

PROYECTO FIN DE GRADO

TÍTULO: Estimación del tamaño de peces en imágenes estereoscópicas con aprendizaje automático y procesamiento de la imagen.

AUTOR: Andrés Alberto Cid Ortiz

TITULACIÓN: Ingeniería de Sonido e Imagen

TUTORA: Juana María Gutiérrez Arriola

DEPARTAMENTO: Departamento de Ingeniería Telemática y Electrónica

VºBº TUTOR/A

Miembros del Tribunal Calificador:

PRESIDENTA: Elena Blanco Martín

TUTORA: Juana María Gutiérrez Arriola

SECRETARIO: Nicolás Sáenz Lechón

Fecha de lectura: 23 de Julio de 2024

Calificación:

El Secretario/La Secretaria,

Agradecimientos

A todos aquellos que me han acompañado en este viaje, desde Cáceres, pasando por Madrid y el Chami, hasta la redacción de esta memoria.

A mis compañeros de piso, Adri, Txalo, Miguel, Tomás, Arturo, Mireia, y ahora Santi, todos con los que he convivido habéis sido un gran apoyo para que haya podido sacar esto adelante.

A la gente de la uni, especialmente a Carlitos, Sara, Isa y Patri. Sin vosotros las horas hubiesen sido más lentas y probablemente no estaría escribiendo esto.

A mi tutora, Juana, gracias por ser tan paciente y estar siempre dispuesta a ayudar en lo que he necesitado.

A mi familia, por haber seguido confiando en mí hasta el final a pesar de cualquier piedra que se haya puesto por el camino.

Y, en definitiva, gracias a cualquier persona que no estoy mencionando y haya contribuido a que llegue hasta aquí y sea quien soy hoy.

Resumen

La estimación del tamaño de los peces es un tema fundamental para los procesos productivos que tienen lugar en el campo de la acuicultura. Su relevancia radica en que, mediante este dato, se puede conocer el calibre del pienso que se le suministrará a un determinado tanque de peces.

Este trabajo parte del sistema de visión estereoscópica desarrollado por María Castillo Moral en su Trabajo Fin de Máster: *“Sistema de estimación de la biomasa en piscifactorías mediante imágenes RGB”* [1]. Este sistema consiste en segmentar y detectar la cabeza, cola y centroide de los peces, estimar la inclinación y profundidad a las que están ubicados y finalmente hallar su tamaño en base a estos parámetros.

La aportación original de este proyecto consiste en extrapolar la estimación de tamaño de objetos con forma de pez impresos en 3D, a peces reales ubicados en tanques, en concreto, rodaballos. Para ello, se ha empleado un sistema de *Machine Learning*, particularmente el modelo SAM (Segment-Anything-Model) desarrollado por Meta-AI, en su versión MobileSAM. Este modelo ha sido combinado con técnicas de procesamiento de imagen para conseguir segmentar rodaballos aislados en tanques acuícolas.

El proceso ha sido integrado mediante una serie de algoritmos en los que se detectan los rodaballos, se encuentran las correspondencias de los peces entre las dos imágenes correspondientes a cada una de las cámaras estereoscópicas, y finalmente se estima el tamaño conociendo el contorno de los peces y los parámetros de calibración de las cámaras de visión estereoscópica.

El sistema desarrollado permite visualizar la distribución del tamaño de los peces en el tanque. Para comprobar el funcionamiento del sistema se han analizado 20 imágenes, en las que los resultados obtenidos son coherentes con el tamaño real de los objetos presentes en estas.

Abstract

Estimating the size of fish is a crucial issue for the production processes in the field of aquaculture. Its relevance lies in the fact that this data allows for determining the appropriate feed size for a specific tank of fish.

This work builds on the stereoscopic vision system developed by María Castillo Moral in her master's Thesis: "Biomass Estimation System in Fish Farms Using RGB Images." This system involves segmenting and detecting the head, tail, and centroid of the fish, estimating the inclination and depth at which they are located, and finally determining their size based on these parameters.

The original contribution of this project consists of extrapolating the size estimation of 3D-printed fish-shaped objects to real fish located in tanks, specifically turbot. To achieve this, a Machine Learning system has been employed, particularly the SAM (Segment-Anything-Model) model developed by Meta-AI, in its MobileSAM version. This model has been combined with image processing techniques to segment isolated turbot in aquaculture tanks.

The process has been integrated through a series of algorithms that detect the turbot, find the correspondences of the fish between the two images from each of the stereoscopic cameras, and finally estimate the size by knowing the contour of the fish.

The developed system allows visualization of the size distribution of fish in the tank. To verify the functionality of the system, 20 images have been analyzed, and the results obtained are consistent with the actual size of the objects present in them.

Índice de figuras

Fig. 2.1. Comparativa entre la imagen original de un girasol y la imagen segmentada.	4
Fig. 2.2. Detección de objetos en una carretera. Fuente: [7]	5
Fig. 2.3. Entradas y salidas del modelo. Fuente: [8]	6
Fig. 2.4. Proceso de codificación y decodificación para la generación de máscaras. Fuente: [8]	6
Fig. 2.5. Motor de datos de SAM. Fuente: [8]	7
Fig. 2.6. Arquitectura de red de YOLO. Fuente: [12]	8
Fig. 2.7. Aplicación de la geometría epipolar en un sistema de dos cámaras rectificado.	14
Fig. 2.8. Relación entre píxeles y tamaño real.	15
Fig. 4.1. Diagrama de bloques del sistema diseñado.....	19
Fig. 4.2. Ubicación de las cámaras en un entorno de pruebas.....	19
Fig. 4.3. Cámara izquierda y derecha con su distancia marcada.	20
Fig. 4.4. Comparativa entre dos frames consecutivos con un evento relevante.	22
Fig. 4.5. Proceso de dedistorsión y rectificación de las imágenes. Fuente: [1]	23
Fig. 4.6. Comparativa entre una imagen antes y después de la corrección.	23
Fig. 4.7. Imagen de un tanque de rodaballos con todas las máscaras procesadas por SAM. Cada color indica una máscara	25
Fig. 4.8. Cuadrícula de puntos de muestreo para <i>points_per_side</i> = 64.....	27
Fig. 4.9. Reprocesamiento de la imagen sobre la segunda capa, para <i>crop_n_layer</i> = 2, <i>crop_n_points_downscale_factor</i> = 2.....	28
Fig. 4.10. Máscaras de la imagen izquierda con 32 puntos de muestreo y ninguna capa adicional.....	29
Fig. 4.11. Máscaras de la imagen izquierda con 64 puntos de muestreo y 2 capas adicionales.	29
Fig. 4.12. Caja delimitadora en formato XYWH. W y H han sido sustituidos por bx y by respectivamente. [32]	31
Fig. 4.13. Correlación cruzada bidimensional. Fuente: [34]	32
Fig. 4.14. Recorte de un rodaballo	33
Fig. 4.15. Región de búsqueda y máxima correlación señalada mediante el punto rojo	33
Fig. 4.16. Región de búsqueda y caja delimitadora del objeto hallado.	34
Fig. 4.17. Correspondencias entre la imagen izquierda y derecha.	35
Fig. 4.18. Correspondencias después del filtrado por ángulo.	36
Fig. 4.20. Antes y después de la umbralización por Otsu en una imagen de baja iluminación.	38
Fig. 4.21. Histograma de la imagen de Fig. 4.17 con umbral determinado por Otsu marcado en rojo.	39
Fig. 4.22. Recorte procesado correctamente.	40
Fig. 4.23. Imagen descartada por contorno no circular.	40
Fig. 4.24. Traslación de un par de puntos P1 y P2 a P1' y P2'.	42

