velocidad y capacidad para procesar grandes volúmenes de datos, especialmente en entornos dinámicos al aire libre donde las condiciones son inestables.

La auto-localización y el mapeo, componentes clave del SLAM, emplean técnicas que se dividen en dos grupos: probabilísticos y no probabilísticos. La mayoría de los enfoques actuales se basan en métodos probabilísticos, utilizando la estimación bayesiana y destacando el uso de Filtros de Kalman (FK) [13] para procesar la información obtenida a través de los sensores, un ejemplo importante es [14]. Aunque se trata de un modelo estadístico, hoy en día es una de las herramientas más óptimas para la estimación de valores, además de su simplicidad en la implementación. Sin embargo, se enfrenta a limitaciones como la falta de cierre de bucle y retos en sistemas no lineales. Para superar estas limitaciones, se han adoptado filtros como Unscented Kalman Filter (UKF) y extendido de Kalman (EKF) [15–17], que permiten la linealización de sistemas no lineales y se han convertido en una solución estándar para implementaciones de SLAM. La Figura 1.2 intenta ilustrar el proceso SLAM, en el que un modelo de movimiento actualiza la pose del robot utilizando datos de odometría o velocidad. La información de los sensores se integra mediante un modelo de observación, que refina esta estimación junto con un bloque de fusión. El filtrado compara los puntos de referencia con las observaciones actuales para corregir la posición del robot y actualizar el mapa con nuevas características, mejorando continuamente la precisión del mapa y la localización del robot.

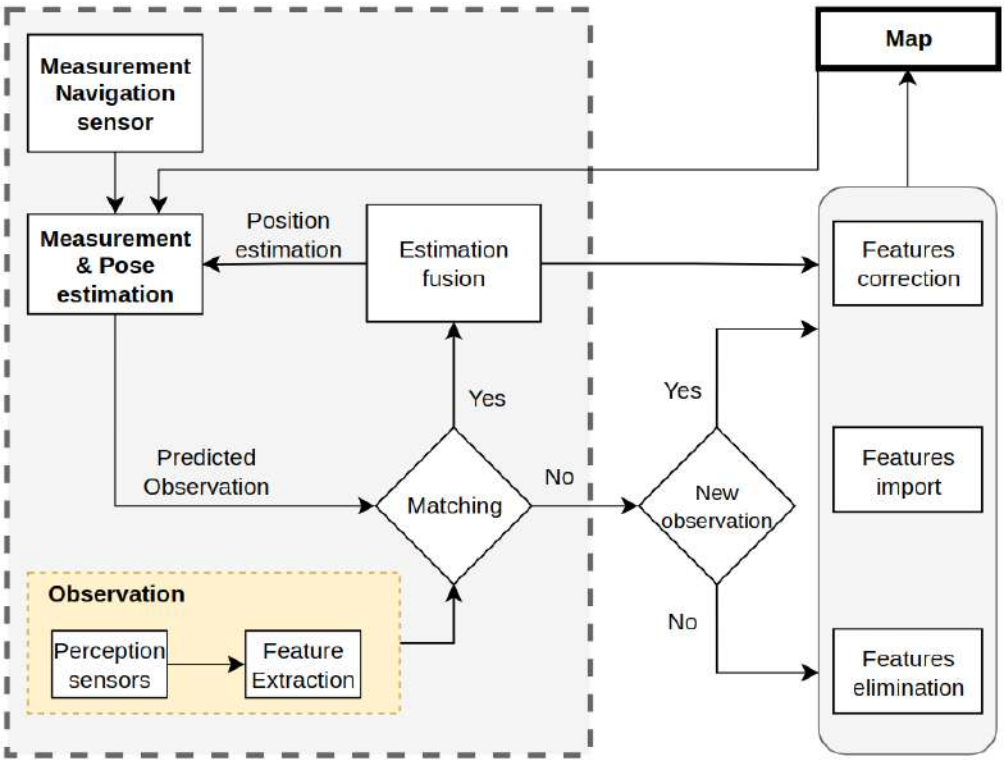


Figura 1.2: Diagrama de la estructura general de los sistemas SLAM.

El SLAM tiene dos raíces principales: SLAM basado en LiDAR y SLAM basado en visión por ordenador. Mientras que el SLAM basado en LiDAR ha sido ampliamente implementado en robots inteligentes y productos comerciales, produciendo mapas simples que a menudo limitan la interpretación del entorno por parte del robot, el SLAM visual se ha desarrollado para superar estas limitaciones. Mediante técnicas de visión por computadora, este sistema es capaz de reconstruir mapas 3D detallados utilizando cámaras, lo que proporciona una gran cantidad de información visual. Este enfoque se beneficia de la mejora de los sensores visuales, lo que permite una variedad más amplia de aplicaciones y una implementación robusta, precisa y relativamente sencilla. La Figura1.3 muestra los componentes clave del Visual SLAM, que consiste en el seguimiento, el mapeo, la optimización global y la relocalización. El visual SLAM se clasifica principalmente en monocular, RGB-D y estereoscópico. El uso de una cámara para el problema se denomina SLAM monocular [18], la combinación de un sensor de profundidad y una cámara monocular se denomina SLAM RGB-D [19], y el uso de dos cámaras se denomina SLAM estereoscópico [20].

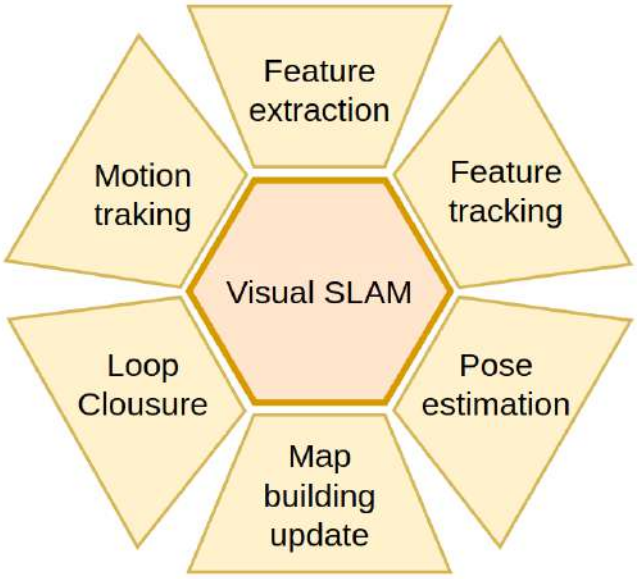


Figura 1.3: Ilustración que muestra los componentes clave del sistema Visual SLAM

1.1.2. Inteligencia Artificial en SLAM

La motivación de este trabajo surge de observar las posibilidades del Deep Learning [21] en el contexto del SLAM [22], destacando cómo el Deep Learning puede revolucionar el campo del Visual SLAM proporcionando soluciones innovadoras a los retos tradicionales. La intersección del aprendizaje profundo con el Visual SLAM se divide en áreas clave como la estimación de profundidad, correspondencia de características, ajuste de paquete (Bundle Adjustment), la segmentación semántica, la estimación de la pose de la Cámara y el flujo óptico, cada una ofrece oportunidades únicas para avanzar en la tecnología SLAM, como se muestra en la Figura 1.4.

El aprendizaje profundo ofrece varias oportunidades para mejorar y optimizar los componentes del Visual SLAM. A continuación, se enumeran las áreas clave que se ven afectadas por el Deep Learning:

- **Correspondencia de características:** Las técnicas de correspondencia de características basadas en CNN han demostrado resultados de vanguardia en varios conjuntos de datos [23]. Sin embargo, el aprendizaje de correspondencias de características óptimas para etapas posteriores, como el Bundle Adjustment sigue siendo un problema abierto.

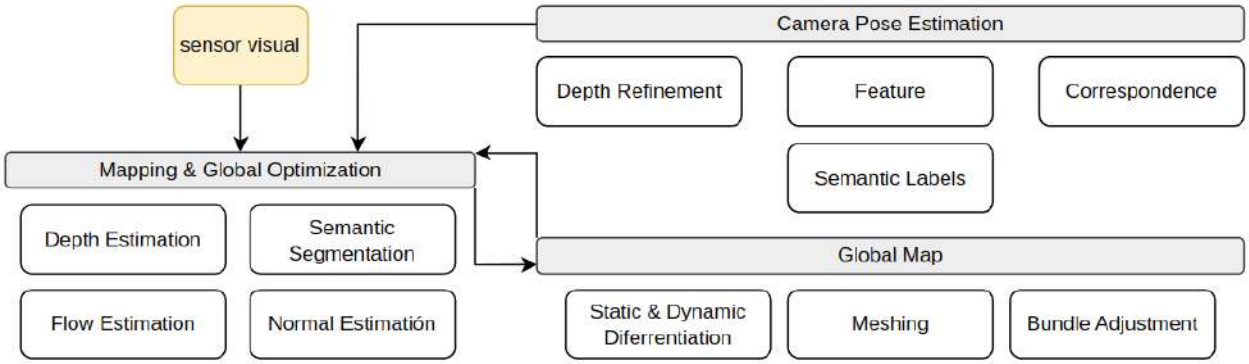


Figura 1.4: Diagrama áreas derivadas del Deep Learning y oportunidades para SLAM.

- **Bundle adjustment:** Aunque todavía no existe una solución madura para el Bundle adjustment basado en CNN, hay intentos iniciales que buscan modelar restricciones de proyección de forma diferenciable [24], lo que abre una nueva area de investigación para la optimización del SLAM.

- **Estimación de la pose de la cámara:** La localización precisa dentro del mapa, descrita por la pose de la cámara, es fundamental para el éxito del SLAM. El Deep Learning ha permitido el desarrollo de métodos que pueden recuperar eficientemente la pose de la cámara de utilizando pipelines basados en características [25].

- **Segmentación Semántica:** La capacidad de segmentar la imagen en partes semánticamente significativas es esencial para comprender el entorno, con aplicaciones que van desde la robótica hasta la conducción automatizada. El aprendizaje profundo ha mejorado significativamente la precisión y eficiencia de la segmentación semántica [26].

1.1.3. Integración de la inteligencia artificial con VSLAM

La combinación del Deep Learning y Visual SLAM ha transformado la forma en que los robots autónomos, desde cuadrúpedos y drones hasta vehículos exploradores, que interactúan con su entorno y lo comprenden, ver 1.5. La capacidad de generar mapas detallados de entornos desconocidos y determinar su propia posición dentro de esos mapas es fundamental para la navegación autónoma [27]. Sin embargo, la dinámica y comple-

jidad de los entornos dinámicos plantean importantes retos, especialmente a la hora de distinguir rápidamente entre elementos estáticos y dinámicos.

Figura 1.5: Las plataformas autónomas: un robot cuadrúpedo [1], un vehículo terrestre [2], y un dron [3], todos equipados con sensores visuales, están demostrando ser plataformas óptimas para explotar el Visual SLAM.

Aunque existen sistemas notables como DynaSLAM [28] y DS-SLAM [29] que han mejorado el rendimiento del Visual SLAM en entornos dinámicos, siguen enfrentándose a retos en diversas condiciones, como los elevados requisitos computacionales, la baja latencia, la presencia de objetos multiescala y la pérdida de información. Estos retos afectan a la precisión del mapeo y la localización. También requieren dispositivos pesados con gran potencia de cálculo, lo que limita su aplicación en entornos dinámicos.

1.1.4. Sistemas de Navegación Autónoma

Continuando con este reto, la integración de Deep Learning con Visual SLAM ha demostrado tener varias áreas de crecimiento, una de las cuales es el diseño de arquitecturas para las redes neuronales más ligeras. Entre los ejemplos más destacandos se encuentra la red ResNet [30] que destaca por su estructura simplificada que facilita el aprendizaje de características. ResNet18 es una configuración de esta red que tiene 18 capas, incluidas cuatro capas convolucionales en cada módulo, conectadas por una capa convolucional inicial y una capa completamente conectada al final. Esta configuración permite a ResNet18 aprender eficientemente de los datos sin el sobreajuste habitual de modelos más profundos. La Figura 1.6 ilustra la arquitectura de ResNet18 [31], destacando la secuencia de operaciones desde una convolución inicial de 7x7, pasando por la normalización

por lotes y la agrupación máxima de 3x3, hasta las capas convolucionales que definen su capacidad de aprendizaje distintiva. Configurando el número de canales y bloques restantes, se pueden desarrollar variantes de la ResNet, como el más profundo ResNet152 [32], para satisfacer las necesidades específicas de las aplicaciones de visión por computadora.

La red YOLO [33] y Mask R-CNN [34] también han contribuido significativamente a este campo, YOLO permite la detección rápida de objetos y Mask R-CNN la segmentación detallada en entornos dinámicos. Aunque estos enfoques siguen requiriendo importantes recursos computacionales y se enfrentan a problemas de precisión en aplicaciones de exteriores, la optimización de las redes de segmentación ha demostrado mejoras significativas en la eficiencia computacional, lo que permite aplicaciones más viables en sistemas embebidos compactos. Un ejemplo destacado es el uso de YOLO en drones, que, aunque inicialmente se limitaba a entornos interiores, demuestra el potencial de la segmentación para la identificación de objetos en tiempo real [35].

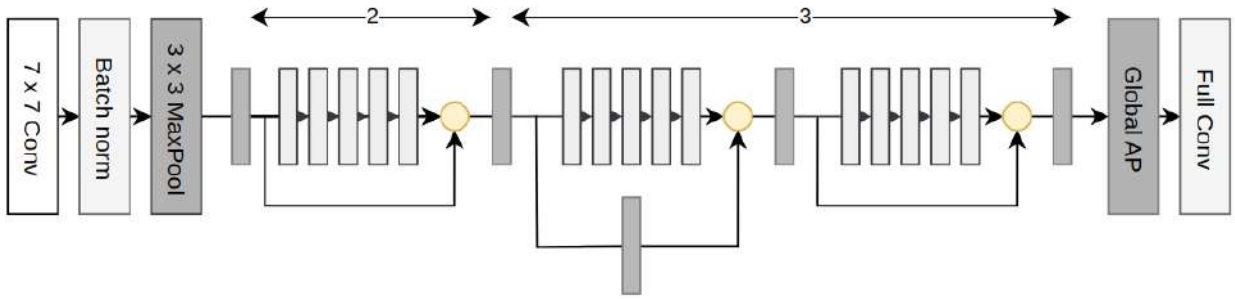


Figura 1.6: Arquitectura de ResNet18

Este trabajo se centra en superar estas limitaciones mediante el desarrollo de un sistema Visual SLAM integrado con una red de segmentación semántica optimizada diseñada específicamente para un sistema que pueda ser soportado por vehículos autónomos como drones, vehículos terrestres y robots cuadrúpedos. Nuestro enfoque busca adaptar estas tecnologías avanzadas para su implementación efectiva en la plataforma Jetson Xavier mediante la optimización de la arquitectura embebida para lograr un equilibrio entre precisión, eficiencia computacional y adaptabilidad en entornos dinámicos.

1.2. Definición del problema

La localización y el mapeo mediante sistemas Visual SLAM en entornos dinámicos, así como la integración de Deep Learning para la segmentación semántica, suponen un reto debido a los elevados recursos computacionales necesarios. Estos requisitos limitan la implementación de estos sistemas en computadoras embebidas, como la Jetson Xavier. El análisis anterior muestra, que si bien el Deep Learning promete ser una solución en ambientes dinámicos, su aplicación práctica en ambientes de exterior y en robots móviles se enfrenta a importantes limitaciones.

