



Centro Universitario de la Defensa en la Escuela Naval Militar

TRABAJO FIN DE GRADO

*Creación de una plataforma para la aplicación de sistemas de
visión artificial a la localización y seguimiento de objetos en el
agua*

Grado en Ingeniería Mecánica

ALUMNO: Víctor Recio Rico

DIRECTORES: José Antonio González Prieto

CURSO ACADÉMICO: 2023-2024

UniversidadeVigo



Centro Universitario de la Defensa en la Escuela Naval Militar

TRABAJO FIN DE GRADO

*Creación de una plataforma para la aplicación de sistemas de
visión artificial a la localización y seguimiento de objetos en el
agua*

Grado en Ingeniería Mecánica
Intensificación en Tecnología Naval
Cuerpo General

UniversidadeVigo

RESUMEN

Este Trabajo de Fin de Grado (TFG) se enfoca en la aplicación de técnicas de Inteligencia Artificial (IA) al campo de la visión estereoscópica, que implica la captura sincronizada de vídeos con múltiples cámaras. El objetivo del desarrollo es usar técnicas que permitan resolver las tareas de detección, clasificación y seguimiento de objetos en vídeos (*tracking*), estimando sus posiciones en 3 dimensiones (3D) a partir de la información de las cámaras. Para resolver las tareas de localización y seguimiento de objetos, se emplea el algoritmo *YOLO8*, conocido por su eficiencia en este tipo de tareas en tiempo real. Debido a este algoritmo, las redes neuronales convolucionales (*CNN*) son fundamentales en este trabajo. Por otra parte, se requiere la calibración y sincronización de los dispositivos de grabación para lograr la estimación precisa de las coordenadas 3D de los objetos detectados. Este TFG busca desarrollar una solución completa que abarque desde la detección, clasificación y seguimiento hasta la estimación de su posición 3D para medir distancias entre ellos. Esto implica la creación de un *dataset*, entrenamiento de modelos (*YOLO8* para detección, clasificación, *tracking* y una red neuronal directa para generar un mapa entre las coordenadas de los objetos detectados en las imágenes y sus posiciones reales en 3D), y una validación exhaustiva de los resultados obtenidos.

PALABRAS CLAVE

Visión estereoscópica, Calibración, Red neuronal, Seguimiento (*tracking*), *YOLO8*

AGRADECIMIENTOS

Me gustaría agradecer al profesor José Antonio González Prieto el apoyo y dedicación que ha proporcionado durante todo el proceso de desarrollo de mi Trabajo de Fin de Grado. Gracias a su paciencia y compromiso que ha dedicado desde el inicio de este proyecto, he podido alcanzar los objetivos propuestos, adquiriendo nuevos conceptos sobre el ámbito de la inteligencia artificial que desconocía.

CONTENIDO

Contenido	1
Índice de Figuras	3
Índice de Tablas.....	6
1 Introducción y objetivos	7
1.1 Contexto y motivación	7
1.2 Objetivos y alcance	8
1.3 Estructura de la memoria	8
2 Estado del arte	11
2.1 Visión artificial	11
2.1.1 Definición	11
2.1.2 Contexto histórico.....	12
2.1.3 Aplicaciones y tipos de problemas a resolver.....	13
2.1.4 Deep Learning	14
2.1.5 Redes convolucionales.....	15
2.1.6 Algoritmo YOLO.....	18
2.1.7 Seguimiento dinámico de objetos en vídeos.....	21
2.2 Visión estereoscópica.....	25
2.2.1 Concepto de la visión estereoscópica	25
2.2.2 Origen de la visión estereoscópica	26
2.2.3 Percepción de la profundidad	27
2.2.4 Calibración.....	30
2.2.5 Problemas comunes de la visión estereoscópica	32
2.2.6 Aplicaciones de la visión estereoscópica.....	33
2.2.7 Aplicaciones específicas de la visión estereoscópica en el entorno marítimo.....	34
2.2.8 Algoritmos de coincidencia estéreo	36
3 Desarrollo del TFG.....	39
3.1 Dispositivo de grabación portátil empleado.....	39
3.2 Herramientas de desarrollo empleadas	40
3.2.1 Anaconda (Miniconda)	40
3.2.2 FFmpeg	40
3.2.3 Google My Maps	40
3.2.4 Label Studio.....	41
3.2.5 VSCodium	41

3.2.6 Ultralytics HUB	41
3.2.7 Google Colab	41
3.2.8 Entorno de trabajo VIDEO-IA-CUD	41
3.3 Problemas a resolver durante el desarrollo del TFG	42
3.4 Sistema de identificación de un objeto en una posición fija	43
3.4.1 Obtención de imágenes	43
3.4.2 Etiquetado de imágenes	44
3.4.3 Entrenamiento	45
3.4.4 Seguimiento (Tracking) del objeto en un vídeo con posiciones en los distintos ejes.....	47
3.5 Sistema de identificación de objetos dentro de un recipiente con agua	48
3.5.1 Obtención de imágenes	48
3.5.2 Etiquetado de imágenes	49
3.5.3 Entrenamiento	50
3.6 Proceso de calibración de los dispositivos de grabación	52
3.6.1 Calibración mediante una red neuronal directa	53
3.6.2 Calibración mediante OpenCV	54
3.7 Obtención de vídeos con <i>tracking</i> , <i>frame</i> y posición 3D	55
3.7.1 Entorno de laboratorio	55
3.7.2 Entorno en el ambiente marítimo	57
4 Resultados y validación	65
4.1 Aplicación y análisis del modelo	65
4.1.1 Resultados y validación de los modelos entrenado	65
4.1.2 Resultados y validación del modelo en el entorno del laboratorio	67
4.1.3 Resultados y validación del modelo en el entorno marítimo.....	68
4.2 Resultados y validación general	71
5 Conclusiones y líneas futuras	75
5.1 Conclusiones y objetivos alcanzados	75
5.2 Líneas futuras	76
6 Bibliografía.....	79
Anexo I: Implicaciones Sociales, y/o Económicas, y/o Ambientales	83
Anexo II: Reflexiones Éticas y Sociales	85
Anexo III: Glosario de siglas y acrónimos	87
Anexo IV: Script de OCR.....	89

ÍNDICE DE FIGURAS

Figura 2-1 Campos de la inteligencia artificial [2].....	12
Figura 2-2 Experimento de neurofisiólogos [3]	12
Figura 2-3 Diferencia gráfica entre las diversas tareas [6].....	14
Figura 2-4 Funcionamiento <i>Deep Learning</i> [7]	15
Figura 2-5 Línea temporal de los hitos más importantes de la historia de CNN [9].....	16
Figura 2-6 Primera etapa de la convolución [10]	17
Figura 2-7 Ejemplo de agrupación máxima (<i>Max Pooling</i>) [12]	18
Figura 2-8 Ejemplo de la estructura de una red convolucional [13]	18
Figura 2-9 Evolución de las diferentes versiones del algoritmo YOLO [14]	19
Figura 2-10 Gráfico resumen del algoritmo YOLO [16]	20
Figura 2-11 Ilustración de cambios de apariencia que dificultan el seguimiento de objetos [21] ...	22
Figura 2-12 Estructura FairMOT [24].....	24
Figura 2-13 Ejemplo del uso del algoritmo <i>ByteTrack</i> [25].....	25
Figura 2-14 Proceso en la visión estereoscópica [26]	26
Figura 2-15 Estereoscopio de <i>Wheatstone</i> [27].....	27
Figura 2-16 Esquema del funcionamiento de una cámara de tiempo de vuelo (<i>ToF</i>) [29].....	28
Figura 2-17 Proceso de rectificación [31]	29
Figura 2-18 Ejemplo de mapa de disparidad y mapa de profundidad en una imagen [32].....	29
Figura 2-19 Descripción general de un sistema de visión estereoscópica [33].....	30
Figura 2-20 Observación de un objeto P desde dos cámaras [34].....	30
Figura 2-21 Transformación de un sistema 3D del mundo en coordenadas 2D de la imagen a través de una cámara [36].....	31
Figura 2-22 Sistemas utilizados para la conducción autónoma [38].....	33
Figura 2-23 <i>Ezypilot</i> : dispositivo diseñado para la localización y percepción [39].....	34
Figura 2-24 Concepto del sistema para la detección y <i>tracking</i> de buques [41].....	35
Figura 2-25 Diagrama de bloques del funcionamiento del sistema de visión <i>Hammerhead</i> [42] ...	36
Figura 2-26 <i>Semi-Global Matching SGM</i> [43].....	36
Figura 2-27 Algoritmo <i>PSMNet</i> [44].....	37
Figura 3-1 Cámara digital <i>ELP 1080P HD</i> 5-50 mm empleada para la obtención de imágenes y vídeos [45].....	40
Figura 3-2 Proceso realizado en el desarrollo de un modelo en <i>Ultralytics HUB</i> [46].....	41
Figura 3-3 Sistema de coordenadas empleado para la toma de imágenes (autoría propia).....	43
Figura 3-4 Ejemplo de una toma de imágenes simultánea por las dos cámaras con el objeto en la misma posición (autoría propia)	44

Figura 3-5 Imágenes etiquetadas en <i>Label Studio</i> (autoría propia).....	45
Figura 3-6 Probabilidad del objeto identificado sea igual que el etiquetado (autoría propia)	46
Figura 3-7 Resultados obtenidos tras el entrenamiento (autoría propia).....	46
Figura 3-8 Imágenes extraídas del vídeo de <i>tracking</i> (autoría propia).....	48
Figura 3-9 Obtención de imagen con fecha y hora exacta (autoría propia)	49
Figura 3-10 Etiquetado de imagen en <i>Label Studio</i> (autoría propia)	49
Figura 3-11 Reparto de las imágenes que componen el <i>dataset</i> del modelo a entrenar (captura generada por <i>Ultralytics</i> de autoría propia)	50
Figura 3-12 Métrica del modelo (gráfica generada por <i>Ultralytics</i> de autoría propia)	50
Figura 3-13 Gráfica de <i>box loss</i> para cada uno de los <i>epochs</i> (gráfica generada por <i>Ultralytics</i> de autoría propia).....	51
Figura 3-14 Gráfica de <i>class loss</i> para cada una de las entradas (gráfica generada por <i>Ultralytics</i> de autoría propia).....	51
Figura 3-15 Gráfica de <i>object loss</i> durante el transcurso del entrenamiento (gráfica generada por <i>Ultralytics</i> de autoría propia)	52
Figura 3-16 Red neuronal empleada para el proceso de calibración de los dispositivos gráficos (autoría propia).....	53
Figura 3-17 Modelo obtenido tras el entrenamiento de las redes neuronales (autoría propia)	54
Figura 3-18 Diagrama de flujo sobre el proceso de calibración (autoría propia).....	55
Figura 3-19 Proceso de calibración para el cálculo de la matriz y coordenadas (autoría propia)....	55
Figura 3-20 Imagen capturada desde la cámara izquierda, mostrando la trayectoria de la pelota y las coordenadas 3D en ese momento (autoría propia).....	56
Figura 3-21 Imagen capturada desde la cámara derecha, mostrando la detección del objeto (cuadrado blanco) como la detección del objeto al mismo instante en la cámara izquierda (cuadrado rojo) (autoría propia)	56
Figura 3-22 Puntos notables de la dársena de la ENM para la ejecución del <i>script</i> y eje de coordenadas (autoría propia).....	58
Figura 3-23 Resultado de la regresión en 3D de los puntos <i>GPS</i> obtenido tras el entrenamiento de la red neuronal por puntos notables (autoría propia)	58
Figura 3-24 Distancias y posiciones <i>GPS</i> de barcos y veleros (autoría propia).....	59
Figura 3-25 Posiciones de barcos y veleros en el <i>frame</i> 12 en <i>My Maps</i> (autoría propia).....	60
Figura 3-26 Malla obtenida con respecto a la cámara derecha (autoría propia)	61
Figura 3-27 Malla obtenida con respecto a la cámara izquierda (autoría propia)	61
Figura 3-28 Disposición del <i>grid</i> tras la obtención de los puntos en <i>My Maps</i> (autoría propia)	62
Figura 3-29 Resultado de la regresión en 2D y 3D de los puntos de la malla obtenido tras el entrenamiento de la red neuronal (autoría propia)	62
Figura 4-1 Problemas de oclusión y sensibilidad a la iluminación (autoría propia)	66
Figura 4-2 Matriz de confusión generada durante el entrenamiento del modelo (autoría propia) ...	66
Figura 4-3 Posición y <i>tracking</i> del objeto dinámico (autoría propia)	67

Figura 4-4 Imagen de posición, distancias y <i>tracking</i> en el <i>frame</i> 24 (autoría propia).....	68
Figura 4-5 Posiciones de barcos y veleros en el <i>frame</i> 24 en <i>My Maps</i> (autoría propia).....	69
Figura 4-6 Trayectoria seguida por los barcos entre los <i>frames</i> 100 y 260 (autoría propia).....	70
Figura 4-7 AIS (en amarillo) de la lancha de instrucción (autoría propia)	71
Figura 4-8 Percepción de profundidad en el eje Y (autoría propia)	72
Figura 4-9 Ejemplo de superposición de objetos (autoría propia)	73

ÍNDICE DE TABLAS

Tabla 2-1 Modelos de *YOLOv8* [18] 21

1 INTRODUCCIÓN Y OBJETIVOS

1.1 Contexto y motivación

El término conocido como Industria 4.0, se encuentra en auge estos últimos años debido al uso de nuevas tecnologías relacionadas con la Inteligencia Artificial (IA) capaces de realizar tareas complejas, como, por ejemplo, la visión artificial. Esta tecnología permite obtener, procesar y analizar imágenes del mundo físico mediante herramientas, con el objetivo de que la información obtenida de todo el proceso pueda ser interpretada para generar los resultados esperados en cada tarea a resolver. La IA hace posible que las máquinas aprendan a partir de los datos que el sistema analiza, de forma que, para conseguirlo, emplean diferentes técnicas como la aplicación de redes convolucionales o el aprendizaje profundo, permitiendo que las computadoras puedan ser entrenadas para realizar una tarea en concreto procesando grandes cantidades de datos y reconociendo patrones en los mismos.

Para comprender el funcionamiento de la visión artificial y sus aplicaciones en entornos 3D donde se pretende posicionar los objetos de un cierto escenario visual, es esencial entender también el problema de la visión estereoscópica. Este término, en el ámbito de la biología, se refiere a la capacidad del cerebro de los seres vivos para fusionar dos imágenes en una, creando la sensación visual de profundidad y permitiendo la visualización de una imagen en 3D, de forma que, en el ámbito de la visión artificial, la visión estereoscópica se emplea en el desarrollo de sistemas capaces de percibir la profundidad y la estructura en 3D. Para llevar a cabo este proceso, se utilizan algoritmos y técnicas que simulan, por ejemplo, la visión binocular humana. La visión estereoscópica en la visión artificial tiene diversas aplicaciones, como la percepción de profundidad en vehículos autónomos o la visión tridimensional aplicada a la robótica en movimiento, como es el caso del uso de herramientas de intervención médica avanzadas, así como en muchos otros campos tecnológicos. Es fundamental que los sistemas de visión artificial tengan la capacidad de comprender la estructura tridimensional para tomar decisiones precisas y llevar a cabo tareas específicas de manera más eficiente.

En el contexto de la Armada, se está experimentando una fase de transformación hacia la denominada Armada 4.0, en la que la inteligencia artificial desempeña un papel fundamental. Para impulsar este proceso, se han establecido el Centro de Inteligencia Artificial (CIA2) y el Centro de Gestión del Dato. Los fundamentos de esta evolución incluyen la automatización de los procesos laborales, la integración progresiva de tecnologías y aplicaciones basadas en datos únicos, de calidad y seguros, una infraestructura sostenible y digitalizada, así como la adaptación ágil de la organización [1]. Estos pilares están detallados en el documento titulado *“Visión del AJAL (Almirante Jefe de Apoyo Logístico) del Arsenal Inteligente”*.

La evolución de los buques de la Flota se encamina hacia la construcción de nuevas embarcaciones en las cuales la inteligencia artificial adquiera la misma relevancia que tiene en diversos sectores de la vida actual. Estos buques están diseñados para convertirse en los más modernos del mundo, incorporando sistemas que abarcan, mediante sus sensores, el dominio marítimo, el litoral, el espacio aéreo y la vigilancia del ciberespacio. La implementación de la inteligencia artificial en estos barcos posibilitará a sus tripulaciones llevar a cabo una gestión eficiente de la fusión de datos, mejorando la conciencia situacional y facilitando la toma de decisiones.

Las futuras fragatas F-110 se perfilan como buques inteligentes, marcando un hito al ser el primer programa naval español diseñado para incorporar un Gemelo Digital. Esto implica la creación de una réplica virtual del buque que recopila información y datos a través de una red de sensores distribuidos en toda la embarcación, dando forma a un sistema ciberfísico. Este enfoque utiliza modelos de comportamiento y tecnologías como el *cloud computing*¹ y *machine learning*, permitiendo respaldar el mantenimiento y la operación incluso a distancia de miles de kilómetros. Además, este enfoque contribuye a mejorar el intercambio de información sobre el tráfico de barcos que transitan por aguas de interés nacional.

Este último aspecto constituye la motivación central de este Trabajo de Fin de Grado (TFG), que tiene como objetivo desarrollar una plataforma basada en la IA destinada a facilitar la localización y el seguimiento de objetos en el agua. La aspiración es crear una herramienta de asistencia a la navegación adaptable a diferentes tipos de buques. Como se ha mencionado previamente, el enfoque se basará en el uso de la inteligencia artificial para entrenar los sistemas mediante la utilización de medios de grabación proporcionados, con el propósito de llevar a cabo la tarea propuesta.

1.2 Objetivos y alcance

El objetivo primordial de este trabajo consiste en desarrollar una plataforma de visión artificial que, empleando técnicas de aprendizaje profundo, tenga la capacidad de medir las distancias entre diversos objetos detectados, así como estimar su velocidad y, en la medida de lo posible, la trayectoria recorrida por cada uno de los objetos localizados. La meta de esta plataforma es facilitar la localización y el seguimiento de objetos en el agua, ya sean buques u otros elementos presentes en el entorno marino. El proceso de desarrollo y construcción se llevará a cabo mediante el uso de dos cámaras de grabación.

Los hitos propuestos para alcanzar con eficacia y éxito el cumplimiento de este TFG son los siguientes:

- Estudio de la visión artificial y de la visión estereoscópica, presentando los problemas que puede suponer esta última a la hora del desarrollo del proyecto.
- Familiarización con el *software* y medios de grabación empleados para la construcción de la plataforma.
- Resolver el problema de la calibración del sistema empleado.
- Aportar una base de datos lo suficientemente abundante de adquisición y etiquetado de objetos mediante el entrenamiento con el empleo del algoritmo *YOLO*.
- Análisis del desarrollo del *software* de visualización con la base de datos obtenida, además de su funcionamiento y rendimiento.

1.3 Estructura de la memoria

Contextualizado el proyecto a realizar, y presentados los diferentes objetivos e hitos que se buscan conseguir con este trabajo, se presenta a continuación la estructura que encuadra la memoria de dicho trabajo. Con esta memoria se aporta una visión clara y ordenada, dividiéndose el trabajo en cinco partes

¹ *Cloud computing*: modelo de servicio que permite acceder y utilizar recursos informáticos a través de Internet, en lugar de depender de recursos locales o de hardware físico.

bien diferenciadas entre sí, donde se detalla los resultados obtenidos y las conclusiones sacadas a partir de estos. De esta manera, el trabajo se divide de la siguiente manera:

- **Introducción y objetivos:** en este primer apartado se introduce un contexto acerca de la importancia que tiene la inteligencia artificial en la actualidad, así como una breve introducción al tema de la visión estereoscópica. Además, se establecen los hitos y objetivos a alcanzar, así como su alcance. Finalmente, se presenta la estructura de la memoria.
- **Estado del arte:** en el segundo capítulo del presente trabajo se estudian los conocimientos teóricos que permitan entender cuáles son los fundamentos empleados en los que se basa el desarrollo práctico del trabajo. El capítulo se divide en dos puntos, subdividiéndose estos en subapartados para que su comprensión sea más sencilla. Se dividen de la siguiente manera:
 - **Visión artificial:** se da una breve introducción e historia sobre la inteligencia artificial, así como sus principales aplicaciones y problemas a resolver. Además, se explican conceptos de gran importancia durante el transcurso del trabajo como son las redes neuronales convolucionales y el algoritmo *YOLO*.
 - **Visión estereoscópica:** incluye una breve introducción del concepto y origen acerca de la visión estereoscópica. Se presentan los problemas que este tipo de técnicas ayuda a resolver empleando la percepción de la profundidad. También se presentan sus aplicaciones, específicamente aquellas relacionadas con el ámbito marino.
- **Desarrollo del TFG:** apartado donde se exponen todas las herramientas y *software* empleados para la ejecución y construcción del sistema. Igualmente, se refleja el proceso que se ha llevado a cabo para el desarrollo y las diferentes fases en las que se encuadra dicho trabajo.
- **Resultados, obtenidos y validación:** tras el desarrollo de la plataforma, se exponen los resultados obtenidos. Se analiza estos resultados y se indica si los objetivos que se han propuesto para el trabajo se han cumplido con éxito.
- **Conclusiones y líneas futuras:** en este capítulo se presentan las conclusiones extraídas una vez que se han analizado los resultados obtenidos. Asimismo, se exponen las líneas futuras de desarrollo e investigación sobre la temática de este proyecto, aspirando a una continuación del mismo.

Para finalizar, en la última sección de la presente memoria se recoge la bibliografía utilizada para llevar a cabo este trabajo, así como los diferentes anexos complementarios necesarios para la comprensión y ejecución del proyecto.

2 ESTADO DEL ARTE

En este capítulo se analiza el concepto de visión artificial y visión estereoscópica. Dentro de la visión artificial, se trata una breve explicación acerca sobre la misma con un contexto histórico. Se desglosan las distintas aplicaciones y tipos de problemas a resolver mediante la visión artificial y se realiza una introducción acerca del concepto de aprendizaje profundo (*Deep Learning*). Seguidamente se trata el concepto de redes convolucionales, así como una presentación del algoritmo YOLO (*You Only Look Once*). Por último, se expone el problema del seguimiento dinámico de objetos en vídeos, mejor conocido como *tracking*.

Posteriormente, se entra en detalle acerca la visión estereoscópica. Al igual que con la visión artificial, se muestra una explicación y contexto acerca de este tipo de aplicaciones, entrando en detalle sobre la percepción de profundidad y calibración. Se presentan los problemas más comunes, así como las aplicaciones englobadas dentro de la visión estereoscópica y más en concreto aquellas específicamente relacionadas con el entorno marítimo. Por último, se muestran algunos de los algoritmos empleados en la localización, clasificación y seguimiento de objetos en vídeos.

2.1 Visión artificial

2.1.1 Definición

En líneas generales, la visión artificial se encuadra dentro del campo de la inteligencia artificial (Figura 2-1), permitiendo a los ordenadores extraer información gracias a la entrada de imágenes digitales visuales y otros tipos de entradas. Con esta información, se pueden obtener medidas y realizar recomendaciones acerca de la información extraída. Con respecto a la IA, que permite a los ordenadores generar decisiones a partir de los datos recopilados, la visión artificial representa la parte destinada a realizar la función de ser “los ojos” de un sistema, es decir, para que el sistema sea capaz de observar y comprender su entorno en 3D.

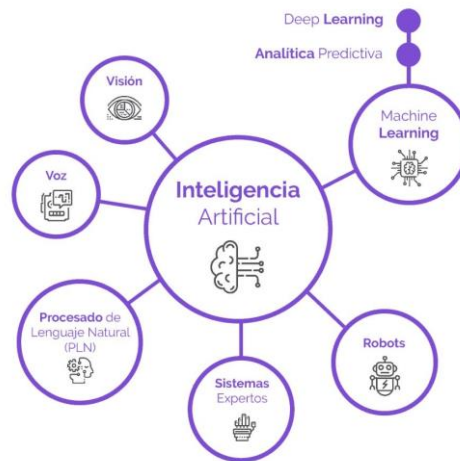


Figura 2-1 Campos de la inteligencia artificial [2]

De esta forma, el funcionamiento de la visión artificial intenta emular sistemas como la visión humana. Sin embargo, el principal problema de la visión artificial frente a la visión humana es que no puede reconocer distancias, diferenciar objetos o reconocer errores de inicio. Los sistemas de IA necesitan ser entrenados con gran cantidad de datos y tiempo de entrenamiento para alcanzar los niveles de precisión requeridos para la tarea que se desea resolver. Los humanos tienen la ventaja del beneficio de toda una vida para entrenar como son los diferentes objetos para distinguirlos, saber si están cerca o lejos, si están en estático o dinámico o si hay algún problema con una imagen. La visión artificial es capaz de entrenar a las máquinas a realizar estas tareas en menos tiempo gracias a cámaras, datos y algoritmos.

Para lograr estos objetivos, se combinan diferentes ideas relacionadas con la IA, como son los modelos de *Machine Learning* denominados *Deep Learning*, es decir, basados en redes neuronales con un gran número de capas, con conceptos relacionados con el uso de redes neuronales convolucionales (*CNN*, *Convolutional Neural Network*). Ambos conceptos se desarrollan en el presente capítulo.

2.1.2 Contexto histórico

El origen de la visión artificial se remonta al año 1959, donde un grupo de neurofisiólogos realizaron un experimento con un gato, al que le mostraron una matriz de imágenes, intentando correlacionar una respuesta en su cerebro (Figura 2-2). Con este experimento descubrieron que en primer lugar respondía a las líneas o los bordes más marcados, significando científicamente que el procesamiento de imágenes empieza con formas simples.

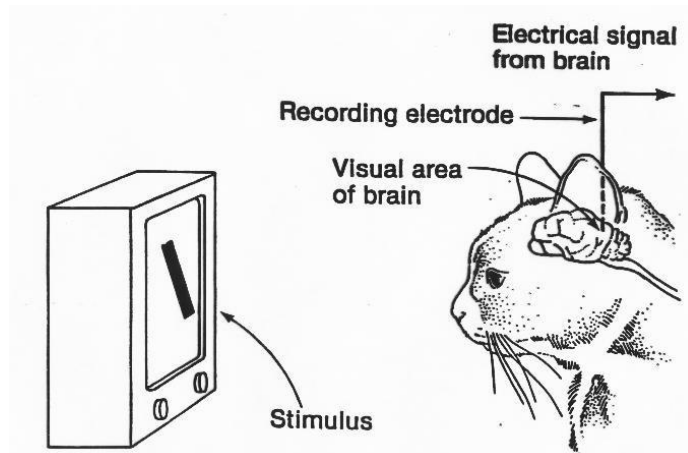


Figura 2-2 Experimento de neurofisiólogos [3]

Al mismo tiempo, se desarrollaba la primera tecnología de escaneo de imágenes, permitiendo a los ordenadores digitalizar dichas imágenes y adquirirlas. Durante la década de los 60, se llevaron a cabo varios hitos. Uno de ellos se alcanzó en el año 1963, cuando los ordenadores eran capaces de transformar imágenes 2D (dos coordenadas) en formas 3D. En 1973, se introdujo la tecnología de reconocimiento óptico de caracteres (*OCR, Optical Character Recognition*). Con esta tecnología se podía reconocer un texto impreso en cualquier fuente o tipo de letra. De manera similar, se introdujo otra tecnología de reconocimiento inteligente de caracteres (*ICR, Intelligent Character Recognition*), siendo capaz de descifrar un texto escrito a mano mediante redes neuronales. Gracias a estas dos tecnologías, hoy en día se aplican al procesamiento documentos, lectura de matrículas de vehículos, pagos móviles y otros usos.

En 1982, el neurocientífico *David Marr*, introdujo algoritmos para que las máquinas detectaran esquinas, bordes, curvas y formas básicas similares, de forma que se establecieron las bases para entender que la visión funciona de forma jerárquica. Además, el científico *Kunihiko Fukushima* fue capaz de desarrollar una red de celdas que reconocía patrones. La red se denominó *Neocognitron*, y en la misma se incluían capas convencionales (también llamadas capas directas), en una red neuronal.

Ya dentro del siglo XXI, en los primeros años de siglo se centraron los estudios en el reconocimiento de objetos, apareciendo las aplicaciones de reconocimiento facial en tiempo real. En el año 2010, se hizo público el conjunto de datos de *ImageNet*. Este conjunto de datos estaba compuesto por millones de imágenes etiquetadas en muchas clases de objetos, proporcionando una base para las *CNN* y los modelos de *Deep Learning*, que son utilizados actualmente. En 2012, un grupo de investigadores de la Universidad de Toronto apuntó a una *CNN* a un concurso de imágenes. El modelo fue denominado *AlexNet*, y consiguió reducir de manera significativa el índice de errores a la hora del reconocimiento de imágenes.

2.1.3 Aplicaciones y tipos de problemas a resolver

A día de hoy, aproximadamente todo lo que nos rodea genera contenido digital. Gracias a la visión artificial, esta tecnología se está llevando en muchos ámbitos y sectores de nuestra vida cotidiana como pueden ser [4]:

- **Banca / seguros:** pueden prevenir pérdidas gracias a la detección de comportamientos irregulares relacionados con robo o fraude.
- **Transporte / logística:** utilizan la tecnología *RFID (Radio Frequency Identification)* para llevar un control y seguimiento con una cámara móvil, sin el uso de infraestructuras.
- **Educación:** simula casos reales prácticos, ofreciendo información de calidad sobre materias y sectores, aumentando la eficacia del aprendizaje.
- **Industria / manufacturas:** previene incidentes y planifica tareas de soporte y mantenimiento, minimizando las pérdidas por interrupción en la producción y costes.
- **Energías:** permite reducir el impacto medioambiental y consumo energético, así como personalizar la oferta de servicios al cliente gracias al análisis de datos e imágenes para el pronóstico de la demanda.

Estas aplicaciones son un porcentaje de la cantidad total que utilizan las tecnologías de visión artificial actualmente. Para llevar a cabo esta tecnología en las diversas aplicaciones citadas, es necesario conocer las tareas de la que se ocupa este campo de investigación. Estas tareas se pueden clasificar desde tareas simples como clasificar una imagen hasta otras más complejas como aumentar la resolución de una imagen. Entre las tareas más importantes, se encuentran las siguientes [5]:

- **Clasificación de imágenes:** esta tarea permite a los ordenadores ver una imagen y clasificarla con precisión a la clase que pertenece. La visión artificial entiende las clases y las etiqueta, como por ejemplo diferenciar entre un perro o un gato.
- **Localización:** en la tarea de localización, la red neuronal es la encargada de proporcionar varios valores a la salida. Estos valores representan una caja, o *bounding box*, que se encarga

de encerrar al objeto principal que aparece en la imagen. Si combinamos esta tarea con la de clasificación, obtenemos la posición en la que se encuentra el objeto de la imagen, además de reconocer un objeto.

- **Clasificación y detección de objetos:** tarea de visión artificial utilizada para detectar y localizar imágenes. La clasificación se ocupa de identificar, ordenar y organizar las imágenes. En el caso de la detección, esta se utiliza en procesos de fabricación e industriales para controlar aplicaciones autónomas y líneas de producción. Las empresas de cámaras domésticas también llevan a cabo esta tarea en sus cámaras, siendo capaces de detectar personas y objetos en tiempo real, proporcionando una alerta temprana a sus clientes.
- **Segmentación:** la segmentación es un algoritmo de visión artificial, capaz de identificar un objeto dividiendo sus imágenes en función de los píxeles que se ven. La segmentación también tiene la capacidad de reconocer si hay un objeto o más en una imagen o marco. A diferencia de la clasificación de objetos, que consiste en crear una caja en las proximidades de la imagen, la segmentación indaga los píxeles, determinando la forma del objeto, facilitando así su análisis y etiquetado. En la Figura 2-3, se puede ver la diferencia de las distintas tareas explicadas hasta ahora.

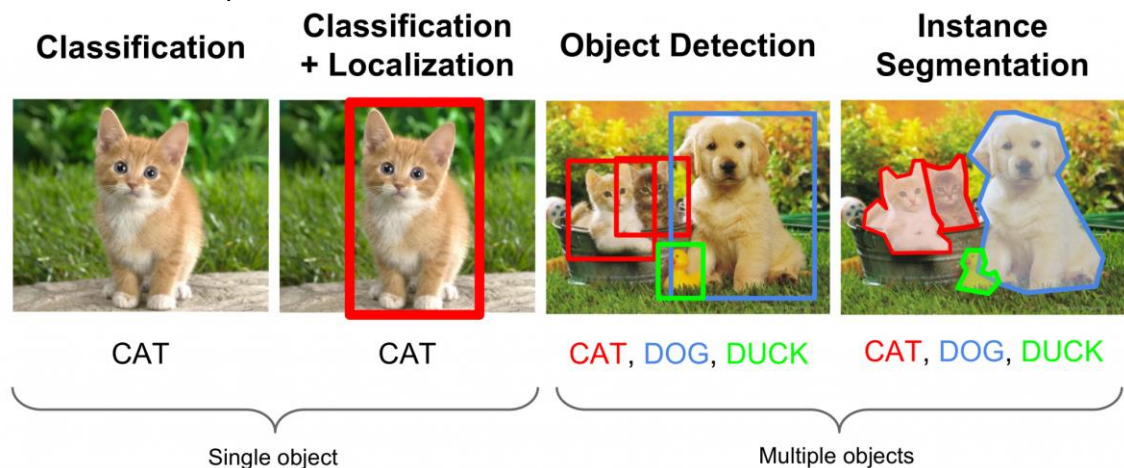


Figura 2-3 Diferencia gráfica entre las diversas tareas [6]

- **Seguimiento de objetos:** en esta tarea se utilizan modelos de aprendizaje profundo (*Deep Learning*) para la identificación y rastreo de los elementos pertenecientes a categorías. El primer elemento del seguimiento de objetos es la detección de objetos. El objeto es encuadrado mediante un marco en su contorno, donde se le asigna un identificador y se rastrea a través de marcos. Es muy útil para supervisar el tráfico en entornos urbanos o marítimos.
- **Detección de puntos de interés:** además de detectar objetos, gracias a la visión artificial se pueden detectar puntos claves o de interés de los objetos detectados. Tomando de ejemplo la detección de personas, permite destacar las partes del cuerpo tales como su cabeza, articulaciones o extremidades.
- **Recuperación de imágenes basada en el contenido:** esta aplicación se ocupa de buscar imágenes digitales específicas en bases de datos de gran tamaño. Se encarga de analizar metadatos como etiquetas, palabras claves y descripciones para llevar a cabo lo requerido por el usuario.