Fig. 4.25. Representación gráfica de un punto trasladado y las distancias euclídeas al otro contorno.	43
Fig. 4.26. Diagrama de flujo del sistema final.....	43
Fig. 5.1. Resultado de la estimación con imagen bien iluminada.....	45
Fig. 5.2. Resultado de la estimación con alta densidad de rodaballos situados en el fondo....	46
Fig. 5.3. Resultado con imagen de buena calidad.....	46
Fig. 5.4. Resultado de la estimación con mala iluminación	47
Fig. 5.5. Comparativa entre detección de un rodaballo aislado y un grupo de rodaballos.....	48
Fig. 5.6. Comparativa entre segmentación cabeza-cola y lateral-lateral.....	49
Fig. 5.7. Segmentación de un rodaballo en agua profunda.....	49
Fig. 5.8. Estimación del tamaño del largo de un móvil.....	50
Fig. 5.9. Estimación de 30 cm a lo largo de un metro.	51
Fig. 5.10. Comparativa del tamaño de dos rodaballos con extensión similar pero situados a distintas profundidades.....	51
Fig. 5.11. Variación del error en función de la profundidad. Fuente: [1]	52
Fig. 5.12. Histograma de frecuencias relativas de los tamaños de rodaballos en intervalos de 0,5 cm correspondientes a una imagen con predominancia de rodaballos situados en la superficie.	53
Fig. 5.13. Histograma de frecuencias relativas de los tamaños de rodaballos en intervalos de 0,5 cm correspondientes a una imagen con predominancia de rodaballos situados en la profundidad del agua.....	53

Lista de acrónimos

CNN: Red Neuronal Convolutacional (Convolutional Neural Network)

GAN: Red Generativa Adversativa (Generative Adversarial Network)

JSON: JavaScript Object Notation

SAM: Segment-Anything-Model

YOLO: You Only Look Once

Índice de contenidos

Resumen	v
Abstract	vii
Índice de figuras	ix
Lista de acrónimos.....	xi
1. Introducción	1
2. Marco tecnológico	3
2.1. Visión por computador	3
2.2. Segmentación y detección de objetos en imágenes	4
2.2.1. Modelos de Deep Learning	5
2.2.2. Técnicas de procesado de imagen	9
2.3. Métodos de comparación de objetos.....	10
2.3.1. Métodos basados en características.....	10
2.3.2. Métodos Basados en Aprendizaje Profundo	11
2.3.3. Registro de imágenes	11
2.3.4. Métodos híbridos.....	12
2.4. Visión estereoscópica	12
3. Especificaciones y restricciones de diseño	17
4. Descripción de la solución propuesta	19
4.1. Comparación manual de vídeos y extracción de frames equivalentes.	21
4.2. Corrección de imágenes	22
4.3. Aplicación del modelo	24
4.4. Búsqueda de correspondencias entre objetos de distintas imágenes.....	30
4.5. Tratamiento aislado de recortes de rodaballos.....	37
4.6. Obtención de contornos	41
4.7. Obtención de puntos extremos	41
4.8. Estimación de tamaño	43
4.9. Sistema final.....	43
5. Resultados	45
5.1. Análisis Global.....	45
5.2. Validación de resultados.....	47
5.2.1. Detección y segmentación de rodaballos.....	47
5.2.2. Estimación de tamaño	50
5.2.3. Estimación de profundidad.....	51
5.3. Representación de resultados.	52
6. Presupuesto	55
7. Impacto del proyecto.....	57
8. Conclusiones	59
8.1. Conclusiones	59

8.2. Trabajos futuros.....	60
9. Referencias	61
Anexos	65

1. Introducción

En el presente Proyecto Fin de Grado, se realiza una implementación de un sistema para la estimación automática del tamaño de rodaballos en un tanque. La acuicultura es una industria en crecimiento que desempeña un papel crucial en la producción sostenible de alimentos marinos. Dentro de este contexto, conocer el tamaño de los peces en los tanques de cultivo es fundamental, ya que determina el calibre del pienso que se suministrará, afectando directamente la eficiencia del proceso de alimentación y el crecimiento de los peces. Obtener medidas precisas del tamaño de los peces es, por lo tanto, esencial para la gestión y el éxito de las operaciones acuícolas.

El uso de técnicas automáticas de visión artificial se ha convertido en una herramienta poderosa para abordar diversos problemas en la acuicultura, incluyendo la estimación del tamaño de los peces. Estas técnicas permiten una monitorización continua y precisa sin la necesidad de intervención manual, lo que ahorra tiempo y reduce el estrés en los animales. Este proceso de estimación de tamaño se realiza normalmente de manera manual, mediante métodos de contactos [1].

En este contexto, se toma como referencia el Trabajo Fin de Máster de María Castillo, quien diseñó un sistema de estimación de biomasa utilizando cámaras de visión estereoscópica [1]. Para ello se parte de una serie de vídeos obtenidos en un entorno real de tanques de rodaballos. Estas imágenes se captaron mediante cámaras de visión estereoscópica con la misma configuración y calibración que la del sistema de estimación de biomasa.

La limitación del sistema de estimación de biomasa es que está reducido a unas condiciones ideales en las que no se tienen en cuenta los problemas que pueden tener las imágenes de tanques de rodaballos reales. Esto se debe a que las imágenes con las que se trabajó están tomadas en un entorno de pruebas controlado en el que las condiciones están determinadas por el propio investigador o desarrollador.

Este proyecto se propone trasladar y adaptar dicho sistema a un entorno práctico, enfrentando las dificultades y variabilidades propias de un entorno acuícola real. Para ello, la principal tarea consiste en llevar a cabo la segmentación de rodaballos en un tanque acuícola de manera automática, puesto que, mediante el dato de la distancia máxima entre pares de puntos del contorno del pez, se puede hallar su tamaño. Otra tarea inicial era la de la sincronización de los vídeos captados por distintas cámaras de manera automática, aunque esta, como se verá en el Capítulo 3 de restricciones del diseño, ha sido limitada.

La segmentación de objetos es un problema clásico de la visión por computador, tradicionalmente abordado mediante técnicas de procesamiento de imagen y aprendizaje automático. Dada la alta densidad de rodaballos en las imágenes y la complejidad de segmentarlos con precisión, se ha optado por una combinación de ambas técnicas, desarrollada en el Capítulo 4.

Es importante considerar que el tamaño de los rodaballos en las imágenes analizadas oscila entre 2,2 y 4,5 cm, ya que estos están en la transición entre fase larvaria y alevinaje, alrededor de los 30 días de edad [2]. Este dato se utiliza en el Capítulo 5 como validación del sistema diseñado.

Este documento se estructura en cinco capítulos principales. En el primero, se realiza un análisis del marco tecnológico sobre el que se desarrolla el proyecto. El segundo capítulo detalla las especificaciones y restricciones de diseño. El tercer capítulo describe la solución propuesta para el problema. A continuación, se presentan los resultados obtenidos y su interpretación. Finalmente, se ofrecen las conclusiones y se discuten las líneas de trabajo futuras.