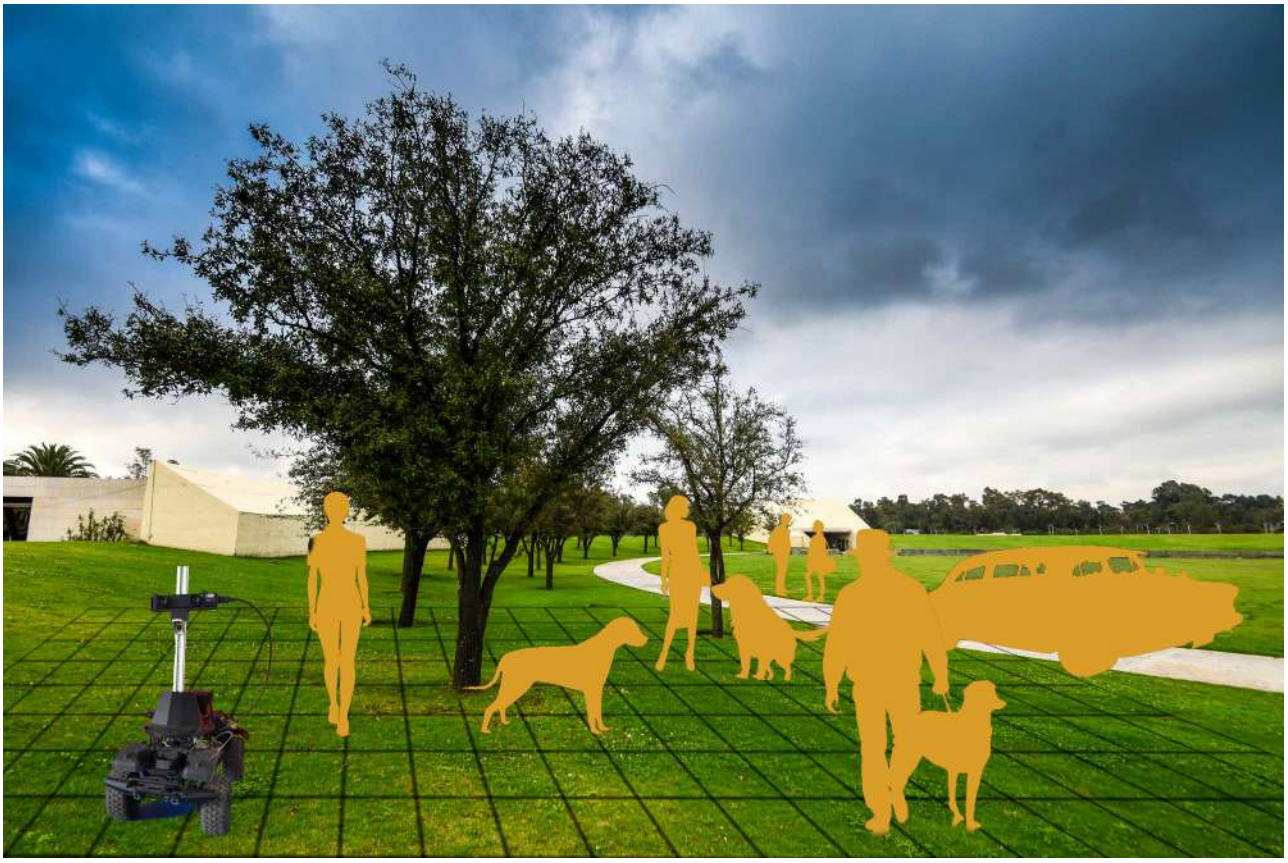


Figura 1.7: Representación de la plataforma de pruebas operando en un parque. Los objetos dinámicos (personas, mascotas, y un vehículo) identificados por el sistema aparecen resaltados en color naranja.

1.2.1. Planteamiento del problema

La principal limitación en la aplicación de estos sistemas radica en su capacidad para procesar eficientemente la información en entornos exteriores sin comprometer la precisión ni aumentar el consumo de recursos computacionales. En particular, la tarea de clasificar entre elementos estáticos y dinámicos en tiempo real, que resulta esencial para la localización y el mapeo en robots móviles, requiere una solución que mejore y equilibre estas partes con un enfoque en la aplicación. Además de la necesidad de adaptar estos sistemas a una plataforma embebida como la Jetson Xavier. Por tanto, se propone explorar métodos innovadores que permitan una segmentación semántica precisa y un procesamiento dinámico del Visual SLAM dentro de los límites de su capacidad computacional para trabajar en tiempo real.

1.3. Justificación y objetivos de la tesis

1.3.1. Justificación

La mejora en sistemas Visual SLAM es importante para ampliar las aplicaciones y la eficacia de los vehículos y robots autónomos en entornos dinámicos y al aire libre. Este trabajo beneficiará a campos como la exploración, la recopilación de datos y los sistemas autónomos.

1.3.2. Objetivo principal

Desarrollar e implementar un sistema Visual SLAM dinámico que integre capacidades de Deep Learning con segmentación semantica. Este sistema se diseñará para ejecutarse a 13-21 Hz en un computador embebido Jetson Xavier, utilizando ROS (Robot Operating System) [36] como marco de trabajo, el sistema será modular y capaz de implementarse en vehículos no tripulados como vehículos terrestres, drones y un robot cuadrúpedo, dándoles la capacidad de mapear el entorno y determinar su ubicación dentro del mapa generado en entornos dinámicos.

1.3.3. Objetivos específicos

- Analizar algoritmos VSLAM del estado del arte e implementarlos en un computador portátil, teniendo en cuenta las diferencias y ventajas de las configuraciones de cámara como RGB-D, estéreo y monocular.
- Investigar arquitecturas de Deep Learning para la segmentación de objetos.
- Implementar y optimizar el algoritmo de segmentación semántica mejor puntuado para que funcione en la Jetson Xavier.
- Implementar y optimizar el algoritmo VSLAM en la computadora Jetson Xavier.
- Adaptar los algoritmos de segmentación y VSLAM para trabajar con ROS.
- Simular el vehículo y su controlador (PX4) para asegurar la comunicación con ROS.
- Integrar el algoritmo de segmentación, VSLAM y simulación para realizar pruebas preliminares.
- Implementar el sistema en un UGV y posteriormente en un dron.
- Desarrollar e implementar el sistema VSLAM en un robot cuadrúpedo para evaluar su rendimiento y adaptabilidad en entornos dinámicos.

1.4. Hipótesis

La optimización de algoritmos VSLAM existentes, combinada con la incorporación de una red neuronal diseñada especificamente para la segmentación de objetos dinámicos y utilizando el framework ROS, permitirá una reducción significativa en el costo computacional asociado a sistemas de visual SLAM dinámicos. Esta mejora permitirá su integración en dispositivos compactos como la Jetson Xavier, mejorando el procesamiento de algoritmos y su aplicación en vehículos no tripulados compactos. Se espera que esta mejora incremente la autonomía de dichos vehículos, liberándolos de la dependencia de sistemas de localización externos, como el GPS, y permitiendo una mejor operación en entornos dinámicos.

CAPÍTULO 2

Marco teórico

2.1. Visual SLAM

Entre los enfoques Visual SLAM, ORB-SLAM [4,37,38] destaca su capacidad para realizar el seguimiento, el mapeo, la relocalización y el cierre de bucles utilizando únicamente sensores visuales. Este sistema emplea características ORB (Oriented FAST and Rotated BRIEF) [39, 40] que facilitan un mapeado detallado y una localización precisa con una dependencia mínima de sensores externos. La Figura 2.1 ilustra la arquitectura general del sistema ORB-SLAM.

Componentes clave de ORB-SLAM

- **Seguimiento:** ORB-SLAM estima la posición del dispositivo detectando y asociando características ORB en secuencias de imágenes, proporcionando una localización continua y precisa.
- **Mapeo Local:** El mapeo local optimiza los puntos clave en un mapa coherente, manteniendo la integridad del mapa para una navegación precisa.

- **Cierre de Bucle:** Identifica y corrige la deriva acumulada para garantizar la coherencia del mapa a lo largo del tiempo, una característica crítica para la fiabilidad del mapa.
- **Relocalización:** Incluye un sistema de reubicación que permite al robot recuperar su trayectoria y posición dentro del mapa tras interrupciones o cuando se inicia en entornos desconocidos.

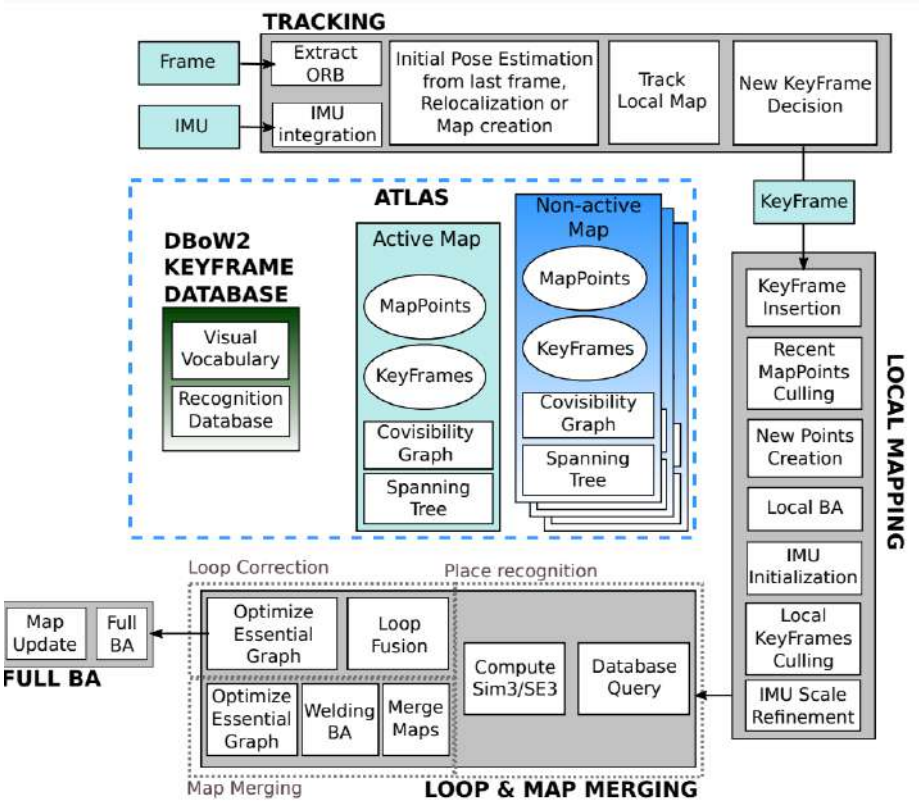


Figura 2.1: Estructura general de ORBSLAM [4].

Aunque ORB-SLAM fue diseñado originalmente para entornos estáticos, su adaptabilidad y flexibilidad lo hacen adecuado para entornos dinámicos. La integración de técnicas de segmentación semántica y redes neuronales ha permitido a ORB-SLAM distinguir y manejar elementos dinámicos del entorno. Esta adaptación destaca el potencial de ORB-SLAM como herramienta versátil para la navegación autónoma en diversos escenarios, desde interiores estructurados hasta entornos urbanos complejos.

2.2. Inteligencia Artificial en Segmentación Semántica

En el campo de la inteligencia artificial aplicada a visión por computadora, el Deep Learning en segmentación semántica juega un papel importante en la interpretación del entorno visual. Este tipo de redes están diseñadas para identificar y clasificar patrones u objetos dentro de imágenes, asignando una etiqueta específica a cada píxel en función de la categoría del objeto al que pertenece. Entre todas las arquitecturas en este campo, DeepLabv3 [41] y LRASPP_MobileNetV3_Large [42]son soluciones muy eficientes, ofreciendo un equilibrio entre precisión de segmentación y eficiencia computacional.

2.2.1. Deeplabv3

Se trata de una arquitectura de segmentación semántica que ha sido mejorada a través de varias iteraciones DeepLab, DeepLabv2, para resolver el problema de la segmentación de objetos a múltiples escalas. Esta red introduce innovaciones como las dilated convolutions, ver Figura 2.2., que son convoluciones normales que tienen una tasa de dilatación que define la distancia entre los valores de un núcleo en serie o en paralelo, lo que ayuda a la red a comprender el contexto multiescala mediante el uso de diferentes tasas de dilatación.

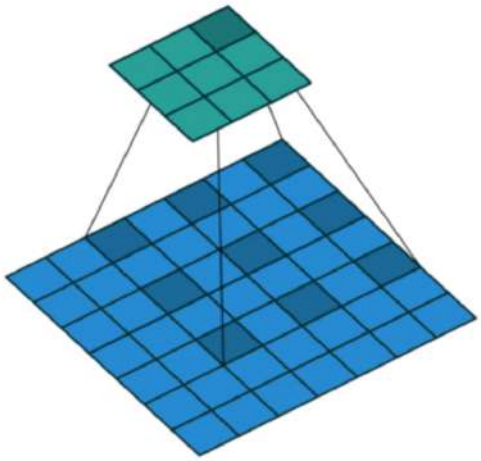


Figura 2.2: Dilate-Convolution 2D con un kernel de 3 y una tasa de dilatación de 2.

Además, DeepLabv3 emplea un bloque de Spatial Pyramid Pooling [43], ver Figura 2.3,

que otorga características a nivel de imagen que codifican el contexto global de la imagen. Este bloque acepta tamaños de entrada arbitrarios y produce salidas de tamaño fijo requeridas por las capas totalmente conectadas (fully connected layers), por lo que la imagen de entrada puede ser de cualquier tamaño. Esto permite no sólo relaciones de aspecto, sino también permite utilizar escalas arbitrarias. Podemos cambiar el tamaño de la imagen de entrada a cualquier escala y aplicar la misma red sin modificaciones. Además, el uso de estas herramientas permite a la arquitectura de DeepLabv3 trabajar con imágenes de tamaño arbitrario y generalizar las características de los objetos a segmentar independientemente de la escala dentro de la imagen, obteniendo así buenos resultados en tareas de segmentación semántica, ver figura 2.4.

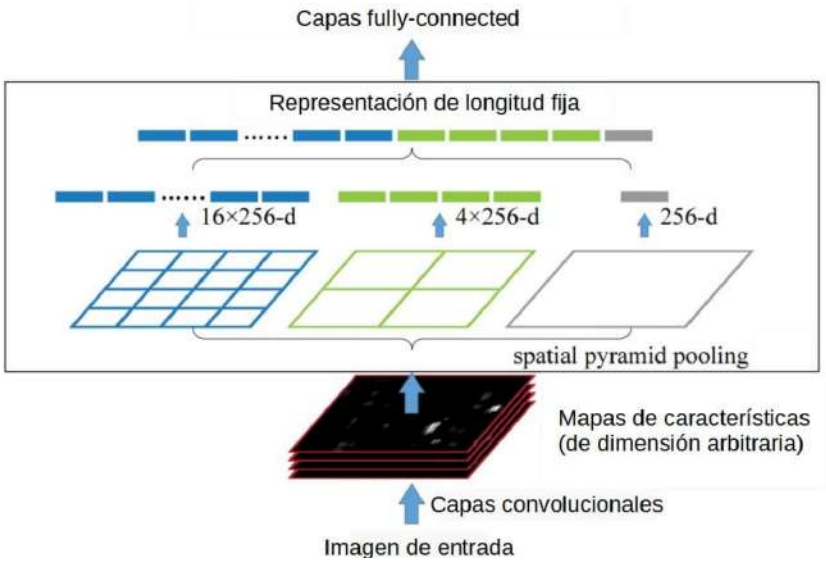


Figura 2.3: Estructura de una red con spatial pyramid pooling, donde 256 es el número de filtros.