2.1.4 Deep Learning

Como se aprecia en la Figura 2-1, el proceso de *Deep Learning* se encuadra dentro de *Machine Learning*, una de las ramas subconjunto de la IA. El *Deep Learning* o aprendizaje profundo, es una técnica de aprendizaje basada en el modelo de red neuronal: se agrupan decenas o incluso cientos de capas de neuronas con el objetivo de aportar mayor complejidad.

El funcionamiento del *Deep Learning* se fundamenta en su uso en redes neuronales estructuradas en forma de múltiples capas secuenciales, que suelen dividirse en las siguiente: *input layer* (capa de entrada de datos), *hidden layers* (capas ocultas concertadas en serie que realizan el procesado de la formación) y *output layer* (capa de salida donde se representa la respuesta obtenida por la red neuronal). Los datos entran por la primera capa, donde diversas neuronas artificiales se activarán o no dependiendo de los datos. Tomando el ejemplo de la Figura 2-4, donde se intenta identificar la presencia de un perro en una imagen, en la primera capa se pueden identificar las diferencias entre las zonas claras y oscuras, descubriendo cuales serían los bordes de la imagen.

Una vez que los bordes han sido identificados, la red pasará la información anteriormente recogida al segundo nivel, nivel donde construye formas simples con los bordes obtenidos de la capa anterior. El tercer nivel de la red combina las formas simples, obteniendo objetos más complejos de identificar. A medida que va pasando la información de capa a capa, esta es cada vez más compleja, es decir, cada vez representa conceptos más abstractos, hasta, finalmente, conseguir la identificación del animal.

Para llegar a este proceso, es necesario que la máquina o el ordenador lleve a cabo un proceso de entrenamiento, donde se debe combinar el aprendizaje supervisado (el algoritmo se guía por las etiquetas proporcionadas por una persona) y el aprendizaje no supervisado (el ordenador es el encargado de encontrar sus patrones para establecer relaciones a partir de los datos que se le han proporcionado).

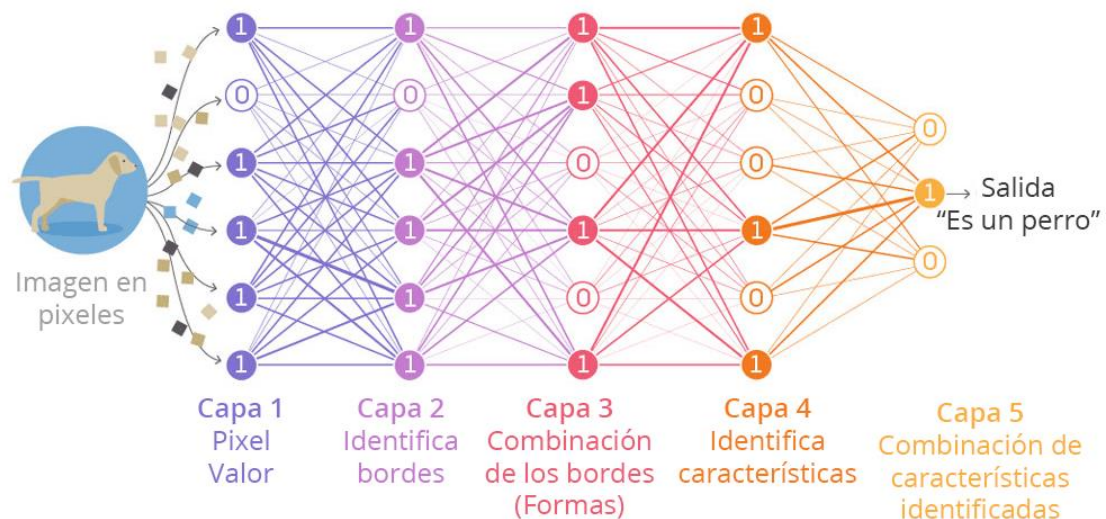


Figura 2-4 Funcionamiento *Deep Learning* [7]

El *Deep Learning* ofrece numerosas ventajas, destacando las siguientes [8]:

- El rendimiento del sistema aumenta su eficacia cuanto mayor sea la cantidad de datos tratados.
- La arquitectura del sistema es flexible, evolucionando y adaptándose a nuevos problemas en el futuro.
- Las características del sistema no son definidas anteriormente, ya que el sistema las deduce, ajustándose a ellas impulsando la automatización de procesos.
- La fiabilidad del sistema se produce de forma automática frente a las variaciones naturales de los datos, produciendo una reducción considerable de los errores.

2.1.5 Redes convolucionales

2.1.5.1 Historia de las redes convolucionales

Kunihiko Fukushima y Yann LeCun fueron los que sentaron las bases para el estudio de las redes neuronales convolucionales con sus respectivos trabajos en 1980. A Yann LeCun, se le atribuye haber

aplicado con éxito la propagación para entrenar redes neuronales pudiendo identificar y reconocer patrones. En la década de los 90, continuó su investigación con su equipo, llevando a cabo *LeNet-5*, que aplica los mismos principios de su anterior investigación para el reconocimiento de documentos. Desde esa época hasta ahora, han surgido diferentes arquitecturas de *CNN* con la introducción de nuevos conjuntos de datos, como *MNIST* (*Modified National Institute of Standards and Technology*) y *CIFAR-10* (*Canadian Institute for Advanced Research*), y competiciones como *ImageNet*, *AlexNet* y *GoogLeNet*. Con la Figura 2-5, se muestra la línea temporal más importantes acerca de las *CNN*.

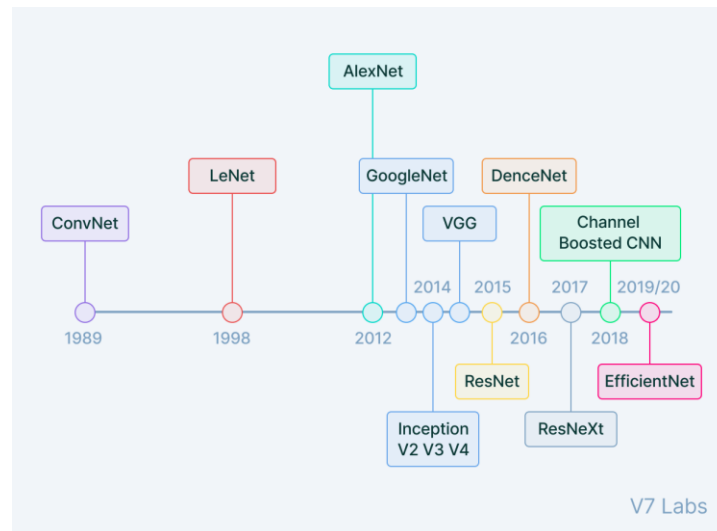


Figura 2-5 Línea temporal de los hitos más importantes de la historia de CNN [9]

2.1.5.2 Concepto de las redes convolucionales

Las redes neuronales convencionales, también conocidas como *CNN*, constituyen un tipo específico de red neuronal artificial diseñado especialmente para analizar datos que presentan una organización en forma de malla, como por ejemplo imágenes, vídeos o señales de audio. Estas redes emplean una operación matemática denominada convolución para identificar características significativas en los datos de entrada y adquirir la capacidad de reconocer patrones dentro de ellos.

Las capas convolucionales de una *CNN* están compuestas por varios filtros o *kernels*, aplicando estos filtros de forma repetida a la imagen de entrada, generando cada uno un mapa de características. La primera etapa de convolución se ve refleja en la Figura 2-6. Los mapas de características se pasan a través de capas de muestreo, también conocidas como capas de *pooling*, donde se reduce el tamaño para que la complejidad computacional de la red disminuya. En el último paso, las capas completamente conectadas de la *CNN* son las encargadas de clasificar la imagen en una o varias categorías. En el apartado 2.1.5.3, se explica con más detalles todas las capas que engloban la arquitectura de las redes convolucionales.

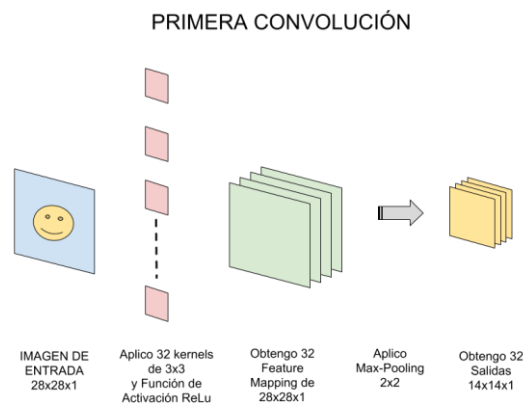


Figura 2-6 Primera etapa de la convolución [10]

Estas redes convolucionales son muy útiles en tareas de visión por computadora, como la clasificación de imágenes, la detección de objetos y la segmentación. Han revolucionado el campo de la visión por computadora gracias a su capacidad de aprender y extraer características de forma automática, además de permitir la creación de aplicaciones prácticas como el reconocimiento facial en imágenes, la clasificación de imágenes en redes sociales y la conducción autónoma en el ámbito de los vehículos.

2.1.5.3 Arquitectura de las redes convolucionales

Las redes neuronales convolucionales se caracterizan por su gran rendimiento en sistemas de datos de imágenes o señales de audio. La arquitectura está compuesta por tres tipos principales de capas [11]:

- **Capa convolucional (*Convolutional Layer*):** esta capa es el principal bloque de creación de una *CNN*, en la que se realizan la mayoría de los cálculos. Para dichos cálculos, requiere componentes como datos de entrada, un filtro y un mapa de características. También existe un detector de características, conocido como *kernel* o filtro, que se mueve por los campos de la imagen para comprobar si la característica está presente. Este proceso es el conocido como convolución. Posteriormente de cada convolución, la *CNN* aplica una transformación lineal rectificada **ReLU Layer**. Esta capa se caracteriza por utilizar la función de activación *ReLU*, donde cada valor negativo en la salida de la capa de convolución se reemplaza por cero.
- **Capa de agrupación (*Pooling Layer*):** también conocida como muestreo, su función es reducir la dimensión mediante la disminución del número de parámetros de la entrada. Al contrario que en la capa de convolución, en este caso el filtro no tiene ningún peso. En este caso, el *kernel* aplica una función de agregación a los valores dentro del campo, llenando la matriz de salida. Hay dos tipos principales de agrupación:
 - **Agrupación máxima (*Max Pooling*):** se encarga de seleccionar el elemento máximo de cada una de las ventanas del mapa de características. Se utiliza más que la agrupación media. En la Figura 2-7, se puede ver un ejemplo de este caso.
 - **Agrupación media (*Average Pooling*):** calcula el promedio de los elementos presentes en la región del mapa de características cubierta por el filtro.

Además, esta capa se caracteriza por la pérdida de información, pero en su beneficio ayuda a reducir la complejidad y mejorar la eficiencia.

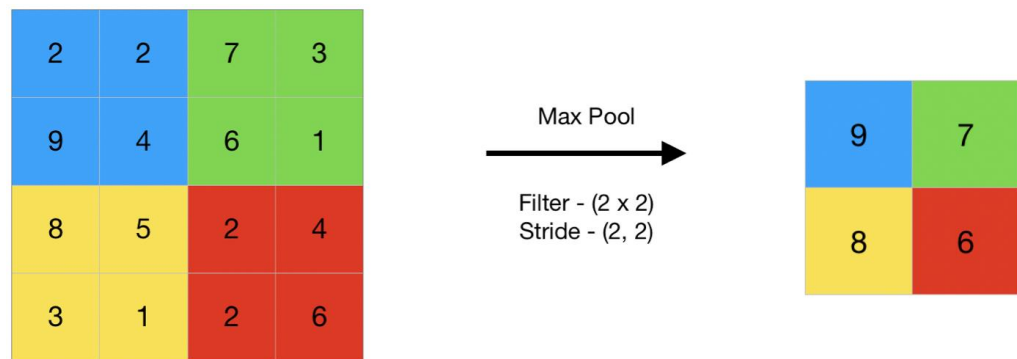


Figura 2-7 Ejemplo de agrupación máxima (*Max Pooling*) [12]

- **Capa totalmente conectada (*Fully-Connected Layer*):** en esta capa se realiza la tarea de clasificación basándose en las características extraídas de las capas anteriores y sus diferentes filtros. Al contrario que las capas convolucionales y de agrupación, donde utilizan función *ReLU*, en la capa totalmente conectada se usa una función de activación *softmax*, que ayuda a clasificar con certeza las entradas y generar una probabilidad entre 0 y 1.

En la Figura 2-8 se aprecia cómo se ha realizado la integración entre las diferentes capas explicadas y las diferentes funciones de activación explicadas en el apartado 2.1.5.3 en el algoritmo *YOLO*.

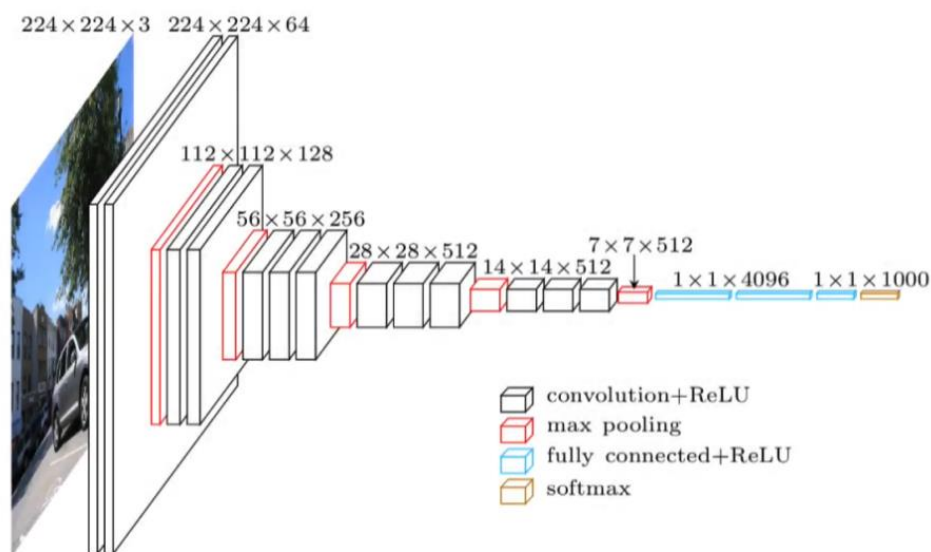


Figura 2-8 Ejemplo de la estructura de una red convolucional [13]

2.1.6 Algoritmo YOLO

YOLO es un algoritmo de detección de objetos en imágenes y vídeos, utilizado en el ámbito de la visión por computadora. Este algoritmo fue presentado por primera vez en 2016 por *Joseph Redmon*, *Santosh Divvala*, *Ross Girshick* y *Ali Farhadi*, en un artículo de investigación llamado “*You Only Look Once: Unified, Real-Time Object Detection*”. Este algoritmo usa una red neuronal convolucional siendo capaz de detectar objetos en una sola pasada en un vídeo o imagen. Esto es lo que le diferencia de otros algoritmos, ya que otros algoritmos necesitan varias pasadas para analizar la detección de objetos de una imagen completa, reduciendo el rendimiento. En este caso *YOLO*, es capaz de realizar detección de objetos en tiempo real con gran rendimiento.

Desde el 2015 hasta el día de hoy, se ha comprobado una evolución de *YOLO* hasta *YOLOv8*. Esta última versión se caracteriza por tener mayor precisión de los modelos anteriores. Es capaz de realizar

diferentes tareas de la visión artificial, como las que se han detallado en el apartado 2.1.3, al mismo tiempo, como pueden ser la detección de objetos, la segmentación y la clasificación de imágenes. Ha sido capaz de obtener un rendimiento mejorado, flexibilidad y eficiencia con respecto a las otras versiones anteriores. En la Figura 2-9, se puede observar la línea temporal de la evolución del algoritmo desde 2015.

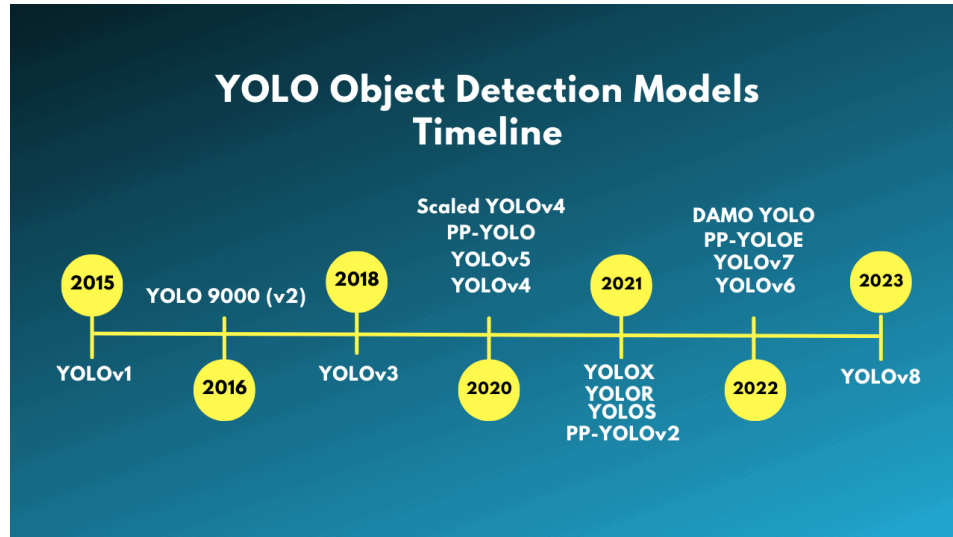


Figura 2-9 Evolución de las diferentes versiones del algoritmo YOLO [14]

2.1.6.1 Funcionamiento del algoritmo YOLO

Para explicar cómo funciona el algoritmo YOLO, vemos una descripción general [15]:

1. En primer lugar, la imagen se divide en una cuadrícula. Cada celda de la cuadrícula dada, se utiliza para predecir la presencia de objetos y sus propiedades, como pueden ser el tamaño, la clase y la ubicación en la imagen.
2. En este paso, lo que se busca es entrenar una red neuronal convolucional en un grupo de imágenes etiquetadas manualmente por medio del usuario. La red neuronal tiene la capacidad para aprender a detectar y clasificar los objetos en las imágenes de entrenamiento.
3. Posteriormente al entrenamiento, la red neuronal se utiliza para el análisis de imágenes o vídeos nuevos, con el fin de predecir la presencia y las propiedades de los objetos en la imagen.
4. A continuación, se lleva a cabo un proceso conocido como “suavizado de la ubicación”, que sirve para mejorar la precisión de la detección de objetos y eliminar falsos positivos.
5. Tras haber realizado el análisis, se muestran los resultados de la detección de objetos en la imagen, proporcionando las clases de los objetos detectados y de las ubicaciones en la imagen o vídeo.

En la Figura 2-10, se hace un resumen gráfico de cómo funciona este algoritmo.

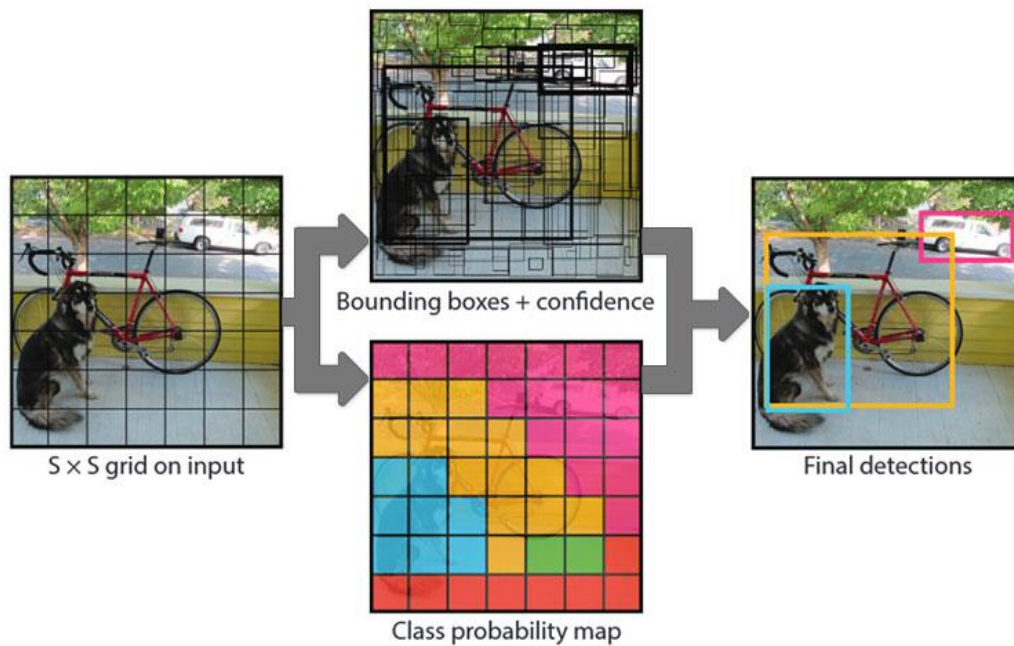


Figura 2-10 Gráfico resumen del algoritmo YOLO [16]

2.1.6.2 Métricas del algoritmo

Como se ha visto anteriormente, en los primeros momentos del entrenamiento, las ventanas generadas por la red vendrán dada por una dimensión y posición incierta. A medida que el entrenamiento se vaya realizando, estas ventanas generadas coincidirán en posición y tamaño con las que el banco de imágenes ofrece como verdad subyacente (*ground truth*). Para entender esto con mejor conocimiento, se detalla a continuación las métricas más importantes del algoritmo [17]:

- **Índice de Jacquard:** también se conoce como intersección sobre unión. Se encarga de medir el área común de solapamiento entre dos ventanas (intersección) y se divide la superficie que se ha obtenido de unir a las dos en una unión. En el caso de que las dos ventanas coincidan, la intersección sobre unión es uno. En el otro caso de que no exista superficie en común, el valor es igual a cero. En diversos sistemas de localización de objetos, se dice que ha habido un positivo, cuando el índice de *Jacquard* entre dos ventanas es superior a 0,5. En caso contrario, es un negativo.
- **Precisión media:** (*mAP*, *mean Average Precision*). Esta propiedad se utiliza para medir el rendimiento de los sistemas durante su proceso de entrenamiento. Es el resultado de la combinación de dos variables diferentes que se explica en este apartado: la precisión (*precision*) y el recuerdo (*recall*).
- **Precision:** esta variable se encarga de cuantificar que ventanas son auténticas, es decir, cuáles de ellas coinciden con los que existen en la verdad subyacente de todas las ventanas detectadas hasta el momento por la red.
- **Recall:** esta variable mide el porcentaje de ventanas de todas aquellas con objetos existentes en la verdad subyacente, han sido detectados por la red. Este valor solo puede aumentar a medida que vamos avanzando a lo largo de todos los datos.

Destacar de que todas las variables que llega a medir el algoritmo, los que juegan un papel fundamental son el índice de *Jacquard* y la precisión media.

2.1.6.3 Algoritmo YOLOv8

Independientemente de la versión del algoritmo YOLOv8, estas se dividen en función de su número de capas y complejidad computacional. Las diferentes versiones pueden ser *nano*, *small*, *medium*, *large* y *extra large*. Con cada variante anteriormente mencionadas, el número de parámetros aumenta

considerablemente, así como la precisión de detección. Sin embargo, en el caso de la velocidad también aumenta, por lo que conlleva a la lentitud del procesamiento y peso. En la Tabla 2-1, se puede apreciar los diferentes valores tanto de tamaño, velocidad y parámetros en las diferentes versiones de YOLOv8.

Model	Size (pixels)	mAP ^{val} 50-95	Speed CPU ONNX (ms)	Speed A100 TensorRT (ms)	Params (M)	FLOPs (B)
YOLOv8n	640	37.3	80.4	0.99	3.2	8.7
YOLOv8s	640	44.9	128.4	1.20	11.2	28.6
YOLOv8m	640	50.2	234.7	1.83	25.9	78.9
YOLOv8l	640	52.9	375.2	2.39	43.7	165.2
YOLOv8x	640	53.9	479.1	3.53	68.2	257.8

Tabla 2-1 Modelos de YOLOv8 [18]

2.1.7 Seguimiento dinámico de objetos en vídeos

Conocido como *object tracking*, es el proceso de localizar y seguir el movimiento de un objeto en múltiples imágenes o vídeos. Gracias a este proceso, varias áreas se ven beneficiadas por usar este tipo de procesos, como son la visión por computadora, la robótica, la vigilancia y la inteligencia artificial. El objetivo principal del *object tracking*, es determinar que trayectoria y posición tiene un objeto en movimiento en cada cuadro de un vídeo. De esa manera, puede analizar su comportamiento y predecir el movimiento que pueda realizar un objeto, permitiendo tomar decisiones basadas en su seguimiento continuo. Existen diferentes algoritmos de seguimiento de objetos, que suelen emplear técnicas de procesamiento de imágenes y visión por computadora para ser capaz de localizar y seguir la posición de un objeto en un entorno visual.

2.1.7.1 Usos y tipos de seguimiento de objetos

El seguimiento de objetos se utiliza en una gran variedad de casos de uso que implican diferentes tipos de secuencia de entrada. Los algoritmos utilizados se ven afectados por diversos motivos, entre ellos el tipo de entrada, que afecta la categoría, los casos de uso y las aplicaciones del seguimiento de objeto. Algunos de los usos y tipos más populares en este ámbito son [19]:

- **Seguimiento de vídeo (Video Tracking):** este caso es una aplicación en la que los objetos en movimiento se encuadran dentro de la información del vídeo. Esto quiere decir, que los sistemas de seguimiento pueden procesar imágenes en vivo y tiempo real. Los procesos utilizados para ejecutar tareas de *video tracking*, difieren según el tipo de entrada de vídeo a la que se dirige.
- **Seguimiento visual (Visual Tracking):** también conocido como seguimiento visual de objetivos, es un tema de investigación en visión artificial, aplicado en una amplia gama de escenarios del día a día. El objetivo en este caso es predecir la posición futura de un objeto visual sin disponer del resto del vídeo.
- **Seguimiento de imágenes (Image Tracking):** este tipo está destinado a detectar imágenes de interés en dos dimensiones. Posteriormente, esa imagen es rastreada en continuo a medida que se mueven en el entorno. *Image Tracking* es ideal para conjuntos de datos con imágenes de alto contraste, asimetría, pocos patrones y múltiples diferencias que se pueden identificar entre la imagen de interés y otras imágenes del conjunto de imágenes.

Destacar que los métodos modernos de seguimientos de objetos se pueden aplicar en transmisiones de vídeo en tiempo real con cualquier tipo de cámara. Una cámara USB (*Universal Serial Bus*) o una

cámara IP (*Internet Protocol*), son utilizadas para llevar a cabo el seguimiento, alimentando los fotogramas a un algoritmo de seguimiento. El salto de fotogramas o el procesamiento en paralelo son métodos comunes que tienen la finalidad de mejorar el rendimiento del seguimiento de objetos con una o varias cámaras en retransmisiones en tiempo real.

2.1.7.2 Problemática del tracking de objetos

Rastrear objetos también tiene sus dificultades, surgiendo problemas en la imagen que hacen que la efectividad de realizar *tracking* de manera correcta disminuya. Varios de los problemas más “clásicos” son los siguientes [20], dando ejemplos de todos ellos en la Figura 2-11:

- **Oclusión:** este problema se presenta cuando un objeto que está siendo rastreado queda oculto en un momento en particular o parcialmente, quedando detrás de una columna u otro objeto.
- **Superposición de objetos:** un ejemplo de este caso ocurre, por ejemplo, en un partido de fútbol. Si tenemos a dos jugadores de fútbol, con camiseta blanca ambos, y uno pasa por detrás del otro, el algoritmo puede verse influenciado y no podría ser capaz de diferenciar que jugador es cada uno, debido a que ambos llevan camisetas del mismo color.
- **Transformación del objeto:** explicado de manera práctica, este fenómeno ocurre cuando una persona, que lleva una camiseta roja, camina de frente. En un momento, decide cambiar su rumbo, dando la vuelta y mostrando que la parte de atrás de su camiseta es de color azul. El sistema sabe que es la misma persona, sin embargo, le confunde que durante el transcurso la camiseta haya pasado de color rojo al color azul.
- **Efectos visuales:** fenómeno que ocurre, por ejemplo, cuando se pasa de una zona soleada a una zona con sombra, generando de esa manera una variación en sus colores. Otro ejemplo de este fenómeno es cuando la luz solar es reflejada en un objeto, como puede ser un coche, produciendo un destello, dificultando su identificación.

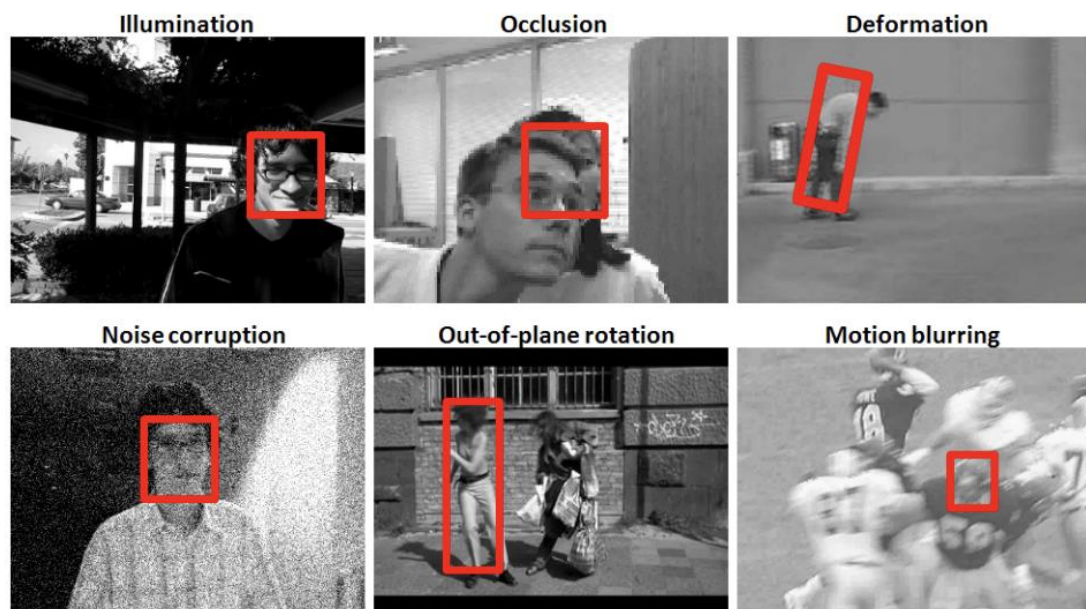


Figura 2-11 Ilustración de cambios de apariencia que dificultan el seguimiento de objetos [21]

2.1.7.3 Niveles de seguimiento de objetos

El seguimiento de objetos se puede definir en dos niveles [22]:

- **Seguimiento de un solo objeto (SOT, Single Object Tracking):** SOT tiene como objetivo rastrear un objeto de una sola clase en lugar de numerosos objetos. En SOT, se define el cuadro delimitador (*bounding box*) del objeto en el primer cuadro. El objetivo que busca el

algoritmo es localizar el mismo objeto haya anteriormente en el resto de cuadros. *SOT* pertenece a la categoría de seguimiento sin detección, ya que se debe asignar manualmente el primer cuadro delimitador al rastreador. Esto quiere decir que los rastreadores de un solo objeto tienen la capacidad de rastrear cualquier objeto que se les proporcione, incluso para un objeto para el cual no ha sido entrenado previamente.

- **Seguimiento de múltiples objetos (*MOT, Multiple Object Tracking*)**: a diferencia de *SOT*, *MOT* se refiere al enfoque que el algoritmo realiza en el rastreo de cada objeto de interés que aparece en el vídeo. En primer lugar, el algoritmo de seguimiento determina el número de objetos en cada cuadro, para realizar posteriormente un seguimiento de la identidad de cada objeto de un cuadro al siguiente, hasta que salen del cuadro. Este nivel de seguimiento es de gran importancia en la conducción autónoma, donde se utiliza para detectar y predecir el comportamiento de peatones u otros vehículos.