2. Marco tecnológico

En este capítulo se describirá el estado actual del campo de conocimiento donde están enmarcadas las técnicas empleadas en el diseño del sistema, así como una breve introducción histórica del mismo. Posteriormente se detallará el funcionamiento de estas técnicas y las principales tecnologías que se emplean para su implementación.

2.1. Visión por computador

“La Visión por Computador, o Visión Artificial, es una rama de la inteligencia artificial, cuyo objetivo es proveer del sentido de la vista a un computador o robot para que éstos puedan interactuar de forma más eficiente en ambientes complejos” [3].

Las primeras investigaciones relacionadas con la visión por computador se remontan a los años 50, con la aparición de los primeros ordenadores de propósito general, aunque debido a las limitaciones técnicas y de recursos que existían entonces, estas se descartaron.

Los primeros indicios reales de la Visión por Computador se dan en los años 60, con el origen de un prototipo automatizado basado en cámaras de visión y sistemas de procesamiento de las imágenes captadas [4]. Con esto se buscaba procesar imágenes digitalizadas para obtener una descripción matemática de los objetos incluidos en la escena. Cabe destacar en este aspecto el trabajo realizado por Lawrence Roberts para la detección de bordes en 1963 [3].

Con el desarrollo de la ingeniería informática, y la creación de procesadores más rápidos y sofisticados, el campo de la visión artificial no ha parado de crecer hasta dar lugar a sistemas capaces de capturar imágenes de forma automatizada y reproducir las características visuales de objetos o espacios a través de la interpretación de los datos que puede realizar un software destinado a esta labor [4].

Actualmente, existen diversas tecnologías que permiten abordar los problemas de la visión artificial de una manera completamente automatizada. Estas tecnologías principalmente se basan en modelos de *Deep Learning*, que es un subconjunto del *Machine Learning*. Estos modelos se basan en redes neuronales de tres o más capas, que intentan imitar el comportamiento del cerebro humano a la hora de analizar datos y extraer información sobre estos [5].

También se siguen empleando técnicas clásicas de la visión artificial, principalmente relacionadas con el procesamiento de la imagen. Estas técnicas se basan en operaciones matemáticas sobre el contenido de las imágenes para facilitar la tarea de extracción de información visual relevante.

Las aplicaciones en este campo son amplias y van desde el análisis de imágenes médicas, al reconocimiento facial o la detección de objetos. Lo que se aborda en este proyecto principalmente se relaciona con las técnicas de segmentación y detección de objetos, así como la comparación de imágenes, puesto que son las más adecuadas para llevar a cabo la tarea de

estimar el tamaño de múltiples objetos en una imagen con una alta densidad de estos mediante un sistema de visión estereoscópica.

2.2. Segmentación y detección de objetos en imágenes

La segmentación y detección de objetos son dos de las principales técnicas empleadas en el campo de la visión por computador para extraer información valiosa de los objetos presentes en la imagen.

La segmentación consiste en dividir una imagen digital en distintos grupos de píxeles (segmentos de imagen) para facilitar la detección de objetos y otras tareas relacionadas [6]. Lo que se consigue con esto es delinear objetos de manera precisa para así poder extraer información como el tamaño, el área, o el tipo de objeto del que se trata. En Fig.2.1 se representa este concepto mediante la imagen de un girasol en la que se muestra primero la versión original de la imagen y posteriormente la segmentada, donde se aprecia como las distintas formas del objeto están marcadas mediante distintas tonalidades de grises.



Fig. 2.1. Comparativa entre la imagen original de un girasol y la imagen segmentada.

Por otra parte, la detección de objetos es una tarea que no se encarga de delinear la forma de estos, sino que se reduce a señalar uno o varios tipos de objetos sobre la imagen mediante datos como las coordenadas x e y de su centro o la caja delimitadora (*bounding box*). Esta se

trata de una caja rectangular cuyo propósito es delimitar la extensión del objeto en la imagen. En Fig.2.2 se muestra la detección de distintos objetos en una imagen de una carretera en la que se pueden apreciar personas, coches, etc.

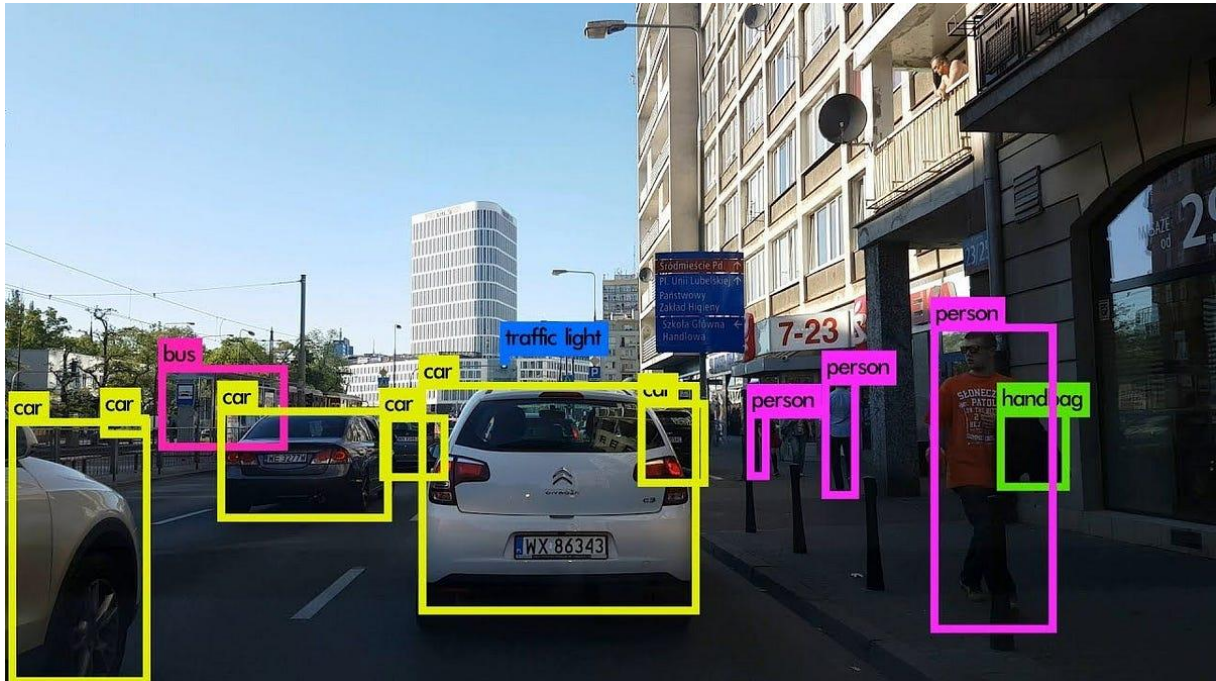


Fig. 2.2. Detección de objetos en una carretera. Fuente: [7]

2.2.1. Modelos de Deep Learning

Como se mencionó anteriormente, las tareas de la visión artificial actualmente se realizan de manera automática gracias a los modelos de Inteligencia Artificial desarrollados para este propósito. En lo que se refiere a la segmentación y detección de objetos existen diversos modelos en la actualidad, aunque hay dos que han destacado por su accesibilidad al ser sistemas de código abierto, además de por su alta precisión. Estos son, por un lado, SAM para la segmentación, y por el otro YOLO para la detección.

