“Prototipo de sistema embebido para la detección de obstáculos mediante redes convolucionales profundas para su integración en un velero autónomo “

Felipe Ayala Valencia

Proyecto de grado para optar al título de Ingeniero Electrónico

Tutor
Ricardo Andrés Velásquez Vélez, PhD
Profesor, Universidad de Antioquia

Universidad de Antioquia
Facultad de ingeniería, Departamento de electrónica.
Programa de pregrado UdeA
Medellín, Antioquia
2024

Cita

Ayala Valencia (2024) [1]

Referencia Estilo IEEE (2020)	[1]	Ayala Valencia F (2024), "Prototipo de sistema embebido para la detección de obstáculos mediante redes convolucionales profundas para su integración en un velero autónomo", Trabajo de grado, pregrado UdeA, Universidad de Antioquia, Medellín Antioquia UdeA, 2024.

Sistemas Embebidos e Inteligencia Computacional (SISTEMIC).

Universidad de Antioquia - www.udea.edu.co
Rector: John Jairo Arboleda Céspedes.
Decano/Director: Julio César Saldarriaga Molina.
Jefe departamento: Eduard Emiro Rodríguez Ramírez.

Tabla de contenidos

Introducción

1.1 Planteamiento del problema

En la última década, el campo de la navegación autónoma ha experimentado avances notables impulsados por el desarrollo de tecnologías como las redes convolucionales (CNN), cámaras LiDAR y sensores de alta frecuencia. Estos avances han suscitado un considerable interés en la comunidad científica y la industria marítima por su potencial para mejorar la eficiencia y seguridad en el transporte marítimo.

La capacidad de un barco autónomo para detectar y evitar obstáculos en tiempo real es esencial para garantizar una navegación segura y evitar colisiones. En este contexto, la detección y evasión de obstáculos se constituye como un aspecto crítico en el desarrollo de sistemas de navegación autónoma.

En el ámbito académico el desarrollo de sistemas de navegación autónoma para barcos y veleros, la detección de obstáculos es crucial para abordar desafíos significativos en la seguridad marítima y la optimización de operaciones. La identificación temprana y precisa de obstáculos en el entorno marino es fundamental para reducir el riesgo de colisiones y accidentes en el mar, lo que a su vez puede mitigar el impacto ambiental y los costos asociados con daños a la embarcación y pérdida de carga. Además, al proporcionar a los barcos un sistema de detección inteligente, se facilita la toma de decisiones informada y rápida por parte de la tripulación, mejorando así la eficiencia operativa y el cumplimiento de regulaciones marítimas.

En el caso de la navegación autónoma en botes, es importante destacar dos aspectos cruciales que deben considerarse al analizar y solucionar este problema. El primer aspecto es el movimiento del barco debido al oleaje. Este fenómeno puede generar inconvenientes como la pérdida de puntos de referencia, mediciones incorrectas de los objetos presentes y una mayor dificultad al incorporar un sistema capaz de manejar estas variaciones. El segundo aspecto es la recopilación de datos relevantes para la red neuronal. A diferencia de otros tipos de desplazamiento autónomo, como el de los vehículos terrestres, un barco enfrenta condiciones de alta humedad, neblina y lluvia, lo que dificulta la distinción de objetos a corta distancia. Además, el reflejo constante de los objetos en el agua y la menor cantidad de contenido multimedia para el entrenamiento de la red neuronal dificultan aún más el proceso. Aunque existen videos con horas de contenido de botes en desplazamiento, gran parte de ellos no muestran obstáculos, o estos se encuentran muy lejos o distorsionados. Conocer estos dos aspectos permite dar un punto de partida y teniendo en cuenta el tipo de desplazamiento naval en donde las correcciones de rumbo se han de realizar con mayor antelación que la de vehículos terrestres ayuda a enfocar el análisis y solución de este trabajo.

En este contexto, las siguientes tecnologías han permitido dar solución a esta problemática; redes convolucionales, han demostrado ser herramientas poderosas en el campo de la visión por computadora y el aprendizaje automático. Estas redes neuronales artificiales son capaces de procesar datos no estructurados, como imágenes, y extraer características relevantes de manera automática, aprendiendo patrones complejos a partir de grandes conjuntos de datos y realizar inferencias con herramientas tecnológicas que son más asequibles como puede ser el uso de cámaras digitales y computadores; Sensores LiDAR, estos sensores permiten obtener un mapeo de puntos 3D con una sola “Cámara”. existen varios tipos, sensores que simplemente realizan la medición de un único punto, y cámaras que permiten obtener una imagen 2D con la información de distancia de los objetos en un ángulo de visión dado siendo ésta una herramienta de un costo elevado comparado con las otras técnicas; Visión estereoscópica, esta técnica consiste en el uso de dos cámaras separadas por una distancia conocida para obtener una visión en 3D de una escena. A través del análisis de las diferencias entre las imágenes capturadas por cada cámara (disparidad), se puede calcular la profundidad o la distancia a los objetos en la escena, siendo esta técnica más barata que las 2 anteriores y de menor complejidad en hardware, sin embargo,

se debe tener en cuenta la distancia máxima en la que pueden medir objetos y la disposición de las cámaras, debido a estas limitaciones no se presenta como una alternativa viable en el contexto de este proyecto.

El objetivo de este trabajo es desarrollar un sistema de detección de obstáculos basado en redes convolucionales para su aplicación en barcos autónomos. Se pretende utilizar técnicas de aprendizaje profundo para entrenar una red CNN capaz de analizar datos provenientes de cámaras ubicadas en los barcos, utilizando un gimbal como plataforma de soporte de la cámara digital y la integración de un sensor LiDAR de punto único de bajo coste. Todo esto ejecutado en un SBC.

Entre las arquitecturas de redes convolucionales más destacadas se encuentra YOLO ("You Only Look Once"), que ha sido ampliamente utilizada en aplicaciones de detección de objetos en tiempo real. En este trabajo, se propone utilizar YOLOv8, una versión mejorada y optimizada de esta arquitectura, para la detección de obstáculos en barcos autónomos.

La utilización de YOLOv8 permite realizar con mayor facilidad la fase de entrenamiento y medir el desempeño de la red debido a su uso extendido, lo que es crucial para la toma de decisiones y la navegación segura. Además, su capacidad para detectar múltiples objetos simultáneamente y clasificarlos en categorías específicas será de gran utilidad para identificar y evitar obstáculos en el entorno marítimo.

Se asignará prioridad a los objetos detectados en función de su cercanía mediante la implementación de hiperparámetros, y se implementará un sistema de seguimiento que involucre una sincronización entre el gimbal y la red neuronal para determinar la distancia a los objetos y mapear su ubicación, todo ello dentro de un prototipo de bajo costo.

La prueba del prototipo se llevará a cabo en un entorno controlado, sin oleaje ni condiciones climáticas adversas, y se evaluará el rendimiento de la red entrenada mediante pruebas con barcos a escala y obstáculos marítimos comunes, como boyas y ramas.

La estrategia de utilizar redes neuronales convolucionales profundas y sensores LiDAR para la detección de obstáculos es prometedora. La combinación de imágenes representativas de obstáculos marítimos con la precisión de la tecnología LiDAR permite un entrenamiento y detección eficientes, enfocándose únicamente en objetos considerados obstáculos. Esto posibilita el uso de un sensor LiDAR de menor capacidad, más económico y con menor coste computacional. La implementación de redes YOLOv8 y estrategias de transferencia de conocimiento facilita la configuración y entrenamiento de la red neuronal, asegurando buenos resultados. Además, integrar un sistema embebido con un LiDAR unidireccional y un gimbal ofrece una solución eficiente y económica en comparación con los costosos radares y LiDARs de barrido.

1.2 Objetivos

1.2.1 Objetivo General

Desarrollar un sistema embebido para la detección de obstáculos en el agua mediante el uso de redes convolucionales profundas. el sistema estará compuesto de un SBC como unidad de procesamiento principal, un sensor LIDAR para determinar la distancia del obstáculo, una cámara digital para la detección de obstáculos y un gimbal que permita reducir el movimiento de la imagen y poder enfocar el LIDAR a los obstáculos encontrados.

1.2.2 Objetivos Específicos

1. Recopilar y seleccionar un conjunto de datos diverso y representativo de imágenes de obstáculos marítimos que abarquen una variedad de condiciones y escenarios, con el fin de entrenar y validar de manera efectiva la red neuronal convolucional YOLOv8 para la detección precisa de estos obstáculos en entornos marítimos.
2. Desarrollar un sistema de detección de obstáculos marítimos basado en la red neuronal convolucional YOLOv8, entrenada con una base de datos representativa y aplicando estrategias de transferencia de conocimiento para mejorar su precisión en la detección de objetos en entornos marítimos.
3. Implementar un algoritmo de enfoque con gimbal y establecer una estrategia de comunicación y sincronización adecuada entre un sensor LiDAR, el gimbal y la red YOLOv8 para lograr la medición precisa de la distancia de los objetos detectados al barco autónomo en tiempo real, optimizando el sistema mediante un análisis de hiperparámetros que prioricen la detección eficiente de los obstáculos más cercanos.
4. Evaluar el desempeño del sistema integrado, tanto en términos de precisión en la detección de obstáculos como en la medición de la distancia proporcionada por el sensor LiDAR, utilizando un conjunto de pruebas específicas en un entorno marítimo simulado o real.

1.3 Contribuciones

En el desarrollo de este trabajo se construyó un dataset de 6000 imágenes representativas del entorno marítimo. El trabajo presenta un prototipo de detección de obstáculos compuesto por una cámara digital, un LiDAR unidireccional y un gimbal. Se entrenó un modelo neuronal basado en YOLOv8 para detectar 10 categorías de objetos marítimos. Se desarrolló un algoritmo para la estimación de rango de los objetos detectados por la red neuronal.

1.4 Contenido

El marco teórico revisa la navegación autónoma, incluyendo tecnologías como radar, sensores LiDAR y redes neuronales, además de la arquitectura y metodología de entrenamiento de la red neuronal y una explicación de SBC (Single-Board Computer). La metodología detallará los pasos para crear el dataset, entrenar la red neuronal, medir rangos a obstáculos e integrar software y hardware mediante impresión 3D y circuitos impresos. En los experimentos y análisis de resultados, se presentará el desempeño del entrenamiento, el seguimiento y medición en entornos controlados y en tiempo real, concluyendo con una evaluación de la efectividad del sistema y propuestas para futuros trabajos.

2. Marco teórico

2.1 Navegación autónoma

La navegación autónoma se refiere a la capacidad de un vehículo para operar y desplazarse sin intervención humana, utilizando una combinación de sensores, algoritmos y sistemas de control para percibir su entorno, planificar rutas y tomar decisiones en tiempo real. Esta tecnología ha ganado un interés significativo en diversas industrias, incluyendo el transporte marítimo [1], debido a su potencial para mejorar la eficiencia operativa, la seguridad y la reducción de costos.

Como se describe en [2] un vehículo autónomo es aquel sistema que cumpla con las siguientes tecnologías:

- Automatización: sensores y sistemas (p. ej. navegación, control de velocidad, etc.).
- Datos digitales: distribución de información, gestión de datos de vehículos automotrices, información del conductor. sistema de información.
- Interfaz de usuario digital (conductor, operador, vehículo) – interacción conductor-vehículo, tarea optimización, mantenimiento.
- Interconectividad: sistemas de gestión de flotas de vehículos, redes de igual a igual, inteligencia artificial, acumulación de conocimiento.

Algunas tareas fundamentales de la navegación autónoma esta encargada de recopilar la información del entorno y tomar decisiones.

- Percepción del Entorno: implica el uso de diversos sensores para captar información del entorno del vehículo. Esta percepción puede incluir la detección de obstáculos, la identificación de rutas navegables y la evaluación de condiciones ambientales.
- Localización y Mapeo: la capacidad de un vehículo para determinar su posición en el espacio y construir mapas del entorno es esencial. Tecnologías como el GPS, el LiDAR y las cámaras se utilizan para crear mapas precisos y actualizados.
- Planificación de Rutas: basándose en los datos percibidos y los mapas generados, el vehículo debe planificar rutas óptimas que eviten obstáculos y cumplan con los objetivos de navegación.
- Control y Ejecución: finalmente, los sistemas de control ejecutan las decisiones de navegación, ajustando la dirección, velocidad y otros parámetros para mantener el curso planificado.

2.2 Detección de obstáculos y sensores de rango

La detección de obstáculos es un componente esencial en la navegación autónoma, tanto en vehículos terrestres como marítimos. Se refiere a la capacidad de un sistema para identificar y localizar objetos en el entorno que podrían interferir con la trayectoria del vehículo, permitiendo tomar decisiones para evitar colisiones y asegurar una operación segura. En la revisión del estado del arte [3], se realiza una recopilación de las tecnologías utilizadas para estos fines, con el agregado de técnicas y algoritmos para procesar información de los sensores y la fusión de estas 2 procesamiento y algoritmos.

Tecnologías de Sensores Utilizados para la Detección de Obstáculos

- Radar (Radio Detection and Ranging): Utiliza ondas de radio para detectar la presencia y distancia de los objetos. Es especialmente útil en condiciones climáticas adversas y para detectar objetos a larga distancia.
- LiDAR (Light Detection and Ranging): Emplea pulsos láser para medir distancias y crear mapas tridimensionales del entorno. Es muy preciso y proporciona información detallada sobre la forma y posición de los obstáculos. Existen diferentes tipos de sensores LiDAR, desde los que realizan

mediciones de un único punto hasta los que capturan imágenes 3D con información de profundidad.

- Cámaras Estereoscópicas: Utilizan dos cámaras separadas por una distancia conocida para obtener una visión tridimensional del entorno. La disparidad entre las imágenes capturadas permite calcular la profundidad de los objetos.

Algoritmos y Técnicas de Procesamiento

- Redes Neuronales Convolucionales (CNN): utilizadas para procesar imágenes y detectar objetos en tiempo real. Modelos como YOLO (You Only Look Once) permiten realizar detecciones rápidas y precisas de múltiples objetos simultáneamente.
- Fusión de Sensores: combina datos de diferentes tipos de sensores (por ejemplo, LiDAR y cámaras) para obtener una representación más precisa y completa del entorno. Esto mejora la robustez y precisión de la detección de obstáculos.
- Filtros de Kalman y SLAM (Simultaneous Localization and Mapping): técnicas utilizadas para mejorar la precisión de la estimación de la posición de los objetos y del vehículo, así como para construir mapas del entorno en tiempo real.

2.2.1 Radar

Los sensores tipo radar son dispositivos que utilizan ondas electromagnéticas para detectar objetos y medir sus características, como distancia, velocidad, dirección y composición. El principio básico de funcionamiento de un radar implica la transmisión de pulsos de energía electromagnética y la recepción de los ecos reflejados por objetos en su trayectoria. Este proceso se basa en la medida del tiempo de vuelo de las ondas reflejadas, permitiendo calcular la distancia al objeto (Ranging). Las antenas juegan un papel crucial, ya que transmiten y reciben las señales electromagnéticas. Dependiendo de la aplicación, los radares pueden operar en diferentes frecuencias, desde microondas hasta ondas milimétricas, cada una con características específicas de alcance y resolución.

Existen varios tipos de radar, cada uno adaptado a distintos entornos y necesidades. Por ejemplo, el radar pulsado (PR) se utiliza para medir distancias precisas, mientras que el radar Doppler es efectivo para detectar velocidades relativas. Otro tipo importante es el radar de apertura sintética (SAR), utilizado en aplicaciones de imágenes de alta resolución para la cartografía y la monitorización ambiental. Además, el procesamiento de señales radar desempeña un papel crucial en la mejora de la resolución y la discriminación de objetivos, mediante técnicas como el filtrado adaptativo y la compresión de pulsos.

Las aplicaciones de los radares abarcan desde usos militares y de defensa hasta aplicaciones civiles como la navegación marítima y aérea, el control de tráfico, la meteorología y la investigación científica. La capacidad de los radares para operar en condiciones adversas como la niebla o la oscuridad los convierte en herramientas indispensables en una amplia gama de escenarios [4].

2.2.2 LiDAR de barrido

LiDAR (Light Detection and Ranging), es un sensor óptico utilizado en aplicaciones de percepción y mapeo en vehículos autónomos y sistemas de navegación. Funciona emitiendo pulsos de luz láser y midiendo el tiempo que tarda en recibir el eco reflejado por los objetos. Con base en el tiempo de vuelo de la señal láser, se calcula la distancia entre el sensor y los objetos.

El proceso de medición implica la emisión de pulsos láser de alta intensidad y la detección de la luz reflejada por los objetos a través de un receptor fotosensible. Utilizando la velocidad de la luz, se estima la distancia entre

el sensor y los objetos mediante el cálculo del tiempo de vuelo.

El sensor LiDAR genera nubes de puntos tridimensionales al combinar el tiempo de vuelo de cada pulso láser con la posición y orientación del sensor. Cada punto en la nube de puntos representa un objeto en el entorno, y contiene coordenadas espaciales (x, y, z) que describen su posición en el espacio tridimensional. La resolución del sensor LiDAR se refiere a la capacidad de distinguir objetos cercanos entre sí, mientras que la precisión se relaciona con la exactitud de las mediciones de distancia realizadas por el sensor. Existen diferentes tipos de LiDAR, como el de estado sólido, que utiliza matrices de fotodetectores, y el mecánico, que emplea un escaneo mecánico giratorio para emitir y recibir pulsos láser en diferentes direcciones.

El sensor LiDAR tiene numerosas aplicaciones, como la generación de mapas tridimensionales, la detección de obstáculos, la navegación autónoma, la monitorización ambiental y la robótica, entre otras.

En el artículo [5] se describe el funcionamiento de estos dispositivos y su aplicación en distintos niveles para navegación de vehículos autónomos junto con el trabajo [6] donde se expone el proceso para fusionar ambos resultados del sensor LiDAR y una red CNN para el mapeo de obstáculos se presenta entonces la segunda opción para la medición y detección de obstáculos.

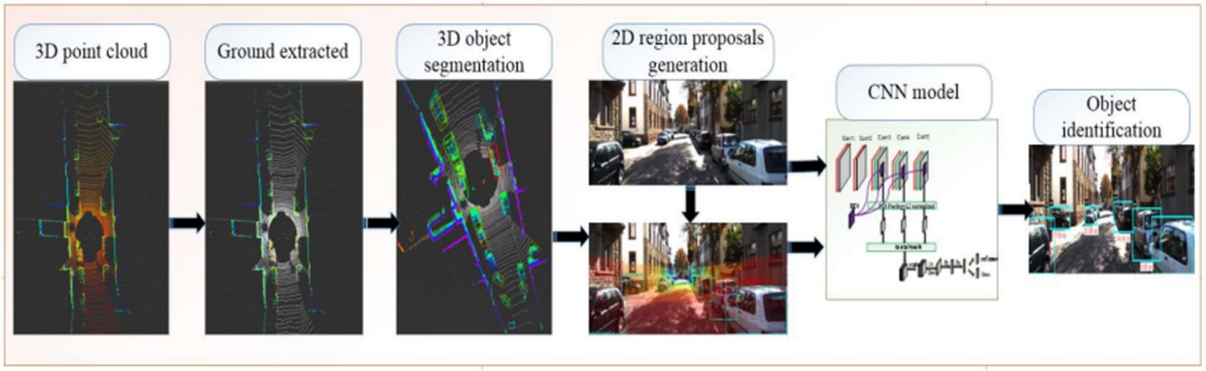


Fig 1. Propuesta de fusión presentada en [6].

Como se puede ver en la figura 1, este método es el más empleado en el estado del arte de detección de obstáculos de vehículos autónomos. Se basa en utilizar un sensor para medir distancia capaz de crear una nube de puntos, se utiliza un sensor LIDAR de alta frecuencia, se combina con la inferencia realizada por una CNN del frame de una cámara digital y se fusionan ambos paquetes de datos. con esto se trata de mapear el entorno y reconocer dentro de la nube de puntos los obstáculos importantes.