2.1.7.4 Algoritmos para seguimiento de objetos en vídeos

Para poder realizar el *object tracking*, y resolver los problemas que se han comentado en el apartado 2.1.7.2, se han desarrollado diversos algoritmos para solucionar este tipo de tareas. Lo que se busca en un algoritmo de detección, es que, en primer lugar, identifique al objeto, manteniendo la “etiqueta” de su identificador con el paso del tiempo, es decir, generando una memoria del seguimiento del objeto y su trayectoria. Los algoritmos más conocidos son [23]:

- ***SORT (Simple Online and Realtime Tracking)***: este algoritmo es una implementación fácil de un marco de seguimiento por detección. *SORT* utiliza la posición y el tamaño de los cuadros delimitadores (*bounding box*) tanto para estimar el movimiento como para la asociación de datos a través de fotogramas. Como detector de objetos se utiliza un *RCNN (Region-Based Convolutional Neural Networks)*. El desplazamiento de los objetos en fotogramas consecutivos se estima mediante un modelo lineal de velocidad constante, siendo independiente de otros objetos y del movimiento de la cámara. Para la asignación de la *ID (Identification)*, se utilizan los nuevos estados de los objetivos, para predecir los cuadros delimitadores que luego se comparan con los cuadros detectados en el marco temporal actual.
- ***DeepSORT***: uno de los inconvenientes que tiene *SORT* son las grandes cantidades de cambios de identidad, a pesar de tener un buen rendimiento general en términos de exactitud y precisión de seguimiento. Para superar este inconveniente, *DeepSORT* consigue reemplazar la métrica de asociación con una métrica que combina información de movimiento y apariencia. En definitiva, se consigue una métrica de distancia de “apariencia profunda”. La idea principal es obtener un vector, que pueda ser utilizado en una imagen determinada.
- ***FairMOT***: es un nuevo enfoque de seguimiento construido sobre la arquitectura de detección de objetos *CenterNet*. Las tareas de detección y reidentificación se tratan por igual en *FairMOT*, que se diferencia de los marcos anteriores de “detección primero, reidentificación secundaria”. Tiene una estructura de red simple, como se puede apreciar en la Figura 2-12, que consta de dos ramas para detectar objetos y extraer características de reidentificación.

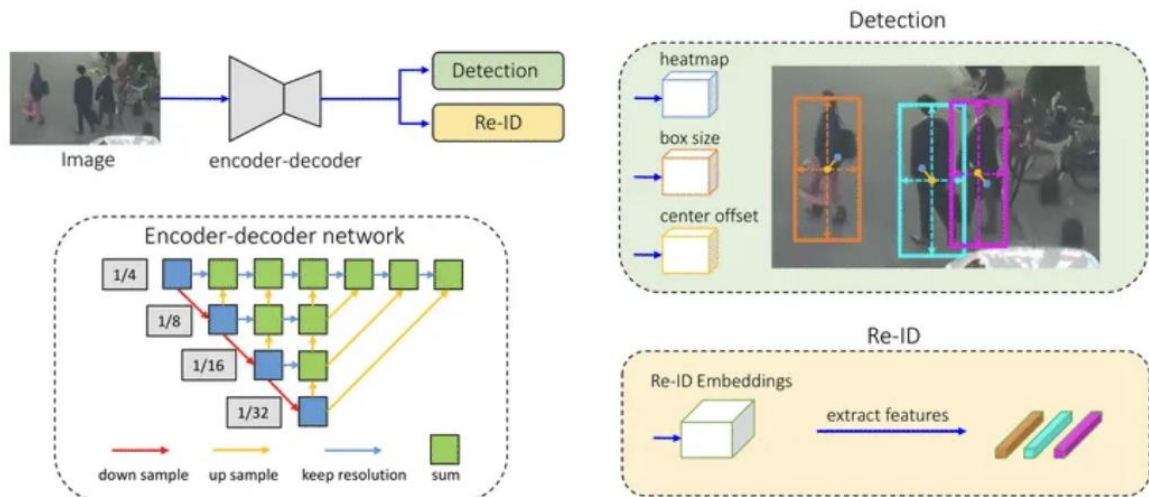


Figura 2-12 Estructura FairMOT [24]

- **TransMOT**: es un nuevo transformador gráfico que organiza las trayectorias de todos los objetos rastreados como una serie de gráficos ponderados dispersos que se construyen utilizando las relaciones espaciales de los objetivos. *TransMOT* utiliza estos gráficos para crear una capa de codificador de transformador de gráfico espacial, una capa de codificador de transformador temporal y una capa de decodificador de transformador espacial para modelar las relaciones espacio-temporales de los objetos. Es un modelo más eficaz que un transformador normal. Para mejorar la velocidad y la precisión del seguimiento, *TransMOT* introduce un marco de asociación en cascada para manejar detecciones con puntuación bajas y oclusiones a largo plazo.
- **ByteTrack**: la mayoría de los métodos de seguimiento obtienen identidades asociando cuadros de detección con puntuaciones superiores al umbral. Esto conlleva a que las puntuaciones de detección bajas se ignoran, lo que genera trayectorias fragmentadas y faltantes de objetos verdaderos. *ByteTrack* es un método de asociación eficaz, que utiliza todos los cuadros de detección, desde puntuaciones altas hasta puntuaciones bajas durante el proceso de comparación. *ByteTrack* lleva a cabo la premisa de que la similitud con los *tracklets*² proporciona una señal sólida para distinguir los objetos y el fondo en cuadros de detección de puntuación baja. Este algoritmo lo que, en primer lugar, hace coincidir los cuadros de puntuación más alta con los *tracklets* en función de similitud de movimiento. Posteriormente, realiza la segunda comparación entre los *tracklets* no coincidentes. Para una mejor comprensión de este algoritmo, en la Figura 2-13 se observa cómo funciona el mismo.

² *Tracklet*: fragmento de la trayectoria seguida por un objeto en movimiento, tal como es construido por un sistema de reconocimiento de imágenes.

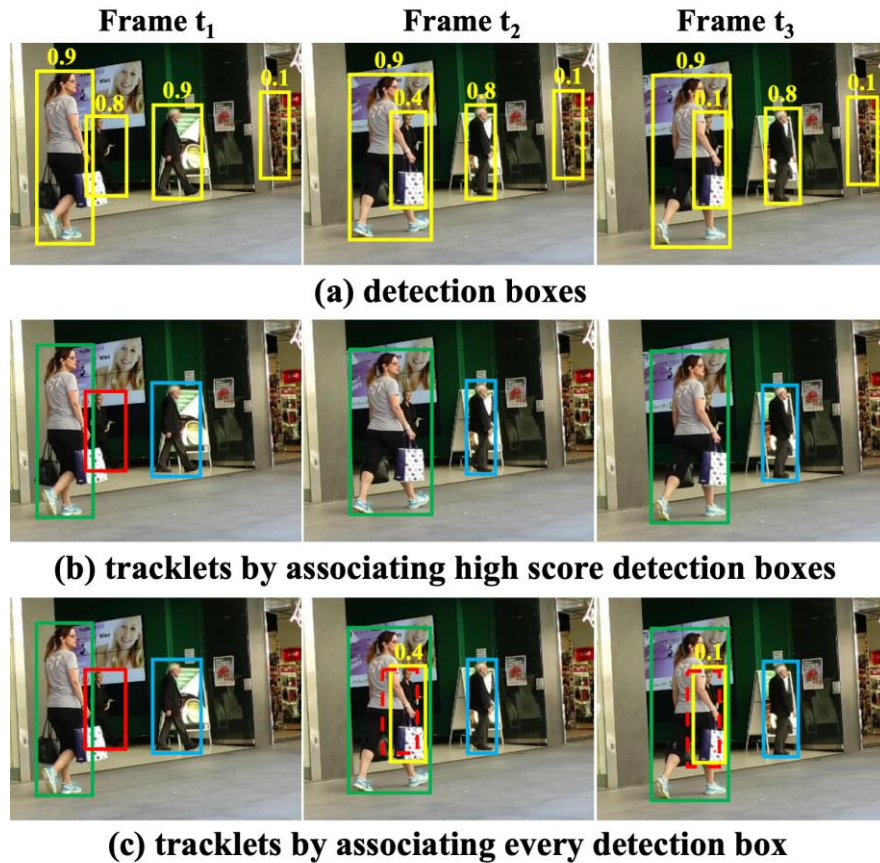


Figura 2-13 Ejemplo del uso del algoritmo *ByteTrack* [25]

2.2 Visión estereoscópica

2.2.1 Concepto de la visión estereoscópica

La visión estereoscópica es la capacidad que tiene un ser vivo de integrar dos imágenes que perciben cada uno de sus ojos en una sola imagen en relieve (Figura 2-14). Esto quiere decir que la percibe de manera tridimensional y con la profundidad suficiente. Este suceso tiene lugar cuando un ojo ajusta el cristalino enfocando un objeto, de forma que ambos ejes ópticos concurren sobre dicho objeto. En el proceso de integración se usan la estereoscopia y la visión binocular. Destacar que la visión estereoscópica puede darse de manera directa o indirecta, como se explica en los apartados 2.2.1.1 y 2.2.1.2, respectivamente.

El proceso de la visión estereoscópica lo realiza el cerebro humano de la siguiente manera:

- Cada uno de los ojos percibe las dos imágenes, y estas son recibidas en el cerebro.
- Al estar los ojos en una posición diferente, ambas imágenes son distintas. El cerebro se encarga de procesarlas y analizarlas adecuadamente.
- Finalmente, una vez que el cerebro ha realizado el proceso de procesarlas y analizarlas, este genera una única imagen en relieve, tridimensional y con profundidad.

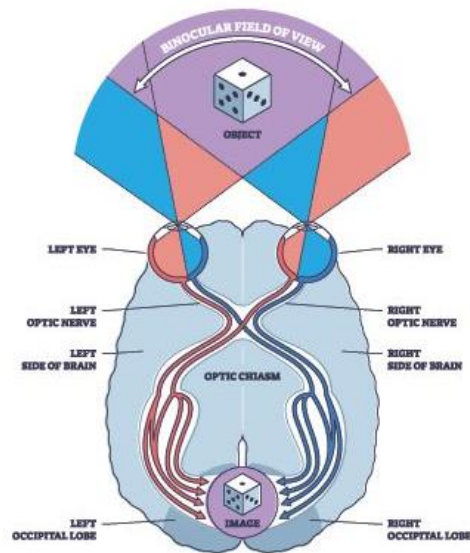


Figura 2-14 Proceso en la visión estereoscópica [26]

2.2.1.1 Visión estereoscópica directa

Este tipo de visión estereoscópica es aquella que permite captar las imágenes en tres dimensiones sin necesidad de recurrir a métodos artificiales o elementos ópticos auxiliares. El ser humano no es capaz de percibir el relieve o la profundidad mediante una sola imagen, debido a que los puntos de la imagen se proyectan en un mismo plano. Para que se perciba la profundidad se deben dar las siguientes condiciones:

- Cada una de las imágenes vistas por cada ojo han de tener pequeñas diferencias entre ellas, observando cada ojo una imagen.
- En algún punto, deben cortarse lo que ve un ojo y el otro de la imagen percibida.
- No se debe exceder el límite de la convergencia natural, que aproximadamente son unos 20 grados.

2.2.1.2 Visión estereoscópica indirecta

En este caso, se recurre al uso de elementos ópticos auxiliares, sustituyendo de manera artificial la visión estereoscópica directa. Para realizar este proceso, se hace uso de un elemento óptico de observación que permita presentar una imagen a cada ojo de manera aislada. El sistema físico-psicológico es el encargado de fusionar el asunto de ambas imágenes, creando un modelo tridimensional. El elemento óptico más utilizado es el estereoscopio, que permite a cada ojo percibir una imagen diferente. Hay diferentes tipos de estereoscopios, pero los más utilizados son los de lentes, de espejos y de espejos y prismas.

2.2.2 Origen de la visión estereoscópica

Para entender el origen de la visión estereoscópica, hay que volver la vista atrás hasta el Renacimiento. Durante esta época, artistas y científicos comenzaron a analizar la anatomía y óptica del ojo, siendo *Leonardo da Vinci* uno de los primeros en sugerir que la visión binocular, que es el uso de ambos ojos, era esencial para la percepción de la profundidad. Posteriormente, el científico británico *Sir Charles Wheatstone*, en el año 1838, propuso el principio de la visión estereoscópica. Para argumentar dicho principio, se basó en que el cerebro fusiona las imágenes de ambos ojos para percibir la profundidad. Además, el científico británico fue el encargado de construir el primer aparato estereoscópico (Figura 2-15), permitiendo presentar dos imágenes ligeramente diferentes a cada ojo. Esta invención sentó las bases para la fotografía estereoscópica. Años más tardes, en 1845, *Sir David*

Brewster fue el encargado de diseñar y construir la primera cámara estereoscópica. Esta cámara disponía de un visor, permitiendo ver las imágenes tomadas por las lentes.

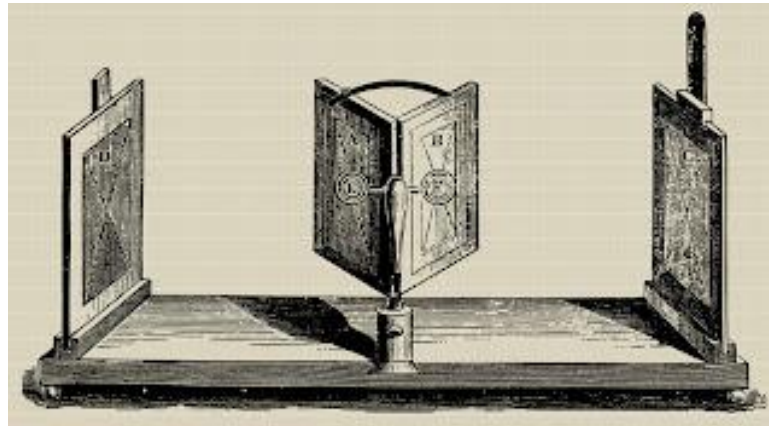


Figura 2-15 Estereoscopio de *Wheatstone* [27]

Oliver Wendell Holmes se encargó de mejorar el diseño del estereoscopio y popularizó su uso en la década de 1860. Al igual que los otros científicos, también fue pionero en construir el estereoscopio de mano más popular del siglo XIX. Estos dispositivos se convirtieron en objetos de entretenimiento, llamando la atención para ser vendidos. Gracias a este auge de los dispositivos, la fotografía estereoscópica ganó popularidad en la segunda mitad del siglo XIX. Fotógrafos como el británico *Roger Fenton* produjeron imágenes estereoscópicas, en las cuales ofrecía vistas en tres dimensiones de paisajes y eventos.

Este fenómeno de la visión estereoscópica fue utilizado en la década de 1950 en el ámbito de las películas y el cine. Se desarrollaron sistemas de proyección de películas en 3D, con películas como “*Bwana Devil*” en 1952 y “*House of Wax*” en 1953. Estas películas marcaron el comienzo de la era moderna del cine 3D, ya que se empezó a dar uso de gafas especiales para crear la ilusión tridimensional. A finales del siglo XX, con el avance de la tecnología digital, se desarrollaron nuevas formas de presentar imágenes estereoscópicas. Las gafas 3D ya eran de uso común para ver contenido estereoscópico, tanto en el cine como en la televisión. Aparecieron dos fenómenos que consiguieron llevar a la visión estereoscópica a un nuevo límite, ofreciendo experiencias inmersivas y tridimensionales gracias a la realidad virtual y la realidad aumentada.

En la actualidad, la visión estereoscópica se puede apreciar en diversos campos como la medicina (para la visualización tridimensional de imágenes médicas), la industria del entretenimiento (en videojuegos y películas en tres dimensiones) y en la educación (con simulaciones interactivas en 3D). Esta tecnología 3D sigue evolucionando, explorando formas de presentar información tridimensionalmente sin necesidad de gafas especiales. Es por ello que sigue siendo un área fascinante de investigación y desarrollo.

2.2.3 Percepción de la profundidad

En primer lugar, hay que tener claro el concepto de percepción de la profundidad. La percepción de profundidad permite recibir el entorno en tres dimensiones, además de medir de forma precisa la distancia hasta un objeto [28]. Principalmente, ésta se basa en la visión binocular, pero también utiliza diversos recursos como señales monoculares para lograr una percepción integrada final. Estos diversos recursos o métodos pueden ser los siguientes:

- **Señales monoculares:** señales de profundidad que pueden percibirse con una sola imagen o cámara. Las imágenes observadas incluyen perspectiva, sombreado, oclusión y gradiente de textura. Todas estas señales ayudan a estimar la profundidad incluso cuando no se dispone de un sistema de visión estereoscópica.

- **LiDAR (*Ligh Detection And Ranging*)**: los sensores *LiDAR* hacen uso de rayos láser para medir la distancia a objetos, proporcionando información acerca la profundidad. Esta información se puede fusionar con datos visuales para una percepción de profundidad más precisa.
- **Cámaras de tiempo de vuelo (*ToF, Time-of-Flight*)**: este tipo de cámaras miden el tiempo que tarda la luz en rebotar en un objeto, y su camino de vuelta hasta la cámara. Es una información muy útil para la estimación de la profundidad. En la Figura 2-16, se observa el funcionamiento de este tipo de cámaras.

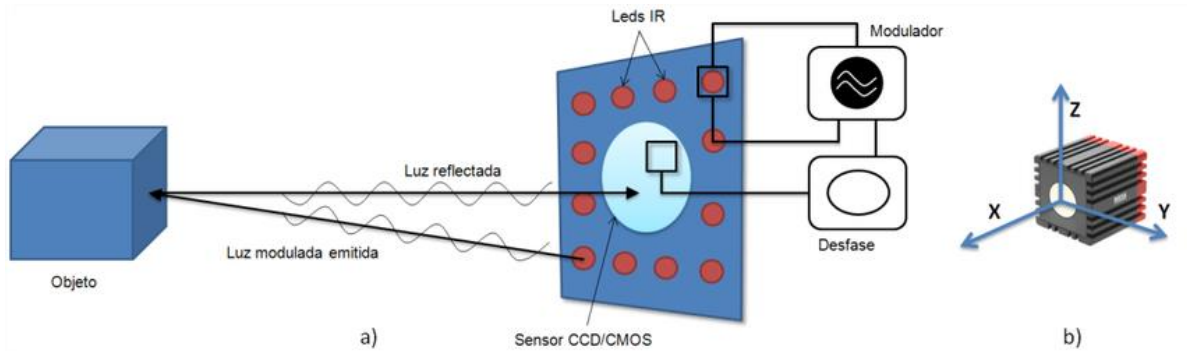


Figura 2-16 Esquema del funcionamiento de una cámara de tiempo de vuelo (*ToF*) [29]

La percepción de la profundidad es decisiva a la hora de realizar tareas como evitar obstáculos, reconocer objetos y comprender escenas en aplicaciones de visión por computadora. Para tener una estimación y percepción de la profundidad efectiva, diversos componentes hacen que esto se cumpla. Los componentes utilizados en visión estereoscópica y en la percepción de la profundidad en visión por computadora son los siguientes [30]:

- **Cámara estéreo**: el sistema de visión estereoscópica se basa en el uso de dos o más cámaras, colocadas a una distancia conocida entre sí. La función que realizan estas imágenes es la captura de imágenes de la misma escena desde diferentes puntos de vista, simulando la percepción que percibe un ser humano desde sus ojos.
- **Captura de imágenes**: las cámaras utilizadas se encargan de capturar imágenes o videos. Generalmente, las imágenes capturadas se denominan imagen izquierda o imagen derecha, según la cámara que ha realizado la captura.
- **Calibración**: las cámaras estereoscópicas deben estar calibradas para calcular con precisión la información de profundidad. Este proceso implica determinar parámetros de la cámara como la matriz intrínseca, los coeficientes de distorsión y los parámetros extrínsecos. Gracias a la calibración se garantiza que las imágenes se puedan rectificar y combinar con eficacia. En el apartado 2.2.4 se detalla todos los parámetros citados anteriormente.
- **Rectificación**: este proceso se basa en una transformación geométrica aplicada a las imágenes capturadas con el fin de alinear las características correspondientes entre ambas imágenes, como queda reflejado en la Figura 2-17. Simplifica el proceso de coincidencia al hacer que las disparidades sean más predecibles.

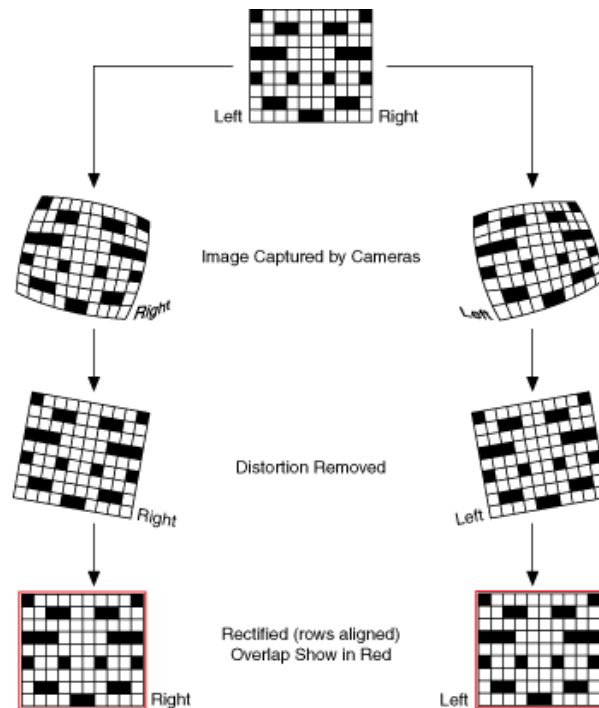


Figura 2-17 Proceso de rectificación [31]

- **Coincidencia estéreo:** este proceso se encarga de encontrar correspondencias entre ambas imágenes. Existen diversos algoritmos de coincidencia estéreo, donde se incluyen la coincidencia de bloques, la coincidencia semiglobal y los cortes de gráficos, que ayudan a encontrar dichas correspondencias.
- **Mapa de disparidad:** el mapa de disparidad es una imagen en escala de grises, donde cada píxel tiene una intensidad diferente, correspondiéndose a la disparidad o profundidad en ese punto. Mientras más cerca este un objeto de la cámara, mayor disparidad tendrá. En el caso de estar más lejos, menos disparidad. En la Figura 2-18, se aprecia tanto un ejemplo de mapa de disparidad como mapa de profundidad.
- **Mapa de profundidad:** derivado del mapa de disparidad, utiliza la distancia entre las cámaras y la distancia focal de las cámaras para llevar a cabo dicho mapa. Calcula la profundidad de cada píxel en unidades del mundo real, en lugar de disparidades.

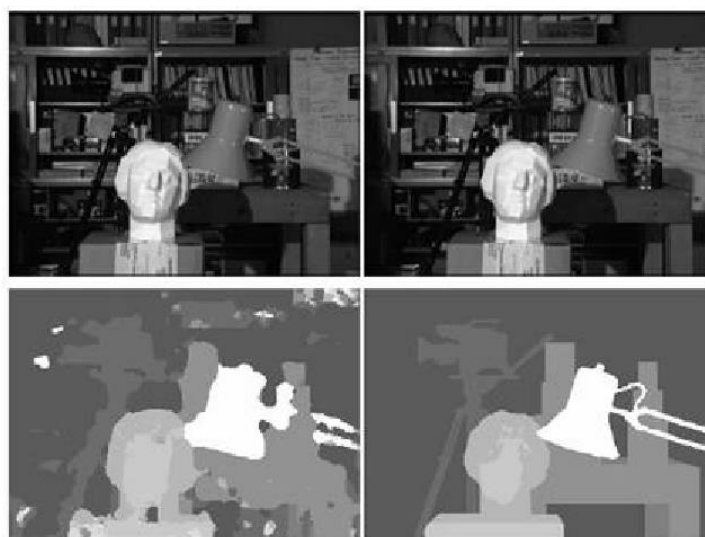


Figura 2-18 Ejemplo de mapa de disparidad y mapa de profundidad en una imagen [32]

- **Hardware:** además de las cámaras, se puede utilizar diferente hardware especializado como cámaras de detección de profundidad como por ejemplo *Microsoft Kinect* o por sensores *LiDAR* para obtener información de profundidad. Estos sensores proporcionan profundidad directamente, sin necesidad de coincidencia estéreo.

En la Figura 2-19, se puede apreciar una descripción general de un sistema de visión estereoscópica para determinar la percepción de profundidad.

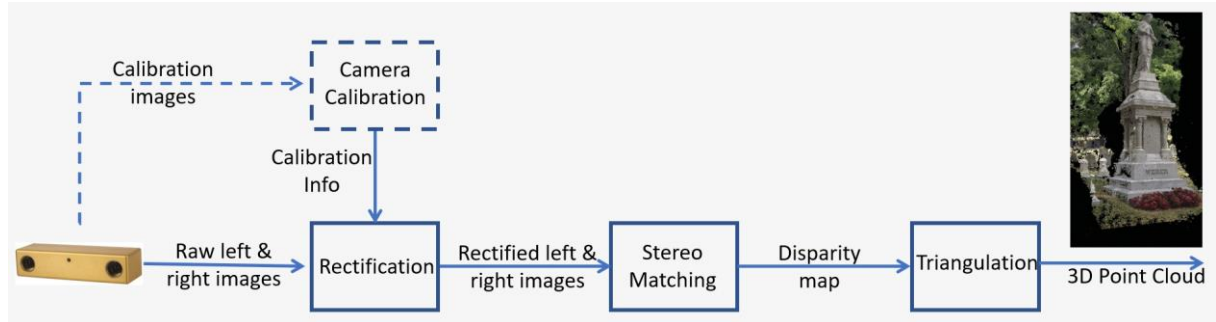


Figura 2-19 Descripción general de un sistema de visión estereoscópica [33]

2.2.4 Calibración

Para imitar la visión binocular, se puede simular empleando dos cámaras trabajando de forma sincronizada, en una configuración que conocemos como sistema de visión estereoscópica. Calibrando de manera correcta cada cámara, estas pueden proporcionar coordenadas 3D de un punto característico en concreto. Con el esquema reflejado en la Figura 2-20, se refleja cómo se observa un objeto P desde dos cámaras.

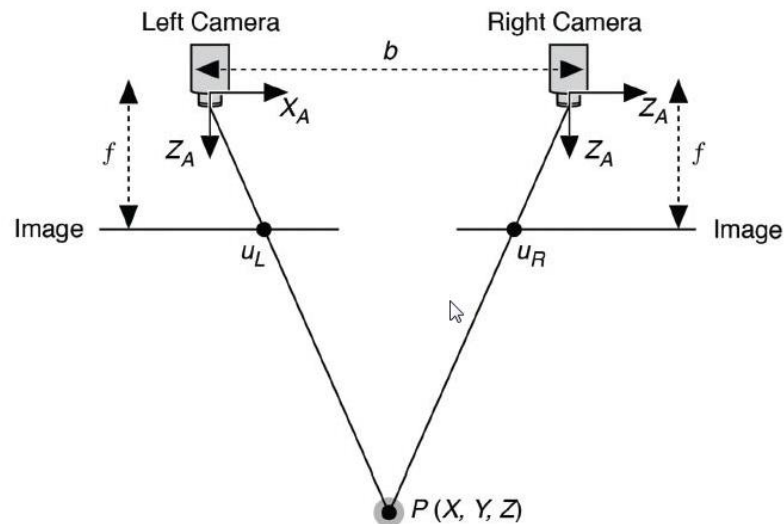


Figura 2-20 Observación de un objeto P desde dos cámaras [34]

La visión estereoscópica se realiza en cuatro etapas con la ayuda de dos cámaras [35]:

1. **Corrección de distorsión:** en esta etapa se eliminan las distorsiones radiales y tangenciales de la lente en las imágenes.
2. **Ajuste de ángulos y distancias entre cámaras en el proceso de rectificación:** el resultado muestra una alineación mediante líneas, indicando que dos imágenes de un mismo plano son coplanares, con sus líneas orientadas en la misma dirección y compartiendo la misma coordenada y.

3. **Proceso de coincidencia de puntos:** etapa que en la que se busca las correspondencias entre los puntos de ambas cámaras. Esto proporciona un mapa de disparidad, donde los valores que aparecen en dicho mapa corresponden a diferencias en la coordenada x de la imagen para un mismo punto desde las dos cámaras.
4. **Reproyección:** al disponer de una disposición geométrica de las cámaras, se puede llevar a cabo una triangulación del mapa de disparidad. Formando un mapa de profundidad, se obtiene la escena 3D deseada.

Para llevar a cabo este proceso, es necesario la calibración de las cámaras. Lo que se busca con esta calibración es la eliminación de las distorsiones de las imágenes y poder establecer una relación entre los píxeles de la imagen y las dimensiones del mundo real. Para eliminar dichas distorsiones, es necesario conocer los parámetros intrínsecos en la matriz K y los parámetros de distorsión. Dicha distorsión puede ser de dos tipos:

- **Radial:** este tipo de distorsión tiene lugar cuando los rayos de luz se curvan más en los bordes de la lente que en el centro óptico de la misma. Para simplificar, las líneas rectas aparecen ligeramente curvadas dentro de una imagen.
- **Tangencial:** en este caso de distorsión, ocurre cuando la lente de la cámara utilizada no está perfectamente alineada, es decir, paralela al plano de la imagen. Esto provoca que la imagen se extienda o se incline más, donde los objetos parecen estar más lejanos o más cercanos del lugar donde se encuentran.

La distorsión se puede capturar mediante cinco parámetros que se encuadran dentro de la matriz de coeficiente de distorsión (1), reflejando la cantidad de distorsión radial y tangencial en una imagen. Los coeficientes k_1, k_2 , y k_3 son los coeficientes de distorsión radial de la lente y los coeficientes p_1 y p_2 son los coeficientes de distorsión tangencial de la lente.

En cuanto a los parámetros intrínsecos y extrínsecos que son necesarios para la eliminación de la distorsión, hay que destacar que los intrínsecos son los específicos de la cámara, que son la distancia focal (f_x, f_y) y los centros ópticos (c_x, c_y). Sin embargo, los parámetros extrínsecos dependen de la posición de la cámara, y están contenidos en la matriz que es conocida como la matriz de la cámara (2).

$$[k_1 k_2 p_1 p_2 k_3] \quad (1)$$

$$\begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \quad (2)$$

El proceso de la calibración se lleva a cabo mediante un algoritmo, que se encarga de calcular la matriz de la cámara (2), utilizando los parámetros explicados anteriormente. Los parámetros extrínsecos son los encargados de transformar un sistema de coordenadas 3D del mundo al sistema de coordenadas 3D de la cámara. En el otro lado, los parámetros intrínsecos representan la transformación para proyectar el sistema de coordenadas 3D de la cámara en las coordenadas 2D de la imagen. En la Figura 2-21, se puede ver esquematizado el proceso de calibración.

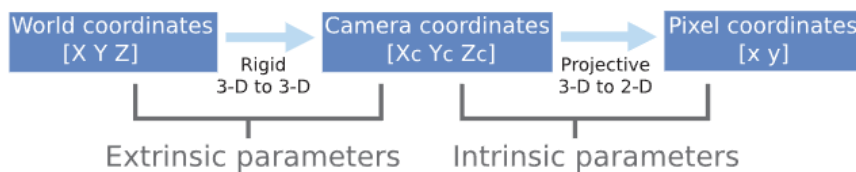


Figura 2-21 Transformación de un sistema 3D del mundo en coordenadas 2D de la imagen a través de una cámara [36]

Con el sistema calibrado, se puede usar para calcular la ubicación 3D de un punto en el espacio tridimensional. Teniendo en cuenta que el objetivo que se busca en este trabajo es realizar un seguimiento (*Tracking*) de un objeto en 3D, una vez realizada la calibración de las cámaras, se puede realizar el seguimiento gracias al uso del conocimiento de los parámetros de las cámaras y su posición relativa. En el transcurso del trabajo, se desarrollará y se estudiará todo lo aplicado en la calibración, además de realizar el seguimiento de múltiples objetos.

2.2.5 Problemas comunes de la visión estereoscópica

En la visión estereoscópica existen limitaciones como la dependencia de la calibración de la cámara o la sensibilidad a las condiciones de iluminación, las cuales juegan un papel fundamental en la calidad del sistema de visión estereoscópica. En este apartado se detallan aquellas limitaciones claves que se pueden encontrar en la visión estereoscópica:

- **Dependencia de la calibración de la cámara:** cada sistema de la visión estereoscópica, requiere en primer lugar del uso de cámaras, que previamente deben ser calibradas. Es fundamental llevar a cabo una calibración precisa para garantizar con éxito que la profundidad se calcule correctamente. En el caso de tener un error en la calibración, la percepción de la profundidad puede verse afectada. La calibración se lleva a cabo como se explica en el apartado 2.2.4.
- **Campo de visión limitado:** existe un campo de visión limitado, que está relacionado con la distancia inicial entre cámaras. Puede llegar a darse la aparición de puntos ciegos o sectores muertos en los que se muy difícil percibir objetos que no estén dentro del campo de visión del sistema.
- **Superficies sin textura y sin características:** los algoritmos de coincidencia estéreo se encargan de buscar características en las imágenes vistas por ambas cámaras. En el caso de carecer de texturas o características, es complejo de igualarlas con precisión, generando errores en la percepción de la profundidad.
- **Oclusiones:** además de ser una problemática en el seguimiento de objetos, como se ha visto en el apartado 2.1.7.2, los objetos que se encuentran ocluidos entre sí en una escena, pueden generar complejidades para la visión estereoscópica. Es muy difícil determinar la profundidad si un objeto oculta a otro.
- **Alcance y resolución limitados:** A más distancia de las cámaras, la precisión de la percepción de profundidad se ve afectada negativamente. Además, las mediciones de la profundidad disminuyen con la distancia, dificultando la percepción de detalles finos en objetos distantes.
- **Sensibilidad a las condiciones de iluminación:** las variaciones de las condiciones de iluminación, de la luz ambiental o de las sombras, juegan un papel importante en la precisión de la visión estereoscópica. Una iluminación que no sea consistente, dificulta la búsqueda de correspondencias entre las dos imágenes observadas.
- **Inexactitud en objetos transparentes o reflectantes:** superficies transparentes o reflectantes causan errores en la visión estereoscópica, ya que estos materiales no reflejan la luz de manera adecuada para la percepción de la profundidad.
- **Uso limitado en exteriores:** el uso de sistemas de visión estereoscópica en entornos exteriores puede verse afectado por luz solar intensa, o por ejemplo con cielos despejados, donde habría una ausencia de texturas.
- **Recursos computacionales:** Los algoritmos que se llevan a cabo en el sistema de visión estereoscópica, pueden llegar a ser computacionalmente intensivos en el caso de imágenes de alta resolución o en transmisiones de vídeo en tiempo real, donde en ciertos sucesos se requiere de un hardware lo suficientemente potente para un procesamiento eficiente.