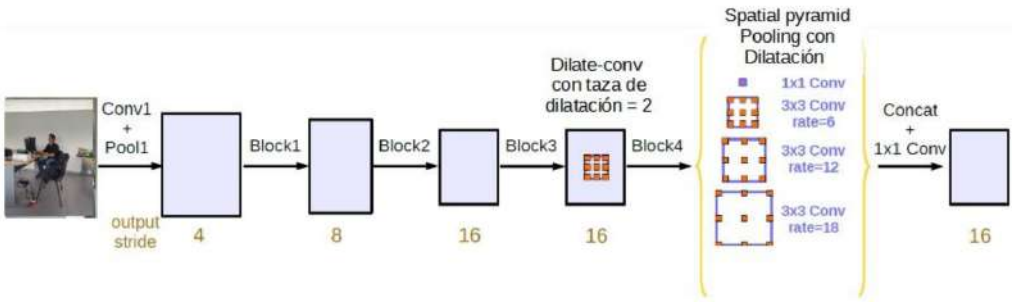


Figura 2.4: Arquitectura de DeepLabv.

2.2.2. LRASPP MobileNetV3 Large

Por otro lado, LRASPP_MobileNetV3_Large es una red neuronal basada en una combinación de técnicas de búsqueda complementarias que se adapta a CPU de teléfonos móviles y dispositivos de bajos recursos con o sin GPU utilizando una combinación de búsqueda de arquitectura de red consciente del hardware (NAS) complementada por el algoritmo NetAdapt. La busqueda MobileNetV3 [44], se definen dos modelos, MobileNetV3Large y MobileNetV3Small. Estos modelos se dirigen a casos de uso de recursos altos y bajos, respectivamente. En este trabajo, la red Lite RASPP (LRASPP) se ha utilizado con MobileNetV3Large [42] tal estructura se muestra en la Figura 2.6. El RASPP Lite, tiene varias mejoras sobre el RASPP, la más importante es el aumento de la velocidad de procesamiento y la reducción de parámetros [44]. La Figura 2.5 muestra la estructura del bloque MobileNetV3, que consiste en una capa de convolución 1x1 (sin filtros no lineales), seguida de una convolución Depthwise, despues se aplica compresión y excitación a la capa residual, por último una convolución 1x1, y concatenamos esta salida con la entrada.

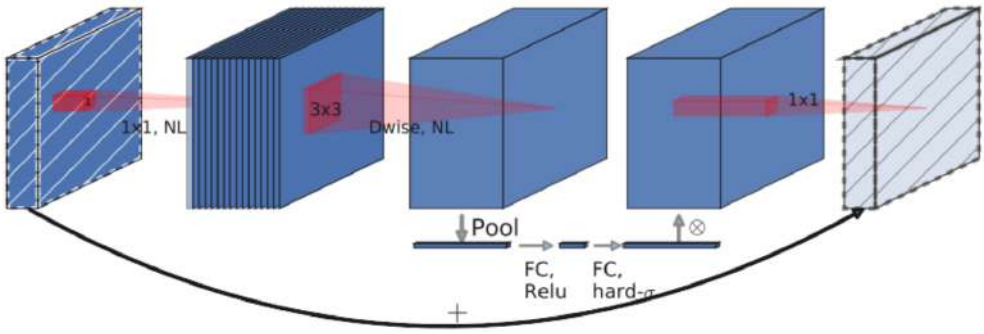


Figura 2.5: Estructura del bloque MobileNetV3.

Estas arquitecturas de redes neuronales representan avances significativos en la capacidad de los sistemas de visión por computadora para interpretar y comprender el mundo visual de forma detallada y eficiente, abriendo nuevas oportunidades para aplicaciones en robótica autónoma, navegación y análisis de imágenes.

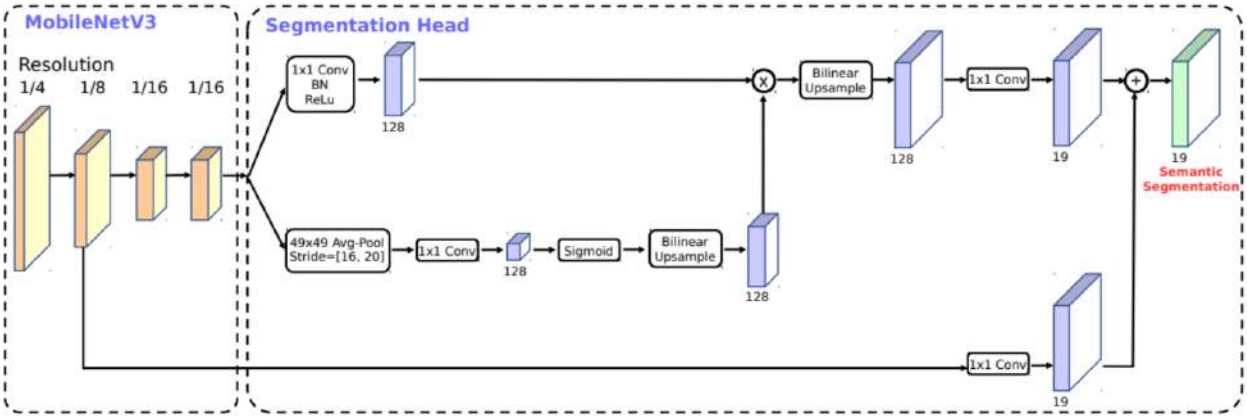


Figura 2.6: En la arquitectura LR aspp MobileNetV3, el bloque de Segmentation Heat proporciona resultados rápidos de segmentación semántica a la vez que combina características de múltiples resoluciones.

CAPÍTULO 3

Metodología

Esta sección presenta y detalla la configuración necesaria para desarrollar el sistema Visual SLAM dinámico. El diagrama de bloques de la Figura 3.1 proporciona una representación visual del flujo de trabajo. El sistema comienza con la fase de adquisición de imágenes, en la que se adquierem datos del entorno utilizando la configuración de la cámara RGB-D o estéreo, seguida de la segmentación semántica de objetos, en la que se identifican los elementos dinámicos y estáticos dentro de las imágenes utilizando un modelo DeepLabv3 mejorado con ResNet50 (véase Figura 3.6). A continuación, la extracción de características ORB (Oriented Fast and Rotated Brief) identifica las características esenciales dentro de las imágenes segmentadas para una navegación precisa. Finalmente, la fase de seguimiento y mapeo utiliza estas características para construir una trayectoria detallada y un mapa del entorno clasificando entre elementos estáticos y dinámicos. Cada uno de estos componentes desempeña un papel fundamental en la mejora de la percepción del entorno y la precisión de la navegación VSLAM.

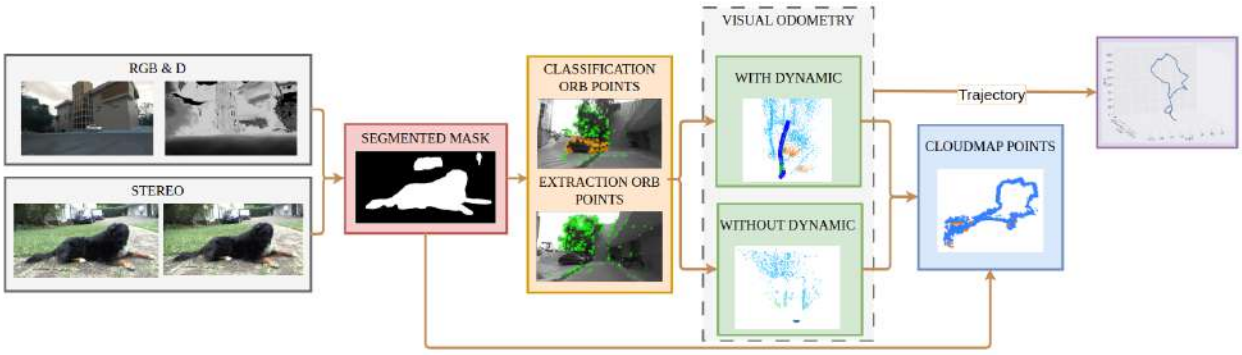


Figura 3.1: Flujo de procesamiento VSLAM dinámico

3.1. Adquisición de Imágenes

Para mejorar la precisión y robustez del VSLAM en entornos dinámicos, decidimos utilizar cámaras estéreo y RGB-D, la cámara ZED [45] ofrece esta solución unificada. Esta elección de combinación se basa en sus distintas ventajas, por lo que la configuración de la cámara depende de las condiciones ambientales y los requisitos de la tarea. La cámara ZED tiene la capacidad de capturar imágenes estereoscópicas y generar mapas de profundidad de diferentes resoluciones. Su diseño integrado permite recoger distintos tipos de información y, gracias a sus bibliotecas, es fácilmente configurable, lo que nos permite adaptarla a nuestras necesidades de adquisición de imágenes.

Figura 3.2: La figura ilustra cómo la cámara ZED combina la funcionalidad de las cámaras estéreo y RGB-D.

3.1.1. Cámara Estéreo

El uso de cámaras estereoscópicas en aplicaciones VSLAM ofrece importantes ventajas, especialmente en entornos exteriores, a diferencia de las cámaras RGB-D, que pueden verse gravemente afectadas por variaciones extremas en las condiciones de iluminación, como la intensa luz solar directa durante el día o la falta de luz por la noche, las cámaras estéreo mantienen un buen rendimiento bajo estas condiciones. Esto se debe a su capacidad para basarse menos en la información de color y más en la geometría espacial entre dos puntos de vista ligeramente diferentes.

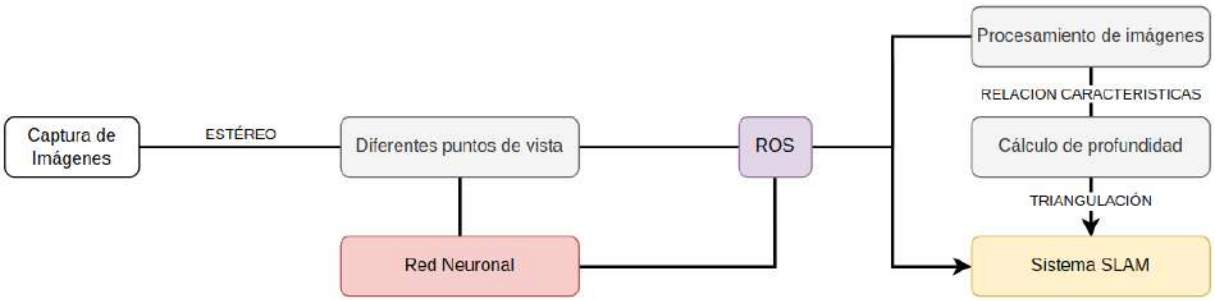


Figura 3.3: Diagrama que detalla el proceso de adquisición y procesamiento de datos de una cámara estéreo para una aplicación VSLAM.

Una de las ventajas de la visión estéreo es su coste relativamente bajo y su eficacia para estimar la profundidad del entorno. Este proceso se lleva a cabo comparando puntos de interés o características similares encontradas en pares de imágenes capturadas por dos lentes separados por una distancia conocida como línea base. Comparando las diferencias de posición de estos puntos entre las dos imágenes (izquierda y derecha), lo que se conoce como disparidad, es posible calcular la distancia a la que se encuentran los objetos en la escena.

La profundidad de cada punto se estima mediante triangulación, una técnica geométrica que utiliza la disparidad entre los mismos puntos en ambas imágenes y la distancia conocida entre las cámaras (línea base). Véase en la Figura 3.3 un diagrama detallado del proceso de adquisición y procesamiento de datos mediante una cámara estéreo, tal como se aplica en VSLAM.

3.1.2. Cámara RGB-D

Las cámaras RGB-D, por su parte, proporcionan una ventaja significativa en escenarios de interior y en situaciones en las que se requiere una percepción de la profundidad más precisa y detallada. Esta configuración captura imágenes en RGB y genera mapas de profundidad combinando tecnologías de infrarrojos y de proyección de patrones. Esta capacidad de proporcionar simultáneamente información de color y mediciones de distancia confiere a las cámaras RGB-D una comprensión más completa y versátil del espaciío. ver Figura 3.5.

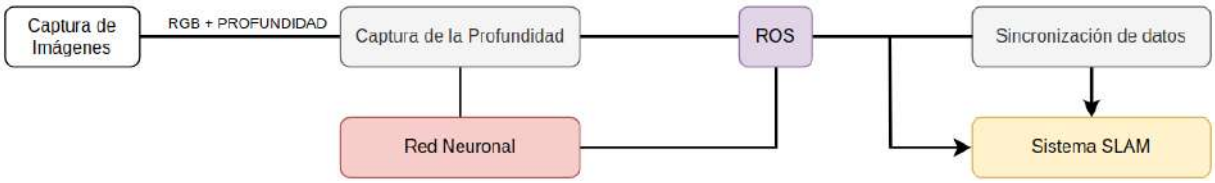


Figura 3.4: Diagrama que muestra el proceso general de adquisición y procesamiento de datos con una cámara RGB-D.

Una clara ventaja de las cámaras RGB-D es su capacidad para generar información de profundidad en tiempo real con un alto grado de precisión. A diferencia de la visión estéreo, que requiere complejos cálculos de correspondencia y triangulación, las cámaras RGB-D pueden proporcionar datos de profundidad directamente, lo que reduce la carga computacional y facilita la implementación en sistemas en tiempo real. Esto es especialmente importante en entornos controlados o interiores, donde la iluminación estable y la ausencia de luz solar directa permiten que la tecnología de infrarrojos funcione de manera óptima.

Además, las cámaras RGB-D superan algunas de las limitaciones inherentes a la visión estéreo, como la dificultad de tratar con superficies con poco texturizadas o la necesidad de una calibración cuidadosa. Al proporcionar un mapa de profundidad directo, estos dispositivos pueden capturar y modelar objetos y superficies con gran detalle, proporcionando una comprensión más completa y matizada del espacio. Esto tiene un valor importante en aplicaciones como la navegación robótica en entornos domésticos, donde la capacidad de detectar obstáculos y esquivarlos rápidamente es fundamental.