SAM

SAM (Segment-Anything-Model) es un modelo cuyo objetivo consiste en generar máscaras de objetos a partir de solicitudes (*prompts*), como pueden ser cajas delimitadoras, puntos en la imagen o incluso se ha llegado a probar con ciertos resultados iniciales en forma de texto [8]. Esto se ilustra en Fig. 2.3.

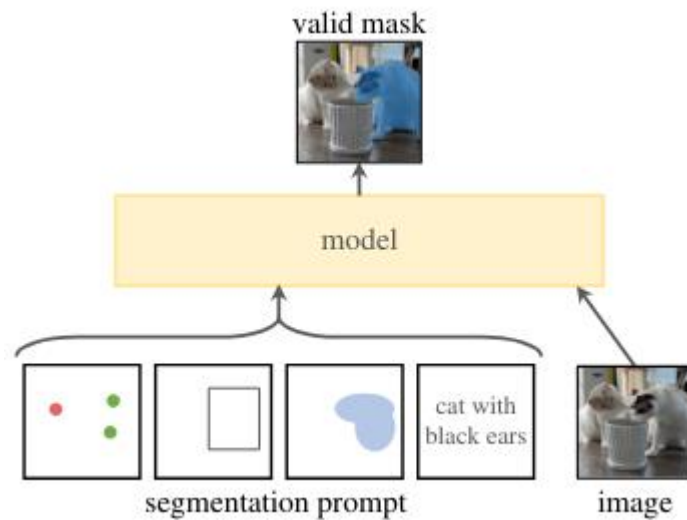


Fig. 2.3. Entradas y salidas del modelo. Fuente: [8]

La forma de conseguir esto se basa en dos codificadores y un decodificador ligero de máscaras. El primer codificador toma como entrada una imagen y la codifica en forma de vectores de características de esta y el segundo es un codificador de solicitudes. La salida de estos dos se combina en un decodificador ligero que genera las máscaras finales. En Fig. 2.4 se muestra este sistema.

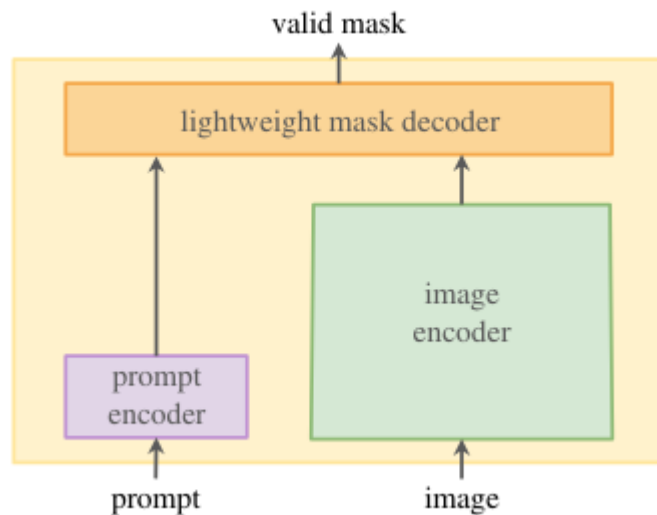


Fig. 2.4. Proceso de codificación y decodificación para la generación de máscaras. Fuente: [8]

SAM (Segment-Anything-Model) se trata de un modelo que se enmarca en los llamados modelos funcionales, que son sistemas basados en aprendizaje autosupervisado. Este tipo de aprendizaje se basa en etiquetar los datos de entrenamiento de forma autónoma [9]. Además, SAM, es un modelo pre-entrenado que no necesita de un entrenamiento posterior para ajustar el sistema a tareas de segmentación específica, lo cual se conoce como modelos *Zero-*

Shot [8]. El sistema está lo suficientemente generalizado como para abordar tareas de segmentación sobre una amplia gama de imágenes.

Para llevar a cabo esta generalización, el modelo debe haber sido entrenado previamente sobre un conjunto de datos lo suficientemente grande. El enfoque clásico para lograr esto es obtener datos en línea, pero debido a la escasez de datos en formas de máscaras, se opta por un enfoque que consiste en usar su propio motor de datos. Este motor se basa en que el modelo inicialmente tenga una mayor supervisión para generar las anotaciones, pero estas se vayan automatizando a medida que es entrenado por un conjunto de datos más amplio. El proceso consiste en el bucle que se muestra en Fig. 2.5.

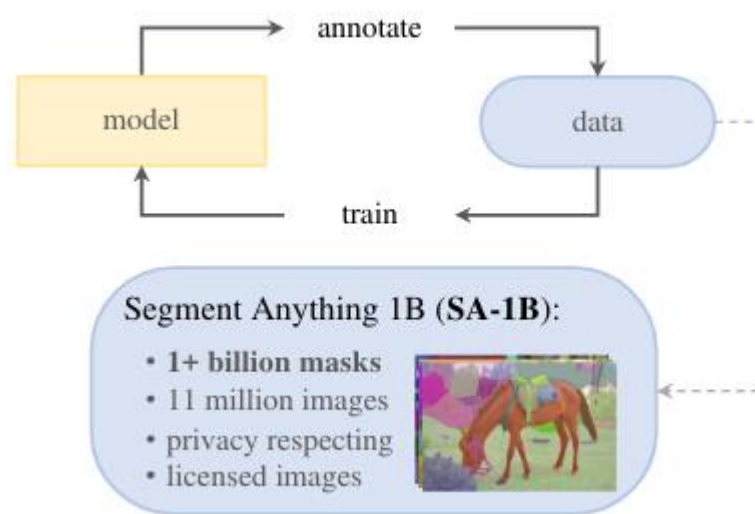


Fig. 2.5. Motor de datos de SAM. Fuente: [8]

El conjunto de datos final, SA-1B, incluye más de mil millones de máscaras extraídas de 11 millones de imágenes licenciadas y que preservan la privacidad [8]. Los resultados del modelo sugieren que SAM puede usarse directamente con ingeniería de solicitudes para resolver una variedad de tareas que involucran objetos y distribuciones de imágenes más allá de los datos de entrenamiento de SAM [8]. Por ello, SAM podría usarse perfectamente para las tareas de detección y segmentación de rodaballos en una imagen.

YOLO

YOLO, que significa "You Only Look Once" (Solo Miras Una Vez), es un modelo pre-entrenado de *Deep-Learning*, cuya finalidad es la detección de objetos en tiempo real. Se enmarca en los modelos de aprendizaje supervisado, cuya forma de aprender se basa en conjuntos de datos de entrenamiento que sirven para orientar al modelo a predecir el resultado esperado [10].

Es conocido por su velocidad y precisión, lo que lo hace altamente eficiente para diversas aplicaciones. Estas son algunas de sus principales características [11]:

- **Modelo de Detección Unificado:** A diferencia de los métodos tradicionales que aplican el modelo a una imagen en múltiples ubicaciones y escalas, YOLO aplica una única red neuronal a la imagen completa. Esta red divide la imagen en regiones y predice simultáneamente las cajas delimitadoras y las probabilidades para cada región.
- **Velocidad:** YOLO es extremadamente rápido porque plantea la detección de objetos como un único problema de regresión, directamente desde los píxeles de la imagen hasta las coordenadas de las cajas delimitadoras y las probabilidades de clase. Esto le permite procesar imágenes en tiempo real.
- **Alta Precisión:** YOLO logra una alta precisión gracias a su innovadora arquitectura de red, que equilibra eficazmente la precisión de la localización y la confianza en la clasificación. Esto reduce significativamente los errores de fondo en comparación con otros algoritmos de detección.
- **Generalización:** El modelo se generaliza bien a nuevos dominios y entradas inesperadas debido a su visión holística de la imagen durante el entrenamiento. Esto permite que YOLO detecte una variedad de objetos incluso en imágenes complejas y desordenadas.