- **Costo y complejidad:** el montaje de un sistema de visión estereoscópica con cámaras calibradas puede llegar a ser costoso y complejo. Los requisitos requeridos de hardware, incluidas las cámaras y equipos de calibración, es un inconveniente para ciertas aplicaciones.

Se trata, por tanto, de una serie de limitaciones que se encuentra presente en los sistemas de visión estereoscópica, donde la percepción de la profundidad o la correspondencia entre las imágenes observadas pueden verse afectadas por las limitaciones mencionadas durante este apartado.

2.2.6 Aplicaciones de la visión estereoscópica

Como se ha explicado en apartados anteriores, la visión estereoscópica es un fenómeno visual basado en la capacidad que el cerebro ejerce para percibir la profundidad al procesar las imágenes recibidas por ambos ojos. Las aplicaciones de la visión estereoscópica son diversas y abarcan diferentes campos y ámbitos. Numerosas aplicaciones en las que está presente la visión estereoscópica son [37]:

- **Robótica:** en este campo, la visión estereoscópica juega un papel importante, ya que ayuda a cumplir ciertas características como localizar y verificar. La velocidad como la precisión son muy importante para los robots modernos. Esto se garantiza por el uso de la tecnología de visión estereoscópica. Gracias a esto, los robots pueden ser utilizados en numerosas aplicaciones, como la colocación de productos mediante la identificación de su geometría y la rápida localización asociada.
- **Conducción autónoma:** varios fabricantes de vehículos conocidos están actualmente probando en sus vehículos de última generación los sistemas 3D para el uso cotidiano. La visión estereoscópica guía de forma segura a través del tráfico, y combinándolas con otros sensores, como pueden ser láser o radar (Figura 2-22), el futuro de la conducción autónoma es muy prometedor.

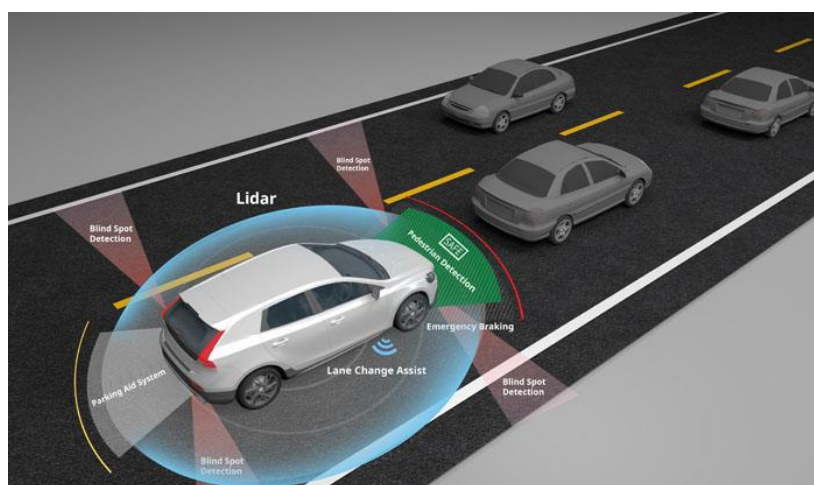


Figura 2-22 Sistemas utilizados para la conducción autónoma [38]

- **Almacenamiento:** uno de los campos donde la visión estereoscópica es muy eficaz y en la que lleva muchos años presente es en el almacenamiento. Hoy en día, numerosas empresas reciben una numerosa cantidad de productos en sus almacenes. La visión estereoscópica ayuda a mantener el orden de estos productos, gracias al uso de sistemas de gestión para mejorar la precisión en la identificación y seguimiento de productos. Además, se utiliza para la lectura y clasificación de códigos de barras o etiquetas tridimensionales, ayudando de esta manera los procesos de almacenamiento y distribución.
- **Agricultura y construcción:** esta tecnología se empieza a ver en nuevos ámbitos como la agricultura y la industria de la construcción. El mapeo de superficies cultivadas, el control del ganado, el seguimiento de los cultivos, el uso de grúas sin conductor y robots

perforadores, son diferentes actividades en las que está presente la visión estereoscópica, agilizando y mejorando la eficacia tanto de la agricultura como de la construcción.

- **Medicina:** en la medicina es de uso común la visión estereoscópica, que sirve de ayuda a los médicos. En numerosos campos de la medicina, como la cirugía asistida por computadora o el diagnóstico de imágenes, el papel que desempeña esta tecnología es fundamental. En el caso de la cirugía asistida por computadora, proporciona a los cirujanos una percepción tridimensional de los tejidos y órganos, permitiéndoles realizar procedimientos más precisos y eficaces.
- **Navegación dentro de espacios cerrados:** durante las últimas décadas, los problemas en la navegación exterior se han ido resolviendo con el auge de la tecnología de la navegación como por ejemplo el caso del tráfico motorizado. Sin embargo, en espacios cerrados sigue siendo un desafío, ya que señales como el *GPS (Global Positioning System)*, señales que se transmiten por satélites, no logran atravesar una estructura de hormigón. La visión estereoscópica puede ayudar en este campo gracias a la posibilidad de determinar la posición dentro de espacios cerrados, combinando con datos *GPS* externos. Un dispositivo que se está desarrollando es *Ezypilot* (Figura 2-23), que es un sistema de localización y percepción para robots móviles autónomos y otros vehículos autónomos de baja velocidad.



Figura 2-23 *Ezypilot*: dispositivo diseñado para la localización y percepción [39]

2.2.7 Aplicaciones específicas de la visión estereoscópica en el entorno marítimo

Dada la capacidad de poder percibir la profundidad mediante la fusión de imágenes, la visión estereoscópica es utilizada en numerosos campos. En particular, su aplicación en el entorno marítimo sigue incrementándose, desde su uso para realizar cartografías submarinas hasta sistemas de ayuda a la navegación avanzados. Gracias a esta técnica, no solo se ofrece una visión tridimensional del mundo submarino, además proporciona una compresión más detallada y precisa de los entornos marítimos. En este apartado, se estudia de manera breve, las distintas aplicaciones específicas de la visión estereoscópica en el entorno marítimo, destacando el papel decisivo a la hora de mejorar la seguridad, eficiencia y conocimiento del ecosistema marino.

- **Inspección de cascos de buques:** debido al mantenimiento periódico en los que está sometido el casco de un buque, existen diversos vehículos submarinos no tripulados con la capacidad de inspeccionar mediante la automatización. Para ello, la visión estereoscópica juega un papel muy importante debido a que gracias a las imágenes obtenidas por las cámaras binoculares proporcionan al vehículo la información necesaria como el posicionamiento, la navegación y la cartografía del casco.

- **Cartografía submarina:** el estudio del fondo marino mediante fotogrametría³ consiste en la toma de imágenes subacuáticas mediante dispositivos sumergibles. Tras esta toma de imágenes, se realiza posteriormente un proceso, generando un modelo tridimensional de la zona submarina en las que se ha realizado la toma de datos.
- **Seguimiento de fauna marina subacuática:** al simular la percepción tridimensional del fondo subacuático, permite monitorear de manera más precisa y detallada los movimientos y comportamientos de las especies marinas. Por ello, se ha convertido en una herramienta esencial para estudios biológicos y de conservación en entornos acústicos. Gracias a esto, se contribuye a preservar en un largo periodo de tiempo los ecosistemas acuáticos.
- **Detección y *tracking* de embarcaciones:** hoy en día, existen sistemas de visión estereoscópica para la detección y seguimiento de barcos, permitiendo aumentar la seguridad de la navegación. El sistema hace que los barcos se detecten mediante imágenes tomadas por dos cámaras diferentes mediante un algoritmo de detección (Figura 2-24). Los buques detectados se rastrean durante varios segundos para obtener el rumbo que llevan.

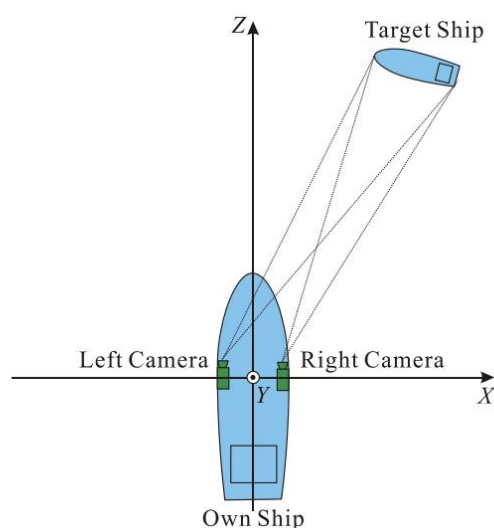


Figura 2-24 Concepto del sistema para la detección y *tracking* de buques [41]

- **Navegación de buques autónomos de superficie:** números buques autónomos utilizan dos componentes claves para llevar a cabo la visión estereoscópica. Uno de ellos es el sistema de visión *Hammerhead*, que es capaz de detectar peligros por encima de la línea de flotación. El otro sistema empleado es el de control *R4SA* (*Robust, real-time, reconfigurable, robotic system architecture*), que se encarga de aplicar comportamientos de navegación. En la Figura 2-25, se puede observar cómo funciona el sistema de visión *Hammerhead*.

³ Fotogrametría es la técnica que estudia y define de manera precisa la forma, dimensiones y posición en el espacio de un objeto cualquiera, utilizando medidas hechas sobre una o varias fotografías de ese objeto. [40]

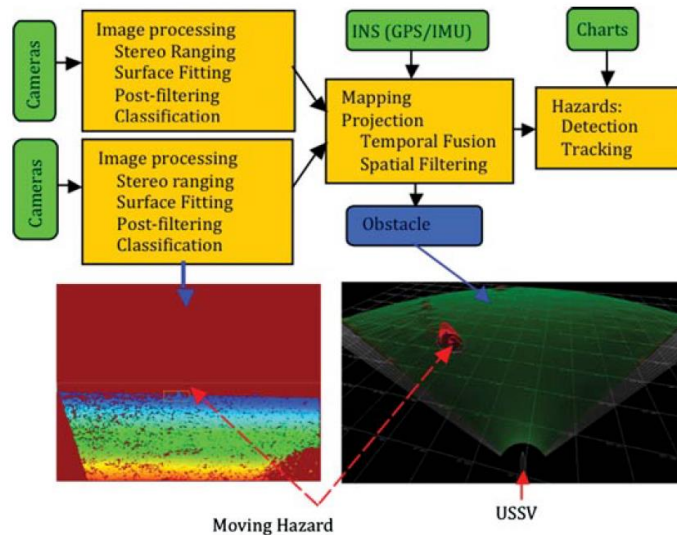


Figura 2-25 Diagrama de bloques del funcionamiento del sistema de visión *Hammerhead* [42]

2.2.8 Algoritmos de coincidencia estéreo

Existen varios algoritmos de coincidencia estéreo que se utilizan para la localización, clasificación y seguimiento de objetos en vídeos. Algunos de los algoritmos más comunes son:

- **Block matching (Emparejamiento de bloques):** *Block Matching* es un algoritmo capaz de estimar el movimiento entre dos imágenes o fotogramas de vídeo comparando bloques de píxeles. El bloque hace coincidir este bloque de píxeles entre fotogramas moviendo el bloque sobre una región de búsqueda.
- **Graph Cuts (Cortes de Grafo):** el procedimiento de corte de grafos es un método de segmentación de imágenes, empleado para resolver con eficiencia problemas de bajo nivel surgidos en la visión artificial como la suavidad de imágenes y la resolución de problemas de correspondencia estereoscópica.
- **Semi-Global Matching (SGM):** es un algoritmo muy extendido para la correspondencia de imágenes que se utiliza en varias aplicaciones, como por ejemplo aplicaciones en tiempo real. Es conocido por su capacidad para manejar problemas de oclusión y proporcionar resultados precisos. El proceso de este algoritmo se refleja en la Figura 2-26.

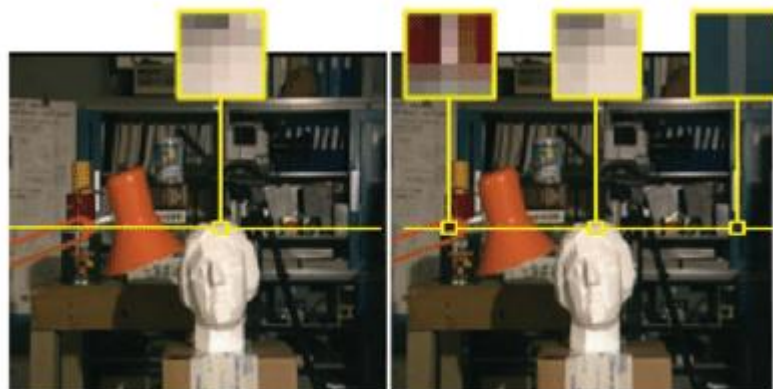


Figura 2-26 *Semi-Global Matching SGM* [43]

- **SURF (Speeded Up Robust Features):** el método *SURF* es un algoritmo rápido y robusto para la representación y comparación de imágenes. El principal interés de este método reside

en la rapidez de sus cálculos mediante filtros de caja, permitiendo aplicaciones en tiempo real como el seguimiento y el reconocimiento de objetos.

- **FAST (Features from Accelerated Segment Test):** es un algoritmo de detección de esquinas, que se utiliza para extraer puntos de características, donde posteriormente se utiliza para el rastreo y el mapeo de objetos en muchas tareas de visión por ordenador. La ventaja de este método es su eficiencia computacional.
- **PSMNet (Pyramid Stereo Matching Network):** consiste en una red piramidal de coincidencia estéreo que consta de dos módulos principales: agrupación piramidal espacial y CNN 3D. El módulo de agrupación de la pirámide espacial aprovecha la capacidad de la información agregando diferentes escalas y ubicaciones para formar un volumen costes. En la Figura 2-27, se puede apreciar el algoritmo.

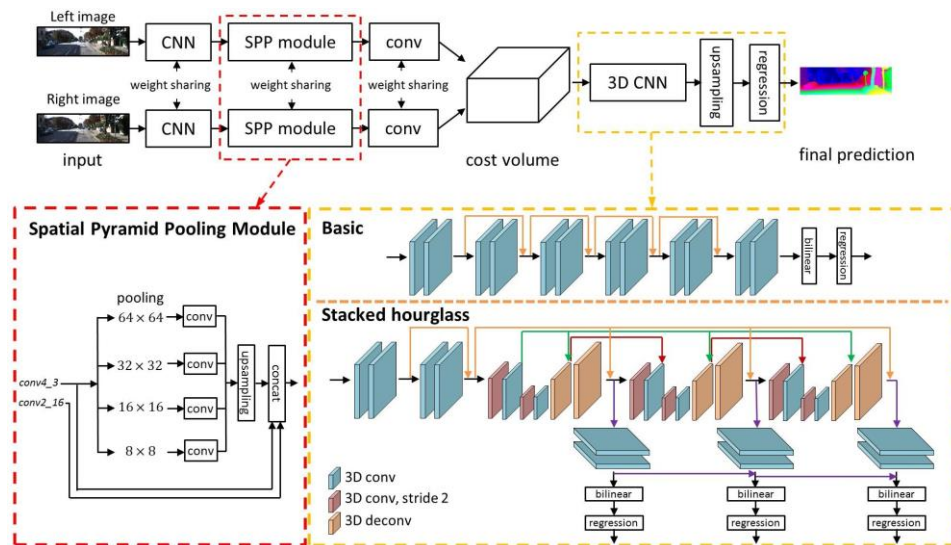


Figura 2-27 Algoritmo PSMNet [44]

Estos son diversos algoritmos de la coincidencia estéreo. La elección del mismo depende de diversos factores, como la complejidad de la escena, los recursos computacionales disponibles y los requisitos de precisión.

3 DESARROLLO DEL TFG

En el presente capítulo se presentan los pasos que se han llevado a cabo para la realización de este trabajo. Se introducen los diferentes dispositivos que se han utilizado para la ejecución del mismo, así como las herramientas de *software* empleadas durante el transcurso del proyecto. Además, se expone la justificación de los algoritmos empleados para la detección de objetos y el seguimiento de objetos en un vídeo. En este caso, se detalla, con más información, la justificación de su elección para el modelo de detección de objetos a entrenar. Por último, se detallan los distintos procesos de calibración, etiquetado y entrenamiento a los que ha sido sometido nuestro modelo, a partir del *dataset*⁴ que se ha elaborado en la confección del trabajo.

El objetivo principal es, a partir del *dataset* elaborado, resolver el problema de la calibración de los dos dispositivos de grabación, así como el cálculo de distancia y la trayectoria seguida por un objeto en el agua o de un objeto en un entorno donde se pueda seguir su trayectoria y sus diferentes posiciones con respecto al centro de un eje de coordenadas establecido.

3.1 Dispositivo de grabación portátil empleado

Para el cumplimiento del presente trabajo, se ha hecho uso de dos cámaras de grabación portátil. Las cámaras empleadas son de la marca *ELP*, concretamente el modelo de 1080 *HD* (*High Definition*) 5-50 mm (Figura 3-1). Este dispositivo de entrada de vídeo es capaz de conectarse a un ordenador y realizar capturas de imágenes o vídeos. Las características de la cámara son las siguientes:

- Resolución de 8 megapíxeles. Esto quiere decir que captura imágenes con una resolución de 8 millones de píxeles.
- Lente de enfoque múltiple. Gracias a esto, la cámara tiene una lente que puede ajustarse a lograr un enfoque nítido en diferentes distancias. El enfoque múltiple permite adaptarse a diversas condiciones de grabación.
- *HD* 5-50 mm: el rango de distancia focal variable de la cámara va desde 5 mm hasta 50 mm. La distancia focal influye en el ángulo de visión y la capacidad de acercamiento.

En general, las características de esta cámara es la adecuada para aplicaciones que requieren alta resolución y flexibilidad en el enfoque, como la captura de imágenes detalladas a diferentes distancias. Es por esto que se considera que esta cámara es idónea para el desarrollo de la parte práctica de este TFG.

⁴ *Dataset*: colección de datos organizados y estructurados que se utilizan para realizar análisis, investigaciones o tareas relacionadas con la informática y la ciencia de datos.



Figura 3-1 Cámara digital *ELP 1080P HD 5-50 mm* empleada para la obtención de imágenes y vídeos [45]

3.2 Herramientas de desarrollo empleadas

Para el desarrollo del modelo de aprendizaje, se han empleado diversas herramientas de *software* que cumplen con los siguientes factores: funcionalidad, costo, facilidad, seguridad, integración y soporte técnico. A continuación, se realiza una breve introducción acerca de las herramientas seleccionadas y de las prestaciones que ofrecen cada una de ellas.

3.2.1 Anaconda (Miniconda)

Anaconda es una plataforma de distribución y gestión de paquetes de código abierto, con el fin de simplificar la implementación y administración de entornos de desarrollo en lenguajes como *Python* y *R*, además de proporcionar herramientas para el análisis de datos, ciencia de datos, la inteligencia artificial y la programación en general.

Para el desarrollo de este TFG, se ha utilizado *Miniconda*, que es una versión ligera y minimalista de la distribución de *Anaconda*. A diferencia de *Anaconda*, *Miniconda* proporciona solo lo esencial, que son el gestor de paquetes *Conda* y su sistema base.

3.2.2 FFmpeg

FFmpeg es una herramienta de *software* y una biblioteca multimedia utilizada para procesar, grabar, transmitir y convertir archivos de audio y vídeo. Es muy versátil y puede manejar una extensa variedad de formatos de archivo multimedia, convirtiéndola de esta manera en una herramienta muy útil para el trabajo con contenido multimedia. En el caso de este trabajo, se ha utilizado este *software* para la verificación de la sincronización de la grabación de dos vídeos a la vez de distintas fuentes. *FFmpeg* puede capturar y grabar vídeos y audio desde distintas cámaras, encargándose de mantener las grabaciones sincronizadas entre ellas al máximo número de *frames* por segundo que el sistema puede mantener, siendo muy útil para la realización del *dataset* elaborado en el presente trabajo.

3.2.3 Google My Maps

Google My Maps es una herramienta que permite a los usuarios crear de manera fácil mapas sobre los mapas bases de *Google Maps*. Con este *software*, se pueden añadir marcadores, líneas, formas y capas personalizadas a un mapa. Para este TFG va a servir de ayuda a la hora de tomar diferentes puntos notables de un sitio geográfico, además de importar puntos *GPS*, permitiendo el análisis sobre estas posiciones en el mapa.

3.2.4 Label Studio

Label Studio es una plataforma de código abierto para la creación y gestión de etiquetas en datos utilizados para entrenar modelos de aprendizaje automático. Tiene diversas características como anotación de texto, anotación de audio y anotación de series de tiempo, pero en este trabajo nos centramos en el etiquetado de imágenes y vídeos. Para el etiquetado de imágenes, proporciona plantillas y herramientas para realizar anotaciones en imágenes. Además, permite exportar el *dataset* etiquetado de las imágenes en distintos formatos, dentro de los cuales se encuentra *YOLO*, lo que es de gran interés para el trabajo, ya que facilita el entrenamiento del modelo.

3.2.5 VSCodium

Es un editor de código fuente similar a *Visual Studio Code*, que fue desarrollado por *Microsoft*. A diferencia de *VSCodium*, *VSCodium* es una versión de código abierto sin pertenecer a *Microsoft*, por lo que puede ser instalado en cualquier sistema operativo. Este editor recoge diversas características importantes para los códigos empleados en el proyecto, con el fin de ejecutar los *scripts* anexados al final del trabajo, entrenar, aplicar y analizar el modelo de aprendizaje a desarrollar.

3.2.6 Ultralytics HUB

Ultralytics HUB es una organización que se especializa en el desarrollo de *software* y herramienta para la investigación y aplicación de modelos de *Deep Learning*. Desde principios del año 2023, permite el entrenamiento de modelos de aprendizaje profundo basados en el algoritmo *YOLOv8*. Esta plataforma permite el desarrollo de modelos de aprendizaje, con la capacidad de entrenar un modelo e identificar patrones a partir de los datos introducidos. Cabe destacar, que esta plataforma incluye gráficas en tiempo real, a medida que el modelo se entrena, siendo importante para el usuario, ya que puede tomar decisiones, optimizar sus modelos y obtener resultados superiores. El proceso que realiza *Ultralytics HUB*, para el entreno de un modelo, es el que se ve en la Figura 3-2:

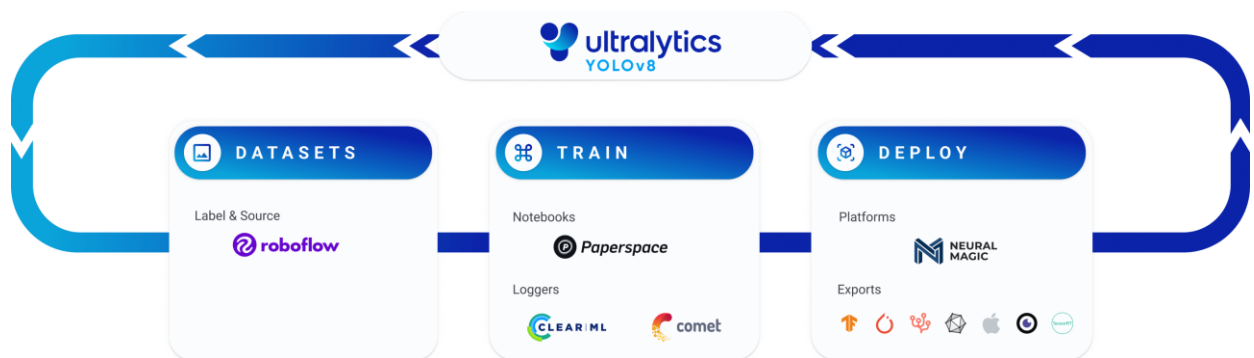


Figura 3-2 Proceso realizado en el desarrollo de un modelo en *Ultralytics HUB* [46]

3.2.7 Google Colab

Google Colab o *Google Colaboratory*, es un servicio de acceso gratuito en la nube proporcionado por *Google*, permitiendo ejecutar y escribir código en *Python* directamente en un entorno basado en la nube. *Colab* utiliza los cuadernos de Jupyter, que permite combinar celdas de código, texto, imágenes y gráficos en un solo documento interactivo. Se caracteriza por el acceso gratuito a recursos de hardware, como *GPU* (*Graphics Processing Unit*), siendo muy útil para soportar tareas con gran carga de computación. Además, gracias a que tiene muchas bibliotecas de *Python* preinstaladas, como *TensorFlow* o *PyTorch*, se puede trabajar con los modelos desarrollados en *Ultralytics HUB*.

3.2.8 Entorno de trabajo VIDEO-IA-CUD

Para poder llevar a cabo los diferentes objetivos propuestos para este proyecto, se ha hecho uso del empleo de una serie de *scripts* en código *Python*, que se encargan de realizar diferentes tareas que son

necesarias durante el desarrollo, tales como la captura sincronizada de vídeos, la extracción de imágenes para la creación del *dataset*, el entrenamiento del modelo basado en *YOLO*, etc. Esta colección de *scripts* se encuadra en el entorno *VIDEO-IA-CUD*, que se está desarrollando en el CUD-ENM, para dar soporte a proyectos ambientados en el campo de la aplicación de la visión artificial al control del tráfico marítimo. Entre los diversos *scripts* empleados, se encuentran los siguientes:

- *test_camera*: se emplea antes de realizar cualquier toma de vídeos, comprobando si la cámara asignada para la tarea tiene los parámetros correctos y la entrada *USB* es la indicada.
- *video_capture_tool*: una vez que el funcionamiento de las cámaras es el correcto, este *script* da el nombre deseado a los vídeos, eligiendo el directorio donde se van a guardar, así como establecer la duración de la grabación, los fotogramas por segundo (*FPS*, *Frames per second*) y la resolución en píxeles.
- *video_extract_tool*: con un vídeo ya capturado y almacenado en un fichero, extrae los *frames* que componen el vídeo, y con la aplicación de un modelo entrenado (basado en *YOLO*), genera las etiquetas de los objetos.
- *model_train_tool*: se emplea para entrenar un modelo a partir de un *dataset* de imágenes y las etiquetas asignadas en el *Label Studio*.
- *video_track_tool*: este *script* es empleado para realizar la comprobación de que el modelo entrenado funciona de manera correcta, siendo capaz de identificar a los objetos dentro del laboratorio o las embarcaciones náuticas de un vídeo seleccionado, mostrando las cajas detectadas en los vídeos obtenidos.
- *video_track_3d_tool*: este *script* se emplea para la visualización y validación de los resultados del interfaz, mostrando los datos de posicionamiento y cinemática de los objetos o embarcaciones en 3 dimensiones a partir de los vídeos obtenidos.
- *ais_capture_tool*: este *script* se trata de una utilidad creada para poder leer los datos del receptor *AIS* (*Automatic Identification System*). A partir de los datos obtenidos del receptor *AIS*, muestra por pantalla todos los mensajes que se van recibiendo en tiempo real.

3.3 Problemas a resolver durante el desarrollo del TFG

Una vez introducido los dispositivos utilizados y las diferentes herramientas de *software* empleadas para la ejecución, en este apartado se presenta una breve introducción acerca de los distintos problemas a resolver durante el desarrollo del trabajo. Cabe destacar que, inicialmente, el entorno utilizado para el progreso del TFG ha sido el Laboratorio de Robótica, utilizando objetos simples como una pelota de *ping pong* y una cubeta de agua con diferentes objetos flotantes, como barcos de juguete y bloques de poliestireno. Además, para la calibración de las dos cámaras, se ha empleado una fotografía de un tablero de ajedrez, ya que este elemento estándar para realizar dicha calibración en distancias cortas, aunque debe tenerse en cuenta que en un espacio exterior de grandes dimensiones no se podría realizar la calibración de manera adecuada mediante este procedimiento. A medida que se va avanzando en el desarrollo de la realización de tareas, se van dando más detalles relevantes sobre estos problemas. De esta manera, los principales problemas a resolver en este trabajo son:

- **Detección y clasificación de objetos en imágenes:** durante esta fase se busca hacer frente al problema de detectar objetos en imágenes. Para ello, el modelo es entrenado previamente para saber qué tipo de objetos tiene que detectar y clasificar. En particular, el modelo a entrenar consistirá en detectar objetos simples como una pelota de *ping pong*, barcos de juguete y bloques de poliestireno.
- **Seguimiento de objetos en vídeo:** una vez que los objetos han sido detectados en las imágenes del *dataset*, se puede realizar un *tracking* a cualquier vídeo donde aparezca el objeto. Para llevar a cabo este tipo de entrenamiento, se hace uso del algoritmo *ByteTrack*, ya explicado en el apartado 2.1.7.4. Este algoritmo atribuye una identidad al objeto que detecta en el vídeo en cada *frame* o fragmento del vídeo. Gracias a esto, el modelo aprende a mantener el seguimiento del objeto durante la grabación.

- **Calibración de los dispositivos de grabación:** para resolver este problema se va a hacer uso del procedimiento basado en un tablero de ajedrez, pero además se verá otro proceso basado en una red neuronal que se encarga de aprender el mapa que resuelve el problema de regresión entre el espacio de coordenadas de las dos cámaras (4D) y la posición tridimensional del objeto (3D). Lo que se busca con este proceso es ajustar y optimizar la configuración de las cámaras, con el objetivo de garantizar una reproducción precisa de lo que se grabe o de las imágenes capturadas mediante las cámaras. Este proceso se utiliza en sistemas de visión por computadora, realidad virtual o en aplicaciones donde se necesite una gran precisión en la captura de imágenes.

3.4 Sistema de identificación de un objeto en una posición fija

Para el desarrollo de este sistema de identificación de un objeto se ha empleado de un modelo de aprendizaje automático simple, con el fin de que el modelo sea entrenado para detectar y clasificar diferentes objetos. Esta fase inicial, al igual que durante el transcurso de las diferentes fases que componen este trabajo, basa su funcionamiento en el algoritmo para detección de objetos *YOLO*, que ya ha sido estudiado anteriormente. El entorno utilizado para realizar esta primera fase ha sido el laboratorio de Robótica, donde se ha desarrollado un sistema con dos dispositivos de grabación, de forma que se establece un sistema de coordenadas de referencia, sirviendo de ayuda para la toma de imágenes del objeto desde una posición fija con el objetivo de establecer su posición. Es necesario conocer la posición exacta del objeto, para crear un *dataset* con los datos de entrada y de salida que van a ser empleados para que el modelo sea entrenado hasta que sepa resolver la tarea planteada. En la Figura 3-3 se puede apreciar el sistema de coordenadas empleado.

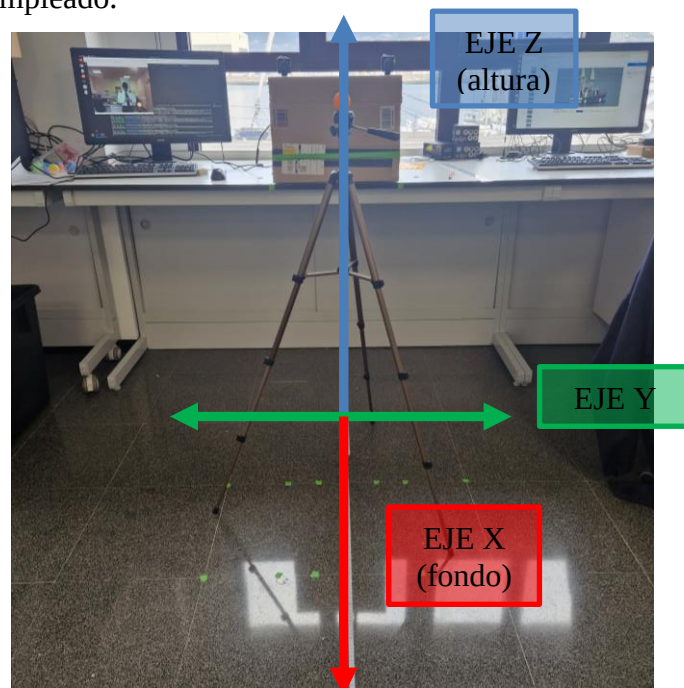


Figura 3-3 Sistema de coordenadas empleado para la toma de imágenes (autoría propia)

3.4.1 Obtención de imágenes

Una vez seleccionado el sistema de coordenadas para la toma de imágenes del objeto desde una posición fija, se empieza con la obtención de las imágenes empleando un *script* que permite tomar capturas instantáneas. El proceso es sencillo y fácil de llevar a cabo. Gracias al uso de las dos cámaras, se ejecuta el código que permita realizar una captura de los dos dispositivos al mismo tiempo, proporcionando dos imágenes cada vez que se ejecute dicho código (Figura 3-4). Es necesario

comprobar que se cumplen ciertos criterios como la tipología, la perspectiva y la calidad de las imágenes tomadas.