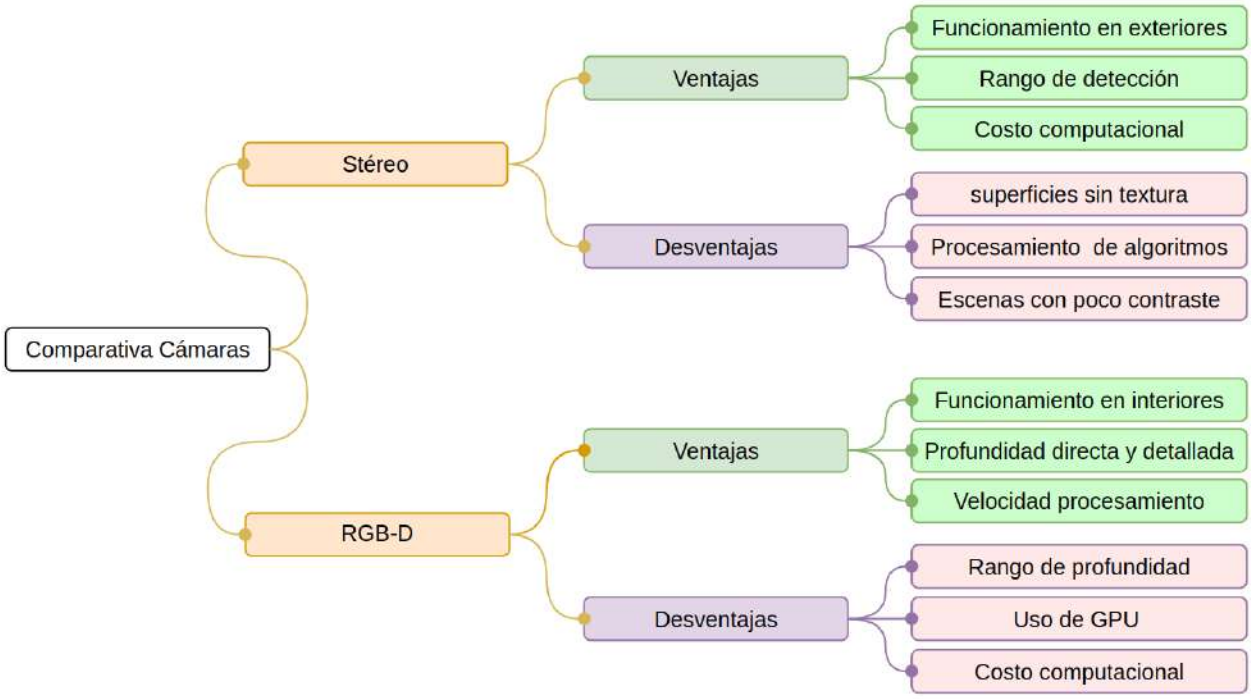


Figura 3.5: Comparación entre Cámaras estéreo y RGB-D

En resumen, nuestra metodología de adquisición de imágenes se basa en aprovechar las ventajas complementarias de las cámaras estéreo y RGB-D para adaptarlas a las características del entorno en que se va a operar y a las necesidades de la tarea que se va a realizar, teniendo en cuenta las limitaciones y ventajas de cada configuración de cámara. De este modo se garantiza que el sistema de VSLAM disponga de una buena entrada de datos independientemente de las condiciones externas. Una vez adquiridas que las imágenes, implementamos un enfoque dual: por un lado, la imagen RGB que es procesada por una red neuronal de segmentación semántica de objetos que distingue entre elementos estáticos y dinámicos de la imagen, mientras que la información sobre la imagen y la profundidad se utiliza para la extracción de características y, posteriormente, para el seguimiento y mapeo. La combinación de cámaras permite a nuestro sistema VSLAM funcionar de forma óptima en condiciones variaciones de iluminación y en exteriores, así como manejar de manera eficiente la complejidad de los entornos dinámicos y no estructurados, mejorando así la eficacia de la aplicación en diversos de contextos prácticos.

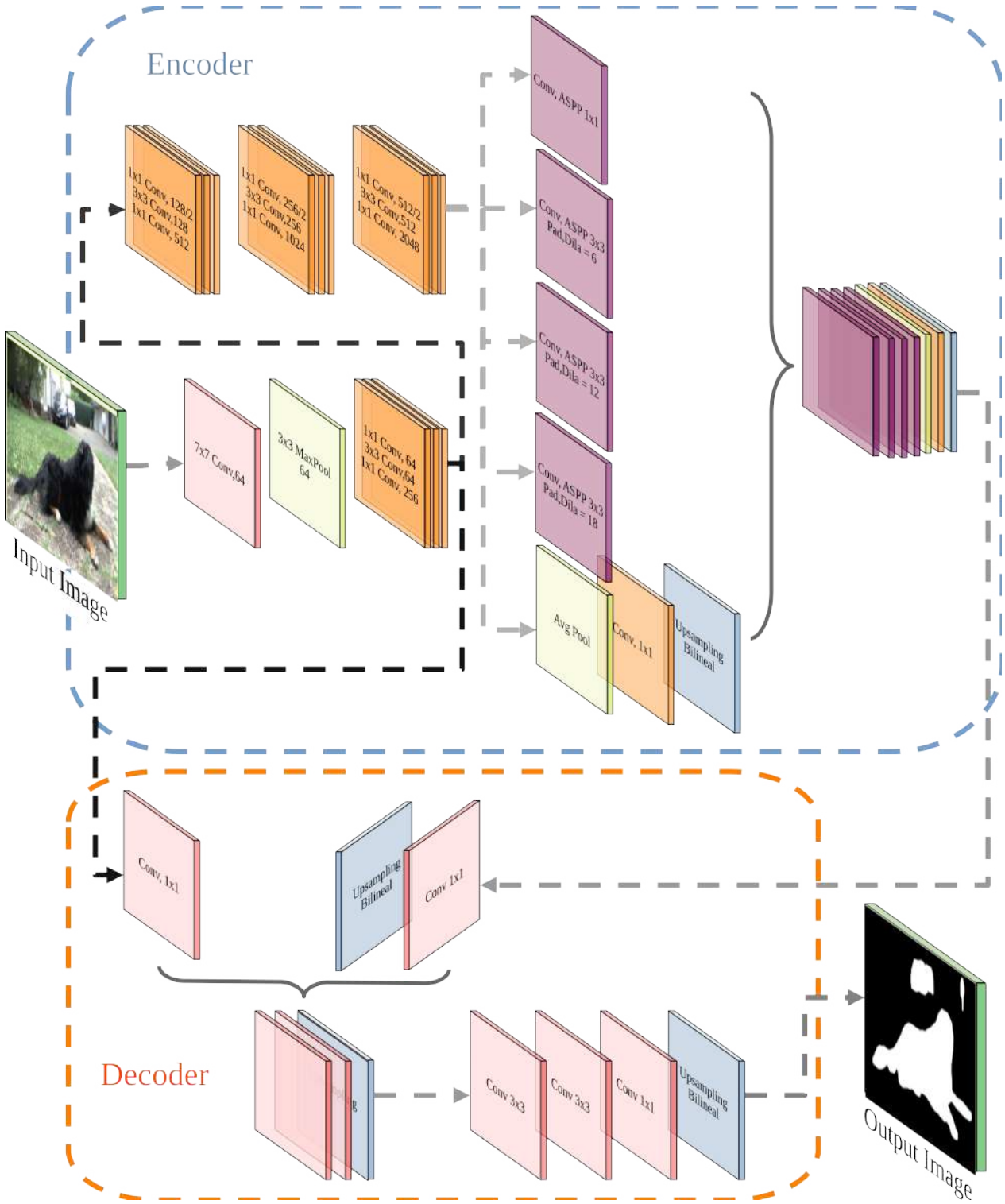


Figura 3.6: Estructura y componentes de DeepLabv3 + ResNet50.

3.2. Segmentación Semántica de Objetos

Implementamos una arquitectura que combina DeepLabv3 con ResNet50 tras compararla con otras arquitecturas de aprendizaje profundo. Esta decisión se basó en su mejor rendimiento en pruebas específicas. Utiliza una estructura de codificador-decodificador, donde el codificador extrae mapas de características de las imágenes de entrada y el decodificador los refina mediante upsampling, reconstruyendo la información espacial para obtener segmentaciones precisas de los objetos. (ver Figura 3.6).

La implementación de la convolución atrous en DeepLabv3 es clave para comprender contextos en múltiples escalas usando tasas de dilatación variables, lo que se consigue sin aumentar el costo computacional. Esta técnica, junto con el Pooling Piramidal Espacial (SPP) y Átrous Spatial Pyramidal Pooling (ASPP), hace que DeepLabv3 sea eficaz en la segmentación de objetos en diferentes escalas.

El modelo DeepLabv3-ResNet50 se entreno en el conjunto de datos MS COCO [46], que contiene una amplia variedad de imágenes, junto con un reentrenamiento en clases especificas de PASCAL VOC [47]. Esto se realizó para garantizar que el modelo sea apto para identificar y segmentar una amplia gama de objetos en diferentes escenarios.

Además de los procesos mencionados, es importante destacar la metodología utilizada para entrenar y afinar el modelo. Se utilizaron técnicas de aumento de datos para mejorar la generalización del modelo a nuevas imágenes. Esto incluyó rotar, escalar y cambiar la iluminación de las imágenes de entrenamiento, lo que permitió al modelo aprender a reconocer objetos bajo diferentes condiciones de visión.

Convertimos el resultado de nuestra red de segmentación semántica en una máscara que distingue entre información estática y dinámica para utilizarla en el sistema Visual SLAM. Esta distinción nos permite excluir los objetos dinámicos de las referencias utilizadas para la localización y el mapeado, lo que mejora el rendimiento de nuestro sistema VSLAM en entornos con objetos en movimiento. Esta mejora es fundamental para mantener una localización y un mapeado precisos y eficientes sin perder información de estos objetos dinámicos.

3.3. Extracción de Características ORB

La extracción de características ORB se realiza siguiendo los siguientes pasos: Detección de esquinas con FAST, el primer paso en la extracción de características ORB es la detección de esquinas utilizando el algoritmo FAST. Este método es muy eficaz en la identificación de puntos de interés, que son esenciales para el mapeo y la localización. La puntuación de una esquina potencial, p, se calcula sumando las diferencias absolutas de intensidad entre el píxel candidato y los píxeles en un conjunto S situados en un círculo alrededor de p.:

$$\text{Puntuación}(p) = \sum_{x \in S} |I(x) - I(p)| \tag{3.1}$$

donde $I(p)$ es la intensidad de píxel del candidato a esquina, mientras que x se refiere a cada uno de los píxeles dentro del conjunto S, que está formado por píxeles en un círculo alrededor de p. Este paso permite identificar características que se siguen a través de secuencias de imágenes, permitiendo una estimación precisa del movimiento y la orientación.

Asignación de orientación: Una vez detectadas las esquinas, se asigna una orientación a cada una de ellas para garantizar la invariancia de las características frente a la rotación de la imagen. Para ello, se calcula un ángulo de orientación $\theta(p)$ para cada punto de interés basado en la suma ponderada de las coordenadas de los píxeles circundantes, ajustada en funciónde la intensidad.

$$\theta(p) = \arctan\left(\frac{\sum_{s \in S} s_y I(s)}{\sum_{s \in S} s_x I(s)}\right) \tag{3.2}$$

donde s_x y s_y son las coordenadas del píxel en S relativas al centro p. Este paso permite que los descriptores de características sean robustos a los cambios en la orientación de la cámara o del objeto observado.

Generación de descriptores ORB: Por último, se generan descriptores binarios robustos para las características detectadas. Estos descriptores se crean comparando pares de píxeles alrededor de cada punto de interés según un patrón predefinido por BRIEF, y codificando el resultado de estas comparaciones en una cadena binaria.

$$\text{ORB}(p) = \sum_{n=1}^{N} 2^{n-1} \cdot \text{cmp}(I(p + r_n), I(p + r'_n)) \tag{3.3}$$

donde r_n y r'_n son pares de píxeles seleccionados según el patrón BRIEF, y cmp$()$ es una función de comparación que devuelve 1 si el primer término es mayor que el segundo, 0 en caso contrario.

Cada uno de estos pasos, desde la detección de esquinas con FAST hasta la asignación de orientación y la generación de descriptores ORB (ver figura 3.7), es fundamental para el funcionamiento eficaz de los sistemas Visual SLAM, ya que proporciona los medios para una localización precisa y una representación detallado del entorno.

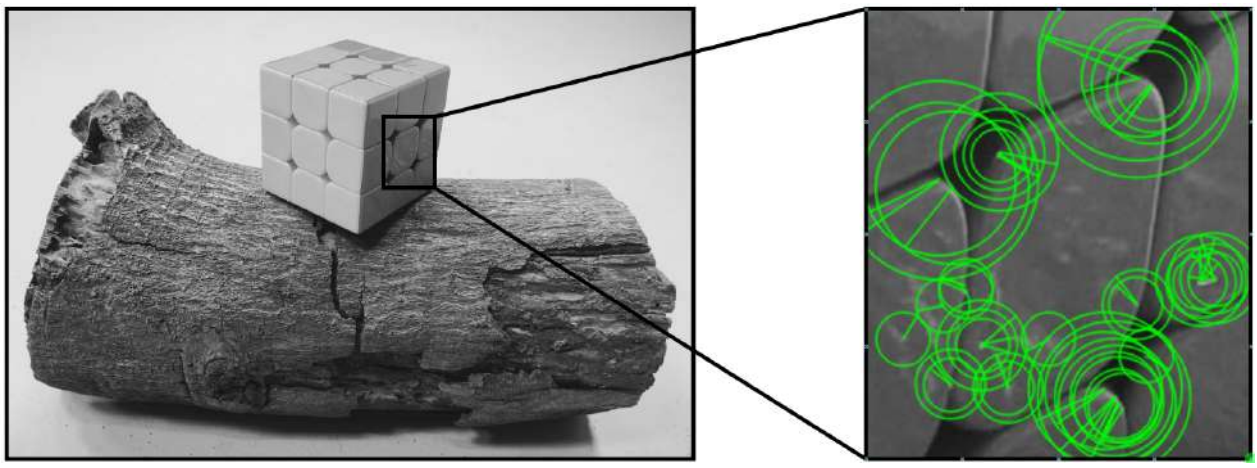


Figura 3.7: La imagen muestra los puntos clave identificados por el detector FAST orientado y BRIEF girado (ORB) en una fotografía. Cada círculo verde representa la ubicación y la escala de un punto clave, mientras que las líneas verdes indican su orientación.

3.4. Seguimiento y Mapeo

En el proceso de seguimiento y mapeo de nuestro sistema VSLAM, se han implementado dos algoritmos para clasificar y gestionar los puntos característicos, y para el procesar y mostrar la nube de puntos. Estos algoritmos aprovechan la información semántica para extraer y gestionar puntos clave de las imágenes, que son esenciales para identificar los puntos de interés y construir un mapa fiable del entorno.

3.4.1. Clasificación de puntos SLAM

El primer algoritmo se centra en la clasificación y gestión de los puntos clave extraídos de las imágenes $\mathcal{F}$ capturadas por el sistema. Utiliza una máscara de segmentación semántica $\mathcal{M}$ para distinguir entre puntos dinámicos y estáticos, asegurando que sólo los puntos estáticos son utilizados en el proceso SLAM, excluyendo posibles elementos dinámicos que pudieran afectar a la precisión del sistema.

Algorithm 1 Tracking

Require: $\mathcal{F}$ (Frames), $\mathcal{M}$ (segmentation mask)
Ensure: $\mathcal{M}_{\text{map}}$

for each $f \in \mathcal{F}$ **do**
  $KP_{\text{ORB}} \leftarrow$ Extract(f)
  for each $kp \in KP_{\text{ORB}}$ **do**
    if $\mathcal{M}(kp) =$ dynamic **then**
      Discard kp
    else
      Incorporate kp into SLAM
    end if
  end for
  Track and Map with KP_{static}
  Classify and store in $\mathcal{M}_{\text{map}}$
end for
return $\mathcal{M}_{\text{map}}$

Esta selección mejora la precisión y fiabilidad del sistema SLAM al evitar la inclusión de elementos dinámicos que podrían comprometer la calidad del mapeado.

3.4.2. Procesado y Nube de Puntos

El segundo algoritmo se centra en el procesado y representación de la nube de puntos, complementando el trabajo del Algoritmo 1. Este algoritmo comienza recogiendo todos los puntos previamente identificados en el mapa, tanto estáticos como dinámicos. La máscara de segmentación semántica $\mathcal{M}$ utilizada en el Algoritmo 1, se utiliza de nuevo para categorizar los puntos.