2.2.3 Visión estereoscópica

Se usa para calcular la distancia a los objetos en su entorno con el uso de 2 cámaras digitales [7]. Dos cámaras estéreo pueden capturar imágenes simultáneamente desde diferentes puntos de vista. A partir de estas imágenes, el sensor utiliza algoritmos de visión por computadora para calcular la disparidad entre los pares de píxeles correspondientes en ambas imágenes. La disparidad representa la diferencia de posición de un objeto en las dos imágenes y proporciona información sobre la distancia relativa del objeto respecto a la cámara. Como se puede observar en la figura 2, donde se observa una imagen 2D en forma de mapa de calor que representa la distancia de cada punto.

Fig 2. Imagen obtenida del sensor de profundidad de la cámara oak-d lite. Ésta tecnología permite la medición de distancia utilizando 2 cámaras o lentes separados a una distancia conocida. Es utilizada como un método de bajo costo para sistemas embebidos. Al usarse en conjunto con redes convolucionales se puede obtener la distancia de objetos que se consideren importantes.

El principio de funcionamiento se basa en la triangulación estéreo, donde se utiliza la información de la disparidad para calcular la profundidad de los objetos en la escena. Se estima la disparidad midiendo el desplazamiento de los puntos de interés en las imágenes estéreo. Cuanto mayor sea la disparidad, más cerca estará el objeto de la cámara.

Una vez obtenida la disparidad, se utiliza la geometría de la cámara estéreo y los parámetros intrínsecos y extrínsecos para convertir la disparidad en una medida de distancia precisa. Estos parámetros incluyen la distancia focal de la cámara, la línea base (distancia entre las dos cámaras estéreo) y la orientación relativa de las cámaras. Con esta información, se puede calcular la distancia en unidades físicas, como metros o centímetros.

El sensor proporciona una nube de puntos en 3D como se puede observar de la figura 2. que representa la geometría de la escena y la ubicación espacial de los objetos detectados. Estos datos tridimensionales se pueden utilizar para aplicaciones como la detección de obstáculos, la navegación autónoma, la realidad aumentada y muchos otros campos.

Se debe tener en cuenta las limitaciones técnicas posee este tipo de soluciones, la cual se presenta a continuación:

1) Disparidad (d):

$$d = xL - xR$$

2) Distancia (Z):

$$Z = \frac{fB}{d}$$

Donde f: Distancia Focal, B: distancia entre cámaras, d: disparidad, suponiendo que:

- Distancia focal (f): 8 mm
- Baseline (B): 120 mm
- Resolución mínima de disparidad (d): 1 píxel

La máxima distancia medible seria de 0.96m, suponiendo una distancia de 1.2m entre cámaras que se instalaría en un velero a escala.

2.2.4 Detección de objetos

La detección de objetos es una tarea fundamental en el campo del procesamiento de imágenes y visión por computadora, que consiste en identificar la presencia, ubicación y clase de objetos específicos dentro de imágenes o vídeos. Este proceso es crucial para aplicaciones que van desde la vigilancia y la seguridad hasta la conducción autónoma y la realidad aumentada. En donde se puede dar cuenta de 2 métodos bastantes comunes [8].

El enfoque de ventana deslizante es un método tradicional utilizado en la detección de objetos que divide la imagen en múltiples ventanas pequeñas y aplica un clasificador en cada una para determinar si contiene un objeto de interés. Este método implica:

- División de la Imagen: la imagen de entrada se divide en regiones superpuestas o no superpuestas de tamaño fijo (ventanas) que recorren toda la imagen con un desplazamiento determinado.
- Aplicación del Clasificador: un clasificador, típicamente una SVM (Support Vector Machine) o un clasificador basado en características como Histogramas de Gradientes Orientados (HOG), se aplica a cada ventana para determinar si contiene o no un objeto de una clase específica.
- Desafíos: es computacionalmente costoso debido a la gran cantidad de ventanas a evaluar, lo cual puede ser impráctico para imágenes de alta resolución y múltiples clases de objetos.

Las redes neuronales convolucionales (CNN) han revolucionado la detección de objetos al permitir un aprendizaje jerárquico de características directamente de las imágenes. Este enfoque se caracteriza por:

- Extracción de Características: utiliza capas convolucionales para extraer características espaciales de las imágenes de entrada, como bordes, texturas y patrones, de manera eficiente.
- Arquitecturas Modernas: incluyen capas convolucionales, capas de agrupación (pooling) para reducir la dimensionalidad, y capas completamente conectadas para la clasificación final.
- Detección de Objetos: integrando capas de detección, como las propuestas de región (region proposals), con el resto de la red para localizar y clasificar objetos en una sola pasada.
- Ventajas: proporciona altos niveles de precisión y eficiencia computacional, especialmente en comparación con enfoques tradicionales como la ventana deslizante.

Mientras que la ventana deslizante es útil para tareas donde la localización precisa no es crítica o para problemas de detección de objetos simples, las CNNs son ideales para aplicaciones donde se requiere alta precisión en la

localización y clasificación de múltiples objetos en una sola imagen.

Las CNNs se utilizan ampliamente en aplicaciones de visión por computadora como conducción autónoma, vigilancia inteligente, reconocimiento facial, diagnóstico médico por imágenes, entre otros, debido a su capacidad para aprender representaciones jerárquicas complejas de las imágenes.

2.2.5 Tracking

Los algoritmos de seguimiento en redes neuronales convolucionales (CNN) se utilizan para rastrear objetos a lo largo del tiempo en secuencias de video o imágenes continuas. Estos algoritmos combinan la capacidad de detección de objetos de las CNN con técnicas de seguimiento para mantener la identificación de objetos individuales a través de múltiples frames. Aquí se describe el funcionamiento básico de estos algoritmos:

1. Detección Inicial: en el primer frame, una CNN detecta y clasifica todos los objetos presentes, asignándoles ID únicos. La red identifica características distintivas de cada objeto, como forma, color y textura.
2. Asignación de ID: a cada objeto detectado se le asigna un ID único. Este ID se mantiene a lo largo de la secuencia de video, permitiendo rastrear cada objeto individualmente.
3. Seguimiento en Frames Sucesivos: en frames posteriores, el algoritmo vuelve a detectar objetos y compara las características de los objetos recién detectados con las de los objetos rastreados en el frame anterior. Técnicas de asociación, como el algoritmo de Hungarian, se utilizan para emparejar los objetos detectados con los ya existentes basándose en la similitud de características y la proximidad espacial.
4. Predicción de Movimiento: algunos algoritmos incorporan modelos de predicción de movimiento, como el filtro de Kalman, para anticipar la posición futura de los objetos. Esto ayuda a mantener el seguimiento incluso cuando los objetos se mueven rápidamente o se ocultan temporalmente.
5. Mantenimiento de ID: al emparejar los objetos detectados con los rastreados previamente, el algoritmo actualiza sus posiciones y características, manteniendo la coherencia de los ID asignados. Si un objeto no se detecta en un frame, su ID puede ser reservado por un número determinado de frames en espera de su reaparición.
6. Corrección y Refinamiento: a lo largo del tiempo, el algoritmo puede ajustar y refinar las características de los objetos rastreados para mejorar la precisión del seguimiento.

Estos algoritmos se utilizan en diversas aplicaciones, como:

- Seguridad y Vigilancia: para rastrear personas o vehículos en áreas monitorizadas.
- Automatización Industrial: para contar y seguir objetos en líneas de producción.
- Deportes: para analizar el movimiento de jugadores y equipos en eventos deportivos.
- Conducción Autónoma: para rastrear vehículos y peatones en sistemas de conducción autónoma.

2.3 Redes Neuronales

Las redes neuronales son modelos computacionales inspirados en el funcionamiento del cerebro humano, diseñados para aprender y realizar tareas complejas mediante la interpretación de datos. Su capacidad para capturar y procesar patrones complejos ha transformado diversos campos, incluyendo la visión por computadora, el procesamiento del lenguaje natural, y más.

Las primeras conceptualizaciones de redes neuronales se remontan a los años 40, cuando Warren McCulloch y Walter Pitts propusieron un modelo matemático de neuronas artificiales [9]. Este modelo inicial sentó las bases

teóricas para el desarrollo de las redes neuronales como herramientas de procesamiento de información.

Las redes neuronales están compuestas por capas de neuronas artificiales que procesan la información mediante operaciones matemáticas. Cada neurona aplica una función de activación a una combinación lineal de sus entradas, lo que le permite aprender representaciones complejas de los datos. Las neuronas están conectadas entre sí a través de pesos que se ajustan durante el proceso de entrenamiento. Estos pesos determinan la fuerza y la dirección de la influencia de una neurona sobre otra, permitiendo que la red aprenda a reconocer patrones y realizar predicciones. En los últimos años, han surgido varias arquitecturas de redes neuronales profundas que han demostrado un rendimiento excepcional en diversas tareas de inteligencia artificial.

2.3.1 AlexNet

AlexNet es una arquitectura de red neuronal convolucional (CNN) que marcó un hito significativo en el campo del aprendizaje profundo y la visión por computadora. Desarrollada por Alex Krizhevsky, Ilya Sutskever y Geoffrey Hinton, fue presentada en 2012 [10] y ganó la competencia ImageNet Large Scale Visual Recognition Challenge (ILSVRC) ese mismo año, demostrando una mejora drástica en la precisión del reconocimiento de imágenes en comparación con métodos anteriores.

Aunque el concepto de redes neuronales se remonta a los años 40, el desarrollo de redes profundas se había visto limitado por problemas como el sobreajuste y la complejidad computacional. AlexNet aprovechó avances en hardware, grandes conjuntos de datos etiquetados y técnicas de regularización para superar estos desafíos.

Las redes neuronales profundas no habían sido ampliamente adoptadas debido a su complejidad y requerimientos computacionales. AlexNet demostró la capacidad de las CNN para aprender representaciones jerárquicas de características visuales a partir de grandes volúmenes de datos. Consta de ocho capas en total, siendo las primeras cinco capas convolucionales seguidas de capas de pooling. Las capas convolucionales permiten que la red aprenda características como bordes, texturas y patrones de diferentes niveles de abstracción. Introduce técnicas como dropout (para evitar el sobreajuste) y batch normalization (para estabilizar y acelerar el entrenamiento). Estas técnicas fueron cruciales para mejorar la generalización del modelo y su capacidad para manejar grandes volúmenes de datos. Utiliza la función de activación Rectified Linear Unit (ReLU), que demostró ser más eficaz en el aprendizaje profundo al abordar el problema del desvanecimiento del gradiente y acelerar el proceso de entrenamiento.

2.3.2 VGG16

VGG16 es una arquitectura de red neuronal convolucional (CNN) desarrollada por Karen Simonyan y Andrew Zisserman en 2014 [11]. Esta red es conocida por su simplicidad relativa y su capacidad para aprender representaciones profundas de imágenes con alta precisión.

A principios de la década de 2010, las CNN estaban demostrando su potencial en tareas de visión por computadora, pero las arquitecturas existentes eran menos profundas y complejas en comparación con las arquitecturas modernas como VGG16.

VGG16 se destacó por su diseño uniforme y profundo, utilizando filtros convolucionales de tamaño pequeño (3x3) y capas de pooling, lo que facilitó el aprendizaje de características complejas en imágenes de gran escala, consta de dieciséis capas, todas ellas utilizando filtros convolucionales de tamaño 3x3 y capas de pooling de 2x2, seguidas de tres capas totalmente conectadas en la parte final para la clasificación. Las capas convolucionales permiten aprender características locales y las capas de pooling reducen la dimensionalidad

espacial, mejorando la invarianza a la translación y reduciendo el número de parámetros en comparación con arquitecturas más simples. Al igual que AlexNet, VGG16 utiliza la función de activación ReLU, que ha demostrado ser más efectiva para el entrenamiento de redes profundas al mitigar el problema del desvanecimiento del gradiente.

2.3.3 ResNet (Redes Residuales)

ResNet, introducida por He et al. en 2015 [12], representa un avance significativo en el diseño de arquitecturas de redes neuronales profundas. A diferencia de las redes convencionales, ResNet aborda el desafío de entrenar redes extremadamente profundas sin sufrir degradación en el rendimiento mediante la introducción de bloques residuales.

A principios de la década de 2010, las arquitecturas como AlexNet y VGG demostraron el potencial de las redes neuronales profundas para tareas de visión por computadora. Sin embargo, a medida que las redes se volvían más profundas, surgieron problemas como el desvanecimiento del gradiente, que limitaban la capacidad de entrenar y optimizar modelos más grandes y complejos.

ResNet propuso el uso de bloques residuales, que permiten que las capas de la red aprendan las diferencias residuales en lugar de intentar aprender funciones completas. Esto se logra mediante la introducción de conexiones de salto, que saltan una o más capas, facilitando el flujo de gradiente durante el entrenamiento y mitigando el problema de la degradación del rendimiento. Cada bloque residual consta de dos o tres capas convolucionales seguidas de una conexión de salto que suma la entrada original a la salida del bloque. Esto permite que la red aprenda tanto las características originales como las diferencias residuales, facilitando el entrenamiento de redes extremadamente profundas. Se caracteriza por su capacidad para manejar redes con más de cien capas, mejorando la precisión y la generalización sin aumentar excesivamente la complejidad computacional.

2.3.4 Yolo

YOLO es un algoritmo de detección de objetos en imágenes que se distingue por su eficiencia y precisión. A diferencia de otros enfoques que requieren múltiples etapas para detectar y clasificar objetos, YOLO realiza la detección en una sola pasada de la red neuronal convolucional (CNN), lo que permite una velocidad de procesamiento más rápida y adecuada para aplicaciones en tiempo real.

El funcionamiento de YOLO se basa en dividir la imagen de entrada en una malla de celdas y predecir cajas delimitadoras (bounding boxes) y probabilidades de clase para cada celda. Cada caja delimitadora está asociada con un conjunto de puntuaciones que indican la confianza de que la caja contiene un objeto y la probabilidad de cada clase. Utiliza una única red neuronal para predecir estos parámetros de salida, optimizando así el proceso de detección y clasificación.

Se hará uso de la arquitectura YOLOV8 la más reciente al momento del inicio del trabajo de investigación, que fue lanzada en enero de 2023, bajo la restricción de que esta arquitectura deberá funcionar en un sistema embebido, portátil, el cual ya tiene otros componentes como el control del velero, telemetría a bordo. Se busca una arquitectura compacta para ahorrar en costo computacional. Como se detalla en [13], donde se menciona las aplicaciones centrales de la arquitectura YOLO desde su versión 1 hasta la versión 8, como puede ser los sistemas robóticos, vehículos autónomos y aplicaciones de monitoreo de video

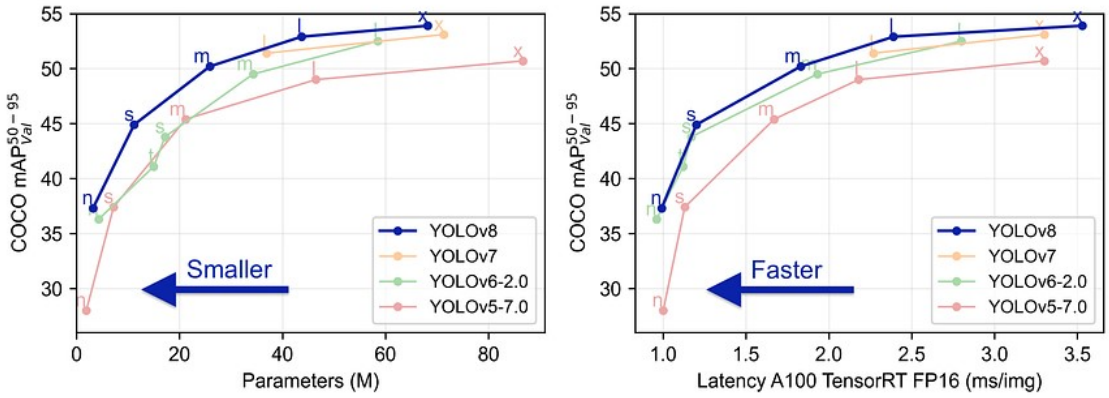


Fig 3. Grafica de rendimiento entre las versiones disponibles de la arquitectura YOLO.

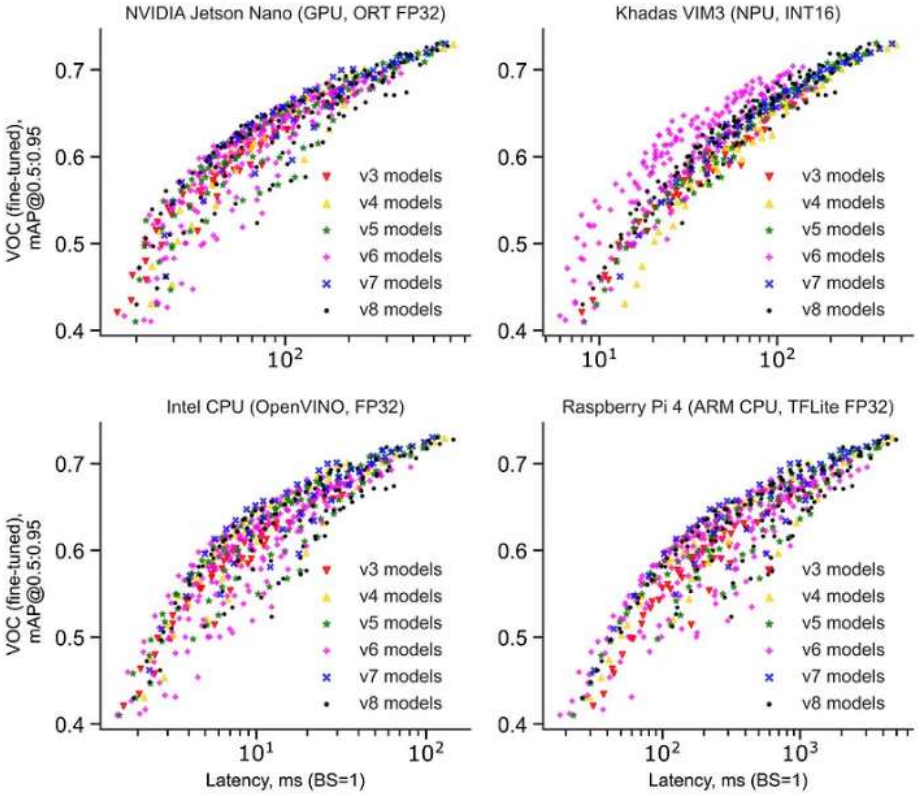


Fig 4. Grafica de rendimiento (tiempo de inferencia) entre las distintas versiones de la arquitectura YOLO.

Las figuras 3 y 4 comparan el rendimiento de diferentes versiones de YOLOv8 en la detección de obstáculos. Se observa que YOLOv8 obtiene los mejores resultados en las pruebas realizadas, aunque la diferencia con otras versiones no es significativa. En la figura 3, se destaca que YOLOv8 requiere menos cantidad de parámetros que la versión 6, la cual ofrece un rendimiento similar. Esta característica es crucial en sistemas embebidos, donde la memoria y la potencia de procesamiento son limitadas. Por lo tanto, la elección de YOLOv8 para la detección de obstáculos en un sistema embebido se justifica por su óptimo balance entre rendimiento y eficiencia.

Esta versión de YOLO permite poder entrenar una versión “nano” para ser implementada en el sistema

embebido además de contar con las últimas mejoras en eficiencia y versatilidad como se menciona en [13]. cuenta con las siguientes mejoras.