Con respecto a la tipología, los datos empleados son fotografías, ya que el uso de vídeos es de mayor complejidad a la hora del procesamiento y etiquetado. En cuanto a la perspectiva, es importante tener una posición fija y constante durante todas las capturas de las imágenes. Gracias a esto, el posterior procesamiento de reconocimiento e identificación de patrones es más efectivo, ya que el modelo es entrenado desde una misma posición sin ser alternado en ningún momento. Y, por último, y lo más importante, es la calidad de las imágenes. Una calidad técnica de las fotografías sirve de ayuda para el aprendizaje del sistema. Es necesario que las cámaras de grabación tengan el enfoque adecuado para localizar e identificar el objeto de manera efectiva, así como tener las mejores condiciones de iluminación y claridad que no puedan afectar al proceso.

Debido a que al entrenamiento de este modelo no es muy complejo, no es necesario añadir más imágenes al sistema porque es capaz de reconocer e identificar de manera efectiva el objeto a detectar y clasificar (una pelota de *ping pong*).



Figura 3-4 Ejemplo de una toma de imágenes simultánea por las dos cámaras con el objeto en la misma posición (autoría propia)

3.4.2 Etiquetado de imágenes

Una vez que se obtiene el número de imágenes deseadas, es necesario etiquetar al objeto en las diferentes fotografías. Para ello se hace uso de la herramienta *Label Studio* ejecutada desde *Miniconda*.

El objetivo del etiquetado es identificar, clasificar e identificar donde está el objeto. *Label Studio* es un software que permite insertar recuadros con diferente tamaño, orientación y de color variable sobre los objetos a procesar. Al programa hay que insertarle unas etiquetas (*labels*), que se establecen en función de los distintos objetos que aparecen en las fotografías analizadas. En este caso, el caso es muy sencillo ya que se cuenta con un solo objeto, donde en las etiquetas se ha denominado como *pong*.

Para insertar la etiqueta es necesario el uso de un código que se implementa en la herramienta. A continuación, se muestra el código empleado para el etiquetado de imágenes en este sistema:

```
<view>
  <Image name="image" value="$image"/>
  <Header value="RectangleLabels"/>
  <RectangleLabels name="label" toName="image">
    <Label value="pong" background="rgba(218, 1, 238, 1)"/>
  </RectangleLabels>
</view>
```

Una vez que está definida la etiqueta a utilizar, se etiquetan todas las imágenes por medio de una aplicación web. *Label Studio* permite sincronizar el directorio donde se encuentran las imágenes, por lo que en caso de que se quieran añadir más imágenes para el proceso de etiquetado, con realizar la sincronización es suficiente, sin necesidad de tener que subir de nuevo todas las imágenes. Con todo esto, se lleva a cabo el etiquetado de los objetos presentes en las imágenes, destacando que es necesario un ajuste preciso a la hora de insertar el rectángulo del etiquetado, ya que sirve de ayuda para que el modelo en el posterior entrenamiento no tenga dudas ni confusiones de que ese objeto es el que se ha etiquetado. A continuación, en la Figura 3-5, se muestra un ejemplo del etiquetado de una de las imágenes:



Figura 3-5 Imágenes etiquetadas en *Label Studio* (autoría propia)

3.4.3 Entrenamiento

Una vez terminado el proceso de etiquetado, se dispone del *dataset* proporcionado por *Label Studio*, de forma que pueda ser empleado para generar el entrenamiento del modelo mediante la ejecución del *script* de entrenamiento de modelos. La versión utilizada para realizar dicho entrenamiento es *YOLOv8n*, que es el modelo más pequeño de este algoritmo y el único que es posible entrenar empleando los ordenadores disponibles en el laboratorio para el desarrollo de los TFG. Al ejecutar el anterior *script*, se automatiza el reparto de nuestro *dataset* de imágenes en tres grupos. Cada grupo tiene un propósito distinto, y cumple con funciones particulares en el proceso de aprendizaje.

El primer grupo que proporciona el *script* es el conocido como *train*. En este grupo se observarán la mayoría de las fotografías, ya que el sistema se encarga de coger este grupo para realizar el entrenamiento. El siguiente grupo se denomina *validation*. En este caso, el sistema usa un grupo de imágenes para validar los resultados. Y, por último, el modelo atribuye un menor porcentaje de imágenes a la fase de *test*. Esta fase nunca se emplea y sirve para poner a prueba al modelo, generando una serie de resultados que permitirán valorar la fiabilidad del comportamiento del modelo respecto a datos de entrada con los que no ha sido entrenado específicamente.

Una vez que el modelo ha sido entrenado, se van generando diferentes gráficas que muestran la predicción y fiabilidad de que el objeto etiquetado es el objeto a identificar y clasificar. También, muestra las imágenes con el etiquetado de antes, pero mostrando en este caso la probabilidad de que el objeto sea una pelota de *ping pong*, siendo 0 lo menos probable y 1 lo más probable. En la Figura 3-6, se puede

apreciar como en este caso el sistema ha identificado de manera precisa el objeto gracias al etiquetado que se realizó anteriormente.



Figura 3-6 Probabilidad del objeto identificado sea igual que el etiquetado (autoría propia)

De las distintas gráficas obtenidas durante el proceso de entrenamiento, se puede destacar las siguientes que aparecen en la Figura 3-7:

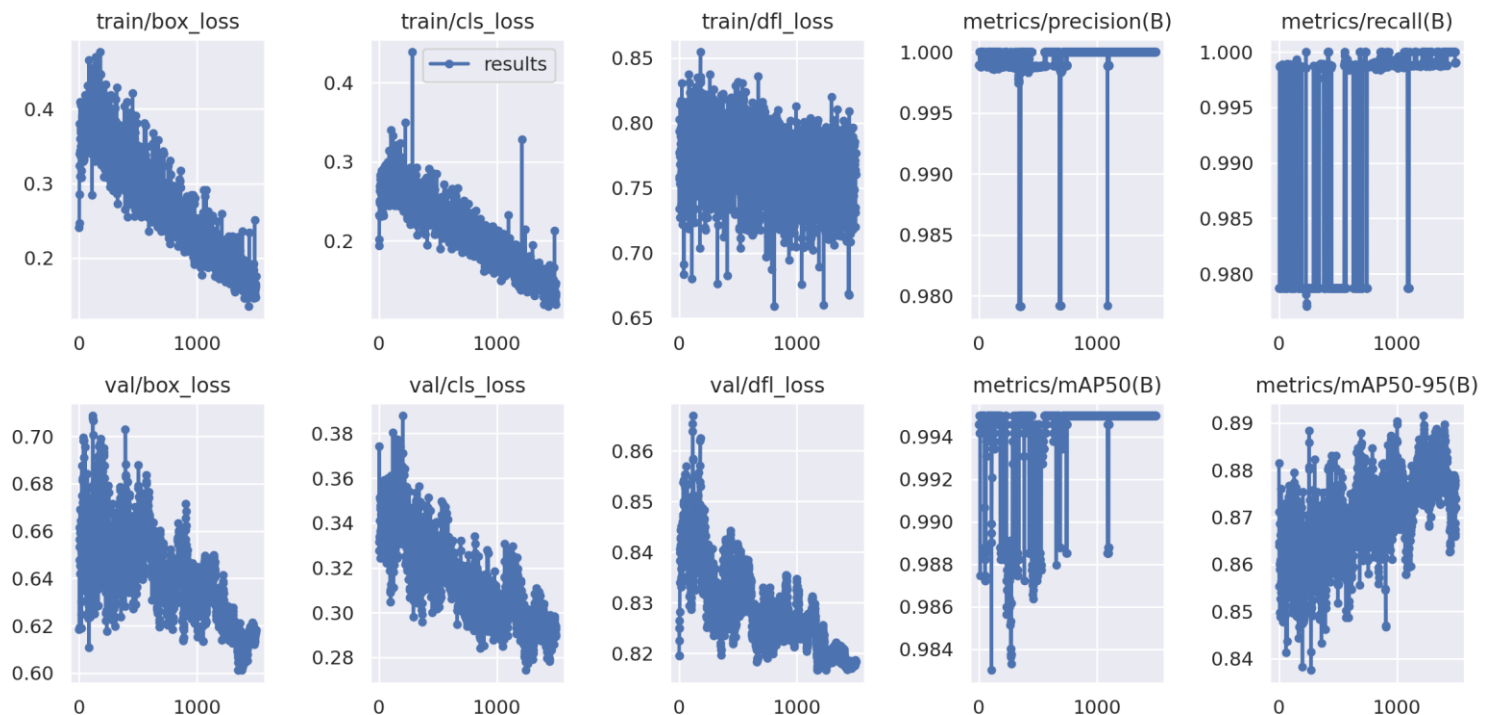


Figura 3-7 Resultados obtenidos tras el entrenamiento (autoría propia)

En esta imagen se puede apreciar como durante el entrenamiento el sistema ha cogido las imágenes de *train* y validación (*val*), dejando las de test reservadas. Empezando por las primeras gráficas que

presenta el parámetro de *box loss* (pérdida de cajas), se puede destacar que durante los primeros *epoch*⁵ o época, su valor aumenta, pero a medida que transcurre el entrenamiento va disminuyendo considerablemente, conformando el entrenamiento del modelo.

Otro parámetro a destacar en el análisis de las gráficas presentes es el *class loss* (pérdida de clase o tipo), que permite analizar la capacidad del modelo para clasificar el objeto detectado según el etiquetado que se ha generado en el *dataset* de entrenamiento. Se aprecia que, excepto en diversos *epochs*, el modelo ha sido capaz de reconocer el objeto. Hay que destacar que, al tratarse de un solo objeto, es más fácil que si se tienen más objetos presentes.

Para terminar con el análisis de las gráficas generadas, aparecen las métricas del algoritmo *YOLO*, que ya han sido explicadas en el apartado 2.1.6.2. En la gráfica de *precision*, el modelo se ha encargado de cuantificar todas las ventanas que son auténticas con respecto a las etiquetadas. Se aprecia que el valor es bastante alto, por lo que se puede determinar que el entrenamiento ha sido fiable en cuanto a precisión. Otro de los parámetros a destacar es el de *recall*, que se encarga de medir el porcentaje de verdaderos positivos en el conjunto de datos. Su cálculo se realiza dividiendo el número de verdaderos positivos por la suma de verdaderos positivos y falsos negativos. El valor obtenido es similar al *precision*, por lo que se destaca que el número de verdaderos positivos es bastante alto.

Por último, tenemos la precisión media (*mAP*). Este parámetro mide el rendimiento de los sistemas de localización de objetos. En cuanto al análisis, destacar que el buen desempeño del modelo debido a que su valor es cercano a la unidad, en ningún momento del entrenamiento ha bajado considerablemente. En el caso de *mAP* acotado entre 0,50 y 0,95 (*mAP50-95*), es menor que en el anterior caso. Esto se debe a que este parámetro tiene en cuenta el *IoU* (*Intersection over Union*) desde 0,5 a 0,95, indicando la precisión media del modelo. En el caso de *mAP*, solo lo tiene en cuenta para un *IoU* de 0,5.

3.4.4 Seguimiento (*Tracking*) del objeto en un vídeo con posiciones en los distintos ejes

Una vez entrenado el modelo con el objeto en diferentes posiciones, pero en una posición fija, es posible hacer un *tracking* de ese objeto en cualquier vídeo, y comprobar la probabilidad del modelo es cercana a la unidad. Si ocurre, significa que el modelo ha sido entrenado correctamente, identificando a la pelota de *ping pong* de manera eficaz y clasificándola como es.

Para realizar el *tracking*, es necesario ejecutar el *script video_track_tool*. Este código permite ver durante el vídeo el seguimiento que realiza el sistema al objeto localizado, proporcionando en todo momento su clasificación con respecto al etiquetado que se realizó anteriormente, así como la probabilidad de que ese sea el objeto con certeza. A continuación, en la Figura 3-8, se muestra dos imágenes de diferentes *frames*, apreciándose el *tracking*.

⁵ *Epoch*: número total de iteraciones de todos los datos de entrenamiento en un ciclo para entrenar el modelo de aprendizaje automático. [47]



Figura 3-8 Imágenes extraídas del vídeo de *tracking* (autoría propia)

3.5 Sistema de identificación de objetos dentro de un recipiente con agua

En este apartado, el sistema cambia con respecto a lo anterior, pero el proceso llevado a cabo es similar. En este caso utilizamos un recipiente de agua, con objetos dentro como son pelotas de *ping pong*, bloques de poliestireno y barcos de juguete. El objetivo de este sistema es entrenar un modelo en un entorno pequeño, para poder aplicarlo de cara al futuro en un entorno tridimensional más grande.

El número de imágenes del modelo a entrenar es de 96 imágenes. El proceso llevado a cabo es parecido, como se comentó anteriormente, pero con las diferencias de que en este caso las imágenes son extraídas a través de vídeos. Con respecto al etiquetado de las fotografías, se realiza de la misma manera, en el software *Label Studio*. Y, por último, el entrenamiento se realiza en otra plataforma, en *Ultralytics*, que proporciona gráficas más detalladas pero similares al caso anterior.

3.5.1 Obtención de imágenes

Para la obtención de imágenes, se empieza grabando varios vídeos. Gracias al *script video_extract_tool*, se crea un *dataset* que contiene 96 de imágenes. Uno de los problemas presentes a la hora de realizar la grabación, es la iluminación y los reflejos de la luz en el agua, que puede llevar a la oclusión de los distintos objetos que están flotando en el agua al ser confundidos con los reflejos.

Otro de los problemas que se han presentado durante el proceso de elaboración de la solución, es el problema de la sincronización entre ambas cámaras. A la hora de la grabación, se ha detectado que ambas cámaras no realizaban las grabaciones de forma sincronizada en todas las tomas obtenidas, lo que supone un problema a la hora de la integrar sus datos para generar la visión estereoscópica. Para solucionar este problema en la aplicación final realizada en el entorno de la ENM (Escuela Naval Militar) y que se presentará más adelante, se ha realizado un *script* (Anexo IV: Script de OCR) que, mediante el uso del software de software *FFmpeg* para conseguir grabar los dos vídeos de forma sincronizada. Para validar esta sincronización se añade la hora en los vídeos que se graban, de forma que posteriormente se emplea la librería *pytesseract* (paquete de *Python* que funciona como motor de OCR de código abierto, con la capacidad de poder reconocer texto en diversas imágenes y convertirlo en texto que se puede apreciar en una fotografía o en una imagen) para realizar la validación de esta sincronización. En la Figura 3-9, se puede apreciar la fecha y hora a la que se ha producido la grabación.



Figura 3-9 Obtención de imagen con fecha y hora exacta (autoría propia)

3.5.2 Etiquetado de imágenes

Una vez extraídas las imágenes del modelo, se procede al etiquetado de las imágenes. A diferencia al etiquetado anterior, en este proyecto se etiquetan 3 tipos diferentes de objetos. Esto hace de que el código que se utiliza para el etiquetado cambie de la siguiente manera:

```
<View>
  <Image name="image" value="$image"/>
  <Header value="RectangleLabels"/>
  <RectangleLabels name="label" toName="image">
    <Label value="pong" background="rgba (218, 1, 238, 1)"/>
    <Label value="block" background="#ff7800"/>
    <Label value="ship" background="#1a5fb4"/>
  </RectangleLabels>
```

Posteriormente a modificar este parámetro de configuración en *Label Studio*, el proceso de etiquetado es igual que antes, con la diferencia de que, en este caso, no aparece solo un objeto como se aprecia en la Figura 3-10. Hay 3 tipos de objetos que se han utilizado para realizar su identificación en la grabación de los vídeos.



Figura 3-10 Etiquetado de imagen en *Label Studio* (autoría propia)

3.5.3 Entrenamiento

Con las imágenes ya etiquetadas, se exporta el archivo de *Label Studio* y se procede a entrenar al sistema en *Ultralytics*. Como se expuso en la prueba de sistema de identificación de un objeto desde una posición fija, el propio software se encarga de hacer un reparto de los datos de entrada, dividiéndolos en *train*, *validation* y *test*. La parte de *train* es la que se lleva el mayor porcentaje de las 96 imágenes que componen el modelo, ya que en esta fase es donde el modelo entrena para la identificación y clasificación de los objetos. El reparto proporcionado por *Ultralytics* para este modelo es el siguiente (Figura 3-11):

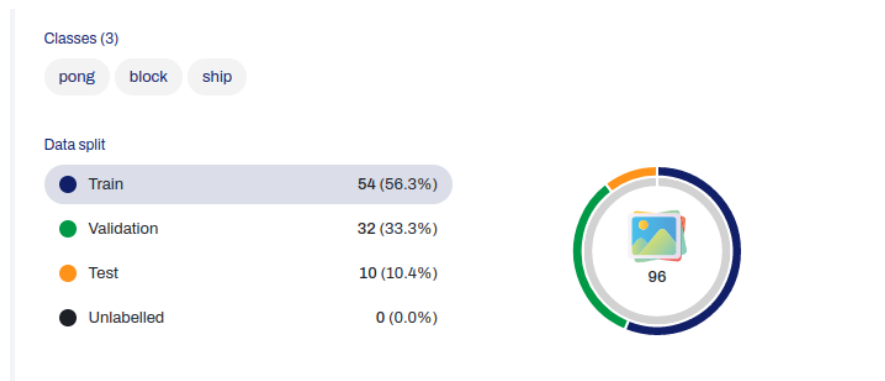


Figura 3-11 Reparto de las imágenes que componen el *dataset* del modelo a entrenar (captura generada por *Ultralytics* de autoría propia)

Con respecto a las gráficas, la primera que se proporciona en este entrenamiento relación los valores de *recall* y precisión, con los resultados del *mAP*. Analizando la Figura 3-12, se observa que el *mAP* obtenido para un *IoU* de 0,5 es considerablemente bueno, pero si lo acotamos entre 0,5 y 0,95, el valor disminuye bastante. En cuanto a los valores de *recall* y *precision* obtenidos desde el *epoch* 85 determina que el modelo es fiable, tanto en clasificación como en detección. El modelo fue propuesto para 1000 *epochs*, pero al sistema le ha bastado con 700 *epochs* aproximadamente para terminar el entrenamiento.

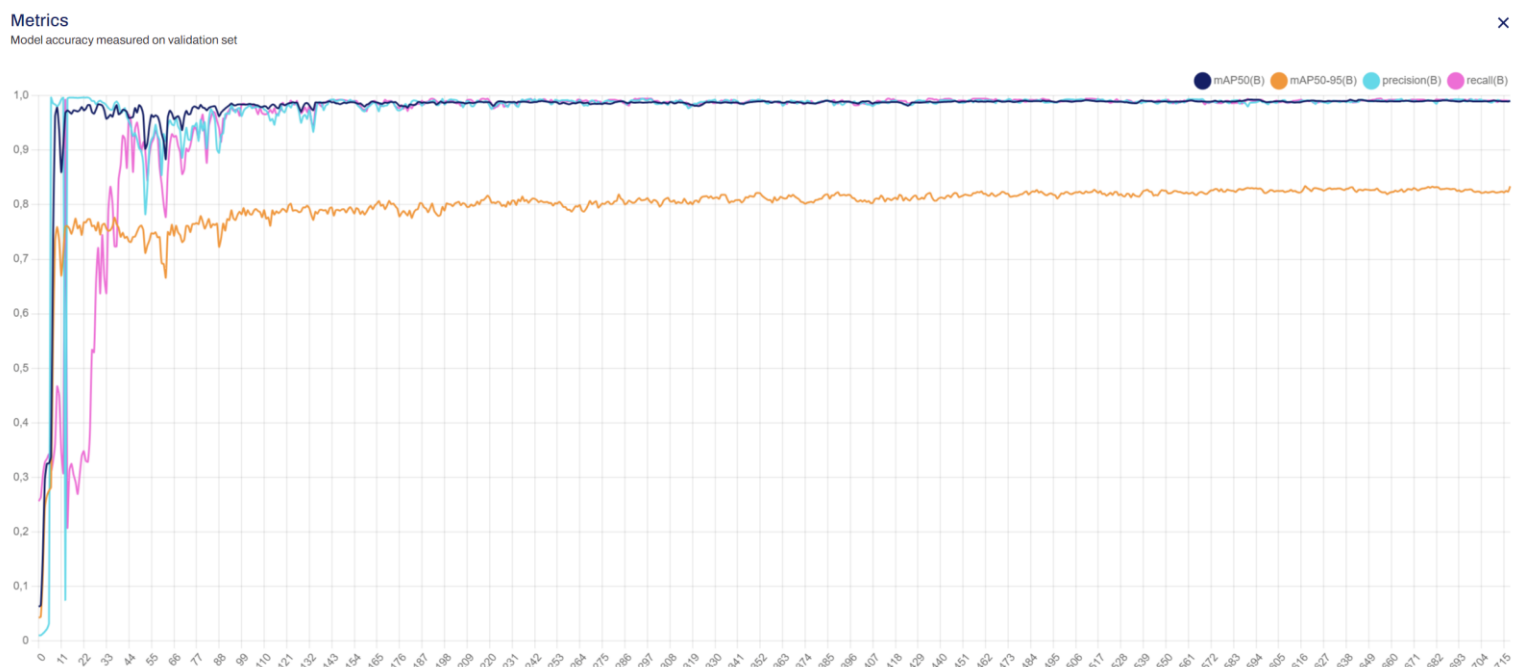


Figura 3-12 Métrica del modelo (gráfica generada por *Ultralytics* de autoría propia)

En la Figura 3-13, se muestra el parámetro de *box loss*, donde se puede observar la disparidad del *box loss* dependiendo si se utilizan las imágenes para la fase de *train* o la fase de *validation*. Observamos que en el caso de *validation*, la gráfica tiende a quedarse en 0,8 tras el entrenamiento, mientras que en el caso de *train*, la tendencia es disminuir a medida que van pasando las etapas del entrenamiento.

Box Loss

Difference between predicted and true boxes

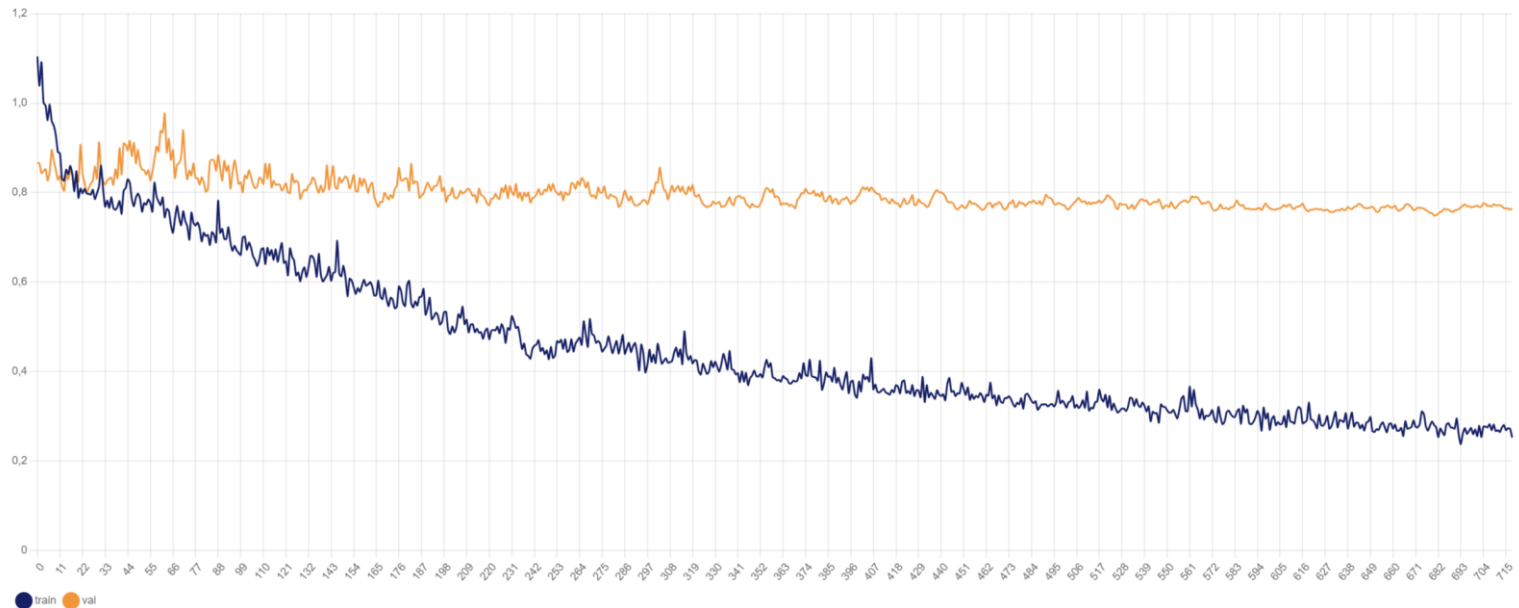


Figura 3-13 Gráfica de box loss para cada uno de los epochs (gráfica generada por Ultralytics de autoría propia)

La Figura 3-14, es la gráfica que representa el parámetro de *class loss*, que como ya se ha citado en el modelo anterior, mide la evolución del error del modelo en el proceso de entrenamiento. Se aprecia que a medida que el entrenamiento transcurre por cada una de las entradas evaluadas, el valor de *class loss* va disminuyendo de manera notable, viendo que el modelo no tiene dificultades a la hora de clasificar los objetos.

Class Loss

The accuracy of the classes in each detection

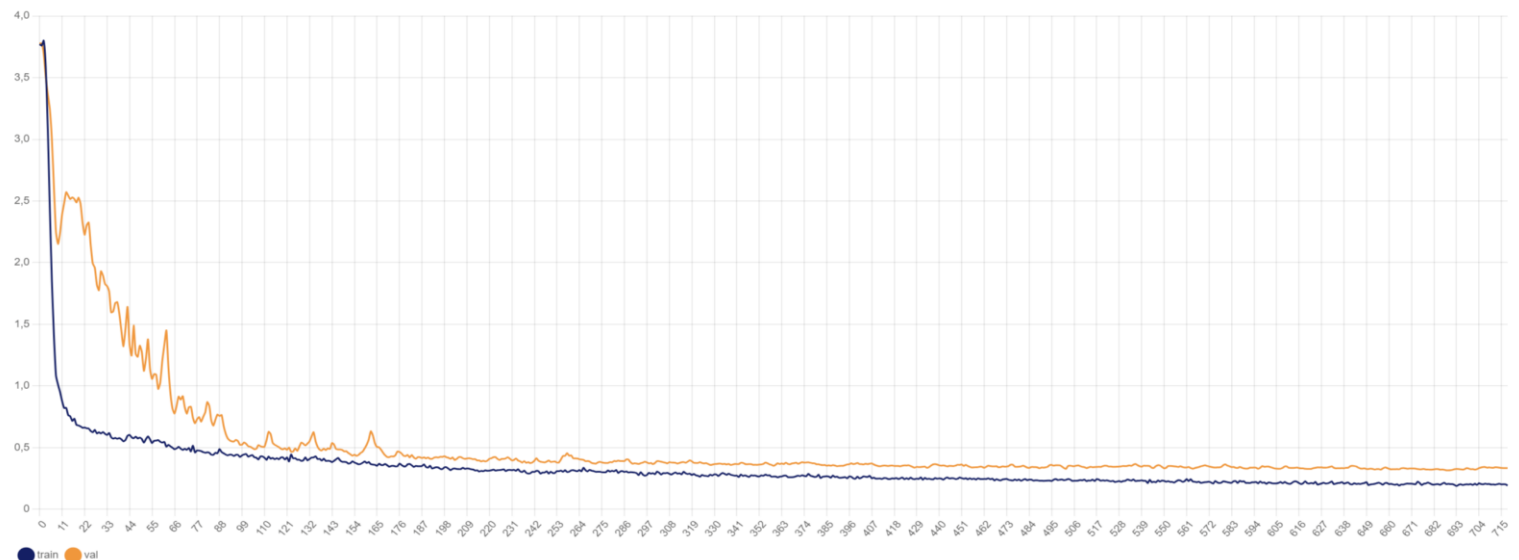


Figura 3-14 Gráfica de class loss para cada una de las entradas (gráfica generada por Ultralytics de autoría propia)

La siguiente y última gráfica que proporciona *Ultralytics* es la de *object loss* (pérdida de objetos), de forma que este parámetro representa la diferencia entre las predicciones del modelo y las etiquetas de los objetos en una imagen. En el caso de que el valor de pérdida se encuentra disminuyendo, esto indica que el modelo está mejorando su capacidad de detectar los diferentes objetos en las imágenes. En el caso contrario, si aumenta, significa que el modelo está memorizando los datos en lugar de aprender donde se encuentra cada objeto. En la Figura 3-15, se aprecia que la tendencia de las gráficas durante el

entrenamiento del modelo se va estabilizando. Al principio del entrenamiento, se aprecia como hay diferentes picos, lo que indica que el modelo está teniendo dificultades a la hora de detectar objetos específicos. Esto se debe a uno de los problemas comentados anteriormente, como es la iluminación que se producía en el agua durante la grabación. Este problema, con ciertos problemas más que han surgido con el entrenamiento de los modelos serán detallados en el siguiente capítulo de esta memoria.

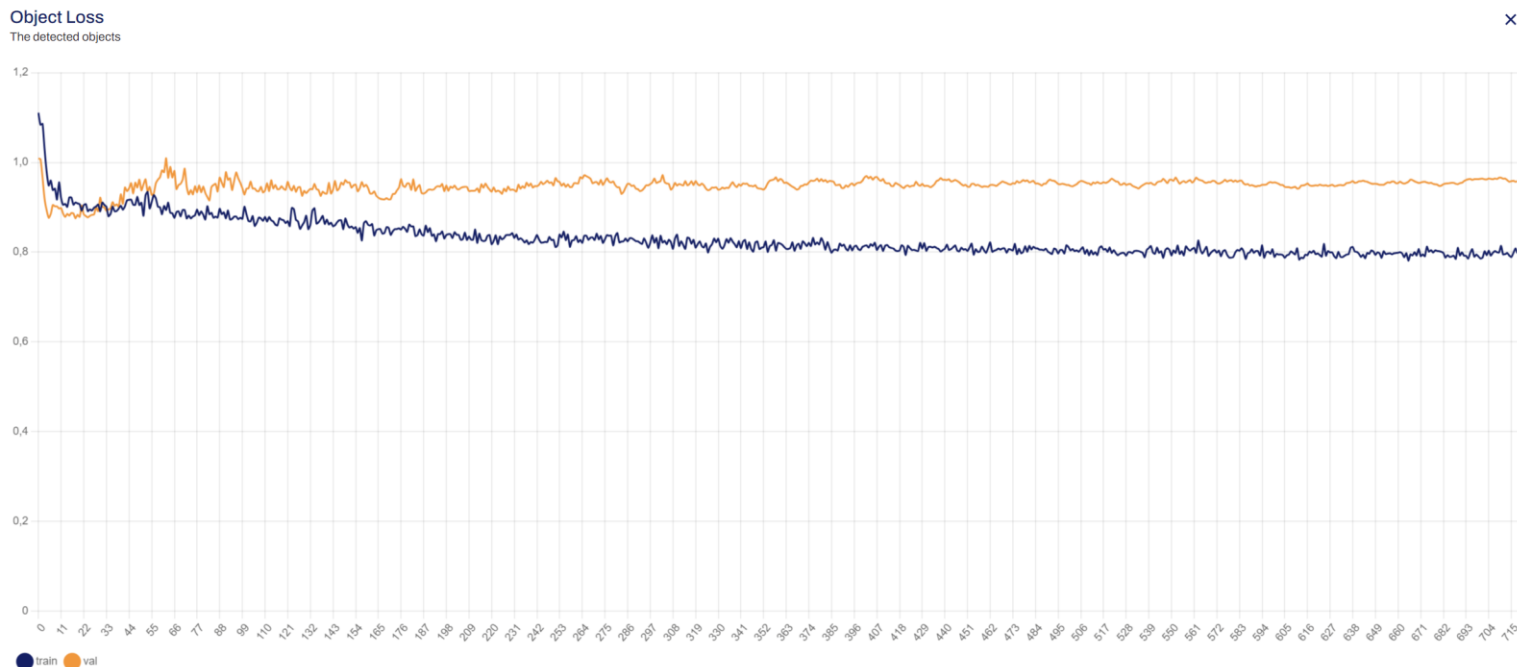


Figura 3-15 Gráfica de *object loss* durante el transcurso del entrenamiento (gráfica generada por *Ultralytics* de autoría propia)

3.6 Proceso de calibración de los dispositivos de grabación

Para realizar la tarea de la calibración de ambos dispositivos de grabación, se va a realizar mediante dos procesos diferentes, contrastando las similitudes y diferencias entre ambos. El primero de ellos se va a realizar gracias a las fotografías que se realizaron en el modelo de la pelota de *ping pong* y al uso de una red neuronal directa, es decir, se va a resolver el problema del mapeado como si se tratase de resolver una regresión entre las coordenadas de los objetos en las imágenes obtenidas por ambas cámaras (4D) y su posición en 3D, así como su función inversa (mapa 4D a 3D). En el segundo caso, la calibración se va a realizar con un tablero de ajedrez. Este proceso es más conocido, debido a que se trata de un procedimiento estándar que está integrado en la librería de *OpenCV*⁶, que proporciona algunas funciones para realizar el proceso de la calibración de las cámaras. Ambos procesos se ejecutan gracias al *script camera_calibration_tool*, de forma que este código permite entrenar la red neuronal que se va a aplicar, y además permite calcular las matrices obtenidas en la calibración que emplea el tablero de ajedrez. La principal diferencia entre ambos procedimientos es que en la que emplea el tablero de ajedrez no es necesario conocer cuál es la posición del tablero, ya que en este caso el proceso de calibración se basa en la búsqueda de los patrones visuales del tablero. Sin embargo, en la calibración por regresión directa, se recogen una serie de posiciones conocidas para poder entrenar una red neuronal que aprende a generar ese mapeado. Por otra parte, el mecanismo del tablero de ajedrez solo puede ser aplicado en entornos visuales a corta distancia, no siendo operativo cuando las distancias son muy grandes, tal y como va a ocurrir en el desarrollo final de este TFG donde se aplicará este procedimiento en vídeos obtenidos en el entorno de la ENM.