La arquitectura de YOLO consiste en una gran red neuronal convolucional en la que las primeras capas convolucionales extraen características de la imagen y las últimas son capas totalmente conectadas que sirven para predecir probabilidades y coordenadas de la salida [12]. Para ser específicos, la red se compone de 24 capas convolucionales seguidas de 2 capas totalmente conectadas. En Fig.2.6. se muestra la arquitectura de la red.

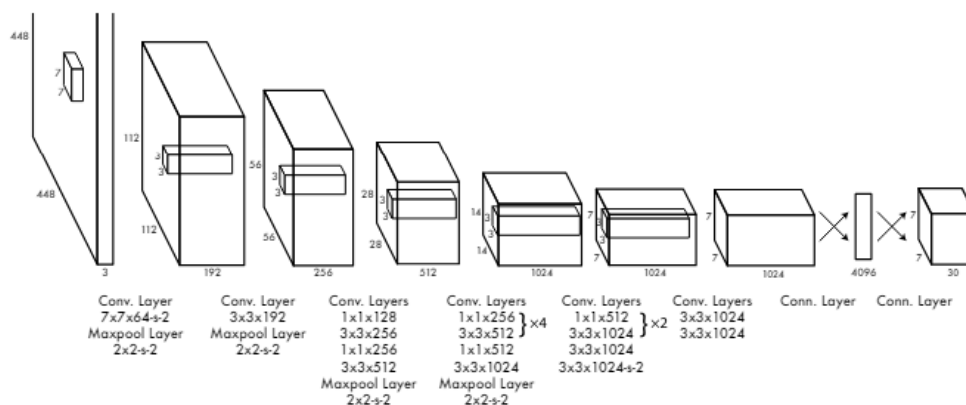


Fig. 2.6. Arquitectura de red de YOLO. Fuente: [12]

El preentrenamiento se hace en base al conjunto de datos *ImageNET*, que contienen 1000 clases de objetos [13]. Esta fase solo incluye la clasificación y se hace mediante las 20 primeras capas convolucionales. Posteriormente se añaden 4 capas más para permitir al modelo efectuar la tarea de detección [12].

El modelo no está pensado para inferir directamente las detecciones como sí era el caso de SAM, especialmente si se trata de imágenes con objetos que no se presentan de forma visualmente directa, por lo que su detección se vuelve más compleja. Pero al tratarse de un modelo de aprendizaje supervisado, puede ser entrenado para un objetivo específico, es decir, se puede realizar un *fine-tuning* [14] sobre este para adaptarlo a la detección de una clase de objeto concreto. Esto se hace utilizando como datos de entrenamiento imágenes etiquetadas del objeto en cuestión.

Por lo tanto, YOLO, podría servir perfectamente para la detección de rodaballos en una imagen, aunque posteriormente habría que realizar una segmentación de estos para poder realizar la estimación de tamaño.

2.2.2. Técnicas de procesamiento de imagen

La forma clásica de abordar la segmentación y detección de objetos es mediante técnicas de procesamiento de imagen, es decir, técnicas basadas en la aplicación de algoritmos para analizar el contenido de la imagen y modificarla según se quiera [15]. Esta metodología sigue teniendo especial relevancia en la segmentación, ya que delinear un objeto de manera precisa requiere de un coste computacional muy elevado si es llevado de forma automática sobre múltiples objetos en una imagen. La segmentación precisa es crucial en aplicaciones donde la exactitud en la delimitación de objetos es fundamental, como en el análisis médico, la vigilancia y la robótica.

Existen diversos métodos para afrontar el problema de la segmentación y detección de objetos, por lo que la técnica a aplicar depende mucho de la imagen concreta con la que se esté tratando y de sus características. Sin embargo, principalmente se pueden dividir en dos grupos [16] :

- Técnicas basadas en el histograma: Estas técnicas aprovechan la distribución de los niveles de intensidad de los píxeles de una imagen, representados en un histograma, para identificar umbrales que permitan separar diferentes objetos o regiones dentro de la imagen. El histograma de una imagen es una representación gráfica de la distribución de las intensidades de los píxeles. Los métodos basados en histogramas, como la umbralización, buscan puntos de corte óptimos que segmenten la imagen en regiones distintas. Estos métodos son particularmente útiles en imágenes con una clara diferenciación de intensidades entre objetos y fondo.
- Técnicas basadas en métodos de agrupación: También conocidos como *clustering*, estos métodos tratan de identificar grupos naturales en los datos de los píxeles según

sus características, como intensidad de color, textura o ubicación espacial. Técnicas como el *K-means*, *Mean Shift* y la agrupación jerárquica agrupan píxeles con características similares para segmentar la imagen en diferentes regiones. Estos métodos son eficaces en imágenes complejas donde los objetos no pueden ser fácilmente diferenciados solo por la intensidad del píxel y requieren una combinación de características para una segmentación precisa.

2.3. Métodos de comparación de objetos

La comparación de objetos en imágenes es una tarea fundamental en el campo de la visión por computadora, con aplicaciones que van desde la búsqueda de imágenes y la detección de duplicados hasta el reconocimiento de objetos y la identificación de patrones. Esta tarea implica evaluar la similitud o las diferencias entre dos o más objetos presentes en imágenes, utilizando diversas técnicas. Las principales técnicas de comparación de objetos en imágenes pueden agruparse en métodos basados en características, métodos de registro de imágenes, métodos basados en aprendizaje profundo, y métodos híbridos.

2.3.1. Métodos basados en características

Los métodos basados en características se centran en la extracción y comparación de características distintivas de los objetos presentes en las imágenes. Estas características pueden ser simples, como bordes y contornos, o más complejas, como descriptores locales.

Descriptores Locales:

- SIFT (Scale-Invariant Feature Transform): Extrae características locales que son invariantes a la escala, la rotación y la iluminación. SIFT es muy eficaz para encontrar correspondencias entre diferentes imágenes del mismo objeto [17].
- SURF (Speeded-Up Robust Features): Similar a SIFT, pero optimizado para ser más rápido. SURF utiliza una aproximación de las características de SIFT, reduciendo el tiempo de cálculo mientras mantiene una alta precisión [18].
- ORB (Oriented FAST and Rotated BRIEF): Combina las ventajas de FAST (Features from Accelerated Segment Test) y BRIEF (Binary Robust Independent Elementary Features) para proporcionar una detección y descripción rápida y eficiente [19].

Características Globales:

- Histograma de Gradientes Orientados (HOG): Captura la distribución de las intensidades de los gradientes de una imagen. Es especialmente útil en el reconocimiento de formas y la detección de objetos [20].
- Momentos de Hu: Se basan en la teoría de momentos y son invariantes a la transformación de escala, rotación y traslación. Se utilizan para la comparación de formas [21].