En el Algoritmo 2, los puntos se sitúan en su posición espacial real, denotada por pos, obtenida por el método p.GetWorldPos(). De este modo contribuyen al modelo detallado del entorno. Aunque los puntos dinámicos no se incluyen en el mapeado principal, su información se conserva para posteriores análisis. Al final del proceso SLAM, el Algoritmo 2 almacena esta información por si fuera necesaria para futuras aplicaciones y análisis detallados.

Algorithm 2 Categorization and Representation

```
MPs ← List of Map Points
for each point p in MPs do
    if p is valid and not a static reference point then
        nLabel ← p.GetDynamicLabel()
        Draw point p at its world position
    end if
end for
When SLAM is complete, save the point cloud
for each point p in MPs do
    if p is valid then
        pos ← p.GetWorldPos()
        nLabel ← p.GetDynamicLabel()
        Write pos and nLabel to the output file
    end if
end for
```

Al implementar estos algoritmos, nuestro sistema VSLAM mejora la clasificación de puntos y la construcción de un mapa detallado del entorno sin perder la información dinámica.

CAPÍTULO 4

Resultados

En esta sección, se presentan los resultados del sistema propuesto, evaluado mediante una combinación de pruebas con conjuntos de datos públicos y pruebas que simulan aplicaciones del mundo real. La evaluación comienza con el conjunto de datos KITTI [48], compuesto por secuencias en entornos urbanos y rurales, y el conjunto de datos TUM [49], con secuencias de una cámara RGB-D en interiores. Estas evaluaciones iniciales sirvieron de base para la comparación con otros sistemas del mismo campo de investigación. Tras estas evaluaciones del sistema en condiciones simuladas y controladas, ampliamos las pruebas a escenarios del mundo real, centrándonos específicamente en la detección de objetos dinámicos. Las pruebas en escenarios exteriores, como estacionamientos y zonas urbanas, nos permitieron evaluar la funcionalidad y adaptabilidad de nuestro sistema en los entornos que mejor reflejaran su uso práctico previsto. Los resultados de estas pruebas nos proporciona información sobre aspectos como la velocidad y precisión del procesamiento en tiempo real, así como la capacidad de identificar y clasificar correctamente objetos dinámicos.

A continuación se describe la configuración del sistema, los resultados de las evaluaciones del conjunto de datos y las pruebas en entornos reales.

4.1. Configuración del Sistema

Para la configuración de la unidad a bordo, elegimos el módulo Jetson Xavier principalmente por su tamaño, bajo consumo de energía y capacidad de procesamiento embebido, ideal para su integración en vehículos autónomos. Este módulo, equipado con un procesador gráfico NVIDIA y una CPU Arm de 64 bits, se destacó como la opción óptima para afrontar los retos experimentales propuestos. La elección del vehículo y su aplicación en distintos terrenos reflejaba la importancia de contar con un sistema modular, fácilmente adaptable y con la instrumentación adecuada; los principales componentes de este sistema pueden verse en la Figura 4.1. El sistema puede navegar automáticamente mediante un programa de planificación de trayectorias, o manualmente con un sistema de radiocontrol.

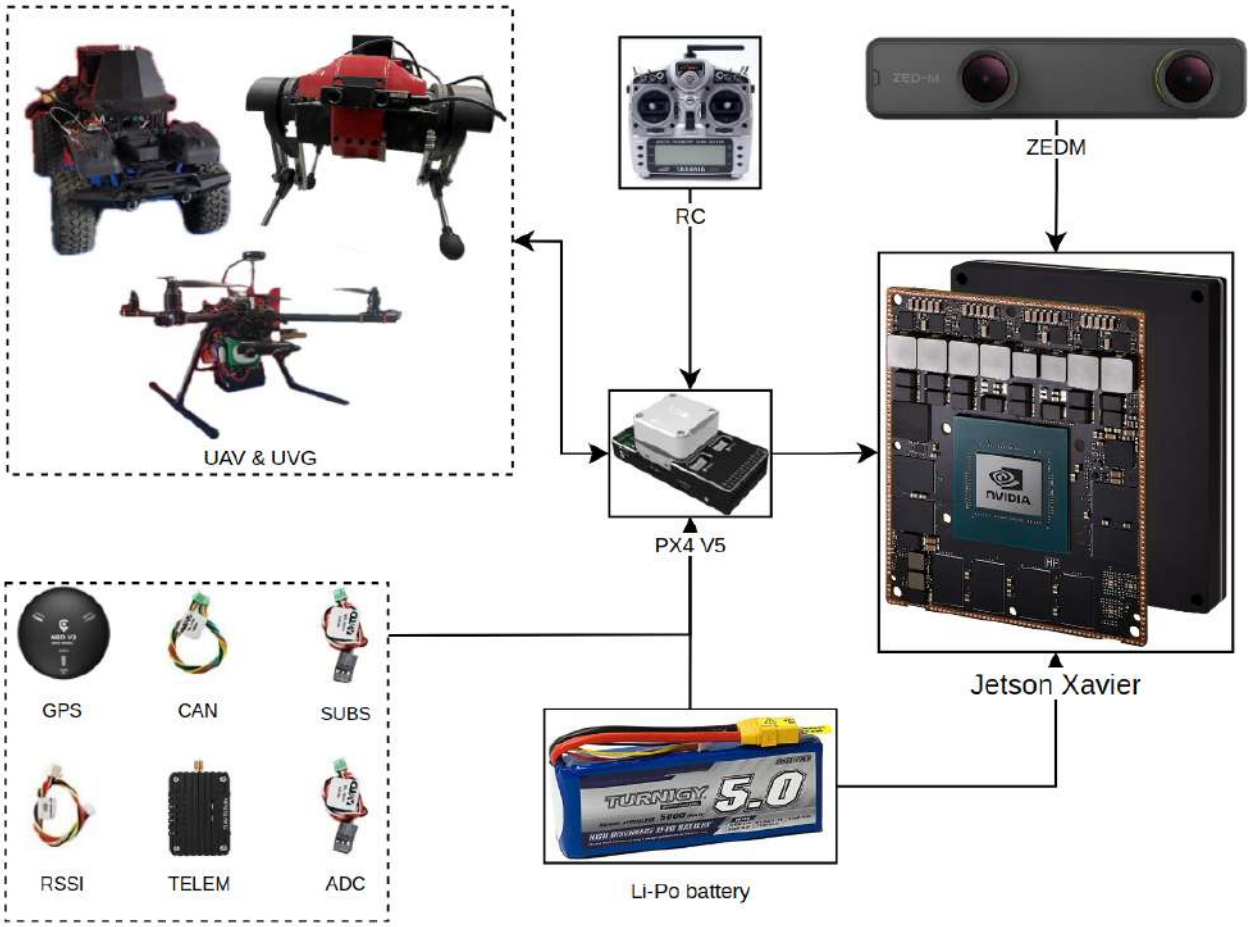


Figura 4.1: Componentes principales del vehículo

Para adaptar sistema VSLAM a la arquitectura Arm64, fue necesario modificar las librerías del sistema operativo y asegurar una configuración óptima con la cámara ZED, ver figura 4.2. El procesamiento se optimizó utilizando CUDA para permitir la aceleración de la GPU. A continuación se describen las configuraciones realizadas en la unidad de procesamiento:

- **SDK JetPack:** La selección de esta versión asegura la compatibilidad con el sistema operativo y proporciona una base para nuestra implementación de prueba.
- **ROS Framework:** La integración de ROS como framework facilita el uso de una amplia gama de herramientas y librerías especializadas en robótica.
- **CUDA y OpenCV:** La actualización a CUDA y la compilación de OpenCV con soporte para CUDA acelera el procesamiento de imágenes y la ejecución de algoritmos de visión por computadora.
- **Pytorch y Python3:** ROS (Melodic) ha sido adaptado para trabajar con Pytorch bajo Python3, eliminando las incompatibilidades causadas por la versión nativa Python2 y permitiendo una integración óptima de las librerías de Deep Learning dentro del ecosistema ROS.

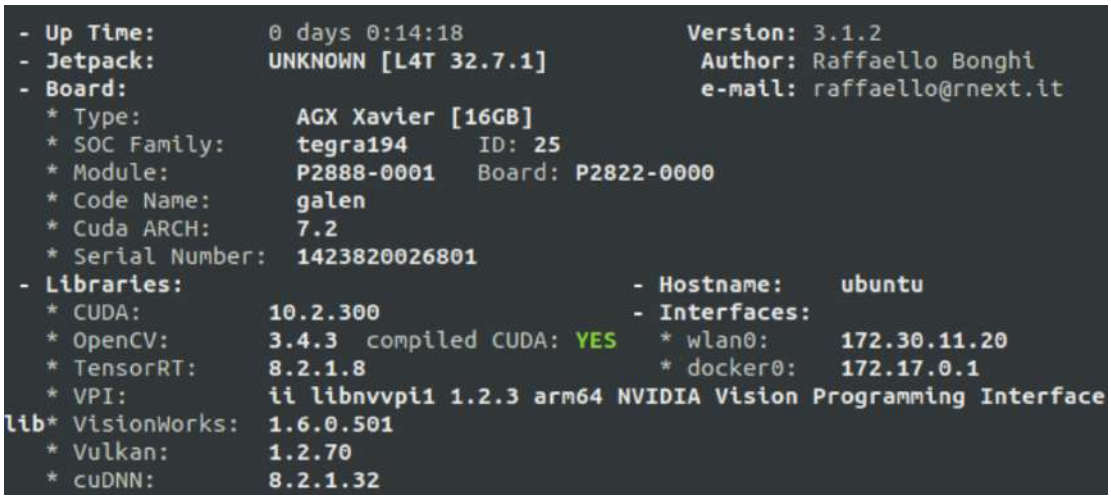


Figura 4.2: Configuración del modulo Jetson Xavier mostrada por jetson-stats [5].

Para la segmentación semantica de los objetos dinámicos, se decidió utilizar las redes DeeplabV3Resnet50 (ver figura 4.4), LRASPPMobileNetV3 y una combinación de ambos modelos mencionados en el marco teórico, véase la figura 2.6 , a continuación se muestra el rendimiento de las redes.

Cuadro 4.1: Comparación de las redes de segmentación con reentrenamiento.

Modelo	**IOU**	**Imagen por segundo**	**Precisión global de pixeles**
DeeplabV3-Resnet50	66.4	29	92.4
DeeplabV3-MobileNetV3	60.3	35	91.1
LRASPP-MobileNetV3	37.58	37.58	89.0

Basandonos en los resultados de la tabla 4.1 y en la calidad de las imágenes segmentadas, elegimos la red DeepLabv3Resnet50. Aunque es la más lenta de las redes, su capacidad para proporcionar una segmentación precisa y completa de los elementos dinámicos, como se muestra en la Figura 4.3, la convierte en la mejor elección.

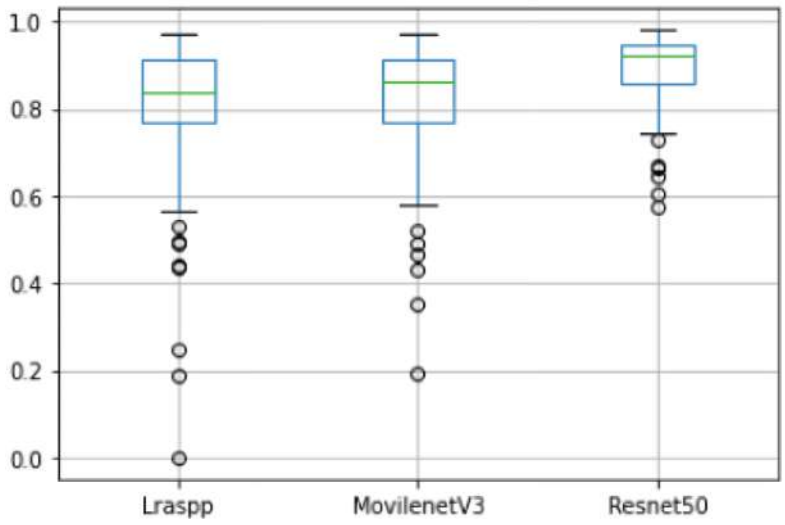


Figura 4.3: Diagrama de dispersión con el índice DICE

Durante la evaluación de nuestro sistema, el entorno ROS nos proporcionó la flexibilidad, robustez y sincronización necesarias para VSLAM. La comparación de rendimiento de nuestro sistema con otros sistemas VSLAM dinámicos se muestra en la Tabla 4.2.

Cuadro 4.2: GPU Used & Processing Time

System	GPU	Processing Time
DynaSLAM	Nvidia Tesla M40	190
STDyn-SLAM [50]	GeForce GTX 1070	333
DS-SLAM	P4000	220
Ours	Jetson Xavier	**80.65**

Nuestro sistema alcanza un tiempo de procesamiento de aproximadamente 13,2 HZ. lo que demuestra su capacidad para gestionar eficientemente las tareas de segmentación semántica y VSLAM. Este rendimiento, que supera a otros sistemas en términos de velocidad y precisión, es especialmente importante en entornos dinámicos donde el procesamiento rápido y la toma de decisiones son de gran importancia para el funcionamiento de los vehículos autónomos.

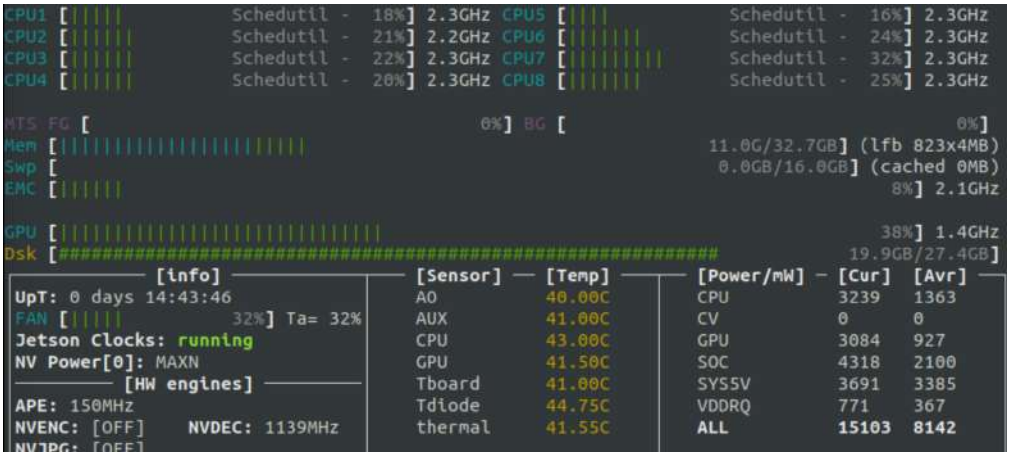


Figura 4.4: Se muestra el uso de recursos utilizando la red de segmentación.

A diferencia de otros sistemas que hacen hincapié en la reconstrucción de escenas 3D en tiempo real, nuestro enfoque da prioridad al procesamiento rápido del mapeo y la localización dejando la reconstrucción 3D para un postprocesamiento, ya que reconocemos que para la reconstrucción 3D inmediata no siempre es necesaria para los robots y vehículos autónomos, y es más probable que sea interpretada por el usuario/operador. En su lugar, nuestro sistema ofrece una visión diferenciada de los objetos dinámicos, proporcionando una comprensión global del entorno se adapta mejor a los requisitos prácticos de la navegación autónoma.

4.2. Evaluación de la precisión de seguimiento

Evaluar la precisión de seguimiento de nuestro sistema VLSAM utilizando los conjuntos de datos KITTI y TUM ha sido importante para validar nuestro enfoque. En primer lugar, validamos el enfoque estéreo con KITTI, que incluyen secuencias en entornos urbanos y rurales, y aplicamos la métrica del error absoluto de pose (APE) para cuantificar la precisión del seguimiento.