⁶ *OpenCV (Open Computer Vision)*: librería de computación visual que se utiliza para el procesamiento de imágenes en *Python*, proporcionando herramientas para el procesamiento de imágenes, reconocimiento de características o el seguimiento de objetos.

3.6.1 Calibración mediante una red neuronal directa

Como se ha comentado anteriormente, este proceso necesita el modelo que se ha descrito en el apartado 3.4. De cada fotografía que hemos obtenido del modelo, tenemos 2 valores de entrada por cada cámara utilizada, siendo en total 4 valores ($u_1, v_1; u_2, v_2$) que representan la posición, en píxeles, del centro de la pelota de *ping pong* en cada uno de los *frames* de ambas cámaras. Con estos valores, entrenamos una red neuronal con diferentes capas lineales y no lineales. En la capa lineal oculta, se puede seleccionar el número de neuronas presente para ir probando diferentes combinaciones y resultados. La función utilizada para desarrollar las capas no lineales es *Leaky ReLU (Rectified Linear Unit)*. Esta función es similar a *ReLU*, pero en este caso, en lugar de tener una pendiente cero para los valores negativos, tiene una pendiente pequeña. Esto hace que la función no se detenga en valores negativos completamente, evitando de esta manera la saturación de la unidad de cálculo. Lo que se busca con esta red neuronal, es conseguir, a la salida de la red, obtener los valores (x, y, z) de la posición en la que estaría la pelota según el modelo entrenado anteriormente, y por otra parte se entrena otra red neuronal que realiza el mapeado inverso, es decir, desde una posición de un punto en 3D, determinar donde debería verse en ambas cámaras. En la Figura 3-16, se observa la estructura de la red neuronal directa empleada para resolver esta tarea.

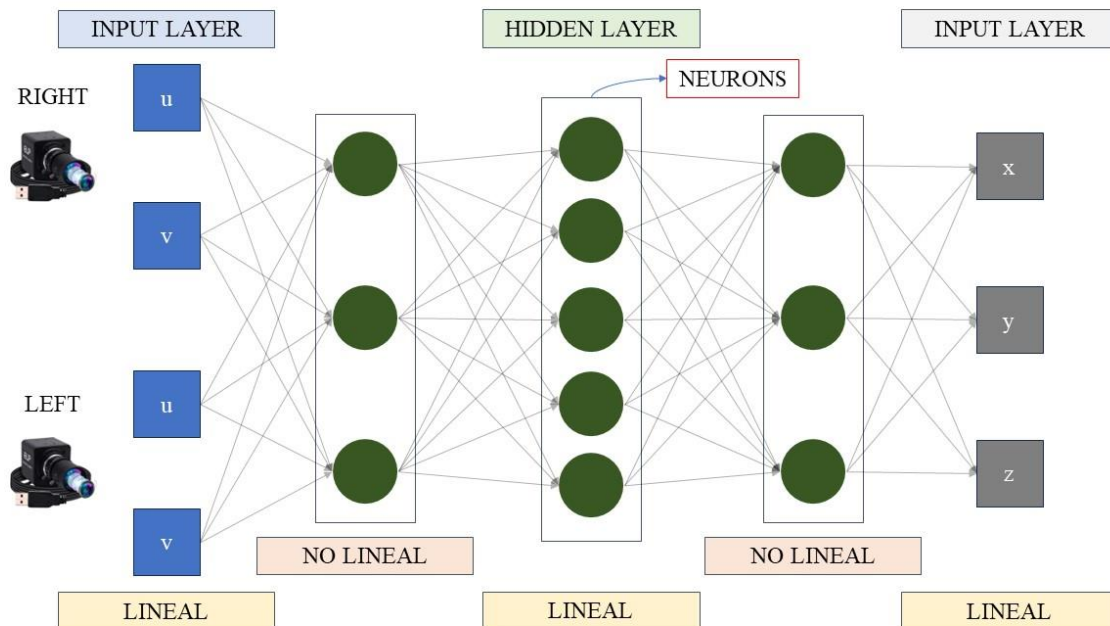


Figura 3-16 Red neuronal empleada para el proceso de calibración de los dispositivos gráficos (autoría propia)

Gracias a este proceso de calibración, se puede observar la posición del objeto con respecto a las posiciones que se insertaron en el modelo de obtención de imágenes de un objeto desde una posición fija. Además, esto permite poder hacer un *tracking* a este objeto en un vídeo, observando en pantalla la posición que tiene el objeto en todo momento.

La red neuronal ha sido entrenada para un número de 50, 100 y 200 neuronas en su capa lineal interna., de forma que cada red neuronal se ha entrenado un total de 50000 *epochs*, lo que quiere decir que cada red ha tenido el mismo “tiempo de entrenamiento”. A continuación, se muestra la comparación de las diferentes redes neuronales entrenadas con respecto a la verdadera posición en la que se encontraba el objeto a la hora de realizar la obtención de las imágenes:

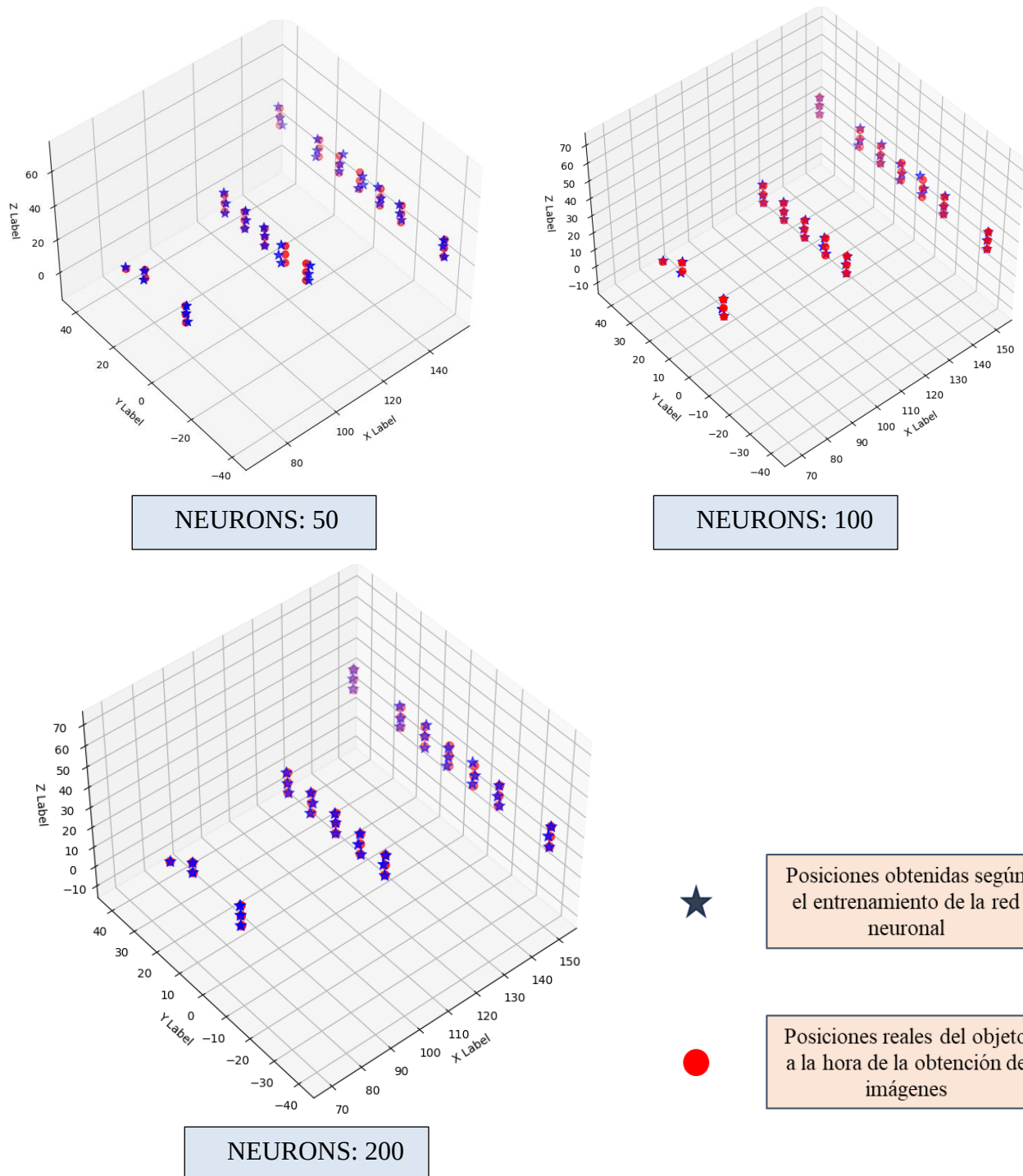


Figura 3-17 Modelo obtenido tras el entrenamiento de las redes neuronales (autoría propia)

Como se aprecia en la Figura 3-17, la red neuronal que se asemeja a las posiciones exactas es la que se ha entrenado con un total de 200 neuronas. En los otros dos procesos, se observa cómo hay mayor error a la hora de estimar la posición con este proceso de calibración.

3.6.2 Calibración mediante *OpenCV*

Como se explicó en el apartado 2.2.4, con el uso de las dos cámaras, se puede realizar el cálculo de las coordenadas 3D de un punto en concreto. Este es el objetivo que se busca con el proceso de calibración. Este proceso se realiza a través de la librería *OpenCV*, que proporciona funciones para la calibración de las cámaras. Para poder realizar y sacar los valores necesarios para poder conseguir las coordenadas, se hace uso de un tablero de ajedrez. La clave del tablero es que todos los cuadrados que

componen el mismo, tienen el mismo tamaño, por lo que el sistema puede realizar mejor la distinción entre ellos y poder sacar los valores.

El proceso a seguir es como el diagrama de flujo que se presenta en la Figura 3-18:

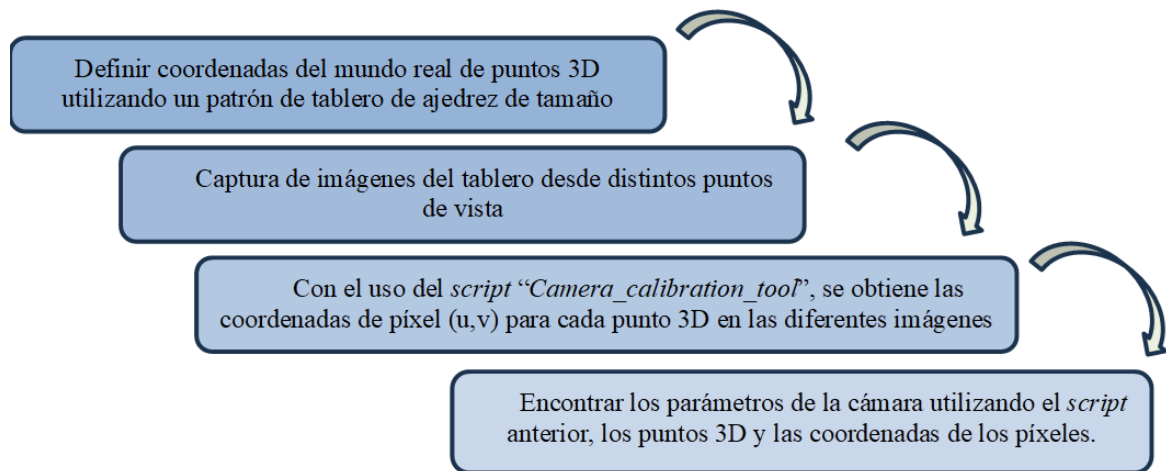


Figura 3-18 Diagrama de flujo sobre el proceso de calibración (autoría propia)

A partir de las imágenes con puntos cuyas coordenadas 2D de la imagen y 3D del mundo, obtenemos la matriz intrínseca 3×3 de la cámara (2), la rotación y la traslación de cada imagen. El proceso que realiza el script utilizado para la calibración con este sistema tendría el resultado que se contempla en la Figura 3-19:



Figura 3-19 Proceso de calibración para el cálculo de la matriz y coordenadas (autoría propia)

3.7 Obtención de vídeos con *tracking*, *frame* y posición 3D

3.7.1 Entorno de laboratorio

Una vez realizada la parte de calibración, se pretende probar el sistema en el entorno de laboratorio con la pelota de *ping pong*. Como ya se explicó, en la red neuronal hay 4 valores de entradas (2 por cada cámara), 3 valores de salida (x, y, z). En la Figura 3-20, se puede apreciar como en la imagen

proporcionada por la cámara situada a la izquierda del sistema, se puede apreciar las coordenadas del objeto en ese momento, así como el *tracking* que sigue desde que empezó a moverse, mediante una serie de puntos que indican su trayectoria. En la Figura 3-21, se muestra la detección del objeto, confirmando que ese objeto es el mismo para ambas cámaras en todo momento. Además, aparece un cuadro rojo que marca el movimiento que hace la pelota en el vídeo obtenido con la cámara izquierda. Ambas imágenes proporcionadas están recortadas, para que se pueda apreciar con precisión los valores obtenidos. Para ver los vídeos completos, se proporciona el enlace de la carpeta para descargar los vídeos:

https://drive.google.com/drive/folders/1iuDQvtKmx194J-Ef_Lh3o5LvTwNvFAK



Figura 3-20 Imagen capturada desde la cámara izquierda, mostrando la trayectoria de la pelota y las coordenadas 3D en ese momento (autoría propia)



Figura 3-21 Imagen capturada desde la cámara derecha, mostrando la detección del objeto (cuadrado blanco) como la detección del objeto al mismo instante en la cámara izquierda (cuadrado rojo) (autoría propia)

Se ha comprobado este proceso de validación de la calibración del posicionamiento en 3D empleando vídeos con diferentes movimientos de la pelota, tanto lineales como circulares o formas cuadradas. En el capítulo de Resultados y validación se introduce los detalles de este proceso, con los resultados obtenidos y comentarios sobre los problemas que han ido surgiendo durante las pruebas de validación realizadas.

3.7.2 Entorno en el ambiente marítimo

Una vez que se ha desarrollado la obtención de la posición 3D, dentro del entorno del laboratorio en el cual el modelo ha sido entrenado tras resolver el problema de la calibración de las cámaras, el siguiente paso es intentar replicar este proceso, pero en un tipo de aplicación real que se relacione con el ámbito marítimo. Para ello, se han obtenido una serie de vídeos del entorno de la ENM junto a Julio Egido Oliver, autor del TFG “Desarrollo de una plataforma de realidad aumentada de bajo coste para la investigación en aplicaciones de visión artificial”, de forma que esta vez el proceso de etiquetado de los distintos objetos presentes en los vídeos se realizó conjuntamente. A diferencia del entorno donde se había trabajado hasta ahora, los objetos serán las lanchas y veleros de instrucción, así como una *rhíb*, todos pertenecientes a la ENM.

Hay que tener en cuenta que, en este entorno, puede darse varias dificultades. La principal es la meteorología. A la hora de realizar la grabación de los vídeos, es importante que la meteorología en ese momento sea la adecuada, para que el modelo que se vaya a entrenar pueda diferenciar las lanchas y veleros que aparezcan. Además, hay que tener en cuenta que la posición de los dispositivos de grabación debe ser fija en todo momento, sin variaciones de orientación, elevación o profundidad, y la configuración de las cámaras debe ser la misma con la que se haya realizado la calibración de ambas. Por ello, el lugar que se ha escogido para la grabación, es la dársena de la ENM. Este lugar es el más favorable ya que, desde la Escalera Monumental de la ENM se puede apreciar toda la dársena con ambas cámaras, contando con puntos notables dentro de la dársena como faros y muelles.

Para poder obtener la posición 3D de los distintos objetos que se llegan a apreciar en los vídeos, es necesario tener el modelo de la calibración de las cámaras en un lugar como en el laboratorio. Para ello, debe hacerse un estudio de la zona de la dársena, recopilando los puntos *GPS* de los mismos, así como los valores de entrada de ambas cámaras ($u_1, v_1; u_2, v_2$) que proporciona *Label Studio* a la hora de realizar el etiquetado de estos puntos en la aplicación. Con la ejecución del *script camera_calibration_tool*, se puede entrenar a la red neuronal, que va a permitir estimar el mapeado entre los puntos 3D y sus posiciones en las imágenes obtenidas por las cámaras.

Empleando la herramienta a *Google My Maps*, se pueden contemplar, tal y como se aprecia en la Figura 3-22, los diferentes puntos que se han escogido para realizar el estudio de la zona, los cuales son proporcionados al código para entrenar la red neuronal. Es necesario que se haga una selección lo más precisa posible, así a la hora de obtener los puntos 3D de las lanchas y veleros sus posiciones obtenidas serán más exactas. Además, es necesario establecer un sistema de coordenadas desde la posición donde se encuentran las cámaras de grabación.

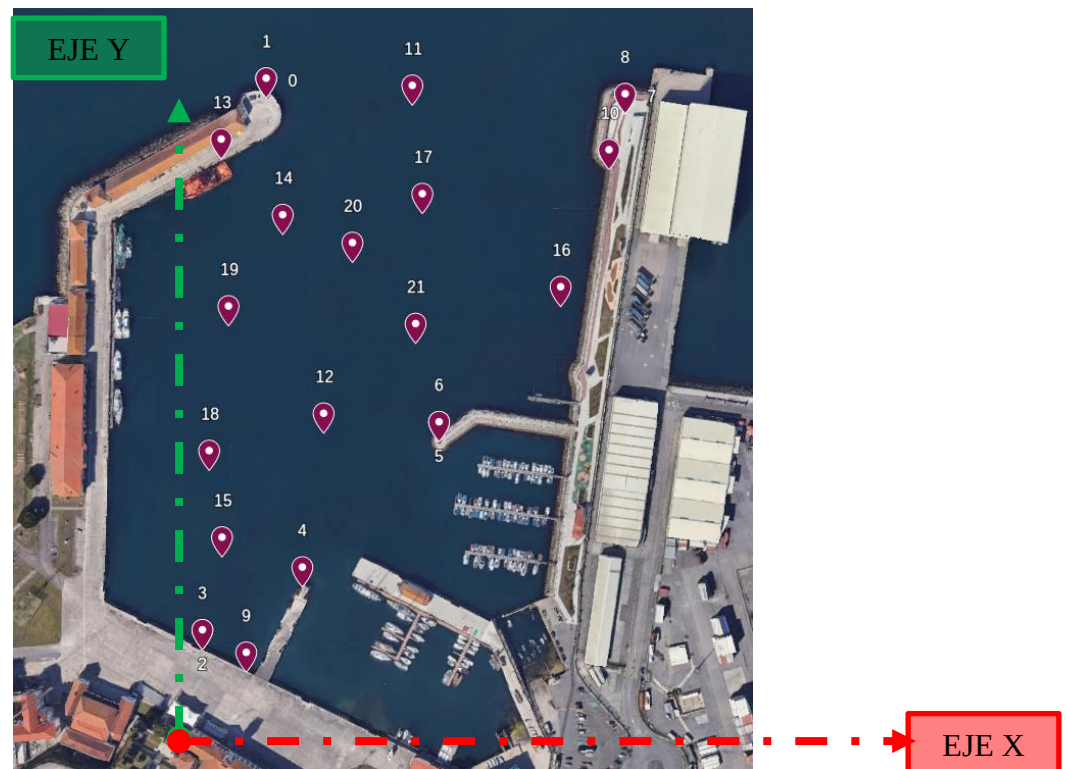


Figura 3-22 Puntos notables de la dársena de la ENM para la ejecución del *script* y eje de coordenadas (autoría propia)

Como ya se visualizó en las gráficas que aparecen en el apartado 3.6.1, el entrenamiento de la red neuronal para el entorno marítimo queda de la siguiente manera (Figura 3-23), donde puede apreciarse la exactitud que se ha obtenido en el entrenamiento de la red neuronal con respecto a la posición donde se encuentra los puntos notables escogidos. El proceso de la red neuronal, es el mismo, 4 valores de entrada (u_1 , v_1 ; u_2 , v_2) y 3 valores de salida (x , y , z). En este caso, se ha escogido 500 neuronas para el entrenamiento de la red neuronal.

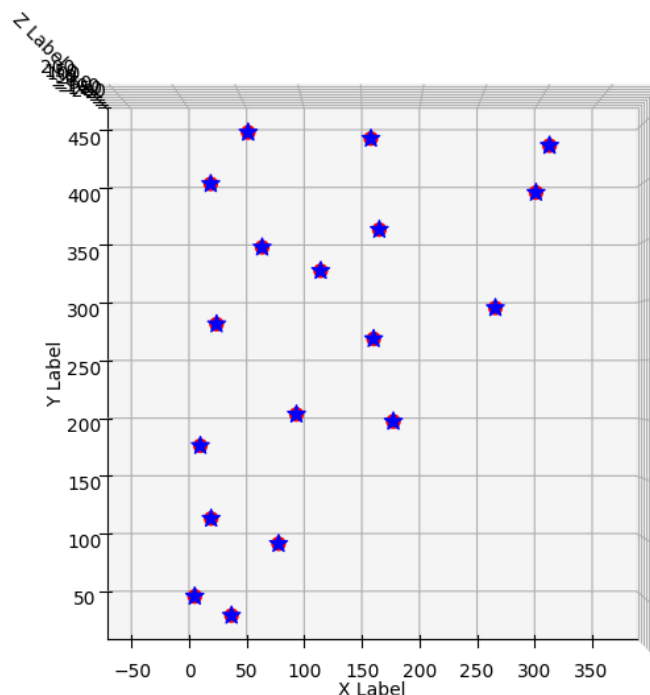


Figura 3-23 Resultado de la regresión en 3D de los puntos GPS obtenido tras el entrenamiento de la red neuronal por puntos notables (autoría propia)

Con la red neuronal y con el *script video_track_3d_tool*, se pueden obtener diversos resultados. Uno de los más importantes es la distancia entre barcos. En un *frame* particular de uno de los vídeos, se ha cogido la lancha de instrucción como el objeto desde donde se van a medir distancias a otras embarcaciones que estén más cercanas. Gracias al código, podemos determinar que, si uno de los veleros está a menos de 50 metros de distancia, el color de la línea será roja. En caso contrario, si un barco o velero se encuentra a más de 50 metros, la línea será verde. Además, se proporciona la posición *GPS* de cada embarcación detectada, así como la posición con respecto a las coordenadas x e y de la imagen. El origen de coordenadas no es donde se encuentran las cámaras, sino en la posición donde se encuentra la farola ya que, facilita la visualización de los resultados.



Figura 3-24 Distancias y posiciones *GPS* de barcos y veleros (autoría propia)

En la Figura 3-24, se puede apreciar lo comentado anteriormente con respecto a las distancias. Observando la distancia entre la lancha de instrucción con respecto al velero que se encuentra más cerca de la misma, el resultado es de 49 metros, estando en ese momento por debajo de los 50 metros de distancia que marca que la línea sea de color verde o rojo. Con respecto a las posiciones, gracias a la interpolación que se realiza con los distintos barcos y veleros que aparecen en la imagen con respecto a los puntos notables de la dársena de la ENM, los cuales han sido empleados para el entrenamiento de la red neuronal, el *script* puede determinar en cada *frame* la posición de los mismos. Esto sirve de ayuda para realizar el cálculo de la distancia relativa entre objetos ya que, al igual que se utilizó *Google My Maps* para observar los puntos notables, en este caso podemos importar los puntos *GPS* obtenidos en el código a este *software*, y ver si la distancia relativa es similar a la obtenida en el vídeo. En el caso de la toma del *frame* que vemos en la imagen anterior, se puede comprobar la distancia relativa con *My Maps*, ya que pasamos de coordenadas 3D a coordenadas 2D.

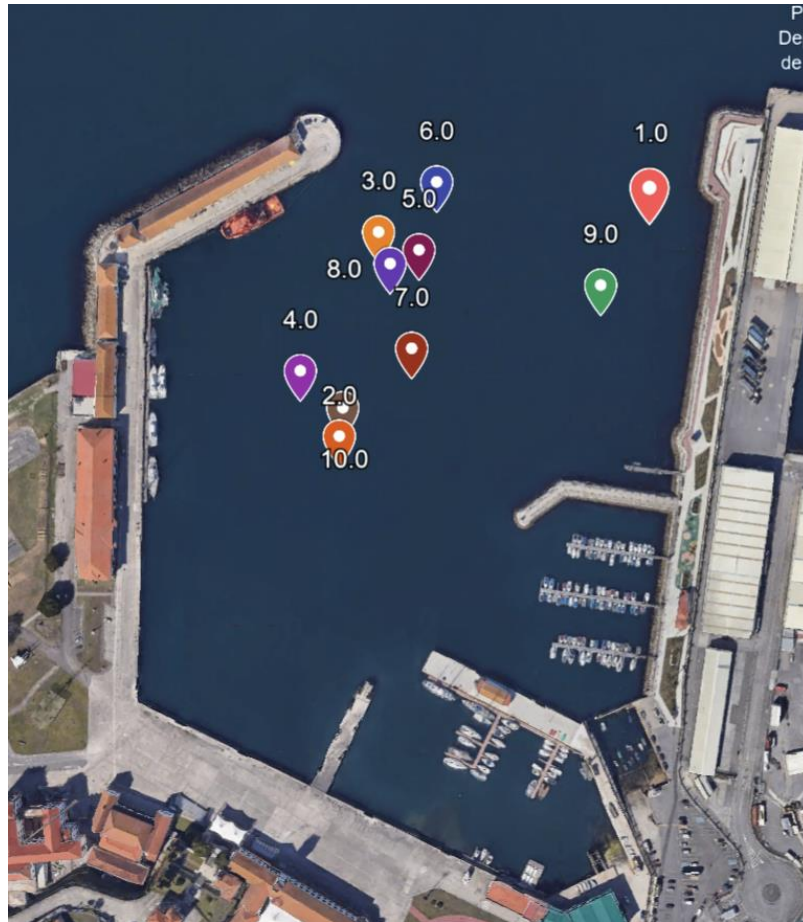


Figura 3-25 Posiciones de barcos y veleros en el *frame 12* en *My Maps* (autoría propia)

El barco desde donde se toma las distancias sería el que tiene el identificador 3.0, tal y como se ve en la Figura 3-25, siendo la lancha de instrucción de la ENM. En la Figura 3-24, se aprecia como el velero más cercano se encuentra a 49 metros. Sin embargo, al medir la distancia el *software* de *Google*, la distancia entre ellos es de 27 metros. En cada distancia obtenida hay un margen de error de 20 a 30 metros con respecto a la que se ha estimado en el vídeo, lo que se considera un error que puede ser corregido modificando el procedimiento de entrenamiento de las redes neuronales de los mapas entre los entornos 4D y 3D.

Para poder corregir esta distancia relativa, se va a entrenar otra red neuronal con el origen centrado en la farola que se encuentra en la explanada de la ENM. Para realizar este entrenamiento, se ha generado una malla de puntos situados dentro de la dársena. El eje Y de la malla, se dirige hacia el norte, y en el eje X, se dirige hacia el este. A partir de tener los ejes ya designados, se han tomado 4 puntos en ellos, y se han aproximado dos puntos de fuga, uno con respecto al norte y otro con respecto al este. Gracias a esto, obtenemos una malla como se observa en la Figura 3-26 (cámara derecha) y Figura 3-27 (cámara izquierda).



Figura 3-26 Malla obtenida con respecto a la cámara derecha (autoría propia)



Figura 3-27 Malla obtenida con respecto a la cámara izquierda (autoría propia)

Con esta malla, se han obtenido coordenadas 2D de las imágenes correspondientes, y al contrario que en el caso de la red neuronal entrenada por puntos notables, en este caso se elimina la coordenada z (que se considera como cero para todas las embarcaciones) dejando el mapeado en un entorno 2D con las coordenadas x e y . Para la obtención de la malla, se ha ejecutado el fichero *camera_test_3D*, que se emplea para visualizar el *grid* de puntos de referencia que se han obtenido de las imágenes para el nuevo modelo de posicionamiento. Además, los puntos obtenidos se han exportado a *Google My Maps*, quedando la situación de los puntos de la siguiente manera como se percibe en la Figura 3-28:

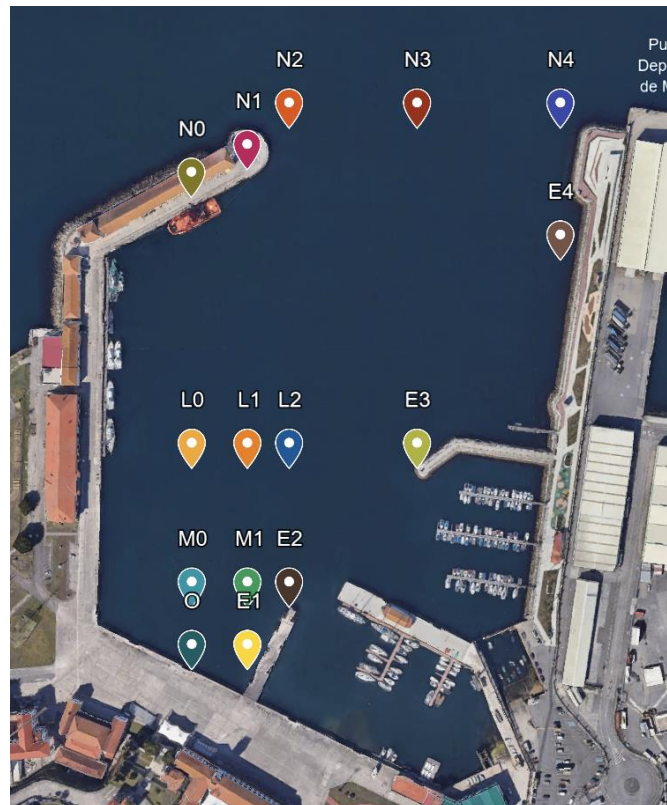


Figura 3-28 Disposición del *grid* tras la obtención de los puntos en *My Maps* (autoría propia)

Para la creación del nuevo modelo de posicionado, se ha hecho uso del código *camera_calibration_tool*. Este código permite entrenar o visualizar el correcto funcionamiento de este modelo. La principal diferencia con respecto al modelo de los puntos notables, es que en este nuevo modelo ya no hay coordenada *z*, y como se ya se comentó anteriormente, se usan aproximaciones puntos de fuga a partir de 4 posiciones conocidas. Para este entrenamiento, se ha seleccionado un total de 200 neuronas para la obtención del modelo, dando como resultado el resultado que se muestra en la Figura 3-29.

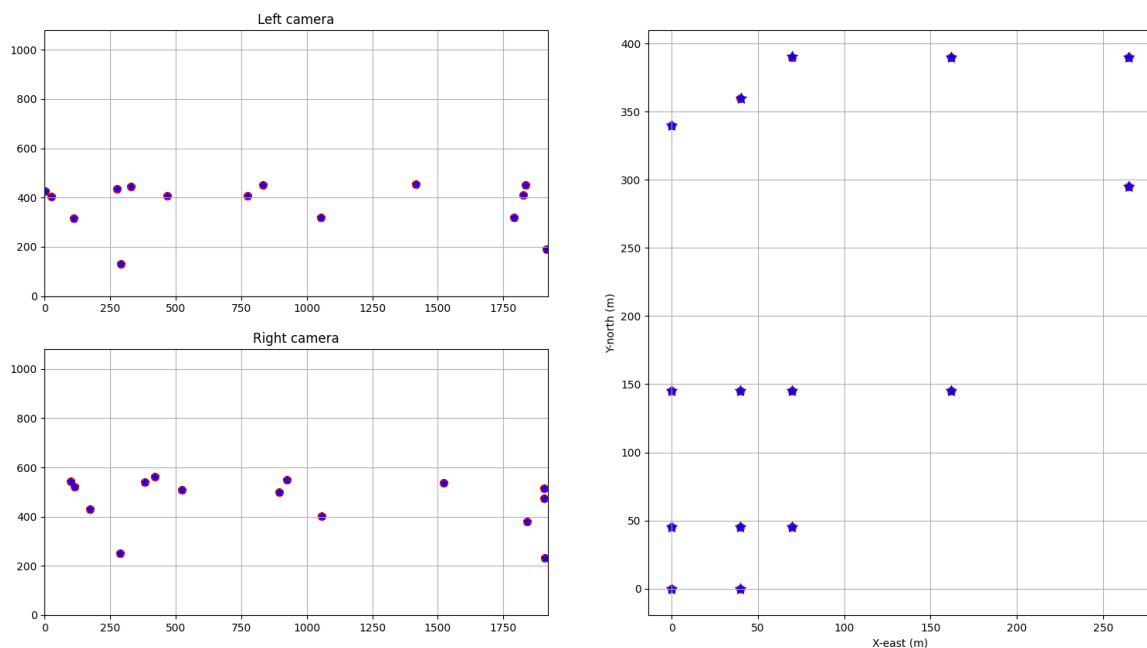


Figura 3-29 Resultado de la regresión en 2D y 3D de los puntos de la malla obtenido tras el entrenamiento de la red neuronal (autoría propia)

En el apartado 4.1.3, se verán aquellos resultados y análisis acerca de este nuevo modelo, detallando también las conclusiones obtenidas con el modelo anterior, haciendo una comparación entre ellos para determinar cuál es el mejor, así como obtener conclusiones para poder hacer mejoras en futuros modelos.