YOLOv8 es una versión mejorada de YOLO que incorpora varias optimizaciones para mejorar tanto el rendimiento como la precisión de detección. Mantiene el enfoque de procesar la imagen en una sola pasada, pero introduce técnicas más avanzadas para mejorar la estabilidad del entrenamiento y la precisión del modelo. Estas mejoras incluyen el uso de promedios móviles para los parámetros entrenados, la aplicación de técnicas de regularización para evitar el sobreajuste, y la inclusión de métricas adicionales para evaluar y mejorar la precisión de la localización de los objetos. Durante la inferencia, YOLOv8 utiliza una combinación de la probabilidad de clasificación, la puntuación de objetividad y una métrica de precisión de localización para calcular la detección final, asegurando así resultados más precisos y fiables

2.3.5 Medida de confianza en CNN

La métrica de confianza en la detección de objetos por medio de redes neuronales es un componente crucial en sistemas de visión por computadora y aprendizaje profundo. Esta métrica se refiere a la probabilidad asignada por la red a la presencia de un objeto en una región específica de una imagen, lo cual afecta directamente la precisión y fiabilidad del sistema de detección. La métrica de confianza en la detección de objetos se representa comúnmente como un valor entre 0 y 1, donde un valor más alto indica una mayor certeza de la presencia del objeto. Esta métrica se deriva de las salidas de la red neuronal, específicamente de la capa de activación final, que utiliza funciones de probabilidad, como Softmax o Sigmoid, para asignar una probabilidad a cada clase de objeto.

2.3.6 Transfer Learning

El aprendizaje por transferencia (transfer learning) es una técnica en el campo del aprendizaje automático y la inteligencia artificial donde un modelo desarrollado para una tarea se reutiliza como punto de partida para un modelo en una segunda tarea relacionada. Esta metodología es especialmente útil cuando se dispone de un conjunto de datos grande y bien etiquetado para una tarea inicial, pero los datos para la nueva tarea son limitados.

Se basa en la idea de que los modelos entrenados en una tarea pueden aprovechar sus conocimientos previos al enfrentar una tarea nueva y relacionada. Hay dos enfoques principales en transfer learning [14]:

Transferencia de Características (Feature Transfer):

- En este enfoque, se toma una red preentrenada en un conjunto de datos grande (por ejemplo, ImageNet) y se utiliza como base para una tarea específica.
- Las capas iniciales de la red (conocidas como capas de características) se mantienen fijas, ya que generalmente aprenden características generales útiles en muchas tareas.
- Las capas finales (capas de clasificación) se adaptan o se entrenan desde cero para la tarea específica.

Aprendizaje por Afinidad (Fine-tuning):

- En lugar de mantener fijas todas las capas de características, se permite ajustar algunas capas específicas durante el entrenamiento en la nueva tarea.
- Esto es beneficioso cuando las características aprendidas en las capas más profundas de la red preentrenada son relevantes para la nueva tarea.

Se consiguen los siguientes beneficios del Transfer Learning en una CNN. Reducción del tiempo de entrenamiento al utilizar una red preentrenada, se puede reducir significativamente el tiempo necesario para entrenar un modelo en una tarea específica, ya que las capas iniciales ya han aprendido características útiles. Mejora del Rendimiento al transferir conocimientos de tareas previas puede mejorar el rendimiento en tareas nuevas, especialmente cuando los conjuntos de datos de entrenamiento son limitados. Generalización Mejorada de las características aprendidas en tareas previas pueden capturar patrones genéricos y útiles, lo que mejora la capacidad del modelo para generalizar en nuevas tareas. Manejo de Conjuntos de Datos Pequeños en situaciones donde hay pocos datos disponibles para la tarea de interés, el transfer learning puede ser crucial para construir modelos efectivos.

2.4 SBCs

El SBC es un tipo de computadora que se caracteriza por tener todos los componentes esenciales en una única placa de circuito. Esto incluye procesador, memoria RAM, almacenamiento, puertos de entrada/salida, y a veces, incluso módulos de conectividad como Wi-Fi y Bluetooth. Aunque suelen ser de menor potencia que las computadoras de escritorio convencionales, los SBCs son ideales para aplicaciones embebidas.

La Raspberry Pi es uno de los SBCs más populares. Algunas de sus características clave incluyen:

La Raspberry Pi utiliza procesadores de bajo consumo, como los de la arquitectura ARM, que son eficientes en términos de energía y adecuados para una variedad de aplicaciones. Los SBCs suelen incluir pines GPIO que permiten la conexión de dispositivos y sensores externos, lo que los hace ideales para proyectos de electrónica y robótica. Los SBCs pueden ejecutar sistemas operativos ligeros, como Raspbian en el caso de la Raspberry Pi, adaptados a sus recursos limitados. Raspberry Pi tiene una gran comunidad de usuarios y una amplia variedad de accesorios y complementos disponibles, lo que facilita el desarrollo de proyectos y la resolución de problemas a través del intercambio de conocimientos.

3. Metodología

En este capítulo se describen los métodos y procedimientos utilizados para desarrollar y validar el sistema diseñado para este trabajo de grado. La metodología se estructura en varias etapas clave, que abarcan desde la recopilación y preparación de datos hasta la implementación de las soluciones propuestas en un entorno práctico. Los pasos principales incluyen la construcción del Dataset con la recopilación de imágenes obtenidas de datasets disponibles en kaggle. Imágenes de frames de fuentes multimedia y de capturas realizadas en el entorno controlado con el uso de las maquetas que se utilizaran posteriormente. El entrenamiento del modelo YOLO v8 para la detección de objetos mediante transfer-learning. Se mencionará las metodologías de tracking para la medición de objetos que se encuentren en pantalla mediante el uso del gimbal como herramienta de enfoque y tracking. El desarrollo de estrategias para la medición del rango a obstáculos mediante la definición de hiperparametros, y la implementación e integración de estas soluciones en el barco realizando la sincronización con todos los componentes.

3.1 Dataset

Para la recolección de datos y creación de la base de datos, se usaron de 3 fuentes, la primera siendo una dataset que se obtuvo de Kaggle “Boat types recognition”, este contiene las clases de; boyas, cruceros, ferris, kayaks y veleros, aproximadamente 1.5K imágenes. Otra fuente fueron videos de YouTube sin derechos de autor, libre para ser usados, videos que tuvieran veleros y boyas en su mayoría, y finalmente la grabación de un par de videos en la Fuente de la Universidad de Antioquia usando maquetas de veleros disponibles en el laboratorio de SISTEMIC. Para un total de 6097 imágenes recolectadas, todas las imágenes durante el proceso de etiquetado son reescaladas a la misma resolución. durante el proceso de etiquetado se crearon un total de 16 clases: Aeroplanos, Botes, Ramas, Puentes, Boyas, Patos, Muelles, Kayaks, Personas, Naufragios, Rocas, Veleros, Cargueros y Surfers. Debido a la poca cantidad de imágenes etiquetadas de algunas clases fueron suprimidas. Reduciendo a 10 clases, los cambios fueron la eliminación de las clases Aeroplanos, Ramas, Patos, Naufragios e Islas, y las imágenes etiquetadas con cargueros se fusionaron con la clase Botes. Finalmente quedo una distribución de 5079 imágenes de entrenamiento (83%), 761 imágenes de validación (12%) y 257 imágenes de prueba (4%). Todo el proceso de etiquetado, creación de clases fue realizado en la plataforma de Roboflow que permite además usar un SDK en Python para entrenar varias versiones también el preprocesamiento y aumento artificial de muestras.

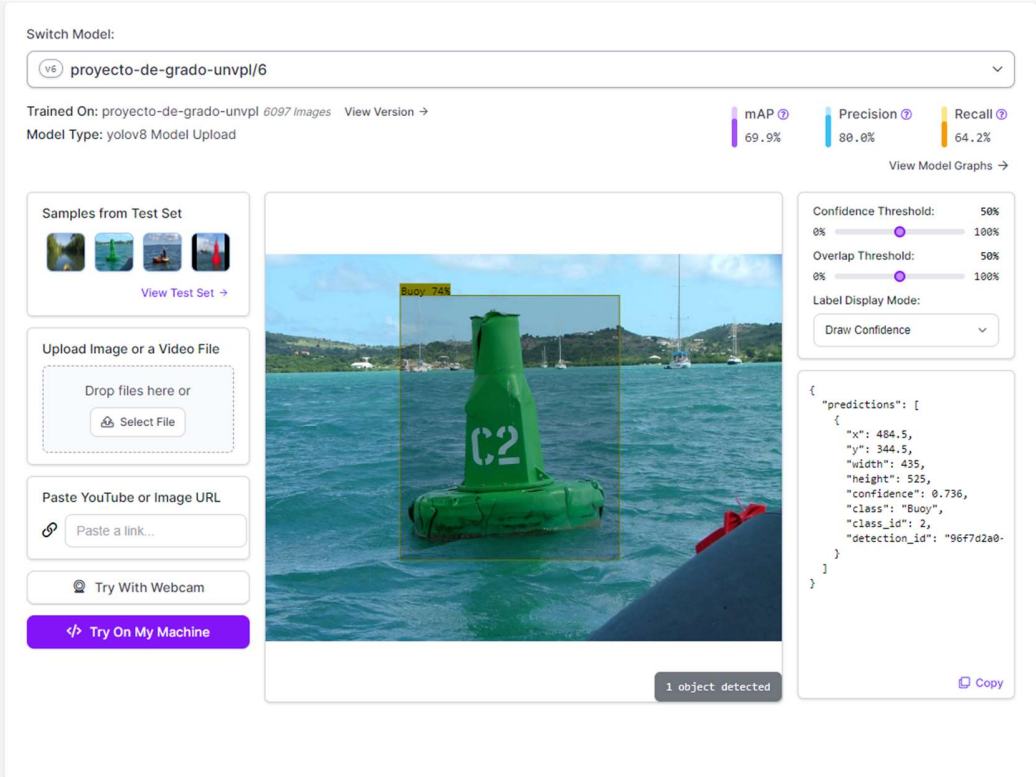


Fig 5. Plataforma para Deployment, etiquetado y entrenamiento de modelos Roboflow.

Para realizar el trabajo de etiquetar las imágenes se usa la plataforma “RoboFlow”, como se expone en la figura 5, plataforma donde se puede realizar cambios en el Dataset mediante aumento artificial de datos, entrenamiento rápido con modelos simples, y visualización de métricas de entrenamiento para su validación.

Clase	N° Imágenes	Porcentaje
Boats	845	5,91%
Briges	37	0,25%
Buoy	1324	9,26%
Dock	97	0,67%
Kayak	576	4,02%
Person	323	2,25%
Rock	282	1,97%
Sail-Boat	10676	74,67%
Surfer	99	0,69%
Wreck	38	0,26%

Tabla 1. Distribución de clases del Dataset.

En la Tabla 1 se muestra la distribución de elementos etiquetados por clase. Se ha puesto especial énfasis en la recolección de un mayor número de muestras para la clase "Sail-boat", ya que el experimento se llevará a cabo utilizando maquetas a escala de este objeto. Este enfoque permite centrar el análisis en el sistema completo, priorizando la detección precisa de la clase "Sail-boat" sobre la detección de múltiples clases por la red. De esta manera, se asegura una evaluación más efectiva del sistema en condiciones que reflejan el objetivo principal del estudio.

3.1.1 Descarga de Datset

1. Instalar la Biblioteca Roboflow:
   !pip install roboflow
   Esto instalará la biblioteca necesaria para interactuar con la API de Roboflow.

2. Importar la Biblioteca Roboflow:
   from roboflow import Roboflow
   Esto importa la clase Roboflow que se utilizará para acceder a los datos del proyecto.

3. Crear una Instancia de Roboflow con la Clave API:
   rf = Roboflow(api_key="xRhu7MLK8D46vKaaBGzP")

4. Acceder al Proyecto y la Versión Deseada:
   project = rf.workspace("udea").project("proyecto-de-grado-unvpl")
   version = project.version(6)
   Aquí, "udea" es el nombre del espacio de trabajo y "proyecto-de-grado-unvpl" es el nombre del proyecto. version(6) selecciona la versión 6 del proyecto.

5. Descargar el Dataset en el Formato Deseado:
   dataset = version.download("yolov8")
   Este comando descargará el dataset en el formato compatible con YOLOv8.

3.2 Entrenamiento de Yolov8

Inicialmente se proponen varias arquitecturas de redes neuronales convolucionales, como VGG16, YOLO, o alguna red diseñada desde cero con el uso de pytorch o tensorflow, para este caso de estudio se ha de tener en cuenta que la red se ejecutara en un sistema embebido por tanto se debe de elegir la red con la mejor eficiencia y que tenga un tamaño más contenido, se elige entonces la Arquitectura de YOLO más específico la versión 8. Esta versión cuenta con varios modelos preentrenados desde la versión “nano”, “small”, “medium”, “large” y “extra-large”, siendo la versión nano la más indicada para ser ejecutada en un SBC.

Para el proceso de entrenamiento, se descarga el Dataset recopilado de la plataforma Roboflow y se procede a entrenar usando como base los pesos de la versión nano como se puede ver en el listings 1, donde se muestra el código necesario para llevar a cabo el entrenamiento.

```
Ejemplo para entrenamiento:
from ultralytics import YOLO

# Load a COCO-pretrained YOLOv8n model
model = YOLO("yolov8n.pt")

# Train the model on the COCO8 example dataset for 100 epochs
results = model.train(data="coco8.yaml", epochs=100, imgsz=640)
```

Listings 1. Código para entrenamiento de modelo.

Aunque las técnicas de transfer learning se puede ejecutar tanto para reentrenar toda la red como solo una parte de ella, generalmente se realiza un reentrenamiento parcial, esto quiere decir que las capas iniciales se congelan, las cuales se encargan de la extracción de características básicas como bordes y texturas, mientras que las capas finales, que aprenden características más abstractas y específicas de la tarea, son reentrenadas. En este caso al descargar el modelo estándar ya entrenado de cualquiera de los tamaños ya sea "nano" o "large" y comenzar a entrenar con un Dataset diferente automáticamente se realiza el congelamiento de las primeras capas y el reentrenamiento de las ultimas capas.

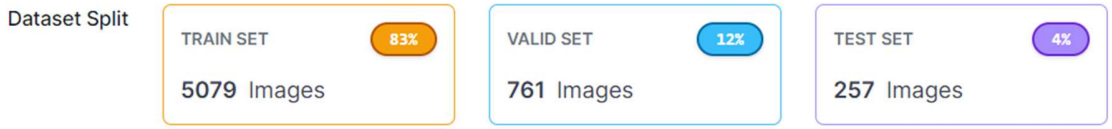


Fig 6. Distribución de imágenes para el entrenamiento.

Para construir modelos de aprendizaje automático robustos y fiables, es crucial distribuir adecuadamente los datos entre los conjuntos de entrenamiento, validación y prueba. Generalmente, el conjunto de entrenamiento debería ser el más grande, representando alrededor del 60-80% del total de datos, lo que proporciona suficiente información para que el modelo pueda aprender patrones significativos. El conjunto de validación, que se utiliza para ajustar los hiperparámetros del modelo y evaluar su rendimiento durante el entrenamiento, suele representar alrededor del 10-20% del total de datos. Finalmente, el conjunto de prueba, que evalúa el rendimiento final del modelo en datos no vistos, generalmente representa alrededor del 10-20% del total de datos. Es importante que los conjuntos de validación y prueba sean independientes del conjunto de entrenamiento para asegurar una evaluación imparcial. En la figura 6 se muestra la distribución de carga para cada una de las 3 fases, teniendo un 83% de imágenes para el entrenamiento, un 13% para validación y un 4% para prueba.

El modelo fue reentrenado durante aproximadamente 310 épocas hasta que no se detectó ninguna mejora dentro del mismo proceso que se encargaba del entrenamiento, utilizando el servidor del grupo de investigación GITA, empleando la API de Roboflow y la biblioteca de Python de Ultralytics, desarrolladores de la arquitectura YOLOv8. Las imágenes utilizadas tenían un tamaño de 640x480 píxeles. Y aquellas que no tuviesen esa resolución nativamente, desde la plataforma de roboflow eran reescaladas a la resolución necesaria.

3.3 Estrategia de tracking de objetos

En una primera instancia, se intentó aplicar un algoritmo de seguimiento que asigna ID únicos a los objetos detectados durante las inferencias, permitiendo identificar y rastrear cada objeto individualmente, incluso si son de la misma clase. Este algoritmo es útil para aplicaciones como el conteo de objetos en movimiento en una determinada área o el seguimiento de personas. La idea era mantener las ID únicas de los objetos a medida que se desplazaban dentro del frame, aplicando similitudes para asegurar la consistencia en su identificación.

Sin embargo, esta implementación fue sustituida por un algoritmo más simple. En este enfoque, se detectan todos los objetos en un solo frame y se enfocan en los centros de los objetos, los valores de los centros serán duplas de puntos de pixeles en horizontal y vertical que posteriormente se procesarán para convertirlos en variaciones del nivel de activación de las señales pwm del gimbal para lograr el enfoque requerido. La razón

para este cambio fue que, al mover el gimbal, se perdía de vista el conjunto de objetos y sus identificaciones únicas. Esto causaba problemas al intentar identificar objetos en frames sucesivos, ya que el algoritmo no podía garantizar que el segundo objeto identificado correspondiera al mismo objeto detectado inicialmente. Por lo tanto, el algoritmo de seguimiento de múltiples objetos no funcionaba correctamente, en este contexto y fue reemplazado por la estrategia mencionada anterior, tiene como ventajas que al realizar la inferencia de un solo frame se puede crear una lista de puntos a los que enfocar el gimbal con la suposicion de que los elementos del entorno marino no tengan una alta velocidad esta estrategia puede pasar por enfocar la posición inicial de estos elementos y que esta posición de medición pueda llegar a medir la distancia de los objetos, por otra parte si los elementos son los suficientemente rápidos o son demasiados se dará el caso en que las mediciones sean incorrectas.

3.4 Estrategias de estimación de rango de obstáculos

El gimbal tiene 2 tareas, la primera es la de contrarrestar el movimiento ocasionado por el oleaje, una cámara fija en el mástil ocasionaría que haya información errónea sobre la posición de los obstáculos al no tener un horizonte fijo, la segunda tarea es la de realizar un enfoque del centro de la cámara al obstáculo al que se mediara la distancia.

El LiDAR por su parte será el sensor encargado de realizar la medición de distancia, valor que será recibido por el microcontrolador y posteriormente procesado por el SBC.

Para la estimación de rango de obstáculos se lleva a cabo el siguiente proceso, después de inicializado el sistema lo que incluye inicializar la captura de video, inicializar el modelo de red neuronal y esperar a la calibración del gimbal, se captura un frame, de este se realiza la inferencia con el fin de detectar obstáculos y sí los hay, delimitarlos con un bounding box, a cada uno guardar las posiciones en el frame de estos elementos junto con el centro de los bounding boxes, se realiza un puntaje medido por los hiperparametros detectados, esto dará el orden con que se realizara el enfoque a cada objeto detectado, una vez ordenado desde el puntaje mayor al menor, se pasa la lista ordenada de puntos a los centros, esta lista que indica punto en frame de la manera X,Y en formato pixeles, se convierte a un nivel de activación de señal PWM para el gimbal, con esta nueva serie de datos se realiza un procedimiento para que las señales del gimbal tengan en cuenta el estado en el que se encuentran, por ejemplo, para no realizar un movimiento de: posición inicial gimbal a objeto repetidamente. Se realiza de objeto a objeto.

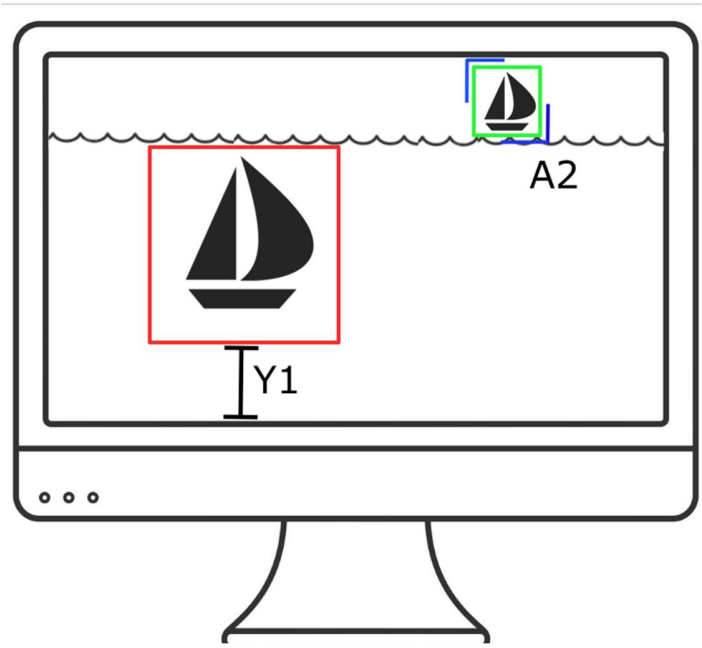


Fig 7. Presentación de los 2 hiperparametros utilizados Y1:H1 distancia del borde del frame al bounding box y A2:H2 el área del bounding box.