2.3.2. Métodos Basados en Aprendizaje Profundo

Con el avance de las redes neuronales profundas, los métodos basados en aprendizaje profundo han revolucionado la comparación de objetos en imágenes. Estas técnicas aprenden a extraer características y a comparar objetos directamente a partir de los datos de entrenamiento, ofreciendo una mayor precisión y robustez.

Redes Neuronales Convolucionales (CNN):

- Modelos preentrenados: Redes como VGG, ResNet e Inception, preentrenadas en grandes conjuntos de datos como ImageNet, se utilizan para extraer características profundas de las imágenes. Estas características pueden ser comparadas usando métricas de distancia como la distancia euclídea o del coseno [22].
- Siamese networks: Consisten en dos subredes idénticas que comparten pesos y se entrenan para aprender una métrica de similitud entre pares de imágenes. Son eficaces para tareas de verificación y comparación [23].
- Triplet networks: Similar a las redes siamesas, pero utilizan tres imágenes (ancla, positiva y negativa) para aprender una representación que maximice la distancia entre el ancla y la negativa, y minimice la distancia entre el ancla y la positiva [24].

Redes Generativas Adversativas (GANs):

Pueden ser utilizadas para generar muestras de imágenes y comparar objetos generados con objetos reales, evaluando la similitud entre ellos [25].

2.3.3. Registro de imágenes

El registro de imágenes es una técnica utilizada para alinear dos o más imágenes del mismo escenario tomadas en diferentes momentos, desde distintos ángulos o por diferentes sensores. Es fundamental en aplicaciones como la fusión de imágenes médicas, la creación de panoramas y la visión estereoscópica.

- Métodos basados en características: Identifican y emparejan características clave en diferentes imágenes. Por ejemplo, SIFT y SURF pueden ser utilizados para detectar y emparejar puntos de interés en las imágenes a registrar [17], [18].
- Métodos basados en intensidad: Comparan las intensidades de los píxeles entre las imágenes para encontrar la mejor alineación. La correlación cruzada normalizada es un método común en esta categoría [26].
- Métodos basados en transformaciones: Utilizan transformaciones geométricas para alinear las imágenes. Estas transformaciones pueden ser lineales (traslación, rotación, escalado) o no lineales (deformaciones más complejas).

2.3.4. Métodos híbridos

Los métodos híbridos combinan técnicas tradicionales basadas en características con técnicas de aprendizaje profundo para aprovechar lo mejor de ambos enfoques.

- Combinación de Descriptores Locales y Características Profundas: Utilizar descriptores locales como SIFT o SURF en combinación con características extraídas de una CNN preentrenada puede mejorar la robustez y la precisión de la comparación.
- Combinación de registro de imágenes y CNNs de detección: Utilizar por un lado redes neuronales preentrenadas para detectar objetos y por el otro, registro de imágenes, puede mejorar la precisión, robustez y velocidad del sistema al realizarse la comparación una vez que se conocen ya los objetos y su posición en la imagen.

2.4. Visión estereoscópica

Un sistema de visión estereoscópica para la estimación del tamaño de objetos emplea tecnología que imita la percepción tridimensional humana para medir las dimensiones de objetos en un entorno. El marco tecnológico se basa en el empleado para el diseño del sistema de estimación de biomasa [1], el cual se sustenta sobre dos elementos:

- Par de Cámaras: Dos cámaras montadas en paralelo y separadas por una distancia fija, conocida como la base estereoscópica. Esta separación es crucial para generar las dos perspectivas necesarias para la estereoscopia.
- Calibración: Las cámaras deben estar calibradas para conocer su orientación y posición exactas. La calibración asegura que las imágenes capturadas por ambas cámaras estén alineadas correctamente.

De esta manera, se generan dos imágenes (izquierda y derecha) sobre las que al aplicar geometría epipolar se puede estimar la profundidad de los objetos presentes en ellas y que así el tamaño medido sea acorde a la realidad.

Para estimar la profundidad Z utilizando los parámetros de un sistema de dos cámaras rectificado, se sigue el principio de triangulación estereoscópica. Aquí está el proceso detallado [27]:

- Captura de Imágenes: Se capturan imágenes de la escena desde dos cámaras separadas por una distancia conocida B (baseline).
- Rectificación de imágenes: Puesto que las imágenes con las que se trabaja tienen una distorsión introducida por las lentes, estas deben ser rectificadas.
- Correspondencia de Puntos: Se identifica un mismo punto físico en ambas imágenes, determinando sus coordenadas p_l y p_r en píxeles, que representan las posiciones horizontales del punto en la imagen izquierda y derecha, respectivamente:
- Disparidad: Se calcula la disparidad d , que es la diferencia en la posición del punto en las dos imágenes: $d = p_l - p_r$.
- Cálculo de Profundidad: La profundidad Z del punto se estima utilizando la fórmula:

$$Z = \frac{f \cdot B}{d} \quad (2.1)$$

Estos parámetros se deducen de una sencilla relación de triángulos semejantes [1]. En la Fig. 2.7 se observa que un sistema de dos cámaras rectificadas tiene cuatro parámetros clave: f , que indica la distancia focal de las cámaras; X_r y X_l , que representan el número de píxeles entre el centro óptico de la imagen y la posición de un mismo punto físico en ambas imágenes, a partir de los cuales se pueden obtener p_l y p_r , que son las coordenadas del punto físico en cada imagen; B , que es la distancia física entre las cámaras; y Z , que es la profundidad a la que se encuentra el punto.

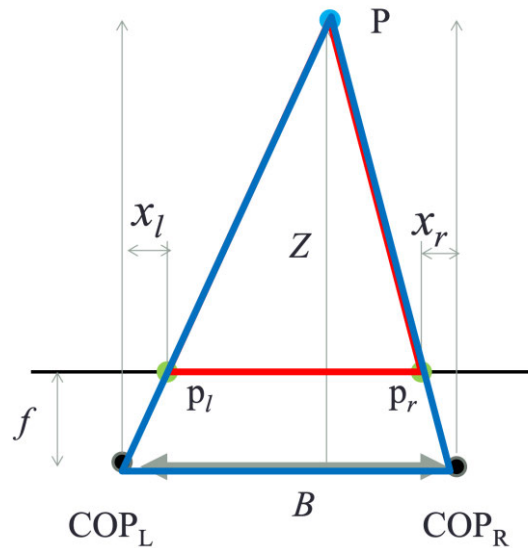


Fig. 2.7. Aplicación de la geometría epipolar en un sistema de dos cámaras rectificado.

Fuente: [1]

Una vez determinada la profundidad de los puntos de interés, se puede estimar el tamaño de los objetos. La profundidad de un objeto en una imagen afecta al número de píxeles que ocupa. Por lo tanto, es necesario convertir el número de píxeles obtenidos a una unidad de medida real, como metros, que refleje su tamaño verdadero [1].

Esta conversión se puede hacer fácilmente utilizando una relación de triángulos semejantes. Como se muestra en Fig. 2.8, el tamaño real del objeto y la profundidad tienen una relación directa con el tamaño del objeto en la imagen y la distancia focal, cuya expresión es:

$$\frac{R}{d} = \frac{r}{f} \quad (2.2)$$

Donde R es el tamaño real del objeto, r es el tamaño del objeto en la imagen, d es la profundidad, y f es la distancia focal.