4 RESULTADOS Y VALIDACIÓN

En este capítulo, se presentan y analizan los resultados obtenidos en las diferentes etapas que componen el desarrollo del trabajo. Se comentarán aquellos resultados que permitan obtener conclusiones tras el análisis de los mismos. Para ello, el presente capítulo se divide en 2 partes. La primera parte del capítulo constituye la aplicación y análisis del modelo desarrollado durante el proceso. En esta parte se trata los resultados obtenidos tras aplicar el modelo tanto en el entorno del laboratorio como en el entorno marítimo. Además, se destacan los problemas surgidos durante el desarrollo del *dataset*, así como la fase del entrenamiento del modelo. En la segunda parte, se procede a una validación y análisis general de los logros conseguidos, teniendo en cuenta todos aquellos parámetros que permiten sacar conclusiones sobre el trabajo realizado.

4.1 Aplicación y análisis del modelo

4.1.1 Resultados y validación de los modelos entrenado

En primer lugar, una de las primeras fases del desarrollo de este trabajo ha sido la obtención de imágenes de la pelota de *ping pong* para, posteriormente, poder realizar el proceso de calibración gracias al modelo entrenado. Los resultados de esta prueba son centrados en la detección y clasificación de un objeto. En este caso, al ser solo un solo objeto, no ha habido ningún problema en la parte del entrenamiento, ya que las condiciones eran las mejores y en un entorno donde la meteorología no influye nada. Al realizar el entrenamiento, se obtiene que al 98%, el modelo detecta de manera efectiva a la pelota, siendo solo un 2% la confusión que se produce a la hora de la clasificación. Se concluye que, en este caso, no es necesario aumentar el número de imágenes para realizar un nuevo entrenamiento del modelo debido a que el entrenamiento realizado cumple de manera efectiva.

La segunda fase del desarrollo consistió en realizar el seguimiento de varios objetos en un recipiente de agua. En esta fase, es donde se han presentado diversos problemas, que han limitado el uso del modelo entrenado. Uno de los problemas principales fue el cambio que se produjo en la configuración del montaje de las cámaras de grabación. Al principio, el montaje estaba pensado para realizar la obtención de imágenes de un objeto en una posición estática. Una vez terminado ese trabajo, la idea era obtener imágenes de objetos en dinámico en un recipiente de agua, por lo que el montaje de las cámaras cambió. Este cambio de montaje no se debió haber realizado ya que, a la hora de realizar la calibración de los dispositivos de grabación, en ningún momento se puede cambiar la distribución de las cámaras, teniendo en todo momento el mismo enfoque, *zoom* y claridad en los dos dispositivos empleados.

Aparte de este problema, podemos destacar diversos problemas que vienen bien para futuros trabajos que se puedan realizar con la misma temática que el presente. En los apartados 2.1.7.2 y 2.2.5, se comentan varios de los problemas que se van a comentar a continuación. La oclusión y la sensibilidad

en las condiciones de iluminación causan errores a la hora de la detección y clasificación de objetos. En este caso, a la hora de detectar objetos, el modelo identificaba de manera errónea objetos en la parte del agua donde había más claridad, así como confunde un mismo objeto, identificándolo de una manera u otra, dependiendo del momento del *frame* del vídeo. A continuación, se muestra una imagen donde se aprecia los problemas de oclusión y sensibilidad a la iluminación (Figura 4-1), y la matriz de confusión con respecto a la clasificación de los objetos (Figura 4-2), donde se muestra la confusión de clasificar un objeto de manera incorrecta.

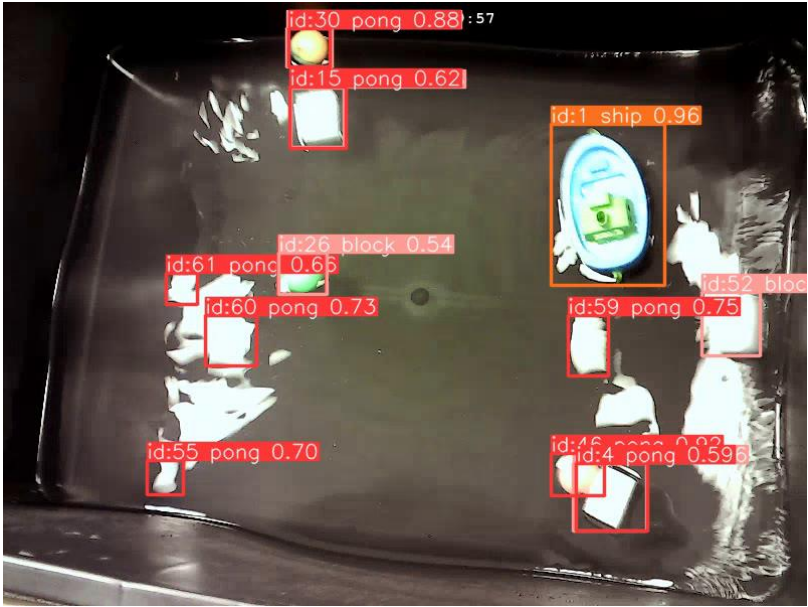


Figura 4-1 Problemas de oclusión y sensibilidad a la iluminación (autoría propia)

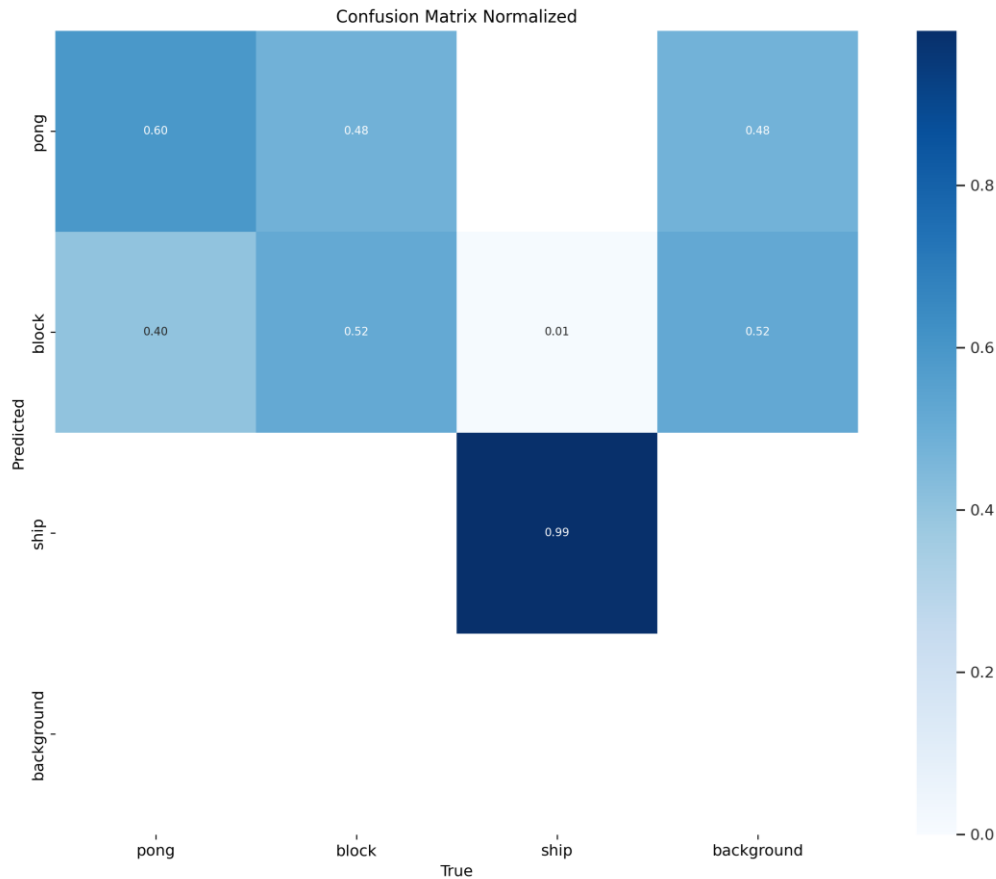


Figura 4-2 Matriz de confusión generada durante el entrenamiento del modelo (autoría propia)

4.1.2 Resultados y validación del modelo en el entorno del laboratorio

Una vez que se realizó la calibración mediante el proceso de redes neuronales, se realizó el proceso de obtener la posición 3D de un objeto dinámico (Figura 4-3). El resultado obtenido fue fructífero, ya que las posiciones obtenidas en todo momento durante la ejecución del vídeo, se asemejan a la posición que tendría el objeto si estuviese en estático en ese mismo momento. Sin embargo, si el objeto sale del entorno en el cual ha sido entrenado, las distancias y la posición no cuadraría, debido a que el sistema solo es capaz de proporcionar de manera correcta los valores de posición dentro del entorno que se le ha proporcionado mediante la obtención de imágenes en distintos puntos del entorno.



Figura 4-3 Posición y *tracking* del objeto dinámico (autoría propia)

Además, durante el desarrollo de esta parte, se apreciaron diferentes sucesos que son importantes destacar. Uno de ellos fue que la grabación de ambas cámaras no estaba totalmente sincronizada en el tiempo en algunas de las tomas realizadas, de forma que una de las cámaras adelantaba, en la trayectoria visual, a la otra. Por otra parte, un segundo problema que se encontró es que, en el mismo vídeo, se aprecia que, a la hora de la detección del objeto, el modelo confunde la pelota con la mano en algunos *frames* del vídeo obtenido, perdiendo de esta manera el *tracking* del objeto que deseamos seguir. Por último, en los vídeos se aprecia como el identificador del objeto en los diferentes *frames* representados con la trayectoria cambian de forma continua, es decir, se pierde la capacidad de realizar tracking del objeto. Esto podría solucionarse empleando una cámara con capacidad de generar un mayor número de *frames* por segundo, ya que el problema surge al intentar seguir un objeto cercano que tiene una alta velocidad relativa que dificulta su seguimiento entre diferentes *frames*. A continuación, se muestra el enlace de la carpeta donde se incluyen un vídeo de cada cámara, grabados en la misma fecha y hora, donde se puede observar los problemas que se ha comentado anteriormente. Para una correcta visualización de ambos vídeos, es altamente recomendable que se ejecute una descarga de ambos, y no se visualice directamente en el navegador, ya que podría darse una importante pérdida de calidad:

https://drive.google.com/drive/folders/1nRR9bwggxEnVmKULT8vRxnPzzA43vNFT?usp=drive_link

4.1.3 Resultados y validación del modelo en el entorno marítimo

Antes de comenzar con el análisis de esta parte, en la Figura 4-4, se puede apreciar como la interfaz ha sido modificada con respecto a la Figura 3-24. Una diferencia con la anterior interfaz es la distancia que marca que la línea sea roja, ya que en esta nueva interfaz la distancia es para 80 metros, no para 50 metros como marcaba la interfaz ambigua. Para una mejor visualización de esta interfaz con respecto a la anterior, se deja a continuación el siguiente enlace donde se pueden descargar los vídeos:

https://drive.google.com/drive/folders/1Umgx8IwWOPxIBw9zPlNaGtHrwyRUDJoX?usp=drive_link

Con respecto al nuevo modelo que se elaboró gracias a la elaboración de una malla 2D de puntos, se puede concluir que las distancias relativas entre los barcos han mejorado, tanto en el vídeo, donde la menor distancia se asemeja más a la realidad, como en *Google My Maps*, que al igual que en el modelo entrenado gracias los puntos *GPS* de la dársena, se puede comprobar si las distancias coinciden con las del vídeo. En este caso, y tras una comprobación en el *software* de *Google*, se llega a la conclusión de que las distancias que se han obtenido concuerdan de manera más correcta. Sin embargo, como se comprobará en la Figura 4-5, se ve que los veleros situados en la derecha no concuerdan su posición *GPS* con respecto a la posición que tienen en los vídeos. Para esta comprobación, se ha tomado el *frame* 24 del vídeo.

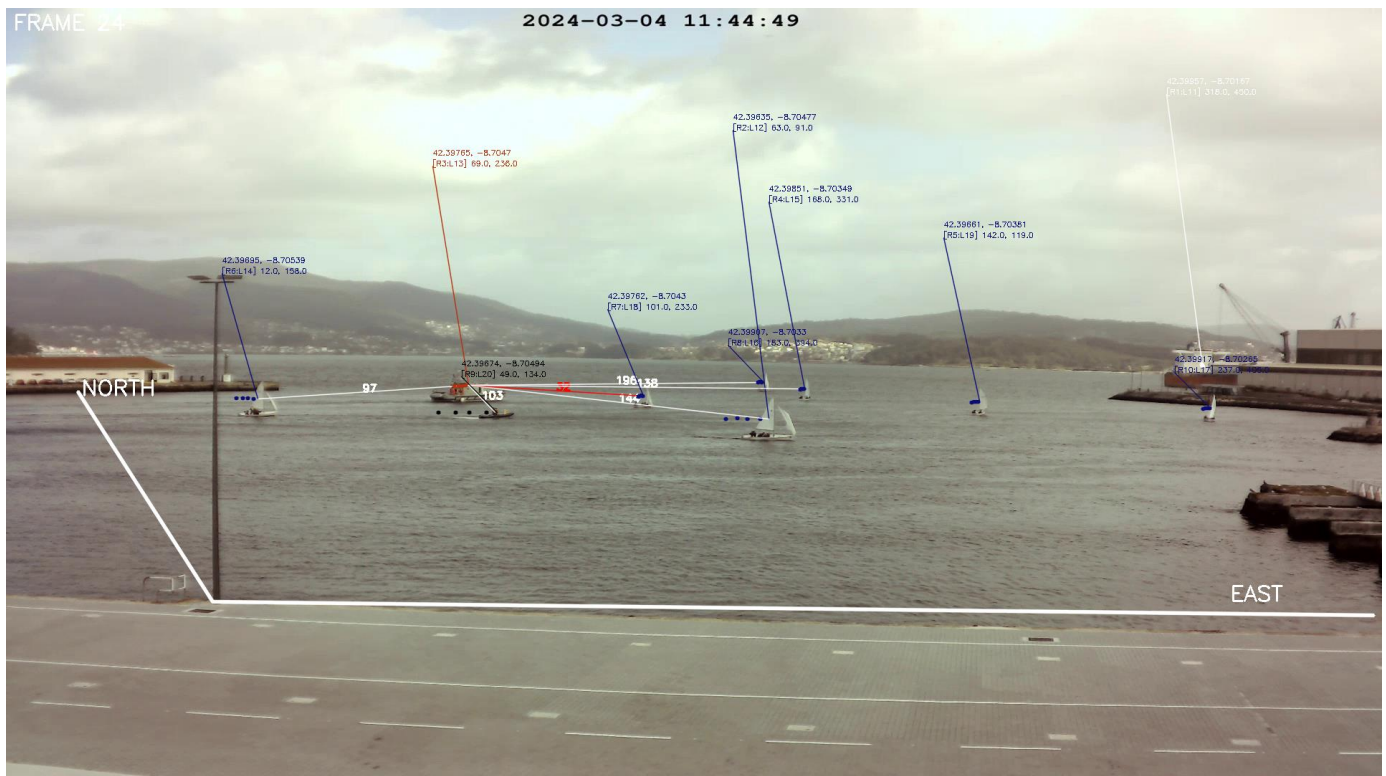


Figura 4-4 Imagen de posición, distancias y *tracking* en el *frame* 24 (autoría propia)

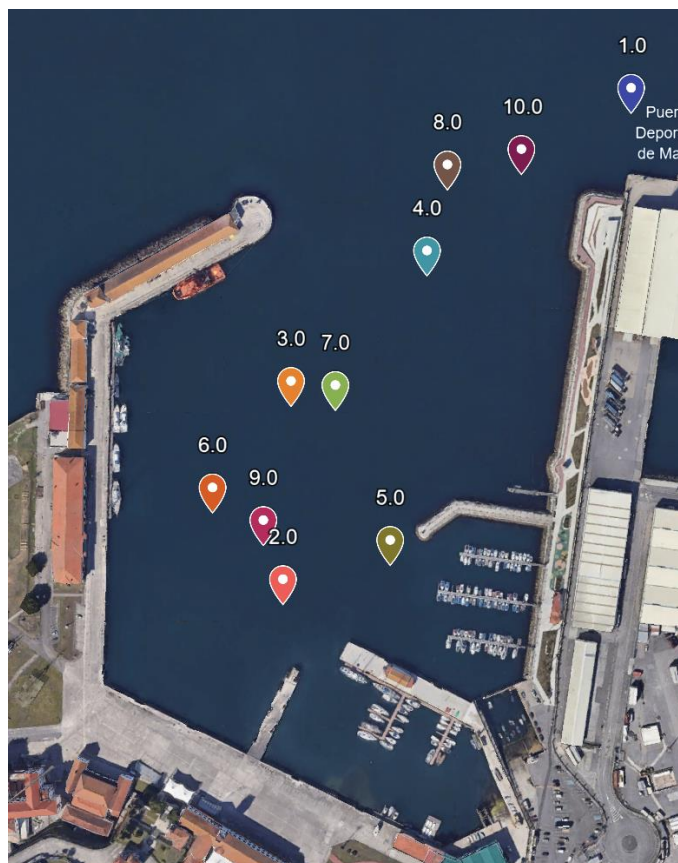


Figura 4-5 Posiciones de barcos y veleros en el *frame* 24 en *My Maps* (autoría propia)

Como se puede apreciar en ambas imágenes, no concuerdan los veleros que se sitúan en la parte derecha del vídeo. Esto se debe a que, a la hora de entrenar la red neuronal, con los resultados obtenidos en la Figura 3-29, se observa como en la parte del centro de la dársena, ningún punto recoge información sobre esta zona. Sin embargo, a la hora de reflejar las distancias de los veleros y la *rhib*, que se encuentran más cerca de la lancha de instrucción, que es el objeto a partir el cual se miden la distancia, se nota una notable mejoría con respecto al modelo entrenado por puntos notables. Además, se observa en todas las imágenes obtenidas de los vídeos de *tracking*, como en la parte de la derecha se ve un mercante. Este mercante no se encuentra dentro del recinto de la ENM, ya que está en el muelle del puerto de la ciudad de Marín. Es por eso, que el sistema no es capaz de colocarlo en ambos modelos de manera correcta, ya que en la zona donde se encuentra no ha sido entrenada.

Hay que destacar también que en el primer modelo entrenado en este ámbito (por puntos *GPS* de la dársena), habría que añadir algún punto más al modelo desde la parte izquierda ya que, a la hora de detectar la posición de los barcos que aparecen en esta zona, encuentra más dificultad con estos barcos que con el resto.

Para ambos modelos se ha llegado a la conclusión de que el problema que más afecta a ambos modelos es la profundidad de las imágenes obtenidas. La percepción de la profundidad ha sido uno de los grandes inconvenientes tanto para la hora de obtener posiciones *GPS* de la zona como a la hora de determinar el *grid* utilizado para el entreno del modelo. Esto se puede observar con las diferentes imágenes extraídas de los vídeos con respecto a la disposición que se muestra en *My Maps* justo en el mismo instante. Debido a la percepción, la perspectiva de que un velero o barco se encuentre a mayor o menor distancias de otra embarcación, cambia con respecto al observador, ya que cada uno puede ver esa percepción diferente al resto. Una de las soluciones que se propone para solucionar este problema, es que ambas cámaras no se encuentren paralelas a la hora de la obtención de los videos, sino que una se encuentre cubriendo uno de los ejes, y la otra cámara orientarla 90° para cubrir el otro eje. De esta

manera, se puede ayudar a la hora de determinar una mejor estimación de la zona donde se encuentran las embarcaciones, dando un mejor resultado de posicionamiento y una mejor exactitud y precisión de distancias entre ellos.

Por otro lado, se puede analizar la trayectoria que han seguido las embarcaciones durante varios *frames* del vídeo. Esto se ha logrado gracias a que el *script* permite guardar la posición de los diferentes barcos que detecta en el vídeo en los *frames* que el usuario requiera. Para hacer esta prueba, se han guardado los puntos *GPS* desde el *frame* 100 hasta el 260 en un rango de 40. Con estos puntos, al igual que se ha hecho en anteriores ocasiones, *Google My Maps* permite insertarlos y crear un mapa con cada uno de los puntos. Además, se puede realizar una línea uniendo los diferentes puntos que identifican al mismo barco. Con esta línea, se puede comprobar la distancia que ha recorrido entre ese rango de *frames*, pudiendo calcular así la velocidad, gracias a la distancia que conocemos con el software y al tiempo que podemos percibir en el vídeo, desde que comienza el *frame* 100 hasta el *frame* 260. En la Figura 4-6, se comprueba que la embarcación de la *rhíb* es la que más distancia recorre, como se puede comprobar en los vídeos adjuntados en el último enlace añadido al presente trabajo. El único barco que no realiza ninguna trayectoria es el mercante, el cual ya se ha comentado anteriormente la problemática surgida con el mismo.

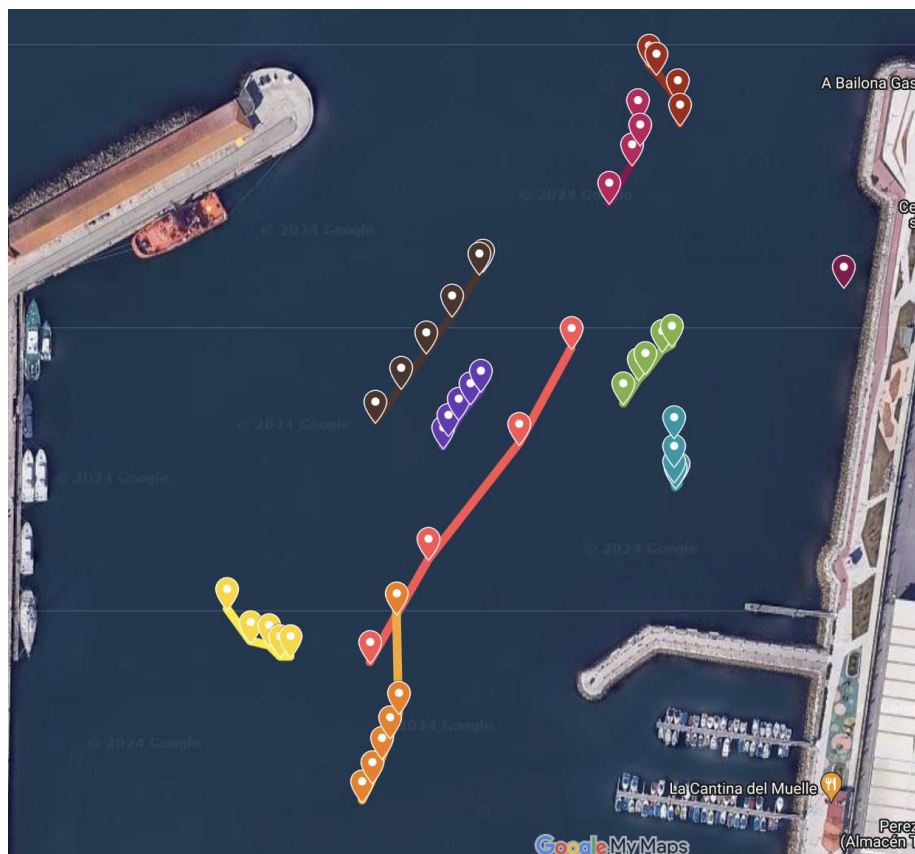


Figura 4-6 Trayectoria seguida por los barcos entre los *frames* 100 y 260 (autoría propia)

Por último, y haciendo uso parte del trabajo de mi compañero Julio Egido Oliver, cuyo título de su trabajo está citado en el apartado 3.7.2, se comprueba con el *AIS* la exactitud a la hora del posicionamiento del buque en un concreto instante. El *AIS* es un sistema de información, que incluye entre otros datos la posición *GPS*, el rumbo y la velocidad de la embarcación. El único buque que transmite en *AIS* en el vídeo, es la lancha de instrucción. Para comprobar el posicionamiento, se ha escogido el *frame* 32 gracias al *script* *ais_capture_tool*. En la siguiente imagen, los valores que se ven en amarillo, son los que son transmitidos por *AIS*, además de proporcionar los valores de *x* e *y* a la hora que marca estos valores (44 min 49 seg). Como se ve en la Figura 4-7, según el *AIS*, la lancha debería estar en la posición (68, 271) a las 11:44:49. Según el modelo entrenado, a esa hora, la lancha se

encuentra en la posición (70, 235). La diferencia con respecto al eje X es mínima, pero con el eje Y sí que se muestra una clara diferencia. Esto se debe a lo citado anteriormente en este apartado sobre la percepción de la profundidad. Esto es una prueba más para ver que la disposición de ambas cámaras debería centrarse una para cada eje. Además, añadir que para que se debería haber realizado la grabación de los vídeos con décimas de segundo, permitiendo una mejor precisión a la hora de tomar los *frames* exactos para la información AIS proporcionada por el buque.

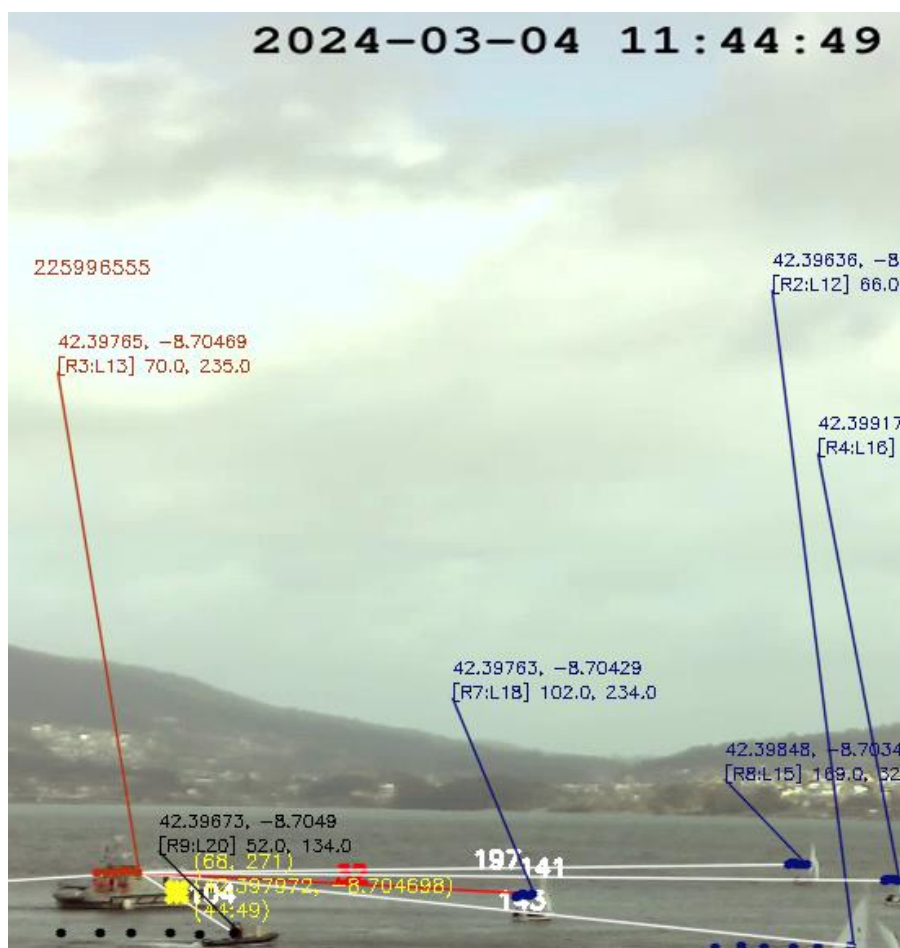


Figura 4-7 AIS (en amarillo) de la lancha de instrucción (autoría propia)

4.2 Resultados y validación general

Una vez evaluado todos los modelos desarrollados en el presente trabajo y los resultados del mismo, es necesario analizar variables que permiten que se haga de una manera más exacta.

En primer lugar, hay que referirse a la base de datos escogida para el entreno de los diferentes modelos. Para este trabajo se han elaborado 3 *dataset* diferentes para llevar a cabo los diferentes modelos creados que se han citado en el apartado de desarrollo. Para este trabajo, los dos primeros *datasets*, “CALIBRATION_PONG” y “WATER_BASIN” se han elaborado con, aproximadamente, una extensión de una centena de imágenes. El último *dataset*, “ESCALERA_MONUMENTAL”, cuenta con 4 imágenes (2 por cada cámara) extraídas de los vídeos realizados para llevar a cabo el proceso en un entorno marítimo, las cuales han servido para asignar y establecer los diferentes puntos notables de la dársena de la ENM para realizar el proceso de pasar de 4D ($u_1, v_1; u_2, v_2$) a 3D (x, y, z), como también la realización de la malla para realizar el proceso inverso, de 2D (x e y , en este caso no hay coordenada z) a 4D. Con el número de imágenes en cada *dataset* se puede llegar a la conclusión que mientras más fotografías contenga la base de datos, el entrenamiento de detección y clasificación de los objetos

aumentaría su eficacia. En el *dataset* “WATER_BASIN”, se comprobó cómo se perdía la detección en ocasiones, pudiéndose solucionar añadiendo más imágenes para entrenar al modelo.

Con respecto a la colocación de los dispositivos de grabación, se ha aprendido que una vez que los dispositivos de grabación son calibrados, no debe cambiarse su orientación. Esto se ha visto reflejado en los resultados del modelo en el recipiente de agua y con el método de calibración con el tablero de ajedrez (proceso mediante *OpenCV*). Además de esto, para este caso se necesitaría unas cámaras con mayor *frames* por segundo, siendo capaz de detectar objetos a mayor velocidad como también se comprobó con las diferentes trayectorias que se realizaron con la pelota de *ping pong*, en la que se perdía por momentos el *tracking*.

Referente a la percepción de la profundidad, es importante tener en cuenta este factor. A la hora de tomar imágenes, la perspectiva y percepción engaña al ojo humano, llegando a engañar donde está el objeto realmente. En el caso de la malla realizada dentro de la dársena, se ve como al fondo de la imagen el tamaño entre cuadrados es muy pequeño con respecto los que se observa cerca del origen del sistema de coordenadas, comprobando que, en un píxel de la imagen, la longitud en ese píxel con respecto al eje Y (orientado al norte) puede llegar a ser de 200 metros aproximadamente. En la Figura 4-8, se puede apreciar como en el origen del sistema de coordenadas hasta el siguiente punto de la malla hay 40 metros. Sin embargo, entre el tercer y cuarto punto del eje Y, la sensación es que hay menos distancia, pero debido a esta percepción, la distancia entre esos dos puntos es de 195 metros.

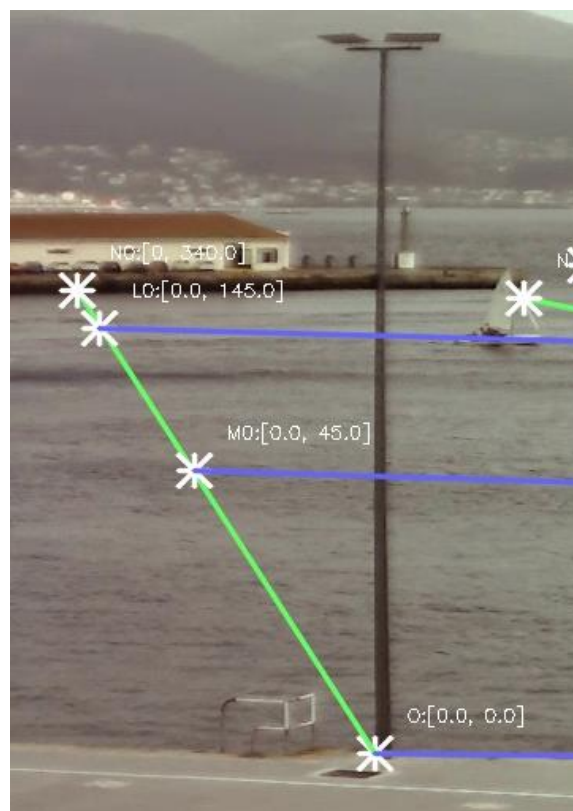


Figura 4-8 Percepción de profundidad en el eje Y (autoría propia)

Una de las variables más importante que se debe tener en cuenta para trabajos de este tipo, es la calidad de las imágenes. Se ha visto casos en el que el reflejo de la luz interfiere al sistema a la hora de detectar blancos, imposibilitando de esta manera una detección y clasificación exacta. Otros problemas como la oclusión y la superposición de objetos han jugado un papel muy importante, que se ha visto reflejado en la última parte del trabajo con los diferentes barcos y veleros navegando en el recinto militar de la ENM. En algunos momentos, como se puede ver en la Figura 4-9, el sistema pierde al objeto (en este caso la *rhíb*) por superponerse uno con el otro (velero), siendo identificado de nuevo al momento de estar claro para el sistema.