Como se muestra en la Figura 7, se proponen 2 hiperparámetros, esto después de realizar un análisis cualitativo de como poder extraer información sobre la distancia de un objeto a la cámara que toma la foto, por ejemplo al tener un entorno nivelado como lo es un cuerpo de agua, se podría decir que cuanto más cerca este el borde inferior del objeto al borde inferior del frame este puede estar más cerca, y saber que tan grande es el objeto podría dar un orden de prioridad a cual debe medirse primero.

1. H1=Y1: Este hiperparámetro se define como la distancia desde el borde inferior del bounding box del objeto hasta el límite inferior del frame capturado por la cámara. Su propósito es proporcionar una estimación de la distancia relativa de los objetos al barco, asumiendo que los obstáculos se encuentran al mismo nivel, pero tienen diferentes alturas. Por ejemplo, en la imagen, el barco grande en el centro tiene un valor Y1 que indica que está más cerca de la cámara en comparación con el barco más pequeño en la parte superior. Por lo tanto, H1 permite inferir cuáles obstáculos están más cerca del barco.
2. H2=A2: Este hiperparámetro se basa en el área del bounding box del objeto. Su función es dar prioridad a los objetos más grandes, considerándolos potencialmente más relevantes o peligrosos. En la figura, el barco grande en el centro tiene un área de bounding box mayor (A2) en comparación con el barco más pequeño en la parte superior. Esto sugiere que el barco más grande puede ser una amenaza más inmediata y, por lo tanto, debería ser priorizado para la medición de distancia.

Estos hiperparámetros permiten al modelo mejorar su capacidad de detección y priorización de obstáculos en el entorno acuático.

3.4.1 Estrategia general

Del diagrama de flujo de la figura 8, bloque principal del programa inicializa la captura de video y la detección de objetos mediante YOLO. Configura hilos para la grabación de video y el chequeo de la señal de salida. Mientras el bucle principal está activo, el sistema captura frames de la cámara, procesa estos frames para

detectar objetos y calcula la posición y el tamaño de estos. Los resultados de la detección se utilizan para generar datos de control y visualización.
En el proceso de configuración del sistema, se inicia con la Inicialización de Componentes. Primero, se establece la captura de video mediante la biblioteca OpenCV, la cual se utiliza para obtener los datos visuales de la cámara digital. Además, se carga el modelo de detección de objetos YOLO, que será fundamental para identificar y clasificar los objetos en los frames capturados.

Durante la ejecución concurrente, se crean dos hilos de procesamiento. El primer hilo se encarga de la grabación del video, asegurando que los datos visuales se registren de manera continua. Simultáneamente, el segundo hilo realiza el chequeo de señales de entrada, permitiendo la interacción con otros componentes del sistema en tiempo real.

En la fase de detección y procesamiento, se capturan los frames del video. Cada frame es procesado para detectar objetos y calcular puntuaciones que ayudan a determinar el objeto más relevante, utilizando los hiperparámetros configurados. Las coordenadas de los objetos detectados se convierten en señales PWM, que se envían a través del puerto serial. Además, se recibe el valor de distancia mediante el sensor LiDAR y se gestiona la conexión con la Raspberry Pi Pico. A partir de un solo frame, se obtienen las coordenadas de todos los objetos detectados, y se desplaza el sistema para medir cada uno de ellos. Los datos visualizados se muestran en un radar y se almacenan para su análisis posterior. Finalmente, en la fase de Finalización, se liberan los recursos utilizados durante el proceso. Los hilos se detienen y se cierran las conexiones seriales, asegurando que el sistema se apague de manera segura y eficiente.

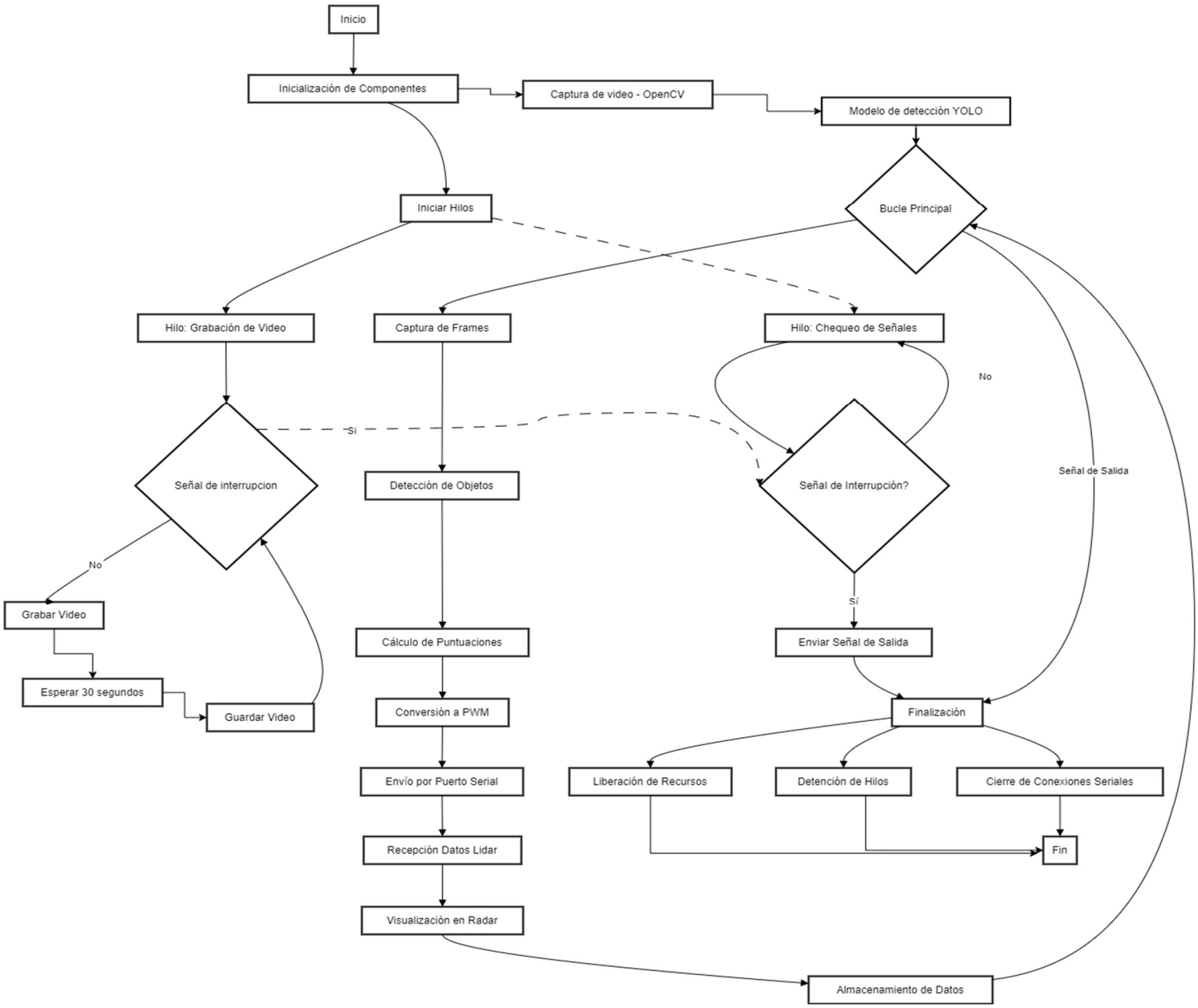


Fig 8. Diagrama seudocódigo de la estrategia general del código.

3.4.2 Manejo del gimbal

El gimbal, un dispositivo crucial para la estabilización de cámaras ofrece varios modos de funcionamiento, entre ellos se puede destacar: autónomo y controlado externamente. En su modo autónomo, el microcontrolador Storm32 se encarga de la estabilización y calibración de la cámara montada en el gimbal. Sin embargo, es el control externo el que permite una manipulación más versátil, esencial para aplicaciones como el control por radio o la captura de videos en drones. Para el manejo de este dispositivo se usarán ambos modos.

El control externo se logra mediante señales de Modulación por Ancho de Pulsos (PWM), funciona modificando el ciclo de trabajo o tiempo en alto de una señal cuadrada de una frecuencia de 50Hz, hablaremos de este ciclo de trabajo como nivel de activación para la señal de pwm dado que es de esta forma en que se maneja en el algoritmo de C implementado. Como es habitual en los controles de este tipo por seal pwm el 7.5% equivale a la posición central del movimiento, en este caso indica la posición inicial a la cual se calibro el gimbal, se describe en el script en C 750 puntos como el 7.5% cuanto menor sea el nivel más tiempo estará en alto la señal y cuanto más alto menor será el tiempo en alto de la señal, gestionadas por un algoritmo escrito en C e implementado en un microcontrolador Raspberry Pi Pico. El proceso de control se divide en dos etapas principales: la conversión de píxeles a ángulos y la subsecuente conversión de estos ángulos a señales PWM.

En la primera etapa, se realiza un cálculo para determinar el ángulo correspondiente a cada píxel, tanto en el eje horizontal como en el vertical. Este cálculo se basa en las especificaciones técnicas de la cámara utilizada. En el caso particular de este sistema, la cámara cuenta con un campo de visión (FOV) horizontal de 55° y un FOV vertical de 42.65°, con una resolución de imagen de 640x480 píxeles. Utilizando estos datos, se obtienen las siguientes medidas angulares por píxel:

- Ángulo por píxel (Horizontal) = 0.085937°
- Ángulo por píxel (Vertical) = 0.088854°

La segunda etapa implica la conversión de estos ángulos a señales PWM. Esta conversión se basa en un análisis empírico del movimiento del gimbal en relación con los niveles de activación PWM. Los experimentos, realizados con una frecuencia PWM de 50Hz, revelaron que:

- Para lograr un movimiento horizontal de 90°, fue necesario un incremento de 200 (950 puntos 9.5% ciclo de trabajo) puntos en el nivel de activación del PWM.
- Para un movimiento vertical de 34° (el máximo permitido con el peso de la cámara y el sensor), fue necesario un aumento de 50 puntos (800 puntos 8% ciclo de trabajo).

De estas observaciones se derivan los siguientes factores de conversión:

- Movimiento horizontal: 90° corresponde a 200 puntos de PWM, lo que da un factor de conversión de 2.22 puntos de PWM por grado.
- Movimiento vertical: 34° corresponde a 50 puntos de PWM, lo que da un factor de conversión de 1.4705 puntos de PWM por grado.

Estas conversiones se pueden expresar matemáticamente como:

$$90° \rightarrow 200$$
$$a° \rightarrow x\ pwm$$
$$xpwm = a° * 2.22, fator\ de\ conversion$$
$$34° \rightarrow 50$$
$$b° \rightarrow y\ pwm$$
$$ypwm = b° * 1.4705, fator\ de\ conversion$$

Donde “xpwm” y “ypwm” son los nuevos valores de activación para ubicar el centro de la cámara en un punto especifico dado por la posición medida en píxel de un frame.

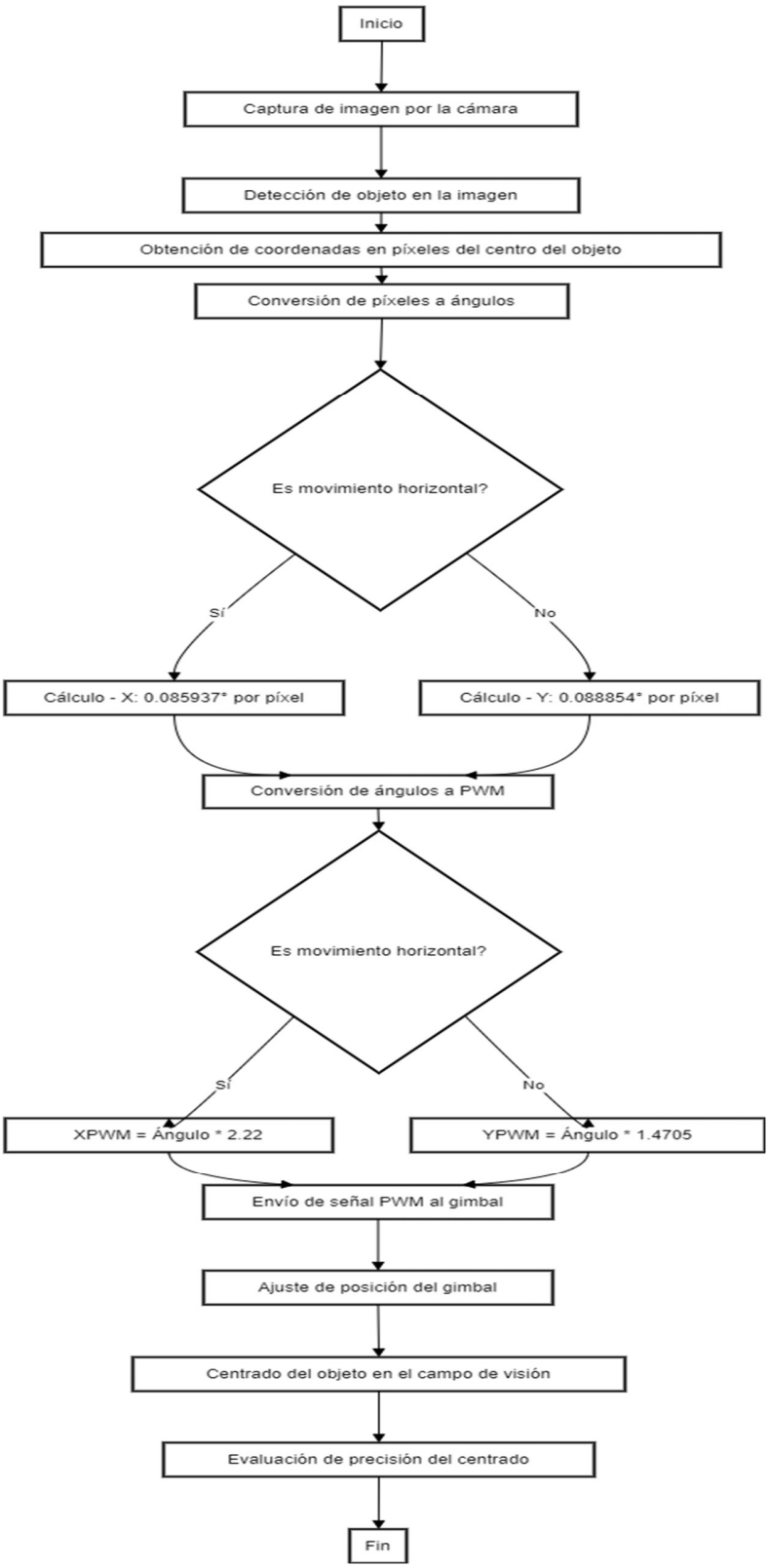


Fig 9. Diagrama de flujo enfoque de gimbal a objeto detectado.

La aplicación práctica de este sistema implica varios pasos como se ve en el diagrama de flujo de la figura 9 y en la figura 10 representativa del proceso. Primero, se detecta un objeto en la imagen capturada por la cámara y se obtienen las coordenadas en píxeles del centro de este objeto. Luego, estas coordenadas se convierten a ángulos utilizando los factores calculados en la primera etapa. Estos ángulos, a su vez, se convierten en señales PWM mediante las fórmulas derivadas en la segunda etapa. Finalmente, estas señales PWM se utilizan para ajustar la posición del gimbal, centrando el objeto detectado en el campo de visión de la cámara.
Para evaluar la eficacia de este método, se realizó una comparación entre la posición del centro del bounding box del objeto detectado antes y después del movimiento de enfoque. Los resultados mostraron una mejora significativa en el centrado de los objetos, lo que demuestra la precisión y utilidad del sistema de control externo del gimbal.
Este enfoque metodológico permite un control estable y preciso del gimbal, ofreciendo una solución efectiva para aplicaciones que requieren un seguimiento preciso de objetos o un control fino del movimiento de la cámara.

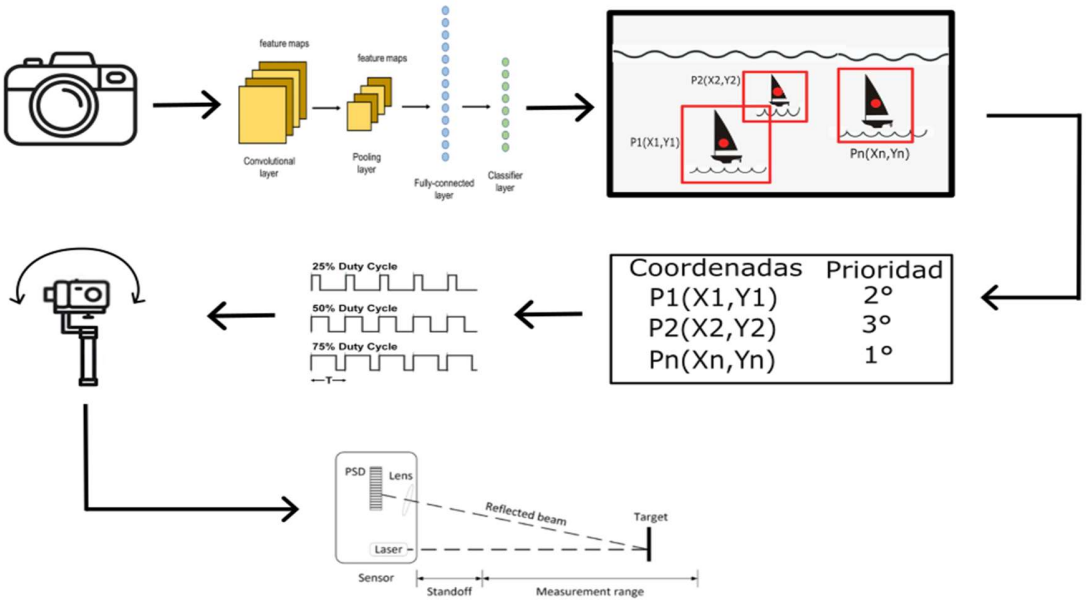


Fig 10. Esquema funcionamiento General del movimiento del gimbal con la detección del centro de objetos en pantalla.

3.4.3 Manejo del LiDAR

El algoritmo de estimación de rango de obstáculos tiene como objetivo calcular la distancia mediante la medición del tiempo que tarda una señal de Pulsos de Ancho Modulado (PWM) en pasar de un estado alto a un estado bajo y viceversa. Que ocurre cuando el sensor LiDAR dispara la señal y esta retorna. Este proceso se repite varias veces para obtener una muestra más precisa y estable.

El modo de acción por tiempo de pulso no requiere interfaz de control ya que el sensor emite la señal desde el momento en que es alimentado, la versión del LiDAR es (LIDARLite v1 “Silver Label”). En el proceso de medición que se expone de la figura 11, se inicia con la Inicialización de Variables. Se establece contadores de iteraciones para controlar el número de mediciones realizadas. La variable Distancia_sample se utiliza como acumulador para las muestras de tiempo, con el fin de calcular la distancia. También se define timeFinal, que

almacena la duración de un pulso, y time1 y time2, que registran los tiempos específicos de transición de la señal.