La capacidad del sistema para distinguir los objetos dinámicos de los estáticos contribuyó a los resultados positivos de la evaluación. Nuestro sistema identifica estos puntos y separa los objetos dinámicos del resto del entorno. Esta separación permite al sistema conservar información valiosa sobre los objetos dinámicos y mejorar la precisión del seguimiento al eliminar las variables en movimiento que podrían distorsionar la percepción y la referencia espacial del sistema. Para ilustrar este proceso, la Figura 4.5 muestra cómo el sistema distingue entre puntos dinámicos y estáticos.

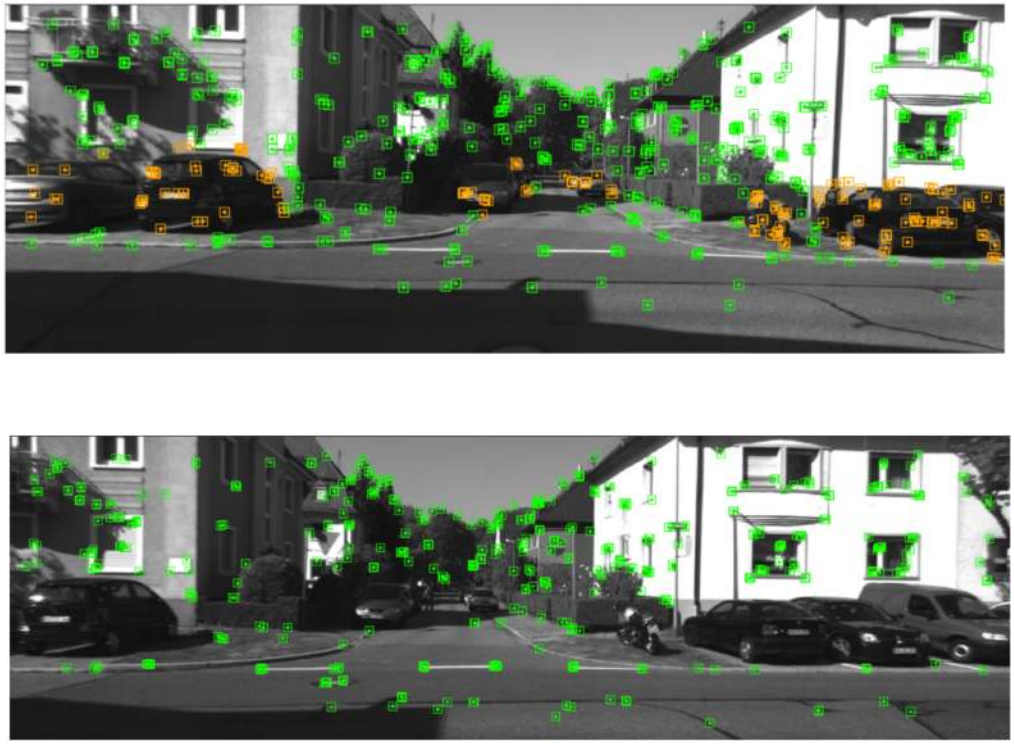


Figura 4.5: Arriba: Puntos dinámicos marcados en naranja, que muestran cómo el sistema identifica y resalta estos puntos en el entorno. Abajo: La misma escena sin los puntos dinámicos marcados, resaltando los elementos estáticos del entorno.

Los resultados de la evaluación de nuestro sistema en el conjunto de datos KITTI se presentan en la Tabla 4.3. Esta evaluación nos permitió comparar nuestro sistema con otros métodos reconocidos, destacando las capacidades de nuestro enfoque en diferentes entornos.

Cuadro 4.3: Comparación del APE en KITTI

SEC	ORBSLAM2		DS-SLAM		DynaSLAM		STDyn		**OURS**	
	MEAN	RMSE	MEAN	RMSE	MEAN	RMSE	MEAN	RMSE	MEAN	RMSE
00	1.339	1.438	3.391	3.715	1.192	1.335	**0.819**	0**.739**	1.237	1.375
01	8.819	9.453	5.212	5.653	11.316	12.160	**2.168**	**2.040**	3.104	3.672
02	5.674	6.977	5.182	5.483	4.825	3.989	**3.096**	**2.692**	5.208	6.305
03	0.696	0.755	0.577	0.605	0.575	0.673	**0.268**	**0.241**	0.585	0.675
04	0.199	0.215	0.564	0.681	0.270	0.270	0.169	0.154	**0.156**	**0.146**
05	0.721	0.810	4.372	4.943	**0.709**	**0.776**	0.836	0.766	0.766	0.873
06	0.841	0.865	4.239	4.510	0.723	0.754	1.735	1.596	**0.719**	**0.744**
07	0.504	0.557	2.915	3.029	0.701	0.701	0.538	0.502	**0.482**	**0.503**
08	3.270	3.534	4.759	5.012	3.295	3.595	3.224	2.781	**3.059**	**2.525**
09	2.765	3.420	7.683	7.824	2.263	**2.715**	1.953	1.826	**1.834**	**1.408**
10	**0.870**	**0.956**	0.905	0.985	1.028	1.145	1.227	1.132	1.132	1.095

En segundo lugar, validamos el enfoque RGB-D con TUM, que incluye secuencias en interiores y objetos en movimiento, utilizando Error Absoluto de Trayectoria (ATE) como metrica de evaluación. Los resultados, mostrados en la Tabla 4.4, muestran la precisión de nuestro sistema en el seguimiento de trayectorias, comparativamente mejor que otros sistemas destacados.

Cuadro 4.4: Comparación del ATE en TUM

SLAM System	**W-Half-Sphere**	**W-XYZ**	**W-RPY**	**W-Static**	**S-Half-Sphere**	**S-XYZ**
ORB-SLAM2	0.351	0.459	0.662	0.091	0.023	0.009
DS-SLAM	0.055	0.043	0.076	0.024	0.039	0.002
DM-SLAM [51]	0.027	0.020	0.061	0.010	0.032	0.003
DynaSLAM	**0.025**	**0.015**	0.035	**0.006**	0.017	0.015
DVO-SLAM [52]	0.029	0.018	0.041	0.006	0.019	0.043
DDL-SLAM [53]	0.033	0.029	0.070	0.052	0.20	0.006
OURS	**0.025**	**0.015**	0.037	0.012	0.018	**0.005**

La configuración RGBD, con secuencias en interiores, nos ha permitido tener una perspectiva más detallada de los elementos dinámicos en el mapeo y seguimiento. En la Figura

4.6 muestra como el sistema detecta los puntos dinámicos junto con la nube de puntos resultante, es posible observar el ruido y la influencia que estos elementos tienen en el proceso. Contrastando esta visualización, la siguiente Figura 4.7 muestra la escena con los puntos dinámicos eliminados, destacando únicamente los elementos estáticos y su correspondiente nube de puntos. Este enfoque da como resultado una representación más limpia y precisa del entorno. Estas visualizaciones son importantes para entender cómo nuestro sistema distingue y gestiona los puntos dinámicos de los estáticos, utilizando las capacidades de la cámara RGB-D.

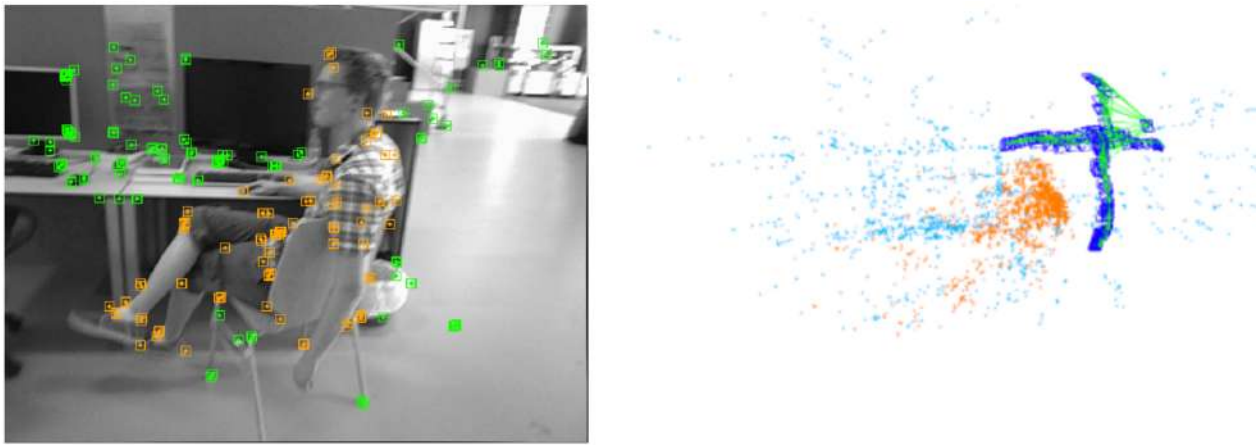


Figura 4.6: Puntos dinámicos en naranja, que muestran su influencia en el ruido de la nube de puntos en un entorno interior.

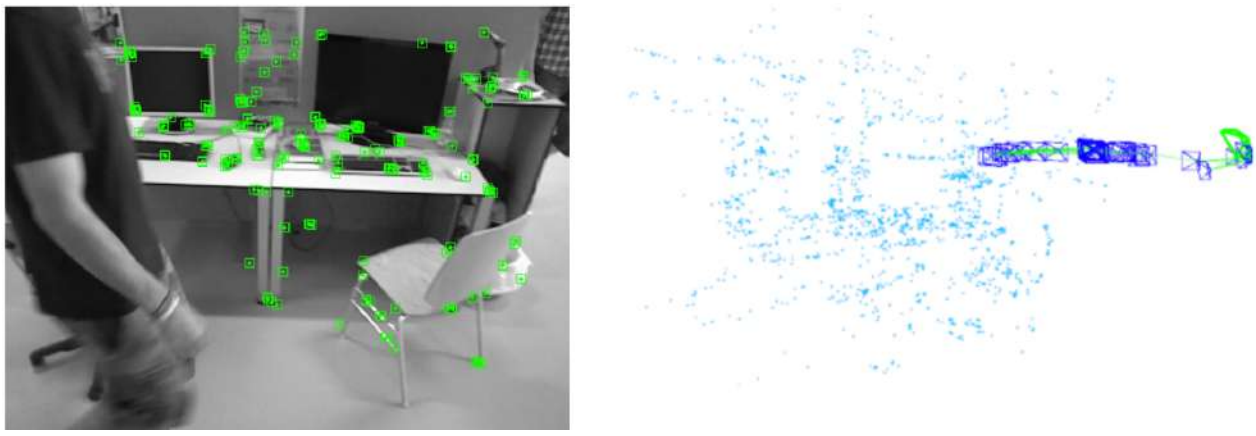


Figura 4.7: Escena sin elementos dinámicos, mostrando un mapa limpio y detallado.

Además, se llevó a cabo una comparación detallada con el sistema ORBSLAM2, resumida en la Tabla 4.5. Esta comparación nos permitió identificar mejoras significativas en varias métricas, lo que refleja la eficacia de nuestro sistema en entornos dinámicos. La Figura 4.8 ilustra la capacidad de nuestro sistema para seguir trayectorias largas y distinguir entre objetos dinámicos y estáticos en el conjunto de datos KITTI, demostrando visualmente las mejoras mencionadas.

Cuadro 4.5: Comparación entre ORBSLAM2 y nuestro sistema

	ORBSLAM2	Mejora de nuestro sistema
	RMSE	**Mejora**
desk_person	0.07	91.3 %
s_halfsphere	0.082	76.9 %
s_static	0.008	27.5 %
w_halfsphere	0.495	94.8 %
w_rpy	0.486	92.2 %
w_stactic	0.416	97.1 %
w_xyz	0.696	97.8 %

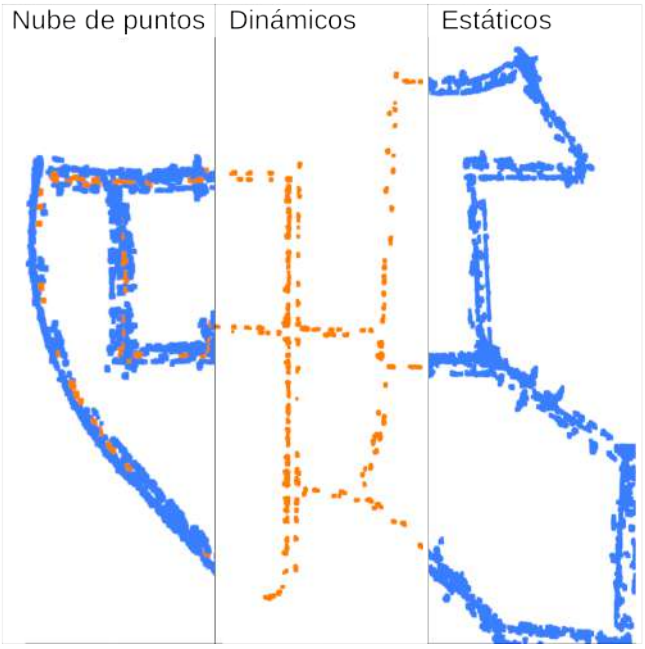


Figura 4.8: Representación de la escena desde tres perspectivas: a la izquierda, una combinación de puntos dinámicos y estáticos; en el centro, sólo puntos dinámicos detectados por el sistema; y a la derecha, sólo puntos estáticos.

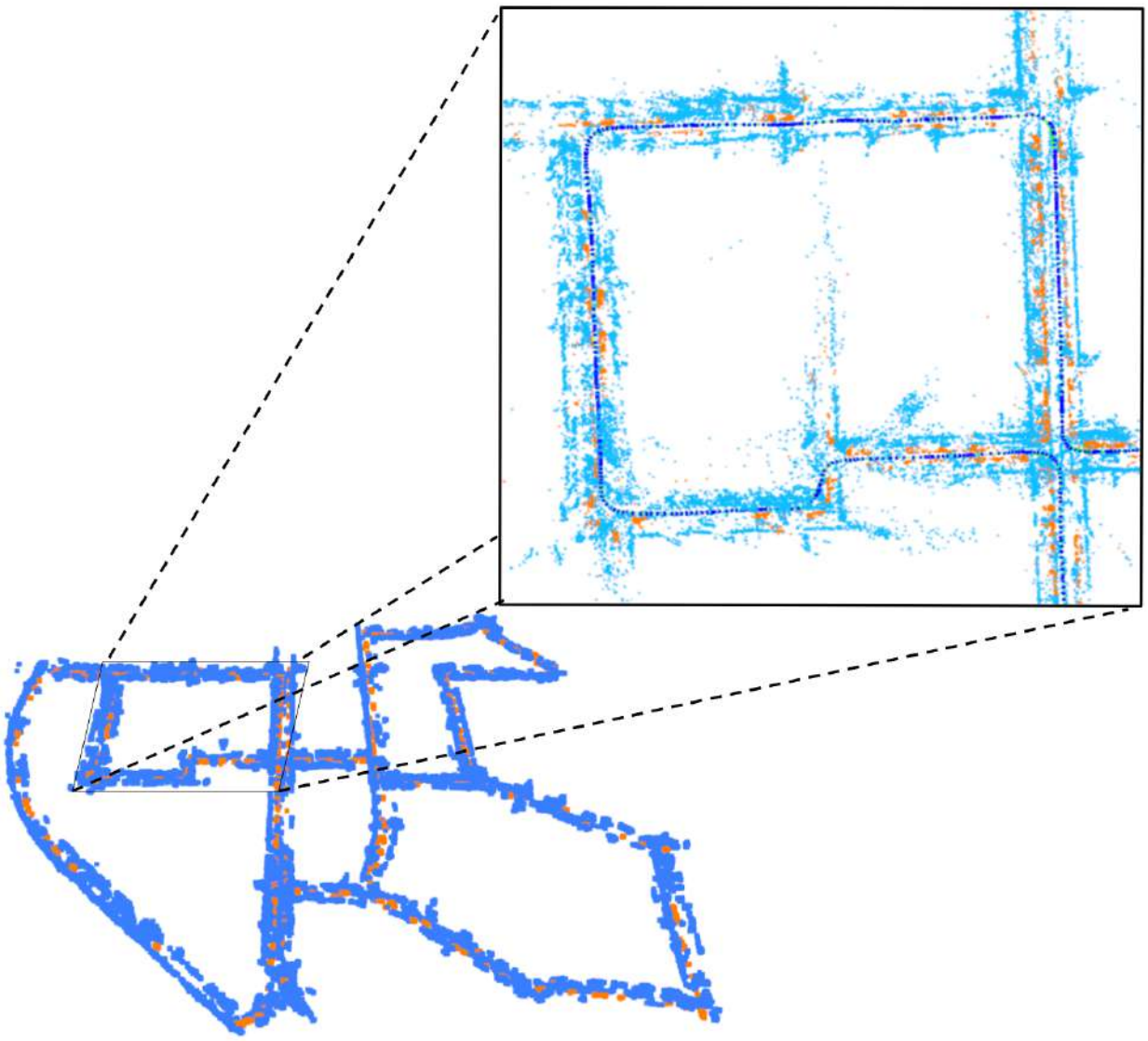


Figura 4.9: Visualización del seguimiento de objetos en el conjunto de datos KITTI.