Figura 4-9 Ejemplo de superposición de objetos (autoría propia)

Con todos los análisis y resultados analizados hasta el momento, se puede llegar a la conclusión que todos los modelos desarrollados cumplen con los requisitos previstos anteriormente a sus elaboraciones con el fin de detectar y *trackear* objetos en entornos marítimos, calculando además la distancia entre ellos.

5 CONCLUSIONES Y LÍNEAS FUTURAS

5.1 Conclusiones y objetivos alcanzados

El uso actual de la inteligencia artificial en el mundo está representado un avance significativo en diversos campos. Uno de ellos es en el marino, como ha quedado detallado en este trabajo. Con la creación de una plataforma en la que se aplique sistemas de visión artificial para la localización y seguimientos de objetos en el agua, se puede llegar a ofrecer numerosas soluciones hoy en día en este ámbito. Esta plataforma puede llegar a ofrecer una solución innovadora para abordar desafíos relacionados con la seguridad marítima y la vigilancia, ya que, gracias a los modelos entrenados, se puede detectar y realizar un seguimiento de las embarcaciones en entornos acuáticos, ayudando de esta manera a disminuir el número de accidentes en alta mar y la protección del medio ambiente.

Una de las conclusiones extraídas con la ejecución de este TFG es la capacidad que puede llegar a tener estos sistemas con respecto a un ser humano. Los sistemas utilizados para el desarrollo de la plataforma trabajan con millones de parámetros y datos para llegar un objetivo final: la automatización de un sistema sin ayuda de una persona. Para una persona sería imposible recopilar grandes cantidades de información y realizar el trabajo en un periodo de tiempo corto como lo hacen estos sistemas. Añadir que gracias a estos sistemas se contribuye al desarrollo de la IA y la visión por computadora, ya que promueve la investigación y la creación de nuevos diseños de algoritmos que se emplean para la detección y seguimientos de objetos, ya sea en un entorno marítimo como, por ejemplo, en un entorno urbano.

Sin embargo, para la creación de sistemas como este, es necesario cumplir con una serie de requisitos. Hoy en día, no se ha encontrado en el mundo comercial de bajo coste un sistema con dos dispositivos de grabación que permita el cálculo de distancias entre objetos, además de la clasificación y el *tracking* de los mismos. Para llevar a cabo esto, es necesario tener en cuenta las condiciones climatológicas donde se está realizando la grabación, las características de las cámaras empleadas, la calidad de las imágenes, la luminosidad con los contrastes de luz, un *dataset* con un gran número de imágenes que sirva de ayuda para realizar un mejor entrenamiento del modelo, etc. Como se puede comprobar, estos sistemas están creados para tareas muy concretas, por lo que es necesario un mayor proceso para llegar al objetivo final que se busca.

Respecto a los objetivos que se marcaron al principio del trabajo, se han cumplido con creces, realizando tareas más allá de las propuestas en un principio. En primer lugar, todo el proceso se llevó a cabo en un entorno de laboratorio ya que, era necesario crear un prototipo fácil de manejar dentro un rango en el que los dispositivos de grabación pudieran grabar sin dificultades los vídeos necesarios. Para ello, se realizó un modelo con el uso de una pelota de *ping pong*, y posteriormente se quería comprobar si el modelo era capaz de detectar objetos flotando en un recipiente pequeño. Con este último surgieron

problemas, ya que la orientación de las cámaras se cambió, además de que se necesitaría unas cámaras más potentes para poder captar los objetos a gran velocidad, ya que se perdía la detección de los mismos.

A continuación, el proceso de calibración se desarrolló de dos maneras diferentes, una de ellas con las imágenes estáticas de un objeto concreto, obteniendo una malla de puntos de posiciones 3D conocidas, y el otro proceso elaborado con un tablero de ajedrez. El único proceso que no tuvo el éxito deseado fue el del tablero de ajedrez, ya que a la hora de pasar de las coordenadas 2D de la imagen a coordenadas 3D, los cálculos con las matrices de las cámaras obtenidas no se consiguieron obtener de manera efectiva debido a la falta de sincronización total en la captura de los dos vídeos en todas las tomas realizadas.

Una vez analizados los modelos creados dentro del entorno de laboratorio, se planteó la tarea de poder implementarlo en un espacio más grande, como es la dársena de la ENM, que es un recinto idóneo en el cual los barcos y veleros salen a navegar cada día, siendo de esta manera objetos fáciles de detectar y de seguir con un modelo entrenado. Al igual que con el proceso de calibración, se realizaron dos métodos para poder obtener las distancias entre barcos, así como las posiciones de los distintos barcos. El primero de ellos se utilizaron puntos notables de la dársena, para la posterior interpolación entre ellos. El segundo método, se realizó una malla o *grid* desde un punto conocido y a partir de 4 coordenadas conocidas, se trazaron puntos de fuga con respecto al norte y al este. Ambos procesos fueron llevados a cabo de manera exitosa, pero con diferencias entre ambos. Las distancias relativas entre ambos, al comprobarse en *Google My Maps*, eran diferentes para ambos métodos, siendo el más preciso el de la malla.

Es importante recalcar que ambos procesos fueron entrenados para una zona en concreto, en este caso el recinto donde navegan las embarcaciones de la ENM. Cualquier otro barco que se encuentre fuera de esta zona, el sistema no va a ser capaz de identificarlo y de obtener sus coordenadas *GPS* de manera correcta. Para estos modelos, solo hubo una problemática en general, y fue la profundidad que se captaba en las imágenes. Una de las soluciones a este problema, como ya se comentó en apartados anteriores, es la disposición de las cámaras de modo que cada una de ellas esté configurada para cada uno de los ejes del sistema propuesto. De esta manera, se puede llegar a disminuir la percepción de la profundidad, y poder detallar de manera más correcta la zona acotada para poder llevar una mejor detección, clasificación y cálculos de distancias entre barcos.

Por último, y más allá de los objetivos propuestos, se pudo realizar una comprobación de la problemática anterior con el uso del *AIS* por parte de la lancha de instrucción. Este sistema puede servir de gran ayuda para mejorar los modelos creados, permitiendo investigar en ello y seguir con el desarrollo de los mismos.

5.2 Líneas futuras

Tras haber finalizado el desarrollo de los varios modelos llevados a cabo, y analizados los resultados y conclusiones de los mismos, el objetivo con este apartado es reflejar líneas futuras para la continuación del presente TFG. Para ello, se presentan una serie de líneas de investigación y distintas cuestiones que pueden servir de base para futuros trabajos, siguiendo con el desarrollo de los modelos y potenciar de esta manera el sistema desarrollado de una forma más exacta. De esta manera, se proponen las siguientes líneas futuras:

- Llevar más allá la clasificación de objetos en el entorno marítimo. Es decir, crear un nuevo modelo, valiéndose del ya existente, donde pueda detectar diferentes objetos presentes en un entorno tan amplio como es el agua, donde pueden verse palangres, bateas, boyas, barcos pequeños que no transmitan por *AIS*, etc. Esto supondría una ayuda a la navegación, mejorando de manera notable la seguridad de todas aquellas personas que navegan en alta mar o en zonas con un gran tránsito de barcos, como son los estrechos, rías o puertos.
- La tecnología de la IA está mejorando y desarrollándose cada año de manera exponencial. Como se aprecia en la Figura 2-9, la evolución de *YOLO* desde su creación no ha parado

durante el paso de los años. El primer modelo fue creado en 2015, y la última versión, *YOLOv8*, data del año 2023. Por ello, sería aprovechable crear un nuevo modelo de aprendizaje, sobre el mismo *dataset* empleado en este TFG, con las versiones venideras de *YOLO* en los próximos años.

- Potenciar el sistema con dispositivos de grabación acordes con la velocidad de las embarcaciones náuticas. Uno de los problemas que se ha presentado en este trabajo, ha sido la pérdida de detección de objetos en el agua debido a su velocidad. Se ha tratado de conseguir el mayor *frame* por segundo de los vídeos realizados, pero en algunos casos es inviable realizar el *tracking* sobre ellos si se encuentran muy próximos a las cámaras y se mueven a altas velocidades.
- Encontrar una adaptación de la plataforma para barcos que puedan ayudar a la investigación y desarrollo del sistema. Para ayudar esta adaptación, el AIS de cada barco beneficiaría a esta indagación, al ser un sistema que proporciona numerosas características como la posición *GPS*, rumbo y velocidad, entre diversas peculiaridades. Además, se podría explorar la integración en drones o vehículos submarinos no tripulados, aumentando de esta manera la cobertura y flexibilidad de la plataforma en vigilancia y exploración marítima.

Para concluir, las técnicas de inteligencia artificial son una de las principales herramientas para el futuro desarrollo de sistemas que sirvan para la automatización en sistemas de detección y seguimiento de objetos en el agua. A medida que la tecnología se desarrolla, los algoritmos también lo hacen, siendo estos más complejos y capaces de resolver, con el tiempo, tareas más complicadas.

6 BIBLIOGRAFÍA

- [1] R. D. InfoDefensa, «La Armada lanza un proyecto para diseñar el arsenal de las fragatas F-110 y los submarinos S-80», Infodefensa - Noticias de defensa, industria, seguridad, armamento, ejércitos y tecnología de la defensa. Accedido: 21 de enero de 2024. [En línea]. Disponible en: <https://www.infodefensa.com/texto-diario/mostrar/4197374/armada-disena-como-sera-arsenal-futuro>
- [2] «CDA Informática - CDA Informática». Accedido: 2 de enero de 2024. [En línea]. Disponible en: https://www.cdainfo.com/index.php?option=com_content&view=article&id=148:el-empleado-de-sistemas-con-perfil-polifuncional&catid=31&Itemid=125&lang=es
- [3] «A Brief History of Computer Vision (and Convolutional Neural Networks) | HackerNoon». Accedido: 2 de enero de 2024. [En línea]. Disponible en: <https://hackernoon.com/a-brief-history-of-computer-vision-and-convolutional-neural-networks-8fe8aacc79f3>
- [4] M. Departamento, «Visión artificial: qué es, cómo funciona y en qué se utiliza», www.elternativa.com. Accedido: 2 de enero de 2024. [En línea]. Disponible en: <https://www.elternativa.com/vision-artificial-ejemplos-aplicaciones/>
- [5] «¿Qué es la visión artificial? - Explicación de la IA y el aprendizaje automático de imágenes - AWS», Amazon Web Services, Inc. Accedido: 2 de enero de 2024. [En línea]. Disponible en: <https://aws.amazon.com/es/what-is/computer-vision/>
- [6] «SensIO | Aprende a utilizar la Inteligencia Artificial en tu día a día, en Castellano.» Accedido: 2 de enero de 2024. [En línea]. Disponible en: <https://sensiocoders.com/>
- [7] Aurora, «Qué es el Deep Learning y sus aplicaciones en la actualidad - ID Bootcamps», ID Digital School - Bootcamps. Accedido: 2 de enero de 2024. [En línea]. Disponible en: <https://iddigitalschool.com/bootcamps/que-es-el-deep-learning-y-sus-aplicaciones-en-la-actualidad/>
- [8] I. CORPORATIVA, «“Deep learning”: un concepto clave para llevar la inteligencia artificial al siguiente nivel», Iberdrola. Accedido: 2 de enero de 2024. [En línea]. Disponible en: <https://www.iberdrola.com/innovacion/deep-learning>
- [9] «Convolutional Neural Networks: Architectures, Types & Examples». Accedido: 2 de enero de 2024. [En línea]. Disponible en: <https://www.v7labs.com/blog/convolutional-neural-networks-guide>, <https://www.v7labs.com/blog/convolutional-neural-networks-guide>
- [10] Na8, «¿Cómo funcionan las Convolutional Neural Networks? Visión por Ordenador», Aprende Machine Learning. Accedido: 3 de enero de 2024. [En línea]. Disponible en: <https://www.aprendemachinelearning.com/como-funcionan-las-convolutional-neural-networks-vision-por-ordenador/>
- [11] «¿Qué son las redes neuronales convolucionales? | IBM». Accedido: 3 de enero de 2024. [En línea]. Disponible en: <https://www.ibm.com/es-es/topics/convolutional-neural-networks>

- [12] «CNN | Introduction to Pooling Layer», GeeksforGeeks. Accedido: 3 de enero de 2024. [En línea]. Disponible en: <https://www.geeksforgeeks.org/cnn-introduction-to-pooling-layer/>
- [13] «¿Qué son las Redes Convolucionales?», Codificando Bits. Accedido: 3 de enero de 2024. [En línea]. Disponible en: <https://www.codificandobits.com/blog/redes-convolucionales-introduccion/>
- [14] «YOLOv8 : Comprehensive Guide to State of the Art Object Detection». Accedido: 4 de enero de 2024. [En línea]. Disponible en: <https://learnopencv.com/ultralytics-yolov8/>
- [15] «Yolo Real-Time Object Detection - Neural Networks». Accedido: 4 de enero de 2024. [En línea]. Disponible en: <https://neuralnetworks.cl/yolo-real-time-object-detection/>
- [16] I. Pascual, «Etiquetado automático de implantes en radiografías dentales - IIC», Instituto de Ingeniería del Conocimiento. Accedido: 4 de enero de 2024. [En línea]. Disponible en: <https://www.iic.uam.es/lasalud/etiquetado-automatico-implantes-radiografias-dentales/>
- [17] R. R. Abril, «Detección de objetos I: YOLO • Un artículo de LMO», La Máquina Oráculo. Accedido: 4 de enero de 2024. [En línea]. Disponible en: <https://lamaquinaoraculo.com/deep-learning/deteccion-de-objetos/>
- [18] «YOLO models for Object Detection Explained [YOLOv8 Updated]». Accedido: 4 de enero de 2024. [En línea]. Disponible en: <https://encord.com/blog/yolo-object-detection-guide/>
- [19] N. Klingler, «Object Tracking in Computer Vision (2024 Guide)», viso.ai. Accedido: 4 de enero de 2024. [En línea]. Disponible en: <https://viso.ai/deep-learning/object-tracking/>
- [20] «Seguimiento de Objetos con Yolo v8 y BYTETrack - Object Tracking», Aprende Machine Learning. Accedido: 4 de enero de 2024. [En línea]. Disponible en: <https://www.aprendemachinelearning.com/seguimiento-de-objetos-con-yolo-v8-y-bytetrack-object-tracking/>
- [21] X. Li, W. Hu, C. Shen, Z. Zhang, A. Dick, y A. van den Hengel, «A Survey of Appearance Models in Visual Object Tracking». arXiv, 19 de marzo de 2013. Accedido: 4 de enero de 2024. [En línea]. Disponible en: <http://arxiv.org/abs/1303.4803>
- [22] «The Complete Guide to Object Tracking [+V7 Tutorial]». Accedido: 4 de enero de 2024. [En línea]. Disponible en: <https://www.v7labs.com/blog/object-tracking-guide>, <https://www.v7labs.com/blog/object-tracking-guide>
- [23] A. Singh, «Top 5 Object Tracking Methods», Augmented Startups. Accedido: 4 de enero de 2024. [En línea]. Disponible en: <https://medium.com/augmented-startups/top-5-object-tracking-methods-92f1643f8435>
- [24] sai venkatesh, «FairMOT — Multi-Object Tracking», Analytics Vidhya. Accedido: 4 de enero de 2024. [En línea]. Disponible en: <https://medium.com/analytics-vidhya/fairmot-multi-object-tracking-386afe930b24>
- [25] B. Le, «An Introduction to BYTETrack: Multi-Object Tracking by Associating Every Detection Box». Accedido: 4 de enero de 2024. [En línea]. Disponible en: <https://www.datature.io/blog/introduction-to-bytetrack-multi-object-tracking-by-associating-every-detection-box>
- [26] «What Is Binocular Vision (Eye Teaming)?», All About Vision. Accedido: 5 de enero de 2024. [En línea]. Disponible en: <https://www.allaboutvision.com/eye-care/eye-anatomy/what-is-binocular-vision/>
- [27] L. H. Rotation, «Left Hand Rotation: 3D en 1838», Left Hand Rotation. Accedido: 5 de enero de 2024. [En línea]. Disponible en: <https://lefthandrotation.blogspot.com/2013/12/3d-en-1838.html>
- [28] «Percepción de Profundidad: ¿qué puede alterarla?» Accedido: 18 de enero de 2024. [En línea]. Disponible en: <https://www.oftalvist.es/blog/percepcion-profundidad-definicion-alteraciones>
- [29] «Figura 2. a) Esquema del funcionamiento de una cámara de Tiempo de...», ResearchGate. Accedido: 18 de enero de 2024. [En línea]. Disponible en: https://www.researchgate.net/figure/Figura-2-a-Eschema-del-funcionamiento-de-una-camara-de-Tiempo-de-Vuelo-b-Sistema-de_fig1_265817241

- [30] Sumitkrsharma, «Stereo Vision and Depth Perception in Computer Vision», Medium. Accedido: 18 de enero de 2024. [En línea]. Disponible en: <https://medium.com/@sumit-kr-sharma/stereo-vision-and-depth-perception-in-computer-vision-8aff8dd04e7c>
- [31] «Stereo Image Rectification - NI». Accedido: 18 de enero de 2024. [En línea]. Disponible en: https://www.ni.com/docs/en-US/bundle/ni-vision-concepts-help/page/stereo_image_rectification.html
- [32] «Figure 3. Disparity map and Depth map for a concrete stereo scene.», ResearchGate. Accedido: 18 de enero de 2024. [En línea]. Disponible en: https://www.researchgate.net/figure/Disparity-map-and-Depth-map-for-a-concrete-stereo-scene_fig1_229592067
- [33] «Cómo diseñar un sistema estéreo integrado personalizado para la percepción de la profundidad». Accedido: 18 de enero de 2024. [En línea]. Disponible en: <https://www.flir.es/discover/iis/machine-vision/how-to-build-a-custom-embedded-stereo-system-for-depth-perception/>
- [34] I. P. English, «The Depth I: Stereo Calibration and Rectification», plainenglish.io/blog/the-depth-i-stereo-calibration-and-rectification-24da7b0fb1e0. Accedido: 16 de enero de 2024. [En línea]. Disponible en: <https://plainenglish.io/blog/the-depth-i-stereo-calibration-and-rectification-24da7b0fb1e0>
- [35] D. Biswas, «Stereo Camera Calibration and Depth Estimation from Stereo Images», Medium. Accedido: 16 de enero de 2024. [En línea]. Disponible en: <https://dibyendubiswas.medium.com/stereo-camera-calibration-and-depth-estimation-from-stereo-images-29d87bc702f3>
- [36] «What Is Camera Calibration? - MATLAB & Simulink - MathWorks España». Accedido: 16 de enero de 2024. [En línea]. Disponible en: <https://es.mathworks.com/help/vision/ug/camera-calibration.html>
- [37] «Industrial Applications – Allied Vision Technologies: Nerian». Accedido: 6 de enero de 2024. [En línea]. Disponible en: <https://nerian.com/applications/industrial-applications/>
- [38] «Qué es un coche autónomo - Blog OfertadeCoches.com». Accedido: 6 de enero de 2024. [En línea]. Disponible en: <https://www.ofertadecoches.com/blog/que-es-un-coche-autonomo/>
- [39] «MOBOTIC - Mobile Robotic Technologies». Accedido: 6 de enero de 2024. [En línea]. Disponible en: <https://www.mobotic.de/en/products/outdoor-localization-perception-unit>
- [40] «Fotogrametría | Ministerio de Transportes, Movilidad y Agenda Urbana». Accedido: 20 de enero de 2024. [En línea]. Disponible en: <https://www.mitma.gob.es/instituto-geografico-nacional/observacion-del-territorio/fotogrametria>
- [41] «article1380811828_Kocak et al.pdf». Accedido: 20 de enero de 2024. [En línea]. Disponible en: https://academicjournals.org/article/article1380811828_Kocak%20et%20al.pdf
- [42] T. Huntsberger, H. Aghazarian, A. Howard, y D. C. Trotz, «Stereo vision-based navigation for autonomous surface vessels», *Journal of Field Robotics*, vol. 28, n.º 1, pp. 3-18, ene. 2011, doi: 10.1002/rob.20380.
- [43] «Figure 2.5. Semi Global Matching, (left) rectified base-and match-image...», ResearchGate. Accedido: 18 de enero de 2024. [En línea]. Disponible en: https://www.researchgate.net/figure/Semi-Global-Matching-left-rectified-base-and-match-image-with-epipolar-line-and_fig4_350485549
- [44] «GitHub - JiaRenChang/PSMNet: Pyramid Stereo Matching Network (CVPR2018)». Accedido: 18 de enero de 2024. [En línea]. Disponible en: <https://github.com/JiaRenChang/PSMNet?tab=readme-ov-file>
- [45] «ELP Cámara web 1080P HD 5-50 mm, objetivo de enfoque variable, USB, compatible con 640 x 480 @ 100 fps, gran angular videoconferencia para Mac/Android/Windows/Raspberry USBFHD01M-SFV (5-50) : Amazon.es: Informática». Accedido: 18 de febrero de 2024. [En línea]. Disponible en: https://www.amazon.es/ELP-compatible-videoconferencia-Raspberry-USBFHD01M-SFV/dp/B07KLMTP65/ref=sr_1_2?__mk_es_ES=%C3%85M%C3%85%C5%BD%C3%95%C3%91&crd=1YUYB2POH2BD1&keywords=elp%2Bcamera%2B5-

50mm&qid=1708250934&s=electronics&sprefix=elp%2Bcamera%2B5-50mm%2Celectronics%2C109&sr=1-2&ufe=app_do%3Aamzn1.fos.5e544547-1f8e-4072-8c08-ed563e39fc7d&th=1

- [46] «ultralytics: Ultralytics YOLOv8 for SOTA object detection, multi-object tracking, instance segmentation, pose estimation and image classification.»
- [47] «¿Qué es Epoch en Machine Learning?», Huawei Enterprise Support Community. Accedido: 5 de marzo de 2024. [En línea]. Disponible en: <https://forum.huawei.com/enterprise/es/%C2%BFqu%C3%A9-es-epoch-en-machine-learning/thread/667232453749784577-667212895009779712>
- [48] «El impacto de la inteligencia artificial en la sociedad | Blog Caja rural de Navarra». Accedido: 9 de marzo de 2024. [En línea]. Disponible en: <https://blogempresas.cajaruraldenavarra.com/el-impacto-economico-de-las-inteligencias-artificiales>
- [49] K. M. M. dice, «Impacto medioambiental de la inteligencia artificial». Accedido: 9 de marzo de 2024. [En línea]. Disponible en: <https://ecologica.life/es/medio-ambiente/inteligencia-artificial-y-el-cambio-climatico/>
- [50] N. Salih, «El impacto medioambiental de la IA: ¿salvadora o destructora?», Eticas Foundation. Accedido: 9 de marzo de 2024. [En línea]. Disponible en: <https://eticasfoundation.org/es/el-impacto-medioambiental-de-la-ia-salvadora-o-destructora/>

ANEXO I: IMPLICACIONES SOCIALES, Y/O ECONÓMICAS, Y/O AMBIENTALES

En el actual anexo se recogen diversas implicaciones sociales, económicas y ambientales que se han llegado a deducir por el desarrollo de este trabajo, como también de las técnicas de inteligencia artificial en general.

En el ámbito social, se profundiza en mayor parte en el Anexo II: Reflexiones Éticas y Sociales, para no tener redundancia acerca de las implicaciones ya tratadas en ese anexo. Solamente destacar la importancia que puede llegar este proyecto a la hora de la seguridad de la ciudadanía como de la seguridad marítima en el uso del sistema desarrollado en buques. También hacer una observación sobre la ciberseguridad, que es crucial para que la información obtenida esté bajo custodia segura.

En cuanto al ámbito económico, la inteligencia artificial está llegando a cualquier ámbito de la vida humana. Al incidir en numerosos sectores, el impacto económico aumenta de manera notable. Muchos estudios afirman que este tipo de tecnología, en el año 2023 ha llegado a mover millones de dólares. *Roland Berger*, una firma de consultoría estratégica a nivel mundial, calcula que la inteligencia artificial puede contribuir a un crecimiento solo en Europa occidental de hasta 420.000 millones de euros para el año 2035 [48]. Además, la Comisión Europea también tiene confianza en esta tecnología, afirmando que tendrá un gran impacto de diferentes maneras a los largos de los años en ámbitos como la Fabricación e Internet Industrial de las Cosas, movilidad y la salud. De igual forma, y en relación al ámbito social, el auge en la economía gracias a la inteligencia artificial ayuda a la creación de puestos de trabajos relacionados con los datos y el comercio.

Con respecto a la plataforma desarrollada en este TFG en cuanto al aspecto económico, se ha comprobado que, con dos cámaras de bajo coste, se puede realizar un gran estudio y análisis sobre la localización y seguimiento de objetos en el agua. Actualmente, es complicado encontrar en el sector tecnológico un sistema que se encargue de corregir la problemática de la visión estereoscópica en objetos a larga distancia, por lo que se desconoce cuánto es realmente el presupuesto de un sistema tan avanzado y desarrollado como puede ser el creado a partir del modelo desarrollado en este trabajo.

Por último, en el aspecto ambiental, existen varias implicaciones que afectan al medio ambiente de manera positiva o negativa. Presentando las positivas, gracias al uso del modelo que se ha desarrollado, se puede ayudar a la localización de objetos flotantes en la mar que pueden influir de manera negativa en al ambiente marino. Al igual que en la seguridad, el sistema puede ser muy provechoso a la hora de realizar un seguimiento a los distintos buques que se encuentren navegando, viendo la trayectoria que realiza cada uno y, evitando así ocasiones entre ellos. Además, la IA ayuda al medio ambiente gracias a la monitorización, análisis y predicción de los cambios ambientales, con el objetivo de implementar soluciones innovadoras y sostenibles tras haber realizado un exhausto estudio sobre cómo puede verse afectado el medio ambiente por diversos factores.

De manera negativa, la inteligencia artificial también tiene su impacto. La huella de carbono es la principal implicación negativa de la IA, ya que esta tecnología consume mucha energía, provocando una mayor contaminación por combustibles fósiles en el aire y la atmósfera si no se hace uso de energías renovables. Para comprobar este gran impacto, la Universidad de *Massachusetts* realizó un estudio sobre los entrenamientos de grandes modelos de IA. Estos entrenamientos, según el estudio, pueden llegar a emitir más de 284.000 kg de dióxido de carbono equivalente, siendo equivalente a cinco veces las emisiones durante la vida útil de un coche [49]. Otros de los problemas del uso de esta tecnología son los centros de datos, que consumen el 2% de la energía global. Su gasto es tan elevado, que incluso en Londres se ha planteado la prohibición de construcción de viviendas hasta 2035, ya que la red eléctrica no sería capaz de hacer frente a las necesidades energéticas básicas por la alta demanda que tienen los centros de datos instalados en la capital inglesa [50].

ANEXO II: REFLEXIONES ÉTICAS Y SOCIALES

Hoy en día, la IA sigue desarrollándose de manera considerable, con grandes cambios significativos, pero planteando algunos dilemas éticos-sociales fundamentales. Es necesario tener estos dilemas presentes a la hora de desarrollar una tecnología tan importante como es la inteligencia artificial.

Las redes neuronales están ganando cada vez más peso para la toma de decisiones de alto riesgo. La conclusión sobre esto es que puede ser beneficioso o no, ya que la tecnología se está desarrollando cada vez más, creando modelos que actúan con mayor exactitud y precisión. Esto hace que cada vez sea más difícil desarrollar mejores modelos que puedan ser verificados y validados para su uso.

En el ámbito de Defensa, cada vez más son los casos donde los nuevos buques y aeronaves no tripuladas utilizan la última tecnología punta para sus sensores. Uno de los objetivos de la Armada, es la defensa de España y de las aguas nacionales. Para ello, cuenta con numerosos buques para preservar la seguridad marítima. Actualmente, uno de los dispositivos más utilizados son los drones, que gracias a sus sensores pueden identificar de manera precisa posibles atentados o ataque hacia nuestro país de barcos enemigos. Gracias a la IA, como por ejemplo con los modelos desarrollados en esta memoria, cabe la posibilidad de que estos drones pueden adquirir los contactos enemigos, haciendo un análisis de ellos y ser capaces desde el buque prever un posible ataque. Esto ayudaría a disminuir en gran medida pérdidas humanas por los ataques que puedan realizar hacia nuestros buques.

En cuanto a la información obtenida gracias al uso de esta tecnología, es primordial la gestión de la seguridad acerca de los datos capturados. La ciberseguridad juega un papel importante, debiéndose establecer medidas que eviten accesos no autorizados. En el caso de que la seguridad falle, la población perdería confianza en la tecnología, ya que podrían utilizar la información de cada persona de manera indebida. Esto puede llegar a suponer un gran problema, y actualmente se llegan a dar casos del mal uso de la inteligencia artificial.

Por otro lado, desde el punto de vista ético, deben establecerse políticas que garanticen que los derechos individuales no sean infringidos. Es necesario que se respeten las leyes y regulaciones vigentes en cuanto al almacenamiento y uso de datos personales tras la captura de imágenes, ya que puede darse un uso del modelo para fines ilegítimos.

Otra de las reflexiones éticas sobre esta plataforma es la equidad en el acceso y beneficio de la misma. La dificultad de encontrar un sistema como el desarrollado es notable, por lo que apenas se puede tener acceso a este tipo de plataformas. Es necesario que se asegure que la plataforma pueda beneficiar a la población de una manera equitativa. Hay que evitar que se produzcan brechas digitales, y garantizar que se pueda adquirir, siendo su uso muy productivo en diversos ámbitos de la vida cotidiana.

En conclusión, el análisis detallado llevado a cabo en este anexo es que el sistema creado en este TFG debe ser utilizado de manera responsable y justa, evitándose de que el modelo pueda ser empleado por organizaciones que no cumplan con las leyes y regulaciones recogidas para el uso de la información y datos personales. Hay que dar un buen uso de la inteligencia artificial, ya que es el futuro hoy en día, llegando a ser muy beneficioso con el cual todos pueden tener privilegios y sacar provecho de esta tecnología punta.

ANEXO III: GLOSARIO DE SIGLAS Y ACRÓNIMOS

AIS: Automatic Identification System

AJAL: Almirante Jefe de Apoyo Logístico

CIA2: Centro de Inteligencia Artificial

CIFAR-10: Canadian Institute for Advanced Research

CNN: Convolutional Neural Network

CUD: Centro Universitario de la Defensa

ENM: Escuela Naval Militar

FAST: Features from Accelerated Segment Test

GPS: Global Positioning System

GPU: Graphics Processing Unit

HD: High Definition

IA: Inteligencia Artificial

ICR: Intelligent Character Recognition

IoU: Intersection over Union

IP: Internet Protocol

Leaky ReLU: Leaky Rectified Linear Unit

LiDAR: Light Detection and Ranging

mAP: mean Average Precision

MNIST: Modified National Institute of Standards and Technology

MOT: Multiple Object Tracking

OCR: Optical Character Recognition

PMSNet: Pyramid Stereo Matching Network

R4SA: Robust, Real-time, Reconfigurable, Robotic system architecture

RCNN: Region-Based Convolutional Neural Network

RFID: Radio Frequency Identification

SGM: Semi-Global Matching

SORT: Simple Online and Realtime Tracking

SOT: Single Object Tracking

SURF: Speeded Up Robust Features

TFG: Trabajo de Fin de Grado

ToF: Time-of-Flight

USB: Universal Serial Bus

YOLO: You Only Look Once

FPS: Frames per second

2D: dos dimensiones

3D: tres dimensiones

ANEXO IV: SCRIPT DE OCR

```
PROJECT_NAME = 'CALIBRATION_PONG' # Name of the project
IMAGE_NAME   = 'DATETIME_OCR/CIRCLE_0_00101.jpg' # Name of the image to be
annotated

import sys, traceback, os, cv2
sys.path.append('./classes/')
from classes.project import Project
import pytesseract

if __name__ == '__main__':
    try:

print('*****')
print('***** DATETIME OCR PROJECT : ' + PROJECT_NAME
+ ' *****')

print('*****')

        projectObj = Project(name = PROJECT_NAME)
        if projectObj.configure():
            image_file_path = projectObj.getFolder('images') + IMAGE_NAME
            if os.path.isfile(image_file_path):
                print('File exist at : ' + image_file_path)
                image = cv2.imread(image_file_path, 1)
                crop_image = image[0:30,290:495,:]

                # Convert the image to grayscale
                gray_image = cv2.cvtColor(crop_image, cv2.COLOR_BGR2GRAY)

                # Apply thresholding to enhance text visibility
                _, threshold_image = cv2.threshold(gray_image, 128, 255,
cv2.THRESH_BINARY)

                # Use Tesseract to do OCR on the thresholded image
                text = pytesseract.image_to_string(threshold_image)

                print('Date time is: ', text)
                cv2.imshow('crop_image', crop_image)
                cv2.imshow('image', image)
                cv2.waitKey()

            else:
                print('File not exist at : ' + image_file_path)

        cv2.destroyAllWindows()
    except KeyboardInterrupt:
        print('KEYBOARD INTERRUPT !!! ', str(traceback.format_exc()))
        sys.exit(0)

    except:
        print('MAIN ERROR : ', str(traceback.format_exc()))
        sys.exit(0)
```