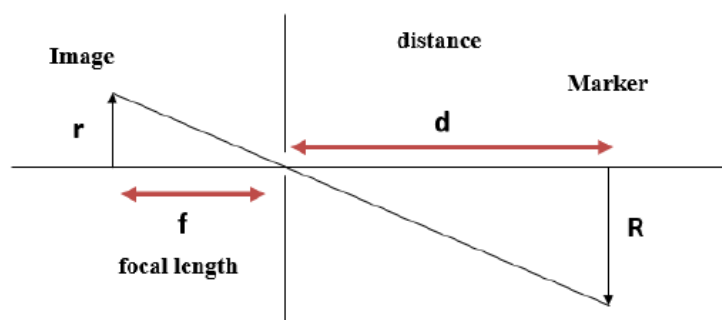


Fig. 2.8. Relación entre píxeles y tamaño real.

3. Especificaciones y restricciones de diseño

El objetivo principal del proyecto es estimar el tamaño de rodaballos en imágenes mediante el sistema de visión estereoscópica desarrollado por María Castillo [1]. Dado que este sistema ya ha sido diseñado, no se modificará, salvo por un módulo que presentaba algunos problemas, como se describe en el apartado 4.6.

Especificaciones:

- **Sincronización de frames:** puesto que los datos iniciales son vídeos de los cuales se deben extraer frames idénticos, la primera especificación consiste en sincronizar estos frames. La sincronización de vídeos se ha tenido que hacer de manera manual debido a la falta de objetos característicos en las imágenes que sirviesen para obtener descriptores de características mediante algoritmos como SIFT. Además, incluso si se encontraran dos frames con una alta similitud basada en un parámetro cuantificable, esto no aseguraría necesariamente que se trate de frames correspondientes.
- **Correspondencias entre objetos:** ya que para que el sistema de visión estereoscópica pueda funcionar, se debe estar referenciando al mismo objeto en ambas imágenes, es necesario buscar las correspondencias del objeto en la otra imagen.
- **Segmentación de objetos:** el sistema de visión estereoscópica requiere que los objetos estén debidamente segmentados para poder estimar su tamaño. Por lo tanto, la tercera especificación crucial es segmentar los objetos correctamente en las imágenes.

Restricciones:

Las principales restricciones se basan en las características de las imágenes de los tanques de rodaballos, así como los conjuntos de datos etiquetados de partida. Estas son:

- Alta densidad de objetos en la imagen.
- Exceso o defecto de iluminación según la zona y la imagen con la que se está trabajando.
- Escasos conjuntos de datos etiquetados con los que poder entrenar un modelo de aprendizaje supervisado.
- Inexistencia de datos conocidos sobre el tamaño real de los peces en el tanque.

Estas restricciones afectan directamente a la solución adoptada ya que hacen que las tareas a realizar se vuelvan más complejas. Además, el último punto afecta al hecho de que el sistema sea validable de una forma empírica.

También existen otras restricciones relacionadas con el entorno de desarrollo, el lenguaje de programación, y el entorno de ejecución:

- El proyecto utiliza únicamente software y librerías de código abierto.
- La implementación se realiza exclusivamente mediante el lenguaje de programación *Python*.
- El proyecto se desarrolla plenamente en el IDE *Visual Studio* y el editor de código *Visual Studio Code*.

Cabe destacar, que el objetivo principal del proyecto es la estimación de tamaño de una manera automática, y para este propósito lo imprescindible es que la detección, la segmentación y las correspondencias entre objetos que sean rodaballos se realice de manera automática, por lo que es en lo que este trabajo se centra.

4. Descripción de la solución propuesta

En este apartado se describe el sistema diseñado, así como la elección de las técnicas empleadas para desarrollarlo. En Fig. 4.1 se muestra un diagrama de bloques del sistema. A continuación, se describen las distintas partes que lo componen.

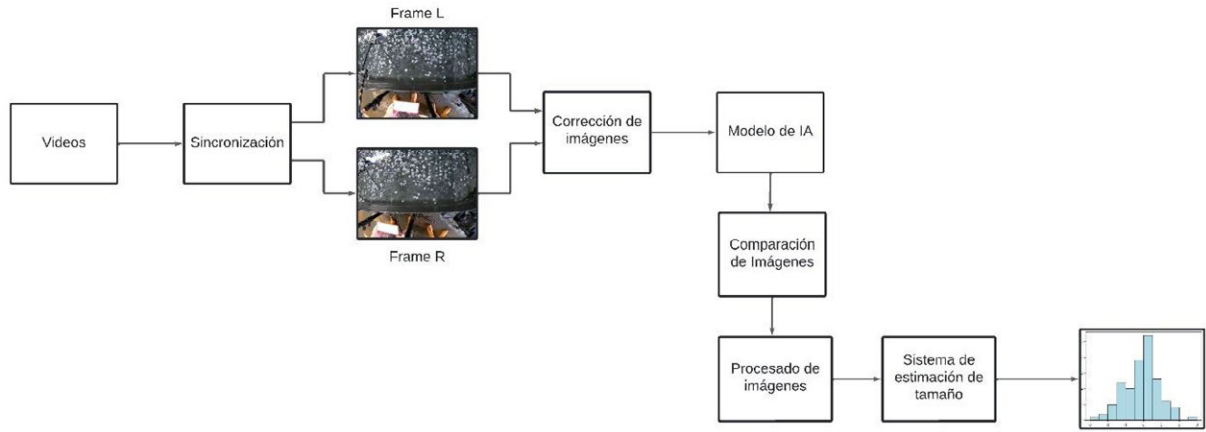


Fig. 4.1. Diagrama de bloques del sistema diseñado

Los datos de partida son 28 vídeos capturados mediante un sistema de visión estereoscópica donde cada uno ha sido tomado por dos cámaras: izquierda (L) y derecha (R). Tal y como se puede observar en Fig. 4.2, están ubicadas por encima del tanque.