4.3. Pruebas en el mundo Real en entornos dinámicos

Además de las evaluaciones de conjuntos de datos estandarizados, nuestro sistema se ha probado en escenarios reales, concretamente en entornos que presentan tanto objetos dinámicos como estáticos. Las pruebas se realizaron en diversos entornos para evaluar la capacidad del sistema para adaptarse y funcionar en diferentes condiciones. La figura 4.10 muestra el proceso y los componentes utilizados para desarrollar las pruebas del sistema en situaciones prácticas.

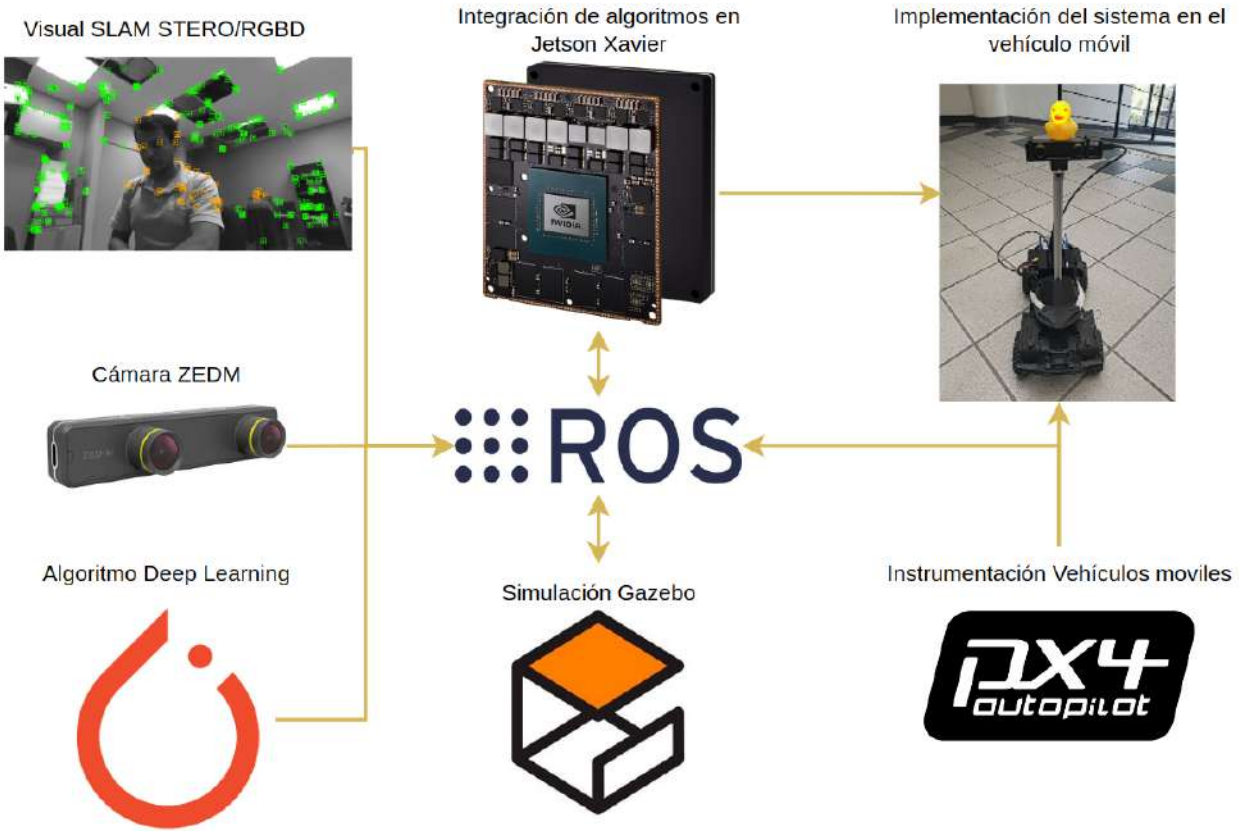


Figura 4.10: Componentes del sistema para las pruebas en escenarios reales

La implementación comienza con la calibración en interiores y exteriores de la cámara ZED para conseguir una percepción precisa del entorno para la segmentación semántica, como se muestra en la Figura 4.11. Esta etapa asegura que el sistema pueda mapear correctamente su entorno y localizar con precisión los objetos para diferenciar entre elementos estáticos y dinámicos, como se ilustra en la Figura 4.12.

Figura 4.11: Cámara ZEDM con su mapa de profundidad y mascará semántica

Figura 4.12: Implementación del VSLAM con la red de segmentación

Utilizamos Gazebo junto con ROS y PX4 para simular el entorno operativo del rover y la imlementación de una inteligencia artificial para el reconocimiento de objetos, lo que nos permite realizar ajustes y optimizaciones antes de las pruebas físicas, ver Figura 4.13 . Esta simulación fue importante para anticipar y corregir posibles errores en un entorno controlado, preparando el sistema para las pruebas en ambientes reales.

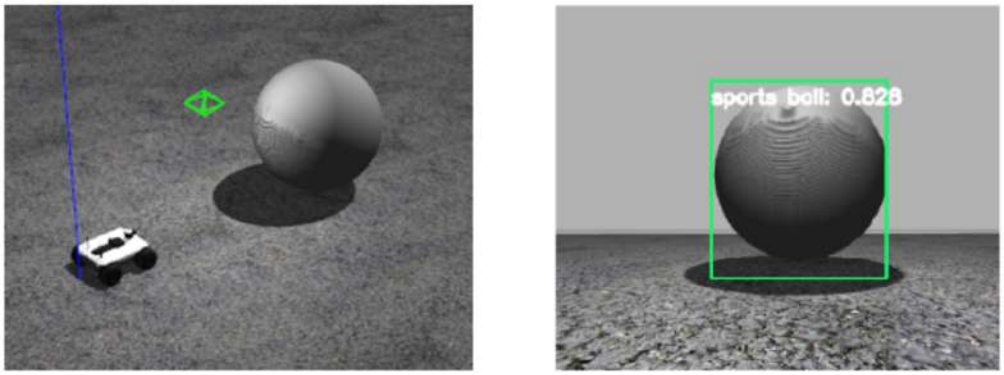


Figura 4.13: Simulación del vehículo móvil en Gazebo y prueba del modelo YOLO y del controlador SITL PX4.

Finalmente, se instaló el sistema completo en diferentes robots móviles (Ver figura 4.14) para pruebas en el edificio Octágono del CIO - Unidad León, donde se simularon escenarios dinámicos para evaluar la interacción del sistema con personas en movimiento, ver 4.15.

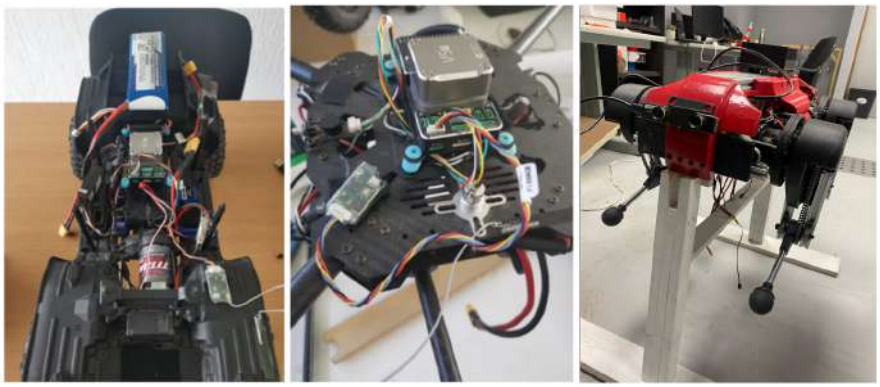


Figura 4.14: Implementación del sistema en diferentes robots móviles.

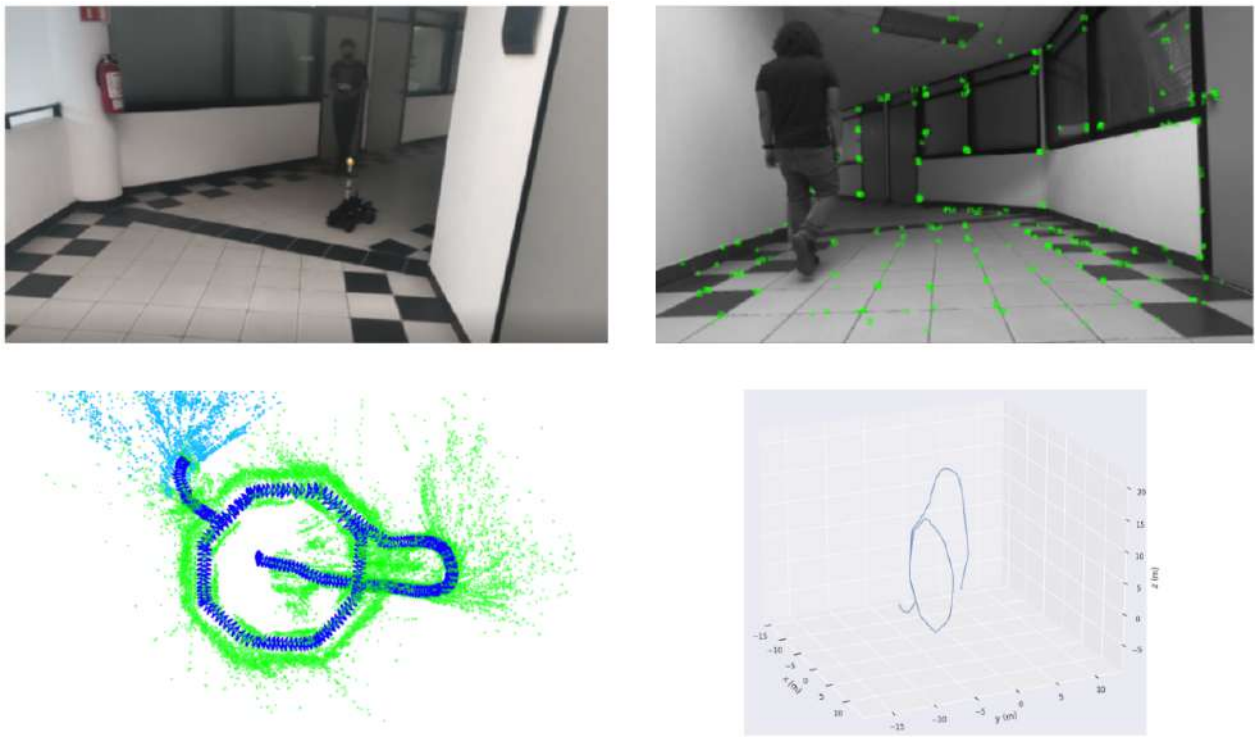


Figura 4.15: Implementación del VSLAM dinamico y nube de puntos en ROS

4.3.1. Pruebas en estacionamientos y entornos urbanos

Las pruebas se realizarón en el campus del CIO (Unidad León) y en un bulevard, entornos seleccionados por sus caracteristicas que presentan retos importantes: El bulevar, con tráfico de peatones y bicicletas además de una gran actividad de vehiculos, y el CIO, en un momento de gran movilidad de estudiantes y doctores, y de vehiculos. Estas condiciones, junto con las variaciones climáticas (nublado en el estacionamiento y soleado en el bulevar), permitieron evaluar el rendimiento del sistema en el seguimiento de trayectorias largas mientas se identificaban y clasificaban objetos de forma dinámica demostrando la robustez del sistema para filtrar el ruido del sistema VSLAM y mantener mapas precisos. Véase la Figura 4.17.

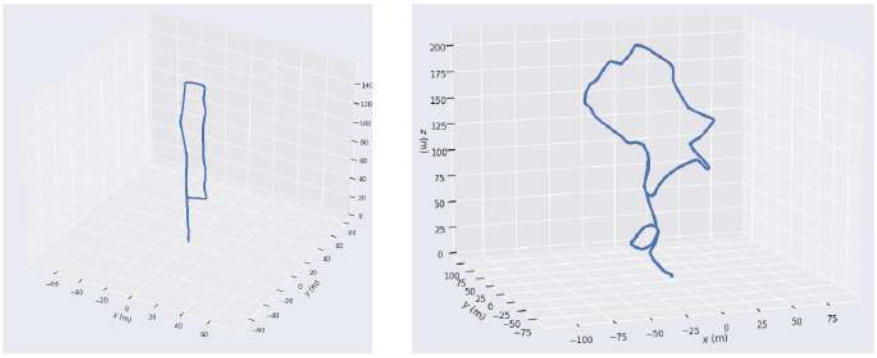


Figura 4.16: Trayectorias de los entornos seleccionados.

4.3.2. Identificación de objetos dinámicos y estáticos

El núcleo del sistema VSLAM dinámico reside en su capacidad para clasificar con precisión objetos dinámicos y estáticos, una capacidad importante para que el sistema VSLAM construya mapas del entorno precisos y fiables utilizando puntos estáticos. En escenarios prácticos con coches y personas, nuestro sistema identificó hábilmente objetos dinámicos (marcados en naranja), y objetos estáticos (marcados en azul). Esta distinción es esencial para construir mapas precisos, sino que también puede ser util para detectar y evitar obstáculos en tiempo real, proporcionando una base sólida para la toma de decisiones en vehículos autónomos y robots móviles. Véase la Figura 1.1.