El siguiente paso es el Bucle de Medición, donde se llevan a cabo las mediciones en un ciclo que itera seis veces. Durante este bucle, se capturan los tiempos de transición de la señal PWM. En la primera iteración, la función simplemente espera que la señal PWM cambie de estado alto a estado bajo y viceversa, sin realizar cálculos de tiempo. En las iteraciones subsiguientes, se mide el tiempo en que la señal PWM permanece en estado alto y luego en estado bajo. Time1 registra el momento en que la señal cambia a estado alto, mientras que time2 captura el momento en que la señal regresa a estado bajo. La diferencia entre time2 y time1 proporciona el tiempo del pulso, el cual se divide por 10 para ajustar la escala temporal antes de ser añadido a Distancia_sample.

Finalmente, se calcula el promedio de las muestras acumuladas en Distancia_sample una vez completadas las seis iteraciones. El valor promedio obtenido se ajusta restando una constante de 20.0 para corregir cualquier desplazamiento sistemático en las mediciones, asegurando así la precisión de los resultados.

La función retorna el valor promedio corregido, que representa la distancia medida. Este valor es más fiable debido a la acumulación y promedio de múltiples muestras, lo que reduce el impacto de cualquier ruido o variabilidad temporal en la señal PWM. Este valor es luego enviado vía serial al SBC que en este momento está esperando la respuesta del microcontrolador con esta medida.

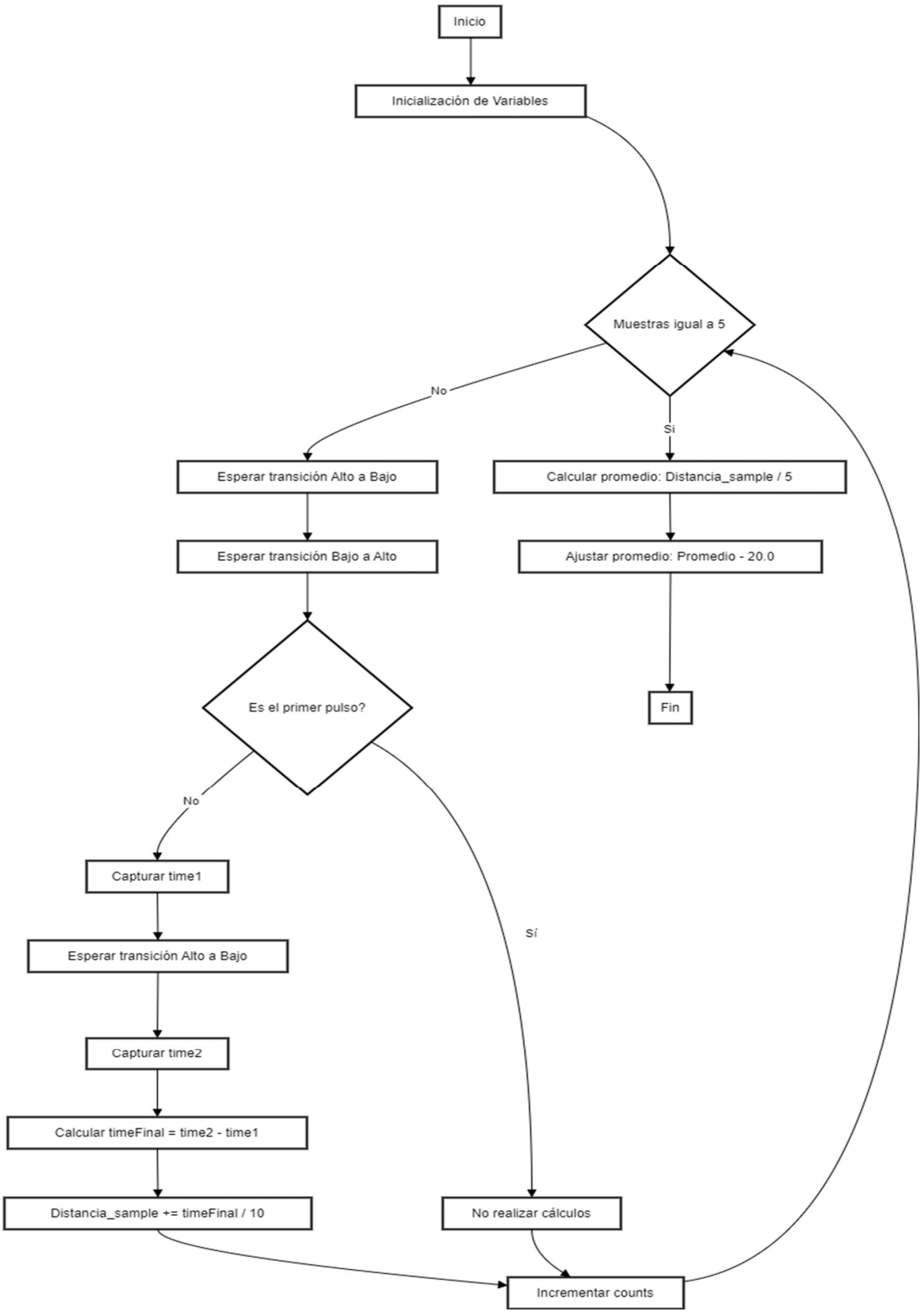


Fig 11. Diagrama de flujo para el funcionamiento en C del sensor LiDAR.

3.4.4 Aplicación completa SBC

Estructura del algoritmo Python SBC que se observa de la figura 12, la cual contiene las 7 tareas que lleva a cabo el algoritmo.

1. Grabación de Video/captura de frame: la función maneja la captura y almacenamiento de video en un archivo. Se ejecuta en un hilo separado para asegurar que la grabación sea continua mientras otras operaciones se llevan a cabo en el hilo principal. Esta función crea un archivo de video en un directorio específico y graba los frames capturados por la cámara. También usa colas para guardar frames que se capturen y modifiquen en el hilo principal, son aquellos frames a los que se les realiza la inferencia, aparte de los frames que se capturan de manera continua los cuales no tienen ningún procesamiento.
2. Inferencia de modelo de red neuronal YoloV8: es la tarea que se ejecuta después de capturar un frame disponible, luego de la inferencia el frame procesado con bounding boxes y los centros de los mismos se agrega a la cola de frames que se guardan como video, por otra parte, se obtiene una lista con los datos de los objetos identificados, como las coordenadas del bounding box, su área y el centro de los objetos.
3. Conversión de Coordenadas de la pantalla a PWM: convierte las coordenadas (x, y) de la pantalla a valores de PWM (Pulse Width Modulation), posterior se realiza un puntaje con los hiperparametros para poder realizar un orden de los centros de los objetos, en esta conversión se procesa las duplas de formato punto-píxel a niveles de activación pwm y luego se hace un arreglo para realizar una secuencia de los centros de los objetos detectados. Finalmente, se envía esta información de niveles de activación PWM mediante serial a la raspberry pi pico.
4. Comunicación Serial: envía datos a través de un puerto serial para controlar dispositivos externos. Interpreta las coordenadas transformadas en PWM y las envía mediante comandos específicos, gestionando la recepción y procesamiento de los datos de retorno.
5. Medicion de rango: una vez enfocado el punto con los datos recibidos se realiza la medición de distancia con el LiDAR, y se devuelve el valor de vuelta al SBC.
6. Conversión de Coordenadas a Ángulos: se recibe del microcontrolador la lista de ángulos y distancias para convertirlo en una lista de valores transformados. Esto facilita la interpretación de posiciones en el espacio angular.
7. Dibujo del Radar: crea una visualización tipo radar que muestra la posición de los objetos detectados en el espacio angular. Utiliza un canvas negro y dibuja círculos y líneas radiales para simular un radar, colocando puntos en las posiciones correspondientes a los ángulos y distancias calculadas.

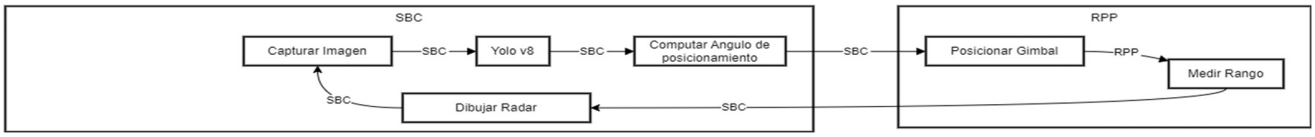


Fig 12. Funciones del script de Python del SBC.

3.5 Implementación

El sistema completo se ubica en un velero a escala, donde se utiliza soportes 3D para el gimbal y la cámara digital, al interior del velero se posiciona el SBC el circuito personalizado y la batería, se carga el script en Python que hace el control interno de todos los componentes y se configura el sistema operativo para ejecutar el programa al iniciar el booteo de la SBC, con esto se logra que al momento de activar la alimentación de la batería el sistema se ejecute. Cuando esto ocurre se deja tiempo para que el gimbal realice la calibración necesaria y se procede a realizar las pruebas en el entorno controlado.

3.5.1 Diagrama de bloques

En el diagrama de bloques de la figura 13 se muestra el proceso de generación y consumo de datos generados, por una parte, se encuentra la cámara digital la cual captura los frames, estos son procesados en el SBC donde se generan los datos de coordenadas para el gimbal y se recibe la información que llega del sensor de medición. Con la inferencia de la red neuronal, los datos calculados se envían al microcontrolador el cual envía señales pwm al gimbal y realiza la medición de distancia con el LiDAR, los datos de este último son procesados y enviados de vuelta al SBC.

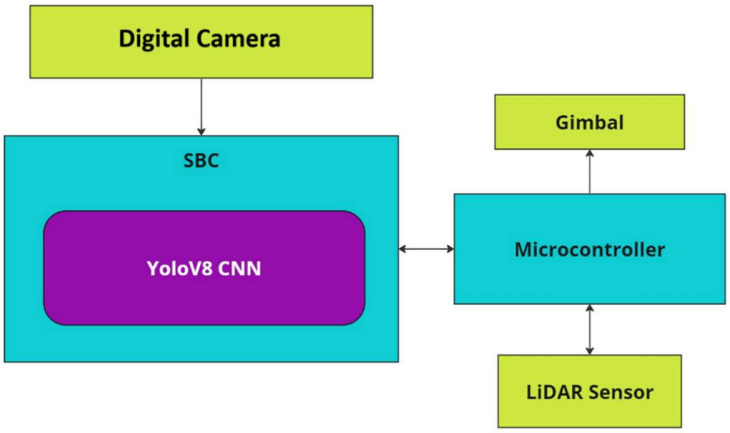


Fig 13. Diagrama de bloques, Detalle de conexiones de datos del microcontrolador, gimbal, cámara digital, Sensor LiDAR y SBC.

3.5.2 Adaptaciones mecánicas

Para este proyecto se requirieron de 2 soportes, uno que sirva para ubicar la cámara y el sensor LiDAR en el LiDAR y otro para sostener el gimbal al mástil del barco, estas piezas se presentan en las figuras 14 y 15 y su montaje final en la figura 16.

Fig 14. Soporte para ubicar el gimbal en el mástil del barco.

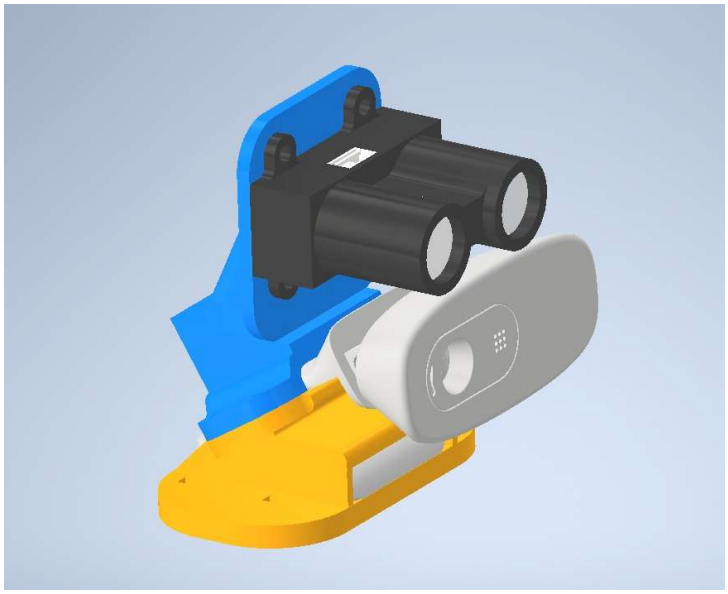


Fig 15. Soporte para la cámara y el sensor lidar, pieza azul y amarilla.

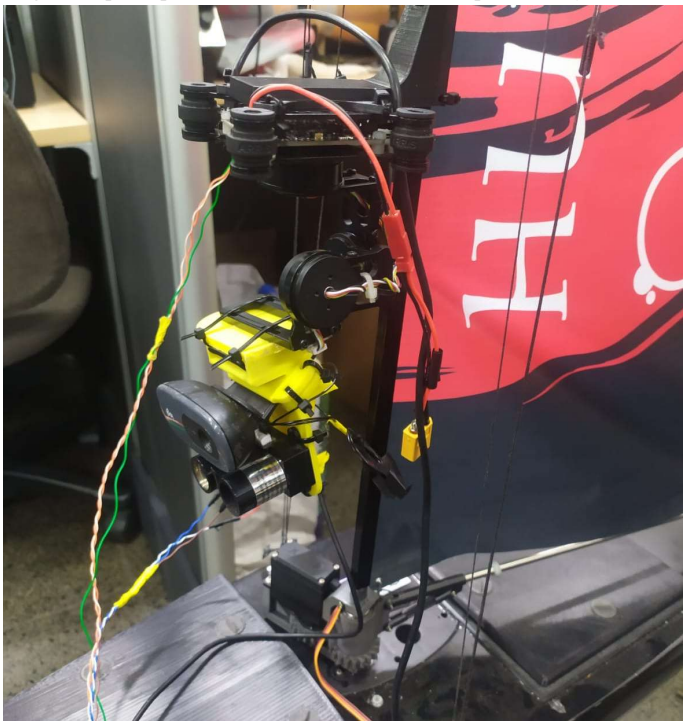


Fig 16. Sistema ubicado en el Velero.

3.5.3 Fuentes, circuitos y requisitos de potencia

Usando el Sistema Embebido proyecto Autosail ya existente; El sistema consta de una SBC VIM3, una pcb propia que tiene conexiones a la sensorica del proyecto, 2 microcontroladores y un regulador de voltaje que es con el cual se alimenta desde una batería todo el sistema. El sistema se alimenta de una sola batería de 8V,

conectada a la PCB personalizada, esta incorpora un conversor tipo boost para elevar el voltaje de la batería hasta los 12v, tensión necesaria para la alimentación de la VIM3 mediante el puerto USB, la cual tiene un consumo de 6W aproximadamente. Se presenta la conexión real en la figura 17.

Fig 17. Sistema embebido del proyecto Autosail. Con el cual se conectará la cámara el LiDAR y el gimbal.

3.6 Métricas

1. Gráficas Precisión-Confianza:
Las gráficas precisión-confianza (también conocidas como curvas de precisión-confianza) son herramientas visuales que se utilizan para evaluar el rendimiento de un modelo de clasificación probabilística. En el eje x se muestra la confianza (la probabilidad asignada por el modelo de que una instancia pertenezca a una clase), mientras que en el eje y se muestra la precisión (la proporción de instancias correctamente clasificadas a un nivel de confianza dado).
Estas gráficas permiten analizar el equilibrio entre la precisión y la confianza del modelo. Un modelo ideal tendría una curva que se mantiene alta en todos los niveles de confianza, indicando que clasifica correctamente la mayoría de las instancias con alta confianza.

2. Matriz de Confusión:
La matriz de confusión es una tabla que resume el rendimiento de un modelo de clasificación en un conjunto de datos de prueba. La matriz muestra el número de instancias que fueron correctamente clasificadas en cada clase, así como el número de instancias que fueron mal clasificadas como pertenecientes a otras clases.
Esta matriz permite identificar las clases en las que el modelo tiene más dificultades y comprender los tipos de errores que comete.

3. Precisión:
La precisión es la proporción de instancias que fueron correctamente clasificadas por el modelo. Se calcula como el número de predicciones correctas dividido por el número total de instancias.

4. Recall:
El recall (también conocido como sensibilidad) es la proporción de instancias positivas que fueron correctamente identificadas por el modelo. Se calcula como el número de verdaderos positivos dividido por el número total de positivos reales (incluyendo los que fueron mal clasificados como negativos).

5. Gráfica Recall:
La gráfica recall (también conocida como curva de recall) es una representación visual del recall del modelo en función de diferentes umbrales de clasificación. El eje x muestra el umbral de clasificación, mientras que el eje y muestra el recall.

Esta gráfica permite analizar la capacidad del modelo para identificar correctamente las instancias positivas a medida que se ajusta el umbral de clasificación.

6. mAP50:
mAP50 (siglas de "Mean Average Precision at 50% Intersection Over Union") es una métrica común para evaluar el rendimiento de modelos de detección de objetos. Representa la precisión promedio del modelo en un conjunto de datos de prueba, considerando un umbral de IoU (Intersection over Union) del 50%.

7. mAP50-95:
mAP50-95 es similar a mAP50, pero en lugar de considerar un único umbral de IoU del 50%, promedia la precisión en un rango de umbrales de IoU entre 0.5 y 0.95. Esto proporciona una evaluación más completa del rendimiento del modelo en diferentes niveles de confianza.

8. F1-score:
El F1-score es una métrica que combina la precisión y el recall en una sola medida. Se calcula como la media armónica de la precisión y el recall. Un F1-score alto indica que el modelo tiene un buen equilibrio entre precisión y recall.

9. Gráfica F1-score:
La gráfica F1-score es una representación visual del F1-score del modelo en función de diferentes umbrales de clasificación. El eje x muestra el umbral de clasificación, mientras que el eje y muestra el F1-score.

4. Experimentos y Resultados

4.1 Análisis resultados entrenamiento de la red

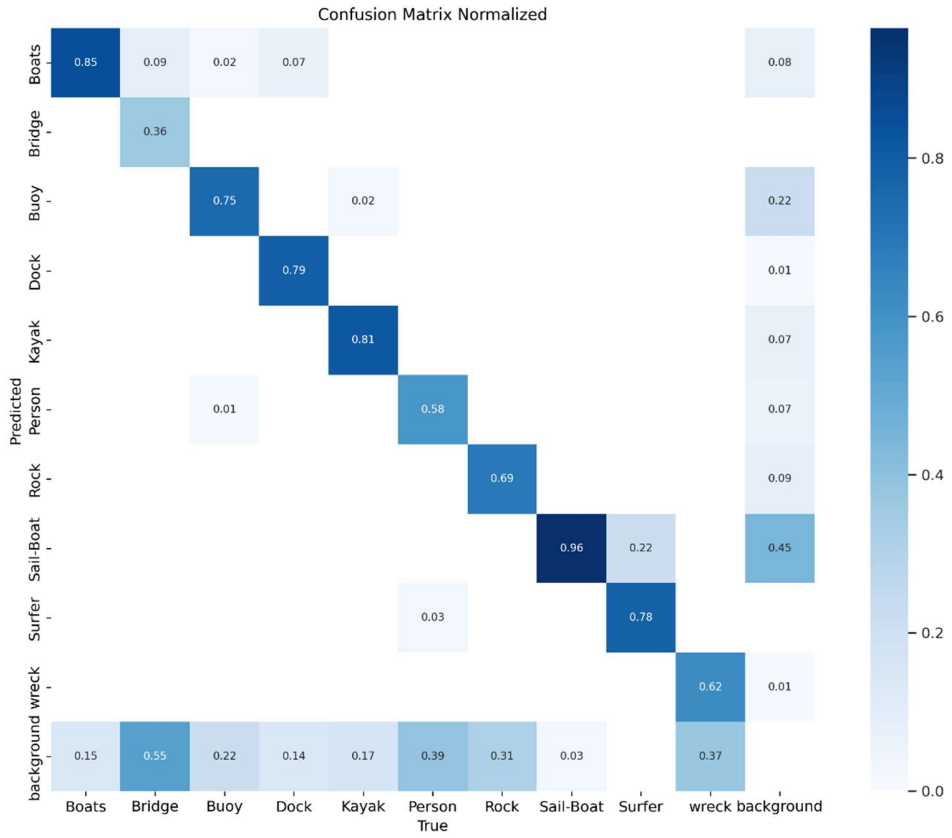


Fig 18. Grafica matriz de confusión.