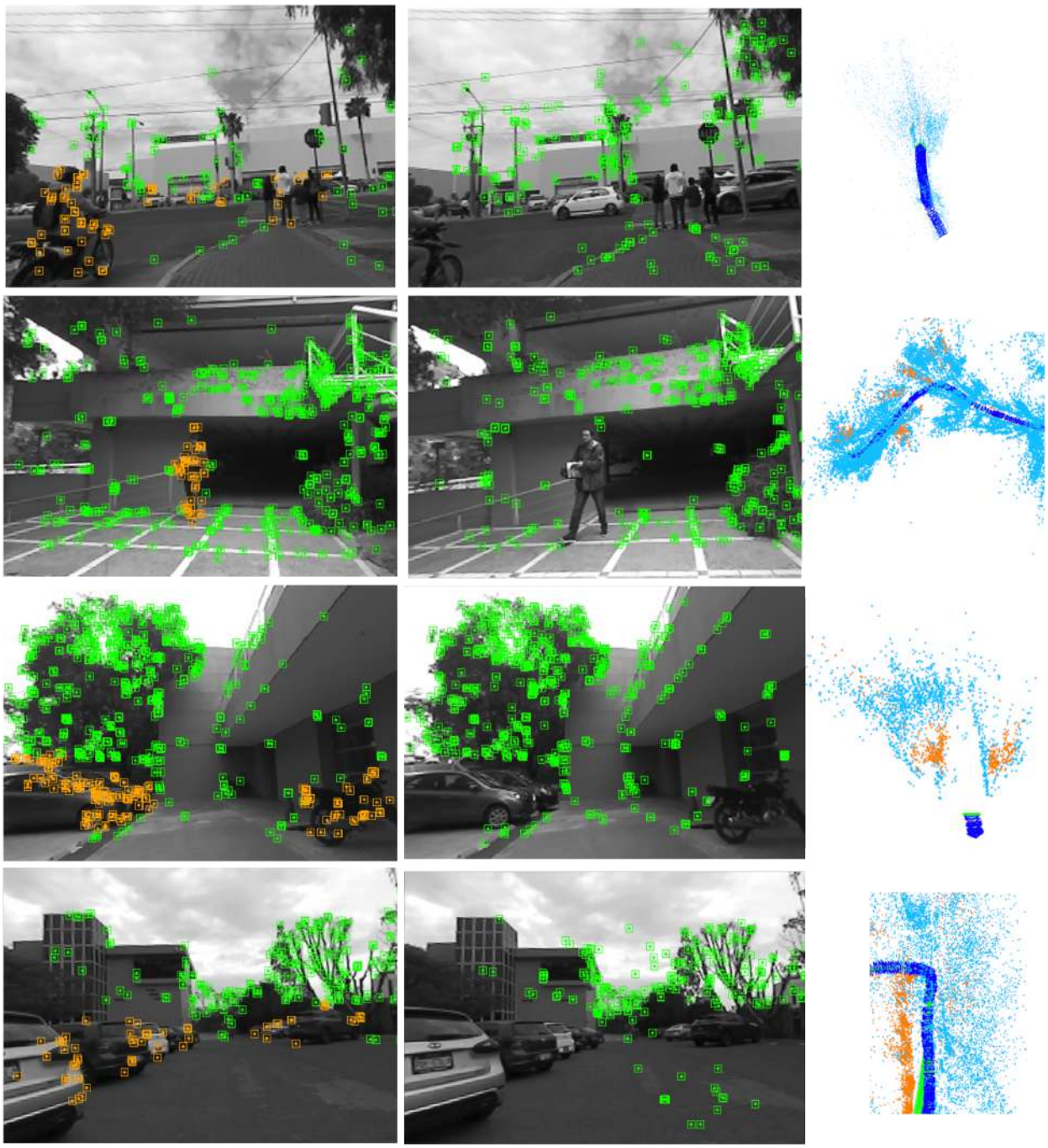


Figura 4.17: Objetos en movimiento y estáticos: a) objetos estáticos y dinámicos, b) sólo objetos estáticos, c) mapa generado a partir de ambos objetos.

CAPÍTULO 5

Conclusión

Los resultados experimentales, que abarcan tanto conjuntos de datos controlados como entornos dinámicos del mundo real, validan la factibilidad y eficacia de nuestro sistema Visual SLAM dinámico. No sólo es adecuado para navegar y mapear entornos dinámicos, sino que también resulta prometedor para aplicaciones más amplias en sistemas autónomos, especialmente en escenarios que requieren adaptación en tiempo real a condiciones cambiantes. Este proyecto ha demostrado la viabilidad de integrar un sistema de VSLAM dinámico en un equipo compacto como el Jetson Xavier, integrando algoritmos de Deep Learning y VSLAM en un entorno ROS. Teniendo en cuenta la velocidad a la que trabaja el Jetson Xavier una vez adaptadas las librerías para su uso, se abre también una brecha para poder seguir generando aplicaciones basadas en estos sistemas móviles. El sistema ha demostrado que puede detectar características de manera robusta y autolocalizarse con exito, al igual que con el algoritmo de Deep Learning que puede trabajar entre 20-30 fps, haciendo posible la integración de otros sensores en caso de ser necesario. También se determinó que la tarea de reconstrucción 3D no es necesaria para la autonomía del vehiculo movil, por lo que el sistema almacena la información que puede ser post-procesada en un ordenador que ejecute ROS y disponga de más recursos.

5.1. Recomendaciones

Basándonos en la eficacia demostrada de nuestro sistema Visual SLAM dinámico tanto en entornos controlados como en el mundo real, recomendamos explorar algoritmos de Deep Learning para mejorar otras áreas del sistema, como las que se muestran en la Figura 1.4. La integración de sensores adicionales podría ofrecer una mejor percepción ambiental, beneficiando a las aplicaciones en condiciones menos visibles. Dada su modularidad y buena implementación en vehículos y robots móviles, sería prometedor adaptarlo a sectores específicos como la agricultura de precisión o la asistencia domiciliaria, adaptando su funcionalidad a las necesidades de cada aplicación. Además, la automatización de su calibración y configuración inicial facilitaría su adopción por usuarios no especializados en el campo, ampliando su accesibilidad y promoviendo la innovación colaborativa, la publicación de resultados y el avance tecnológico en el campo de VSLAM en robótica movil.

Bibliografía

[1] D. Wooden, M. Malchano, K. Blankespoor, A. Howardy, A. A. Rizzi, and M. Raibert, "Autonomous navigation for bigdog," in *2010 IEEE International Conference on Robotics and Automation*, 2010, pp. 4736–4741.

[2] A. Pfrunder, P. V. K. Borges, A. R. Romero, G. Catt, and A. Elfes, "Real-time autonomous ground vehicle navigation in heterogeneous environments using a 3d lidar," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 2601–2608.

[3] S. Ahirwar, R. Swarnkar, S. Bhukya, and G. Namwade, "Application of drone in agriculture," *International Journal of Current Microbiology and Applied Sciences*, vol. 8, no. 01, pp. 2500–2505, 2019.

[4] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. Montiel, and J. D. Tardós, "Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam," *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1874–1890, 2021.

[5] R. Bonghi, "jetson-stats: A package for monitoring and control your nvidia jetson," https://github.com/rbonghi/jetson_stats, 2023.

[6] D. Kiss-Illés, C. Barrado, and E. Salamí, “Gps-slam: an augmentation of the orb-slam algorithm,” *Sensors*, vol. 19, no. 22, p. 4973, 2019.

[7] W. Hess, D. Kohler, H. Rapp, and D. Andor, “Real-time loop closure in 2d lidar slam,” in *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2016, pp. 1271–1278.

[8] A. R. Sahili, S. Hassan, S. Sakhrieh, J. Mounsef, N. Maalouf, B. Arain, and T. Taha, “A survey of visual slam methods,” *IEEE Access*, 2023.

[9] Y. Chen, Y. Zhou, Q. Lv, and K. K. Deveerasetty, “A review of v-slam,” in *2018 IEEE International Conference on Information and Automation (ICIA)*. IEEE, 2018, pp. 603–608.

[10] K. Kamarudin, S. M. Mamduh, A. Y. M. Shakaff, S. M. Saad, A. Zakaria, A. H. Abdullah, and L. M. Kamarudin, “Method to convert kinect’s 3d depth data to a 2d map for indoor slam,” in *2013 IEEE 9th international colloquium on signal processing and its applications*. IEEE, 2013, pp. 247–251.

[11] R. F. Munguía-Alcalá and A. Grau-Saldes, “Slam con mediciones angulares: método por triangulación estocástica,” *Ingeniería, investigación y tecnología*, vol. 14, no. 2, pp. 257–274, 2013.

[12] M. U. Khan, S. A. A. Zaidi, A. Ishtiaq, S. U. R. Bukhari, S. Samer, and A. Farman, “A comparative survey of lidar-slam and lidar based sensor technologies,” in *2021 Mohammad Ali Jinnah University International Conference on Computing (MAJICC)*. IEEE, 2021, pp. 1–8.

[13] R. Olivera, R. Olivera, O. Vite, H. Gamboa, M. A. Navarrete, and C. A. Rivera, “Application of the three state kalman filtering for moving vehicle tracking,” *IEEE Latin America Transactions*, vol. 14, no. 5, pp. 2072–2076, 2016.

[14] A. Gil, O. Reinoso, L. Payá, and M. Ballesta, “Influencia de los parámetros de un filtro de partículas en la solución al problema de slam,” *IEEE LATIN AMERICA TRANSACTIONS*, vol. 6, no. 1, 2008.

[15] G. P. Huang, A. I. Mourikis, and S. I. Roumeliotis, “A quadratic-complexity observability-constrained unscented kalman filter for slam,” *IEEE Transactions on Robotics*, vol. 29, no. 5, pp. 1226–1243, 2013.

[16] M. Tang, Z. Chen, and F. Yin, “Robot tracking in slam with masreliez-martin unscented kalman filter,” *International Journal of Control, Automation and Systems*, vol. 18, pp. 2315–2325, 2020.

[17] I. Ullah, Y. Shen, X. Su, C. Esposito, and C. Choi, “A localization based on unscented kalman filter and particle filter localization algorithms,” *IEEE Access*, vol. 8, pp. 2233–2246, 2019.

[18] Davison, “Real-time simultaneous localisation and mapping with a single camera,” in *Proceedings Ninth IEEE International Conference on Computer Vision*. IEEE, 2003, pp. 1403–1410.

[19] R. Jamiruddin, A. O. Sari, J. Shabbir, and T. Anwer, “Rgb-depth slam review,” *arXiv preprint arXiv:1805.07696*, 2018.

[20] B. Gao, H. Lang, and J. Ren, “Stereo visual slam for autonomous vehicles: A review,” in *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 2020, pp. 1316–1322.

[21] P. P. Shinde and S. Shah, “A review of machine learning and deep learning applications,” in *2018 Fourth international conference on computing communication control and automation (ICCUBEA)*. IEEE, 2018, pp. 1–6.

[22] S. Li, D. Zhang, Y. Xian, B. Li, T. Zhang, and C. Zhong, “Overview of deep learning application on visual slam,” *Displays*, p. 102298, 2022.

[23] W. Dang, L. Xiang, S. Liu, B. Yang, M. Liu, Z. Yin, L. Yin, and W. Zheng, “A feature matching method based on the convolutional neural network.” *Journal of Imaging Science & Technology*, vol. 67, no. 3, 2023.

[24] W. Mai and Y. Watanabe, “Feature-aided bundle adjustment learning framework for self-supervised monocular visual odometry,” in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 9160–9167.

[25] Y. Nakajima and H. Saito, “Robust camera pose estimation by viewpoint classification using deep learning,” *Computational Visual Media*, vol. 3, pp. 189–198, 2017.

[26] H. Pu, J. Luo, G. Wang, T. Huang, and H. Liu, “Visual slam integration with semantic segmentation and deep learning: A review,” *IEEE Sensors Journal*, 2023.

[27] I. Wieser, A. V. Ruiz, M. Frassl, M. Angermann, J. Mueller, and M. Lichtenstern, “Autonomous robotic slam-based indoor navigation for high resolution sampling with complete coverage,” in *2014 IEEE/ION Position, Location and Navigation Symposium-PLANS 2014*. IEEE, 2014, pp. 945–951.

[28] B. Bescos, J. M. Fácil, J. Civera, and J. Neira, “Dynaslam: Tracking, mapping, and inpainting in dynamic scenes,” *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 4076–4083, 2018.

[29] C. Yu, Z. Liu, X.-J. Liu, F. Xie, Y. Yang, Q. Wei, and Q. Fei, “Ds-slam: A semantic visual slam towards dynamic environments,” in *2018 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2018, pp. 1168–1174.

[30] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[31] ——, “Identity mappings in deep residual networks,” in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*. Springer, 2016, pp. 630–645.

[32] M. K. Panda, B. N. Subudhi, T. Veerakumar, and V. Jakhetiya, “Modified resnet-152 network with hybrid pyramidal pooling for local change detection,” *IEEE Transactions on Artificial Intelligence*, 2023.

[33] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[34] K. He, G. Gkioxari, P. Dollár, and R. Girshick, “Mask r-cnn,” in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[35] H. Bavle, P. De La Puente, J. P. How, and P. Campoy, “Vps-slam: Visual planar semantic slam for aerial robotic systems,” *IEEE Access*, vol. 8, pp. 60 704–60 718, 2020.

[36] Stanford Artificial Intelligence Laboratory et al., “Robotic operating system.” [Online]. Available: https://www.ros.org

[37] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, “Orb-slam: a versatile and accurate monocular slam system,” *IEEE transactions on robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[38] R. Mur-Artal and J. D. Tardós, “Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras,” *IEEE transactions on robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[39] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, “Orb: An efficient alternative to sift or surf,” in *2011 International conference on computer vision*. Ieee, 2011, pp. 2564–2571.

[40] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, “Brief: Binary robust independent elementary features,” in *Computer Vision–ECCV 2010: 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part IV 11*. Springer, 2010, pp. 778–792.

[41] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, “Rethinking atrous convolution for semantic image segmentation,” *arXiv preprint arXiv:1706.05587*, 2017.

[42] Y. Sun, B. Pan, and Y. Fu, “Lightweight deep neural network for real-time instrument semantic segmentation in robot assisted minimally invasive surgery,” *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3870–3877, 2021.

[43] K. He, X. Zhang, S. Ren, and J. Sun, “Spatial pyramid pooling in deep convolutional networks for visual recognition,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, no. 9, pp. 1904–1916, 2015.

[44] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, *et al.*, “Searching for mobilenetv3,” in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 1314–1324.

[45] StereoLabs, “Zed - stereo camera for depth sensing,” https://www.stereolabs.com/zed/, 2023, accedido: 2023-09-28.

[46] 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 *European conference on computer vision*. Springer, 2014, pp. 740–755.

[47] 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.

[48] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, “Vision meets robotics: The kitti dataset,” *International Journal of Robotics Research (IJRR)*, 2013.

[49] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, “A benchmark for the evaluation of rgb-d slam systems,” in *Proceedings of the International Conference on Intelligent Robot Systems (IROS)*, Vilamoura, Algarve, Portugal, October 2012.

[50] D. Esparza and G. Flores, “The stdyn-slam: a stereo vision and semantic segmentation approach for vslam in dynamic outdoor environments,” *IEEE Access*, vol. 10, pp. 18 201–18 209, 2022.

[51] J. Cheng, Z. Wang, H. Zhou, L. Li, and J. Yao, “Dm-slam: A feature-based slam system for rigid dynamic scenes,” *ISPRS International Journal of Geo-Information*, vol. 9, no. 4, p. 202, 2020.

[52] C. Kerl, J. Sturm, and D. Cremers, “Dense visual slam for rgb-d cameras,” in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 2100–2106.

[53] Y. Ai, T. Rui, M. Lu, L. Fu, S. Liu, and S. Wang, “Ddl-slam: A robust rgb-d slam in dynamic environments combined with deep learning,” *Ieee Access*, vol. 8, pp. 162 335–162 342, 2020.