La figura 18 presenta una matriz de confusión normalizada, que muestra el rendimiento del modelo de clasificación para diferentes clases de objetos. La matriz compara las predicciones del modelo (eje vertical) con las clases verdaderas (eje horizontal). Cada celda de la matriz contiene un valor que indica la proporción de instancias predichas correcta o incorrectamente para cada clase, normalizado entre 0 y 1.

Ejes y Leyenda

- El eje horizontal representa las clases verdaderas de los objetos: Boats (Barcos), Bridge (Puente), Buoy (Boyas), Dock (Muelle), Kayak, Person (Persona), Rock (Roca), Sail-Boat (Velero), Surfer (Surfista), Wreck (Naufragio) y Background (Fondo).
- El eje vertical representa las clases predichas por el modelo.
- La intensidad del color de cada celda varía de blanco (0) a azul oscuro (1), indicando la proporción de predicciones para cada par de clases.

La diagonal principal de la matriz representa las predicciones correctas, donde la clase predicha coincide con la clase verdadera. Los valores más altos en esta diagonal indican un buen rendimiento del modelo para esas clases específicas.

- Boats (0.85): El modelo predice correctamente los barcos el 85% de las veces.
- Buoy (0.75): Las boyas son correctamente identificadas el 75% de las veces.
- Dock (0.79): Los muelles son correctamente identificados el 79% de las veces.
- Kayak (0.81): Los kayaks son correctamente identificados el 81% de las veces.
- Sail-Boat (0.96): Los veleros son correctamente identificados el 96% de las veces, mostrando un alto rendimiento.
- Surfer (0.78): Los surfistas son correctamente identificados el 78% de las veces.
- Wreck (0.62): Los naufragios son correctamente identificados el 62% de las veces.

Las celdas fuera de la diagonal principal representan las predicciones incorrectas, donde el modelo ha confundido una clase con otra.

- Background (no detecciones): se encuentra una alta tasa de no detecciones por parte de las clases sail-boat y buoy, 0.45 y 0.22 respectivamente.
- La clase surffer se confunde con la de sail-boat (0.22).

La matriz muestra la clase surffer y wreck tiene la tasa más alta de confusión y no detección. Por ejemplo:

- surffer: confunde la deteccion con la clase sail-boat en un 22%
- wreck: no se realiza la correcta deteccion en un 37%.

La matriz de confusión normalizada es una herramienta crucial para evaluar el rendimiento del modelo de clasificación, proporcionando una visión detallada de dónde el modelo tiene éxito y dónde tiene dificultades. Las áreas de alta confusión indican las clases que requieren mejoras en el modelo o en los datos de entrenamiento para aumentar la precisión y reducir los errores.

La mayoría de las clases muestran un buen rendimiento, especialmente los veleros, mientras que algunas clases, como puentes y personas, presentan tasas de confusión significativas. Estos resultados destacan las áreas donde se podría enfocar el esfuerzo de mejora del modelo para aumentar su precisión global.

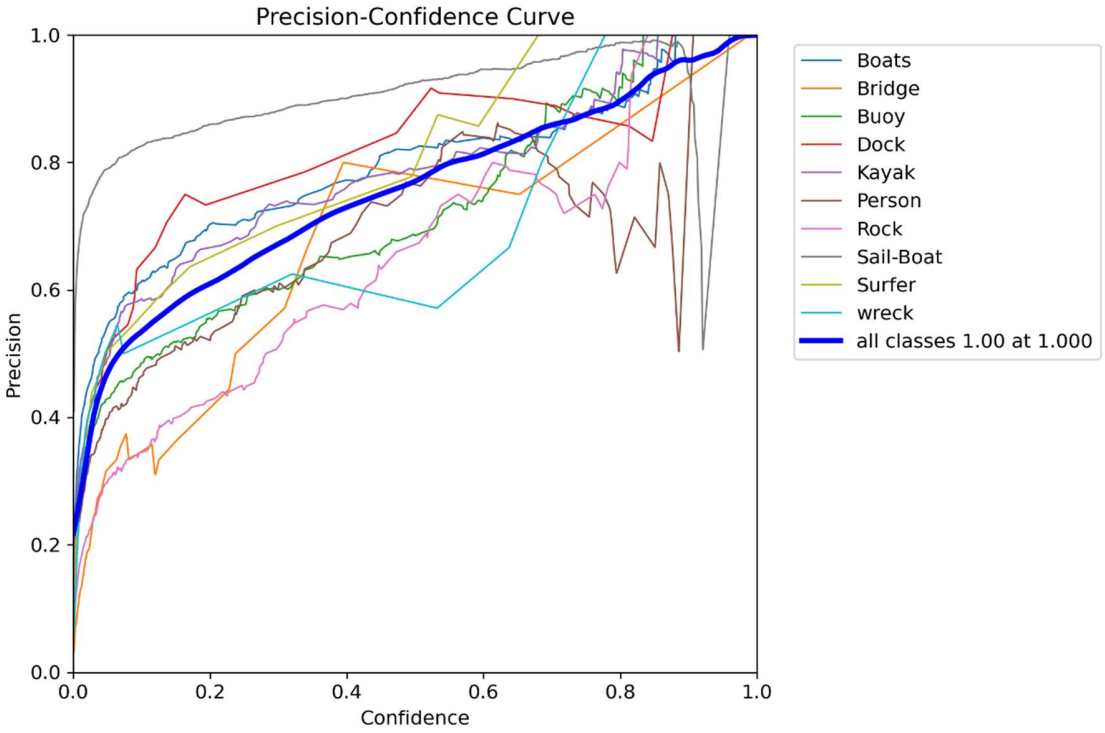


Fig 19. Gráfico Curva de precisión-Confianza.

La Curva de Precisión-Confianza figura 19 muestra la precisión del clasificador en el eje Y y el nivel de confianza en el eje X. Cada línea de color representa una clase diferente de objeto. La línea azul gruesa, etiquetada como "all classes 1.00 at 1.000", representa el rendimiento global del clasificador cuando se consideran todas las clases conjuntamente.

Desempeño por Clase

- Botes (azul claro): La curva de botes muestra una tendencia ascendente constante, alcanzando una alta precisión cerca del 80% en niveles de confianza altos.
- Puentes (naranja): La precisión para la clase de puentes fluctúa considerablemente, indicando una variabilidad significativa en el rendimiento del clasificador para esta categoría.
- Boyas (verde): La curva para boyas es relativamente más suave y muestra una buena precisión a medida que aumenta la confianza.
- Muelles (rojo): Similar a los botes, los muelles tienen una curva ascendente constante, aunque con más variabilidad.
- Kayaks (morado): La curva muestra un rendimiento más errático comparado con otras clases, con caídas significativas en ciertos puntos.
- Personas (marrón): La precisión para personas es moderada y aumenta con la confianza, pero no alcanza niveles tan altos como botes o muelles.
- Rocas (rosado): Las rocas muestran una variabilidad considerable, similar a los puentes.
- Veleros (gris): La curva de veleros es una de las más suaves, alcanzando una alta precisión en niveles de confianza altos.
- Surfistas (amarillo): Los surfistas tienen una curva ascendente constante, similar a las boyas.
- Restos de naufragios (cian): Esta clase muestra una gran variabilidad, con picos y caídas abruptas.

La curva global (línea azul gruesa) sugiere que el sistema de clasificación tiene un rendimiento razonable cuando se consideran todas las clases, con una tendencia general ascendente en precisión a medida que aumenta la confianza. Sin embargo, la variabilidad entre clases indica que el clasificador es más efectivo para ciertas categorías que para otras.

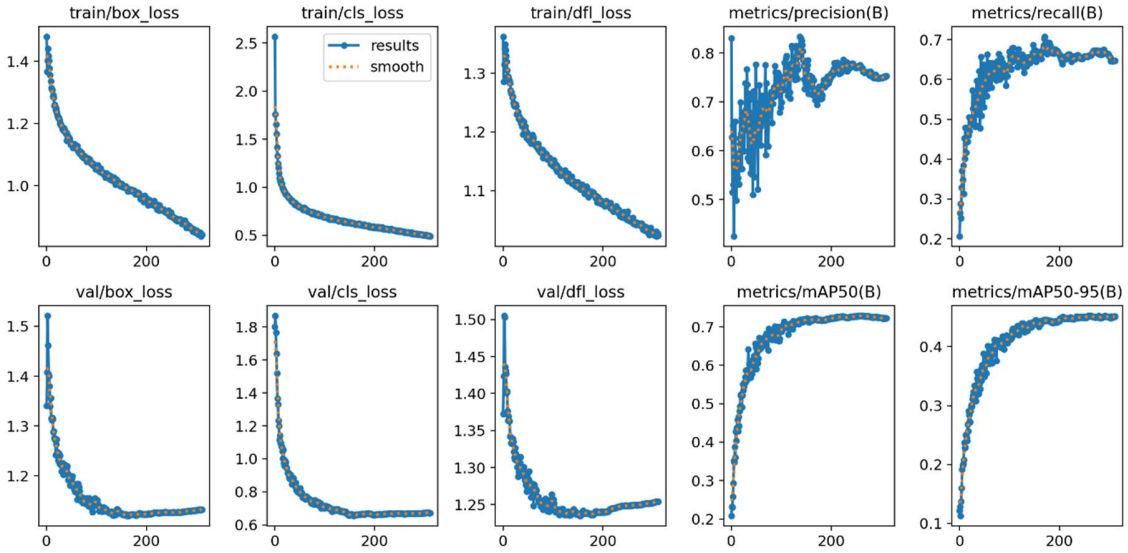


Fig 20. Graficas de rendimiento del modelo.

En las gráficas de pérdida de bounding box figura 20 para el entrenamiento y la validación, se observa una disminución continua a medida que avanzan las épocas. Inicialmente, la pérdida de entrenamiento comienza en aproximadamente 1.4 y desciende de manera constante hasta cerca de 1.0. La pérdida de validación sigue una tendencia similar, comenzando en 1.5 y disminuyendo hasta aproximadamente 1.2. Esto indica que el modelo está mejorando su capacidad para predecir las posiciones de los objetos en las imágenes.

Pérdida de Clasificación (train/cls_loss y val/cls_loss)
Las gráficas de pérdida de clasificación muestran una disminución significativa desde alrededor de 2.5 en el entrenamiento y 1.8 en la validación, hasta aproximadamente 0.5 y 0.6 respectivamente. Esto sugiere que el modelo está mejorando en la clasificación de las diferentes clases de objetos presentes en las imágenes.

Pérdida de Distribución Focal (train/dfl_loss y val/dfl_loss)
La pérdida de distribución focal también muestra una tendencia decreciente tanto en entrenamiento como en validación. En el entrenamiento, la pérdida disminuye de 1.3 a alrededor de 1.1, mientras que, en la validación, la pérdida disminuye de 1.5 a aproximadamente 1.25. La reducción en esta pérdida indica una mejora en la precisión de la localización de los objetos dentro de las cajas delimitadoras.

Métricas de Desempeño
Precisión (metrics/precision(B))
La gráfica de precisión muestra una tendencia oscilante con una mejora inicial, alcanzando un pico alrededor de la época 100, seguido por fluctuaciones. La precisión inicial se encuentra alrededor de 0.6, alcanzando picos cerca de 0.8 en varios puntos, indicando que el modelo es capaz de identificar correctamente las clases de objetos en un alto porcentaje de las ocasiones.

Recall (metrics/recall(B))
La métrica de recall también muestra una tendencia ascendente inicial, estabilizándose después de la época 100. Comienza en 0.2 y asciende hasta aproximadamente 0.7, indicando una mejora en la capacidad del modelo para identificar la mayoría de los objetos presentes en las imágenes.

mAP50 y mAP50-95 (metrics/mAP50(B) y metrics/mAP50-95(B))
Las métricas de mAP50 y mAP50-95, que miden la precisión media promedio a diferentes umbrales de intersección sobre unión (IoU), muestran una mejora continua. El mAP50 comienza en aproximadamente 0.2 y se eleva hasta alrededor de 0.65, mientras que el mAP50-95 comienza cerca de 0.1 y se eleva hasta alrededor de 0.45. Estas métricas indican una mejora consistente en la precisión general del modelo en detectar y clasificar objetos a través de diferentes umbrales de confianza.

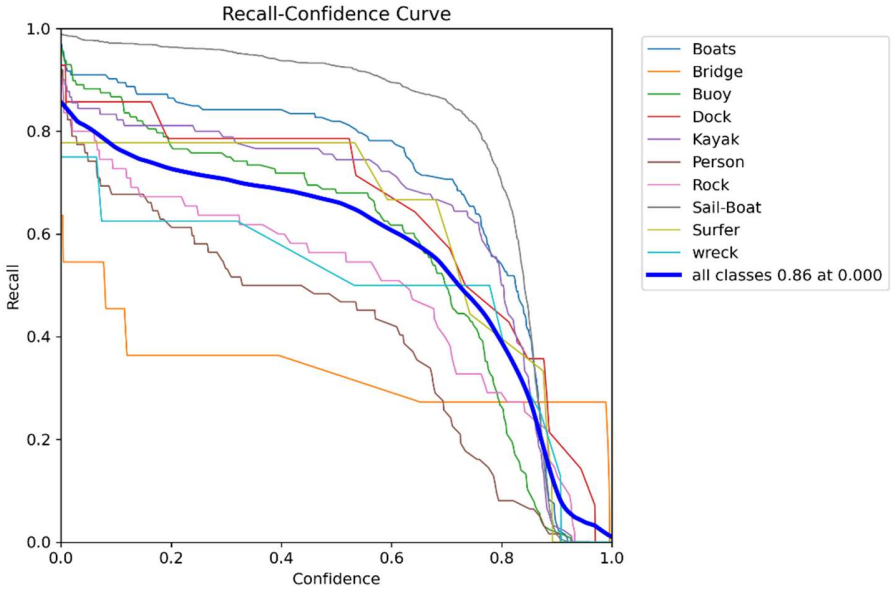


Fig 21. Grafica de Recall del modelo.

La figura 21 se presenta una curva de recuperación-confianza (Recall-Confidence Curve) que ilustra el rendimiento de un modelo de detección de objetos sobre varias clases de objetos en un entorno marítimo. La gráfica compara la capacidad del modelo para recuperar correctamente los objetos (recall) a diferentes niveles de confianza para cada clase específica.

Ejes y Leyenda
El eje horizontal representa el nivel de confianza, que varía de 0 a 1. La confianza es una medida de la certeza del modelo al clasificar un objeto. El eje vertical representa el recall, que mide la proporción de verdaderos positivos identificados entre todos los elementos relevantes en el conjunto de datos, variando de 0 a 1.
La leyenda a la derecha de la gráfica identifica las diferentes clases de objetos detectados por el modelo, las cuales incluyen:

- Boats (Barcos)
- Bridge (Puente)
- Buoy (Boyas)
- Dock (Muelle)
- Kayak

- Person (Persona)
- Rock (Roca)
- Sail-Boat (Velero)
- Surfer (Surfista)
- Wreck (Naufragio)

Cada clase está representada por una línea de color diferente en la curva. Además, una línea azul gruesa representa el rendimiento promedio del modelo en todas las clases, con un valor específico de 0.86 en recall a una confianza de 0.000.

Interpretación de las Curvas
Las curvas muestran la relación entre el recall y el nivel de confianza para cada clase de objeto. A medida que se incrementa la confianza, el recall tiende a disminuir, lo que es esperable ya que mayores niveles de confianza restringen la detección solo a los objetos que el modelo clasifica con mayor seguridad, reduciendo así los verdaderos positivos detectados.

- Boats y Sail-Boat: Estas clases mantienen un alto nivel de recall incluso a niveles más altos de confianza. Esto sugiere que el modelo es muy eficaz en detectar barcos y veleros con alta certeza.
- Bridge: La clase de puente muestra una caída abrupta en el recall al aumentar la confianza, lo que indica que el modelo tiene dificultades para detectar puentes con alta certeza.
- Buoy y Person: Las boyas y las personas también muestran una reducción significativa en recall a niveles de alta confianza, sugiriendo una menor efectividad del modelo en estas clases en condiciones estrictas.
- Kayak, Rock, Surfer y Wreck: Estas clases tienen curvas variadas, con una tendencia general de disminución del recall conforme aumenta la confianza, pero con diferentes tasas de disminución, reflejando la variabilidad en la dificultad de detectar estos objetos por el modelo.

Rendimiento General
La línea azul gruesa que representa el rendimiento promedio de todas las clases indica un recall de 0.86 a un nivel de confianza de 0.000. Esto sugiere que el modelo es capaz de detectar correctamente el 86% de todos los objetos cuando no se aplica un umbral de confianza. Sin embargo, la pendiente descendente de esta curva refleja la inevitable disminución del recall a medida que aumenta la confianza.

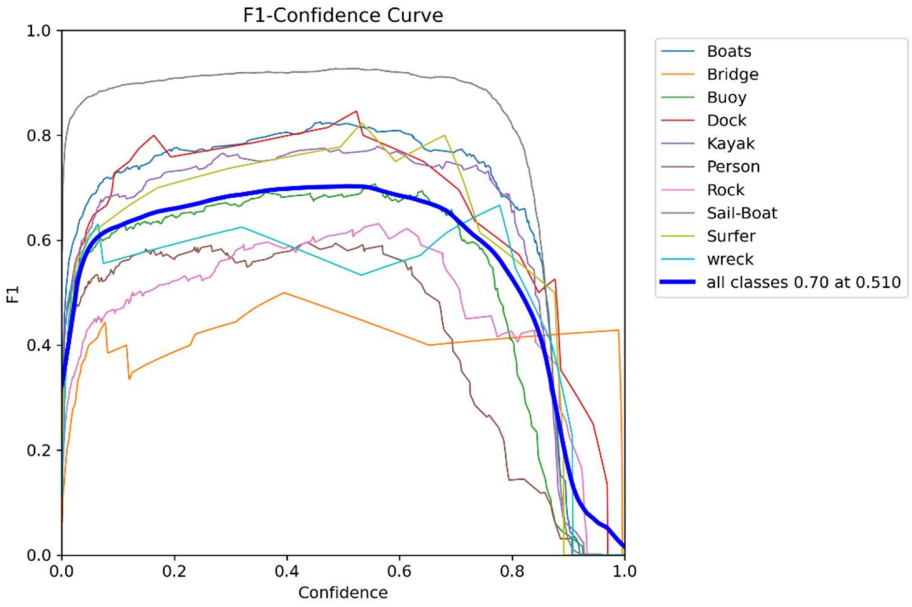


Fig 22. Grafica F1 del modelo.

La figura 22 se presenta una curva F1-Confianza (F1-Confidence Curve), que ilustra el rendimiento de un modelo de detección de objetos sobre varias clases de objetos. La gráfica compara la métrica F1 del modelo a diferentes niveles de confianza para cada clase específica.

Ejes y Leyenda

- El eje horizontal representa el nivel de confianza, que varía de 0 a 1.
- El eje vertical representa el valor F1, que es la media armónica entre la precisión y el recall, variando de 0 a 1. El valor F1 es una métrica que toma en cuenta tanto los verdaderos positivos como los falsos positivos y falsos negativos, proporcionando una visión más equilibrada del rendimiento del modelo.

Cada clase está representada por una línea de color diferente en la curva. Además, una línea azul gruesa representa el rendimiento promedio del modelo en todas las clases, con un valor específico de 0.70 en F1 a una confianza de 0.510.

Interpretación de las Curvas

A medida que se incrementa la confianza, el valor F1 inicialmente tiende a aumentar, alcanzando un pico, y luego disminuye a niveles más altos de confianza.

- Boats y Sail-Boat: Estas clases mantienen un alto valor F1 a través de una amplia gama de niveles de confianza, alcanzando picos altos, lo que sugiere que el modelo es muy eficaz en detectar barcos y veleros con buena precisión y recall.
- Bridge: La clase de puente muestra un valor F1 que no sube tanto como las otras clases y disminuye significativamente a niveles altos de confianza, indicando que el modelo tiene dificultades para detectar puentes con alta certeza.
- Buoy, Dock, Kayak y Surfer: Estas clases presentan un buen rendimiento con picos en valor F1 relativamente altos, pero también muestran una disminución al aumentar la confianza, sugiriendo que el modelo es razonablemente efectivo, pero puede tener dificultades en condiciones estrictas.
- Person, Rock y Wreck: Estas clases tienen curvas variadas con valores F1 más bajos, indicando que el modelo tiene más dificultades para detectar correctamente personas, rocas y naufragios.

Rendimiento General
La línea azul gruesa que representa el rendimiento promedio de todas las clases indica un valor F1 de 0.70 a un nivel de confianza de 0.510. Esto sugiere que el modelo logra un equilibrio razonable entre precisión y recall cuando se aplica un umbral de confianza moderado. La forma de la curva promedio también refleja una tendencia general de alcanzar un pico alrededor del nivel de confianza de 0.5 antes de disminuir.

La mayoría de las clases muestran un buen rendimiento a niveles moderados de confianza, especialmente los barcos y veleros, mientras que algunas clases, como puentes y naufragios, presentan un rendimiento inferior.

4.2 Análisis resultados experimentos en campo

Se procede a realizar 2 pruebas, prueba estática y de campo o dinámica, en ambas pruebas se lleva a cabo un análisis cualitativo de comportamiento del sistema, en cuanto al sistema de enfoque, su precisión relativa tomando en cuenta el punto marcado como destino y su enfoque final, y un análisis cuantitativo en la medición de parámetros de tiempo, como latencia del sistema para la inferencia de frames y tiempo promedio por objeto identificado y enfocado.

4.2.1 Experimento Estático y Analisis de resultados algoritmo de tracking

Se realizo una prueba estática con el fin de observar el funcionamiento de todo el sistema, y corregir errores y calibrar los factores de conversión obtenidos anteriormente. El modelo que se ejecuto es el estándar de YoloV8n.pt, en esta prueba se detectan sillas, botellas de agua y tv's. durante las pruebas realizadas se modificaron los valores de conversión en el eje horizontal, pasando de 2.22 a 1.9 y el del eje vertical de 1.47 a 1.5.

Fig 23. En el centro las inferencias de los objetos, imágenes a los lados movimiento del gimbal para enfocar los objetos detectados.

Como se puede observar en la figura 23, el resultado del algoritmo es bastante satisfactorio. El punto rojo, que indica el centro de la imagen y donde se mediría la distancia con el sensor LIDAR, se aproxima considerablemente al centro obtenido del fotograma utilizado para la inferencia del modelo. En la obtención de la medición de distancia se pudo obtener buenos resultados, donde el sensor tiene una medición con un rango de error de $\pm 10cm$, y un desfase de 20cm que se corrigen al momento de tomar la muestra y su posterior procesamiento con el calculo de promedio de las 5 muestras.

4.2.2 Experimento dinámico

La prueba en la Fuente tuvo 2 momentos, el primero donde el velero se ubicaba en un soporte al borde de la fuente, figura 24 y otro con el velero en el agua figuras 25 y 26, en ambos momentos el experimento fue el mismo, ubicar 2 botes maquetas en la fuente con una distancia máxima de 2 metros debido a limitaciones de espacio, se iniciaría el algoritmo y durante 2 minutos se dejaría todos los botes flotar por la fuente, con algunos cambios en la dirección de los barcos maqueta en algunos momentos durante la prueba. Para el total de la prueba se tuvo en total 6 minutos. Con 4 minutos del caso 1 y 2 minutos del caso 2.

En la prueba dinámica realizada, se observó un buen rendimiento del modelo en la detección de los dos barcos a escala utilizados. No obstante, la precisión de las detecciones no fue óptima. Esta deficiencia se atribuye a las diferencias entre los barcos a escala utilizados en la prueba y los barcos presentes en el conjunto de datos original utilizado para el entrenamiento del modelo.

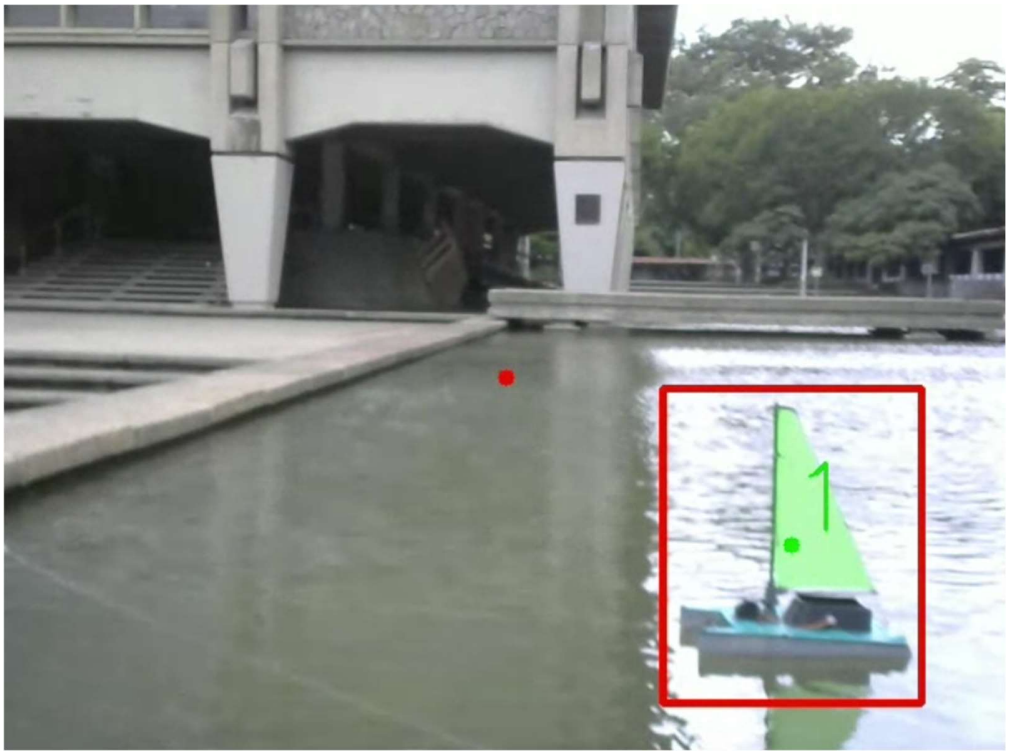


Fig 24. Detección de barcos maquetas.

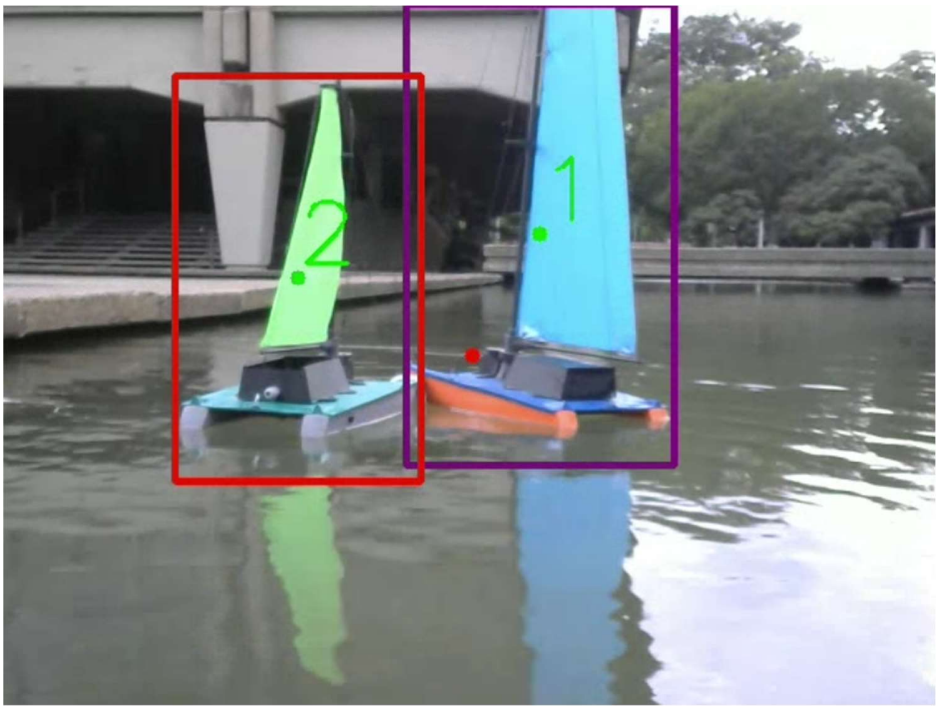


Fig 25. Detección de varios barcos en un mismo frame.

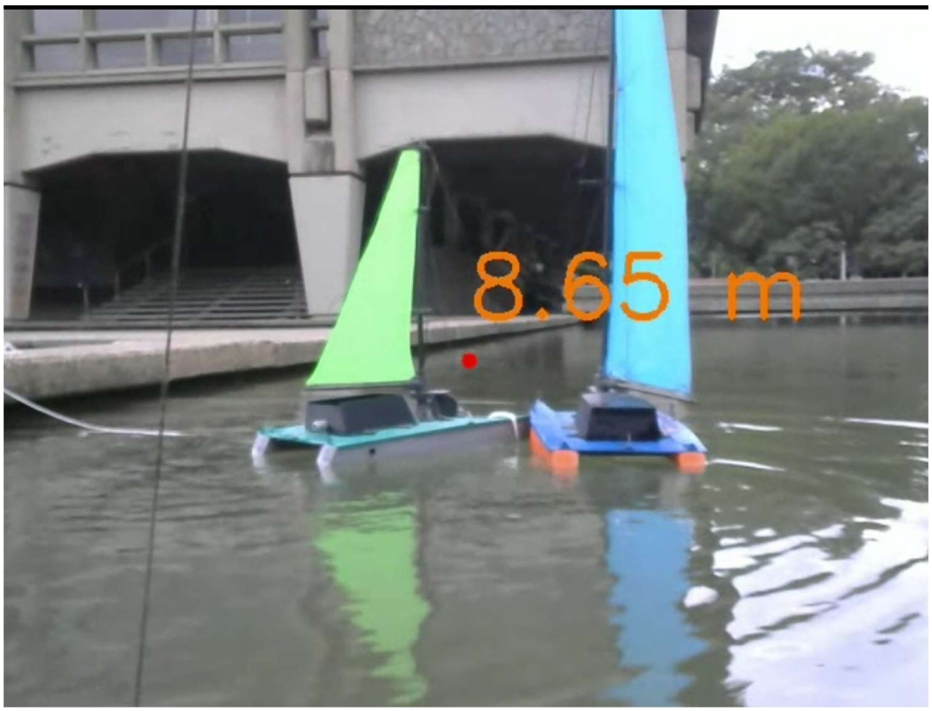


Fig 26. Sensado de distancia tomada, distancia a donde enfoca el punto rojo, distancia calculada del frame anterior.

Durante el análisis de los resultados obtenidos vistos en las figuras 24, 25 y 26, se identificaron varios puntos importantes:

1. Rendimiento de la Red CNN: La red CNN demostró un buen funcionamiento general en términos de

detección de objetos. Sin embargo, se observó que el sistema no es lo suficientemente rápido para medir la distancia de los objetos detectados en tiempo real.

2. Comparación entre Pruebas Estáticas y Dinámicas: Al comparar las pruebas estáticas y dinámicas, se constató que el tiempo de inferencia es un factor limitante. El tiempo de inferencia varía entre 950 ms y 1270 ms, dependiendo de si se detecta un objeto o no. Este tiempo de procesamiento es considerable y afecta la capacidad del sistema para operar en tiempo real.
3. Retardo del Gimbal: Además del tiempo de inferencia, el tiempo que el gimbal tarda en enfocar es otro factor que contribuye al retardo total del sistema. Actualmente, no se puede obtener una medición precisa del tiempo de enfoque del gimbal debido a la falta de una señal de retroalimentación. La conexión con el gimbal se realiza únicamente mediante los GPIO del microcontrolador para regular la señal PWM, lo que limita la capacidad de monitoreo en tiempo real.
4. Estandarización del Tiempo de Estabilización: Para mitigar los problemas causados por estos retardos, se ha estandarizado un tiempo de espera de 500 ms. Este tiempo se considera suficiente para permitir el movimiento y la estabilización del gimbal antes de capturar un nuevo frame con una buena nitidez. Esta estandarización es crucial para asegurar que las imágenes capturadas sean claras y útiles para la detección y medición de distancias.

4.3 Variación de hiperparámetros

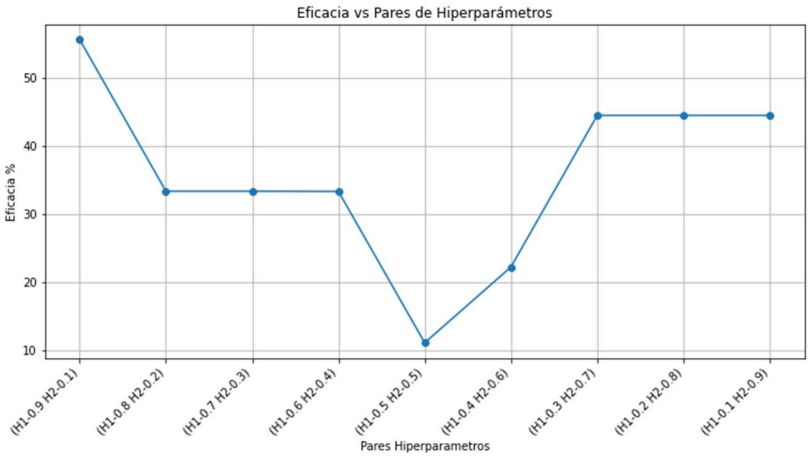


Fig 27. Grafica de la variación de los pares de hiperparámetros.

Para la variación de hiperparámetros la prueba diseñada consistía en ubicar en un frame 10 objetos, los cuales se conoce previamente la distancia desde los objetos hasta la cámara que captura la foto teniendo una lista con el orden del más cercanos hasta el más lejano y al momento de realizar la variación de los hiperparámetros se hace un conteo de cuantos objetos se predice correctamente el orden con el par de hiperparámetros escogidos, es a este conteo y división al que se denominara como eficacia en el experimento de variación de hiperparámetros.

$$Eficacia = \frac{Predicciones\ correctas}{numero\ de\ objetos}$$

La gráfica "Eficacia vs Pares de Hiperparámetros" figura 27, muestra la eficacia del modelo en función de diferentes combinaciones de los hiperparámetros H1 y H2. La eficacia se mide en porcentaje y se presenta en el eje vertical, mientras que los pares de hiperparámetros se presentan en el eje horizontal.
Puntos Clave

1. Mejor Comportamiento con H1 Alto:

La gráfica muestra que los mejores resultados de eficacia se obtienen con un valor alto de H1. Esto indica que cuando se da más importancia a la distancia del borde del bounding box inferior al límite inferior del frame, el modelo tiene un mejor rendimiento en términos de predicción. Este resultado es coherente con la idea de que la distancia relativa al barco es un indicador crucial para determinar la proximidad de los obstáculos.

2. Resultados con H2 Alto:

Aunque un valor alto de H2 también puede resultar en una buena eficacia, la gráfica sugiere que estos resultados presentan ciertas limitaciones. Específicamente, cuando se da prioridad al área del bounding box (es decir, objetos más grandes), el sistema puede cometer errores al identificar obstáculos que no son prioritarios. Esto se debe a que objetos grandes no siempre son los más cercanos o peligrosos, y darles prioridad puede llevar a decisiones erróneas.

3. Combinaciones Específicas de Hiperparámetros:

Las combinaciones de hiperparámetros (H1=0.9, H2=0.1) y (H1=0.3, H2=0.7) muestran una eficacia relativamente alta, superando el 40%. En contraste, combinaciones como (H1=0.5, H2=0.5) y (H1=0.6, H2=0.4) resultan en una eficacia significativamente baja, alrededor del 10-20%.

5. Conclusiones

Después de analizados y evaluados los resultados obtenidos del proyecto respecto al objetivo general de desarrollar un sistema embebido para la detección de obstáculos en el agua mediante el uso de redes convolucionales profundas. A continuación, se presentan las conclusiones detalladas:

Se logró recopilar un conjunto de datos extenso y diverso que incluye imágenes de obstáculos marítimos en diversas condiciones y escenarios. Este conjunto de datos permitió entrenar y validar eficazmente la red neuronal YOLOv8, mejorando la precisión en la detección de obstáculos en entornos marítimos variados. La diversidad de los datos asegura que el sistema pueda generalizar bien a nuevas situaciones, lo que es crucial para la robustez del sistema en operaciones reales.

Se desarrolló un sistema efectivo utilizando la red YOLOv8, que fue entrenada con el conjunto de datos recopilado. Se implementaron estrategias de transferencia de conocimiento para mejorar la precisión de la detección. La red YOLOv8 mostró un rendimiento adecuado en la identificación de obstáculos, cumpliendo con los requisitos del sistema para operar en un entorno marítimo. La elección de esta arquitectura permitió un balance adecuado entre precisión y eficiencia computacional, compatible con las restricciones del sistema embebido.

Se implementó un algoritmo de enfoque con gimbal que mejoró significativamente la precisión en la detección y medición de distancias. Sin embargo, la sincronización perfecta entre el sensor LiDAR, el gimbal y la red YOLOv8 presentó algunos desafíos debido a la latencia en la comunicación. Aunque el sistema logró medir las distancias de los objetos detectados, se identificó que un valor alto del hiperparámetro H2, basado en el área del bounding box, podría introducir errores en ciertas condiciones, y que el mejor par de hiperparametros identificado fue H1=0.9 y H2=0.1, haciendo un mayor énfasis en la distancia relativa entre el barco y los demás objetos propuesta durante el documento. A pesar de estos desafíos, el sistema demostró ser funcional y estable para los escenarios de prueba estáticos en los que se implementó el sistema.

Se realizaron extensas pruebas en un entorno en condiciones reales, donde se evaluó tanto la precisión de la detección de obstáculos como la exactitud de las mediciones de distancia proporcionadas por el sensor LiDAR. Los resultados mostraron que el sistema es capaz de detectar obstáculos con una precisión aceptable y sin embargo la medición de distancia luego de haber realizado el enfoque hacia los obstáculos es impreciso debido a la latencia anteriormente mencionada, haciendo que se mida en muchos casos el fondo debido al movimiento continuo de todos los elementos, el sistema en el velero y los demás obstáculos.

El proyecto logró cumplir con la mayoría de las expectativas planteadas, resultando en el desarrollo exitoso de un sistema embebido para la detección de obstáculos en entornos marítimos. Aunque se presentaron algunos desafíos, particularmente en la sincronización de componentes y la calibración de hiperparámetros, los resultados obtenidos son prometedores y sientan las bases para futuros avances en la navegación autónoma marítima contemplando el uso de componentes de bajo costo. La combinación de tecnologías como YOLOv8 y LiDAR en un sistema embebido demuestra ser una solución viable y eficiente para mejorar la seguridad y precisión en la navegación de barcos y veleros. Por esto se presentan 2 posibles implementaciones a futuro para poder aplicar un sistema completamente funcional.

6. Trabajo futuro

El uso de redes neuronales para la percepción de profundidad [15], [16] y [17] es una dirección prometedora para mejorar la detección y navegación en entornos complejos. Estas redes, como la "Depth Anything", están diseñadas para estimar la distancia de objetos en una escena a partir de imágenes monoculares como se ejemplifica en la figura 28. La percepción de profundidad permite a los sistemas no solo detectar obstáculos, sino también entender su posición relativa en un espacio tridimensional, lo cual es crucial para aplicaciones de navegación autónoma, robótica y realidad aumentada.
La red " Depth Anything ", por ejemplo, ha sido aplicada con éxito en varios estudios y proyectos. Esta red aprovecha la arquitectura de redes convolucionales profundas para generar mapas de profundidad precisos a partir de imágenes bidimensionales. Estos mapas de profundidad pueden luego ser utilizados para navegar en entornos desconocidos, evitar obstáculos y realizar tareas de manipulación con mayor precisión.

Aplicaciones y Referencias

1. Navegación Autónoma: La percepción de profundidad es crucial para vehículos autónomos y drones, permitiendo una navegación más segura y eficiente. Un estudio relevante es el de "Monocular Depth Estimation using Neural Networks" publicado en IEEE Conference on Computer Vision and Pattern Recognition (CVPR) que explora cómo estas técnicas mejoran la navegación autónoma.
2. Robótica: En robótica, la percepción de profundidad permite a los robots interactuar de manera más efectiva con su entorno. "Depth Estimation using Convolutional Neural Networks" es un paper que detalla el uso de estas redes en la robótica, permitiendo a los robots realizar tareas de manipulación complejas.
3. Realidad Aumentada y Virtual: La estimación precisa de la profundidad mejora significativamente las experiencias en aplicaciones de realidad aumentada y virtual. "Deep Learning for Depth Estimation in AR/VR" es un estudio que muestra cómo estas tecnologías se benefician de mapas de profundidad generados por redes neuronales.

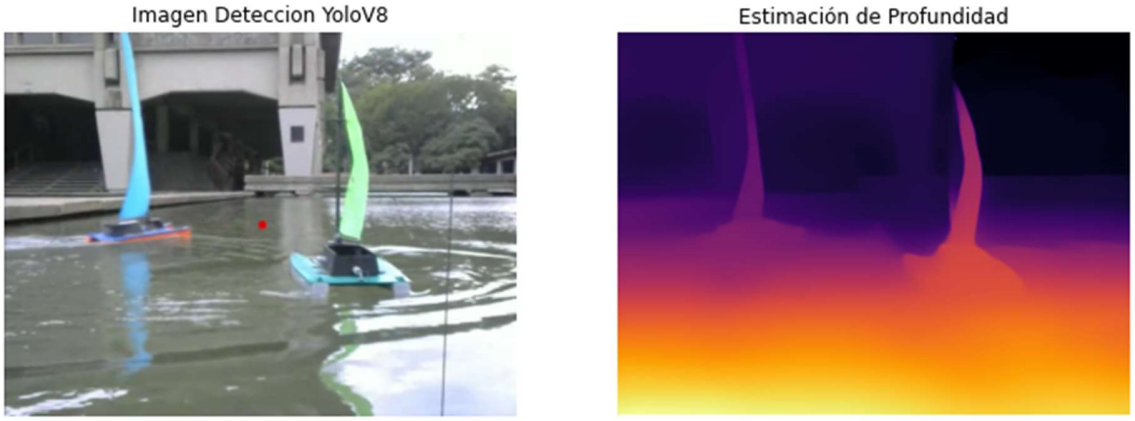


Fig 28. Mapa de profundidad entregado por la red neuronal.

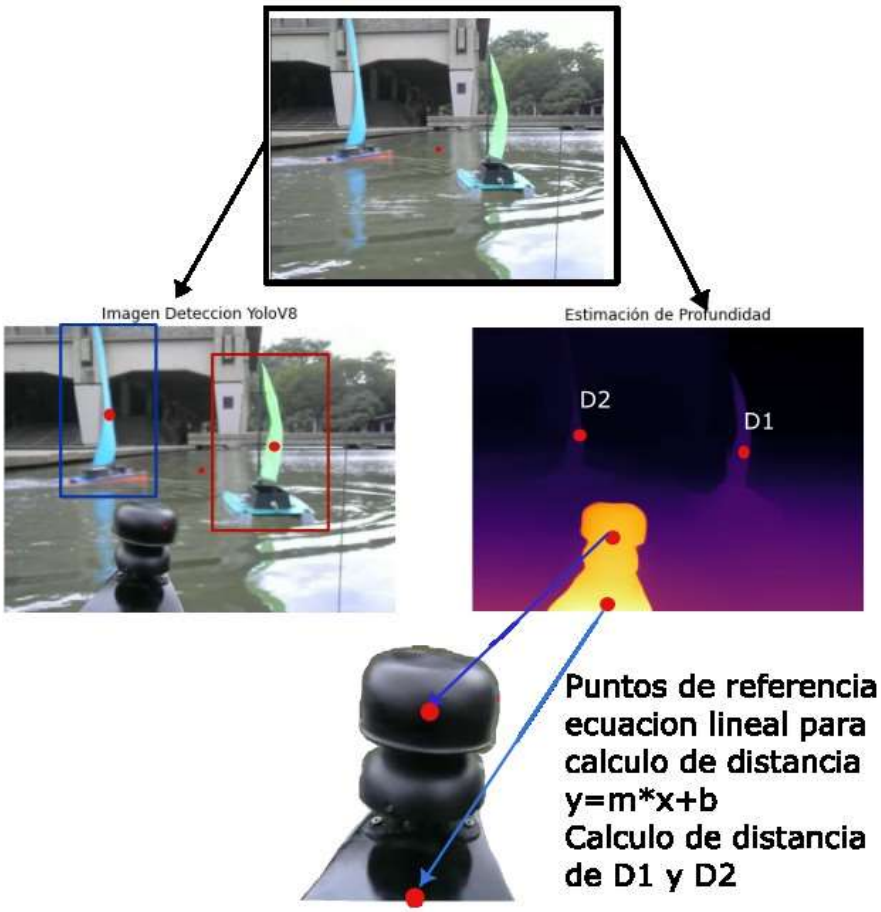


Fig 29. Propuesta de funcionamiento con la red de percepción de profundidad.

Lo que se busca con esta nueva funcionalidad es contar con puntos de referencia ubicados en el bote con los cuales pueda realizarse una predicción de la distancia de los demás puntos no conocidos utilizando interpolación entre las distancias conocida su valor en el mapa de profundidad y los demás puntos ubicados por la red de detección de obstáculos YOLOV8 como se ejemplifica en la figura 29.

Uso de Hardware especializado y de versiones actualizadas de YOLO

Durante el proceso de este trabajo se presenta la más reciente versión de YOLOV10, la cual introduce variaciones enfocadas en mejorar la eficiencia y latencia de la red mediante supresión de no máximos (NMS) durante el entrenamiento y la inferencia, un header más liviano, y el trabajo en paralelo de la aplicación de la convolución de la imagen. En la figura 30 se hace referencia a la mejora en los tiempos de inferencia de esta nueva versión.

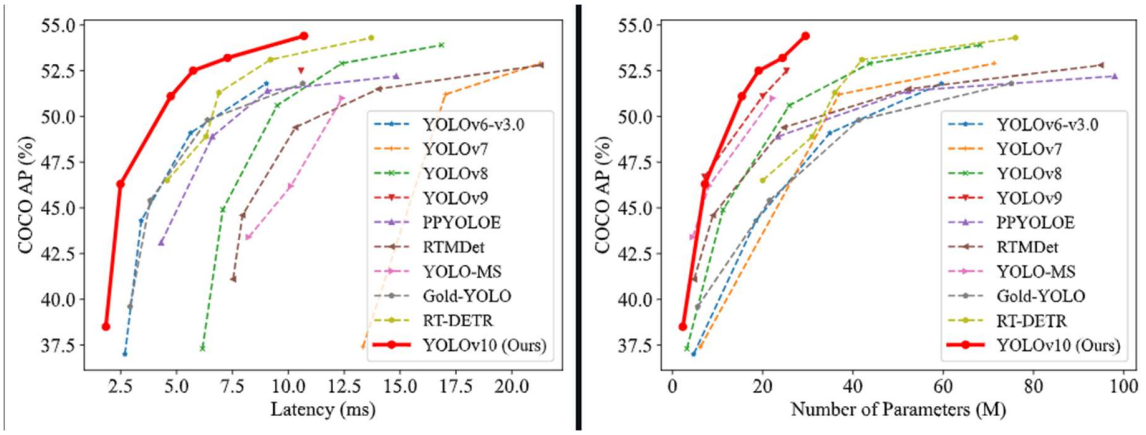


Fig 30. Grafica de la izquierda latencia de inferencia de la red comparada con otros modelos, grafica derecha número de parámetros comparado con otros modelos.

Además, se contempla el uso de hardware computacional especializado, como las plataformas embebidas Jetson Nano o sus versiones posteriores, que aprovechan los núcleos CUDA, ampliamente reconocidos por su eficiencia en unidades de procesamiento. Esto supone una mejora significativa respecto al uso de dispositivos como VIM3, que ejecutan el procesamiento en el CPU sin componentes especializados, lo cual resulta en inferencias más lentas.

Por último, se incluye la integración de un gimbal con puerto USB, lo que proporciona un acceso completo al sistema robótico. Esto permite obtener un feedback en tiempo real para reducir el tiempo de espera necesario para capturar frames estables para las inferencias subsiguientes

Departamento de Ingeniería Electrónica y de Telecomunicaciones

Prototipo de sistema embebido para la detección de obstáculos mediante redes convolucionales profundas para su integración en un velero autónomo

UNIVERSIDAD DE ANTIOQUIA
1803
Facultad de Ingeniería

PRACTICANTE: Felipe Ayala Valencia

ASESORES: Ricardo Andrés Velásquez Vélez

PROGRAMA: Ingeniería Electronica

Semestre de la práctica: 2024-1

Introducción

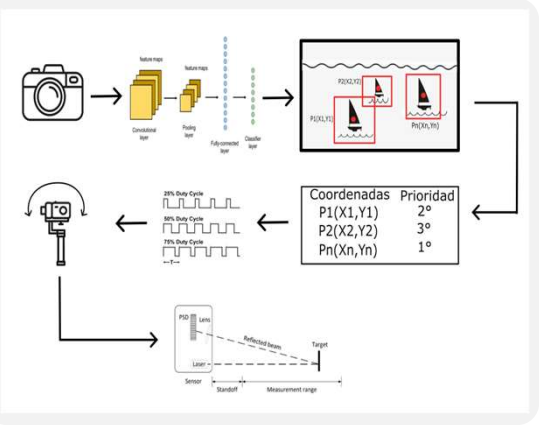
En la última década, el campo de la navegación autónoma ha avanzado significativamente gracias al desarrollo e incorporación de nuevas tecnologías como redes convolucionales (CNN), cámaras LiDAR, Sonares y sensores de alta frecuencia. Estos avances han captado la atención de la comunidad científica y la industria marítima, por su potencial para mejorar la eficiencia y seguridad en el transporte marítimo.

Planteamiento del problema

La detección de obstáculos es crucial para la seguridad en el mar, reduciendo los riesgos de colisiones, de accidentes, mitigando impactos ambientales y los costos derivados de éstos. Un sistema de detección inteligente mejora la toma de decisiones, eficiencia operativa y cumplimiento de regulaciones marítimas. Por tanto, la estrategia de fusionar redes neuronales y sensores LiDAR para la detección de obstáculos es una solución prometedora y de bajo costo para atacar esta problemática.

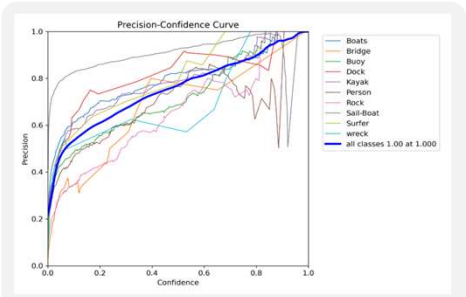
Metodología

Para llevar a cabo el desarrollo del sistema se hizo la construcción del dataset con más de 6 mil imágenes públicas y maquetas. Se realizó el entrenamiento del modelo YOLOv8 con transfer-learning. La estrategia de medición consiste en detectar los objetos de una sola imagen de la cámara, priorizarlos mediante los hiperparámetros e ir apuntándolos de manera individual y sucesiva a cada uno de ellos, para obtener la distancia con el sensor LiDAR, una vez enfocado.



Resultados

La matriz de confusión y la curva de precisión-confianza muestran buen desempeño para veleros y barcos, pero confusiones para las clases surfistas y naufragios. Las pruebas revelan limitaciones en la detección y medición de distancias en tiempo real debido a latencia y enfoque del gimbal. Los ajustes de hiperparámetros muestran buen desempeño en la priorización de objetos



Más información sobre el proyecto

Objetivos

- Recopilar y seleccionar un conjunto de datos diverso de imágenes de obstáculos marítimos para entrenar y validar la red YOLOv8, asegurando detección precisa en variados entornos marítimos.
- Desarrollar un sistema de detección de obstáculos marítimos basado en YOLOv8, entrenado con una base de datos representativa y usando transferencia de conocimiento para mejorar su precisión en entornos marítimos.
- Implementar un algoritmo de enfoque con gimbal y sincronizar un sensor LiDAR con la red YOLOv8 para medir distancias a objetos en tiempo real, optimizando la detección de obstáculos cercanos mediante análisis de hiperparámetros.
- Evaluar el desempeño del sistema integrado en precisión de detección de obstáculos y medición de distancia del LiDAR en un entorno marítimo simulado o real.

Conclusiones

- Se recopiló un extenso conjunto de datos de obstáculos marítimos en diversas condiciones, lo que permitió entrenar y validar eficazmente la red neuronal YOLOv8.
- Se desarrolló un sistema efectivo con la red YOLOv8, entrenada con el conjunto de datos recopilado, y se implementaron estrategias de transferencia de conocimiento para mejorar la precisión de detección.
- Se implementó un algoritmo de priorización mediante el uso de hiperparámetros y enfoque con el gimbal para la medición de distancias mediante la estrategia de medición y enfoque individual.
- El proyecto cumplió la mayoría de las expectativas, logrando desarrollar exitosamente un sistema embebido para la detección de obstáculos en entornos marítimos, observando algunas limitaciones en la estrategia utilizada.

DATOS DE CONTACTO DEL AUTOR:

+57 302-422-6412 | Felipe.ayalav@udea.edu.co | Felipe_a29 | www.linkedin.com/in/felipe-ayala-valencia-8415502aa

Referencias

1. [1] Noel, A., Shreyanka, K., Gowtham, K., & Satya, K. (2019, November). Autonomous ship navigation methods: a review. In *Proceedings of the Conference Proceedings of ICMET OMAN*.
2. [2] Pisarov, J. (2021). Autonomous driving. *IPSI Journal, IPSI BgD Transactions on Advanced Research (TAR)*, *17*(2), 19-27.
3. [3] S. Thombre et al., "Sensors and AI Techniques for Situational Awareness in Autonomous Ships: A Review," in IEEE Transactions on Intelligent Transportation Systems, vol. 23, no. 1, pp. 64-83, Jan. 2022, doi: 10.1109/TITS.2020.3023957.
4. [4] Skolnik, M. I. (1980). Introduction to radar systems (Vol. 3, pp. 81-92). New York: McGraw-hill.
5. [5] H. Woong Yoo et al. "MEMS-based lidar for autonomous driving - e & i Elektrotechnik und Informationstechnik". SpringerLink. https://link.springer.com/article/10.1007/s00502-018-0635-2
6. [6] Zhao, Xiangmo & Sun, Pengpeng & Xu, Zhigang & Min, Haigen & Yu, Hongkai. (2020). Fusion of 3D LIDAR and Camera Data for Object Detection in Autonomous Vehicle Applications. IEEE Sensors Journal. PP. 1-1. 10.1109/JSEN.2020.2966034.
7. [7] Scharstein, D., & Szeliski, R. (2002). A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47(1-3), 7-42.
8. [8] Russakovsky, O., et al. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision, 115*(3), 211-252.
9. [9] McCulloch, W. S., & Pitts, W. (1943). A Logical Calculus of Ideas Immanent in Nervous Activity. Bulletin of Mathematical Biophysics, 5(4), 115-133.
10. [10] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. Advances in Neural Information Processing Systems (NIPS).
11. [11] Simonyan, K., & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv preprint arXiv:1409.1556.
12. [12] He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep Residual Learning for Image Recognition. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).
13. [13] Terven, J., Córdova-Esparza, D. M., & Romero-González, J. A. (2023). A comprehensive review of yolo architectures in computer vision: From yolov1 to yolov8 and yolo-nas. *Machine Learning and Knowledge Extraction*, *5*(4), 1680-1716.
14. [14] Torrey, L., & Shavlik, J. (2010). Transfer learning. In Handbook of research on machine learning applications and trends: algorithms, methods, and techniques (pp. 242-264). IGI global.
15. [15] Eigen, D., Puhrsch, C., & Fergus, R. (2014). Depth map prediction from a single image using a multi-scale deep network. *Advances in neural information processing systems*, *27*.
16. [16] Liu, F., Shen, C., Lin, G., & Reid, I. (2015). Learning depth from single monocular images using deep convolutional neural fields. *IEEE transactions on pattern analysis and machine intelligence*, *38*(10), 2024-2039.
17. [17] Laina, I., Rupprecht, C., Belagiannis, V., Tombari, F., & Navab, N. (2016, October). Deeper depth prediction with fully convolutional residual networks. In *2016 Fourth international conference on 3D vision (3DV)* (pp. 239-248). IEEE.

Agradecimientos

Quiero agradecer, en primer lugar, a mí mismo, por nunca rendirme ante los retos y por la perseverancia que me ha traído hasta aquí. A mis padres, por supuesto, quienes estuvieron ahí en cuerpo, alma y billetera, sin ellos nada habría sido posible. Su amor incondicional, sacrificios y apoyo constante han sido el pilar fundamental de mi éxito. Han sido mi GPS en este viaje llamado vida, recalculando la ruta cada vez que me salía del camino. Soy todo lo que soy gracias a ellos, lo bueno, lo malo y lo inexplicable. Si hay alguien a quien culpar por mi éxito (o mis ocurrencias), son ellos.

A los profesores, esos arquitectos del conocimiento que vieron potencial en el caos de mis ideas y me ofrecieron el andamio para construir mi futuro. Su fe en mí fue como una hipótesis audaz que decidieron poner a prueba, brindándome oportunidades para demostrar que no estaban (completamente) locos al apostar por mí. Han sido catalizadores cruciales en mi reacción de crecimiento académico y personal, tolerando con gracia mis experimentos mentales. Gracias por cultivar mi curiosidad y podar mis ideas más extravagantes, transformando lo que era un jardín salvaje de pensamientos en un ecosistema intelectual más o menos coherente.

A mis amigos, esos cómplices involuntarios en la tragicomedia absurda que llamamos vida universitaria. Por su apoyo emocional y amistad, que han sido como un oasis de cordura en el desierto de la existencia académica. Por ser faros de alegría en los buenos momentos y botes salvavidas en el mar turbulento de los malos. Pero, sobre todo, por su paciencia sisífea y su estoicismo digno de Camus al soportar mis momentos de "genialidad" durante toda la carrera - esos instantes en que, cual Ícaro intelectual, volaba demasiado cerca del sol de la sabiduría, solo para caer en picada hacia el abismo de la incomprensión. Gracias por permanecer a mi lado en este teatro del absurdo, donde mis delirios de grandeza se estrellaban contra la roca de la realidad.

Y, por paradójico que parezca, a mis detractores, esos profetas del fracaso que predijeron mi caída con tanta convicción. Sin su meticulosa labor de sembrar dudas, este triunfo no tendría el sabor agridulce que tanto disfruto. Han sido, sin saberlo, mis entrenadores personales en el gimnasio de la superación, donde cada "no puedes" se transformó mágicamente en una repetición más de "sí puedo". Su escepticismo fue el combustible premium para mi motor de determinación, impulsándome a alturas que ni yo mismo creía posibles.

Finalmente, quiero expresar mi gratitud a todos aquellos que, de una forma u otra, han contribuido a este logro. A mi familia extendida, mentores, y a cada persona que me ha apoyado con una palabra de aliento o un gesto de bondad. Este éxito no es solo mío, sino el resultado del esfuerzo colectivo de todos los que han creído en mí. Gracias a todos por formar parte de mi éxito. Este logro es tan suyo como mío, y espero poder retribuirles en los años venideros todo el apoyo recibido.