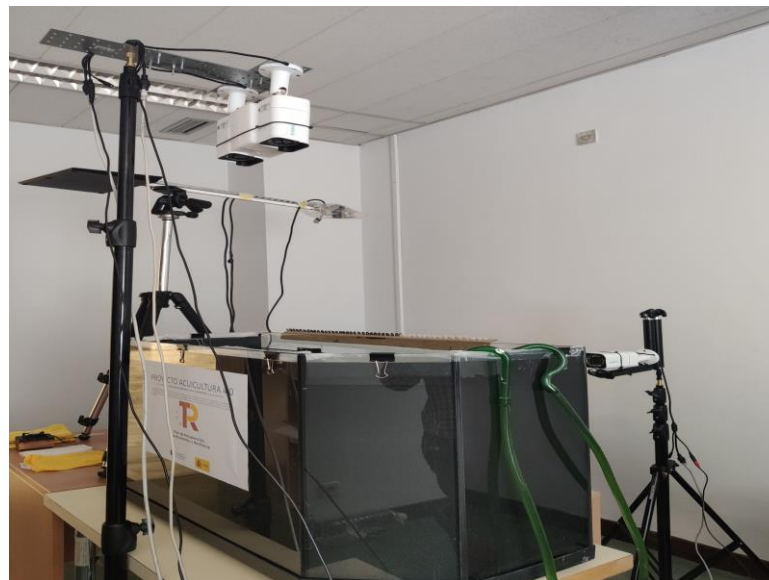


Fig. 4.2. Ubicación de las cámaras en un entorno de pruebas

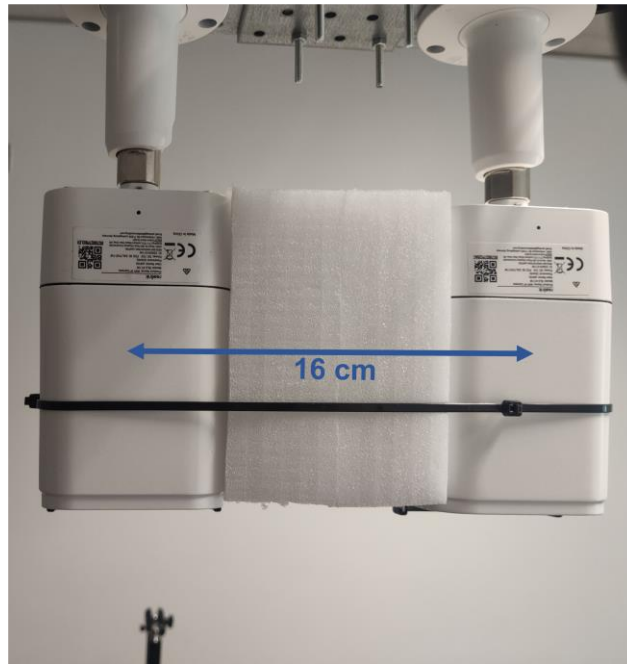


Fig. 4.3. Cámara izquierda y derecha con su distancia marcada.

Los vídeos tienen una duración de 5 minutos. Aunque cada cámara tiene la misma tasa de fotogramas y captura los mismos instantes, los vídeos correspondientes a cada una de ellas no están alineados temporalmente. Por ello, es necesario llevar a cabo una sincronización en primer lugar.

La sincronización se realiza comparando visualmente los fotogramas de los vídeos correspondientes a cada una de las cámaras. El resultado de este proceso es la obtención de los fotogramas L y R que están capturados en instantes de tiempo idénticos.

Las lentes de las cámaras presentan una distorsión, por lo que primero hay que corregirlas. Esta corrección se realiza en base a los parámetros de calibración de las cámaras, que son los mismos que los utilizados en el desarrollo del sistema de estimación de biomasa [1].

Posteriormente, se aplica un modelo de aprendizaje automático para detectar y segmentar los objetos presentes en las imágenes. La previsión incierta de conjuntos de datos para entrenar un modelo de aprendizaje supervisado como YOLO, sumado al entrenamiento intensivo que hubiera requerido para que ofreciera buenos resultados, hizo que se optara por usar el modelo de aprendizaje autosupervisado SAM. Este no es el único motivo de la elección, puesto que las imágenes de los tanques de rodaballos presentan una alta densidad de peces. Esto hace que delimitarlos correctamente mediante un modelo que se basa principalmente en la inferencia a través de la comparación con objetos de la misma clase representara una tarea compleja. La precisión que ofrece SAM, al estar fundamentado exclusivamente en las características intrínsecas de la imagen, resultó ser el motivo principal de la elección. Para reducir el coste computacional, se optó por procesar solo una imagen, en este caso, el fotograma L.

Puesto que es necesario conocer qué objetos del fotograma R se corresponden con los del fotograma L, y las imágenes no están perfectamente alineadas geométricamente, estas son comparadas mediante un algoritmo basado en la correlación cruzada para hallar las correspondencias.

El único problema que presenta SAM es precisamente que se basa únicamente en las características de los píxeles, por lo que no puede abstraer la forma de un objeto en el proceso de inferencia. Esto tiene como consecuencia que el objeto segmentado no se corresponda necesariamente con un rodaballo aislado. Como resultado, se tuvo que diseñar una fase de postprocesado para poder determinar en qué casos se presentaba un rodaballo aislado y en cuáles no.

Finalmente, se realiza la estimación de tamaño en base al sistema desarrollado por María Castillo [1] y se presentan los resultados en forma de histograma donde aparece reflejada la distribución por tamaños de los rodaballos presentes en las imágenes.

4.1. Comparación manual de vídeos y extracción de frames equivalentes.

Debido a la alta densidad de objetos de las imágenes con las que se trata, tal como se ha comentado en las restricciones de diseño, la sincronización de vídeos de forma automática se vuelve una tarea demasiado compleja. Incluso si se consiguiera obtener dos frames con alta similitud en base a los algoritmos que existen para este propósito, no se podría asegurar que estos se correspondan con instantes de tiempo idénticos.

Por lo tanto, la metodología empleada en este caso se basa en la comparación manual de frames. Lo que se hace es buscar dos frames consecutivos correspondientes a una de las cámaras en los que ocurra a lo largo de estos un evento notorio como un movimiento brusco o un cambio de iluminación. En la Fig. 4.4 se muestra una comparativa entre dos frames consecutivos captados por la cámara izquierda en el que en el primero el tanque aparece iluminado por un foco, y en el siguiente este se apaga.

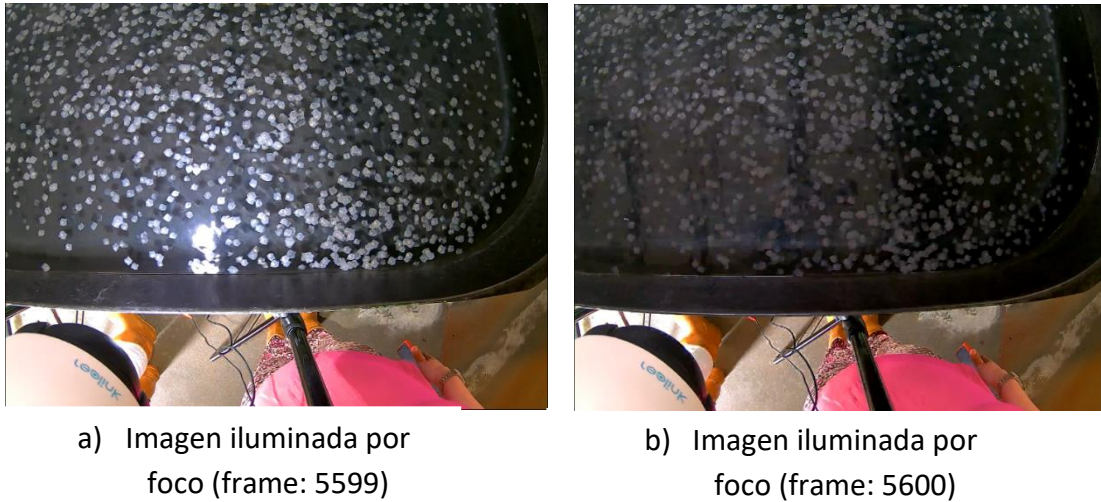


Fig. 4.4. Comparativa entre dos frames consecutivos con un evento relevante.

Las cámaras capturan el vídeo con la misma tasa de frames, de 30 fps, y el desfase entre estas no es asíncrono, por lo que los eventos ocurren a lo largo del mismo número de frames para ambas. Esto permite sincronizar los vídeos identificando los fotogramas correspondientes al mismo evento en ambas cámaras.

Una vez identificado un evento relevante idéntico en las dos cámaras se pueden obtener los frames equivalentes y por lo tanto la diferencia temporal entre ambos. Conocida esta diferencia, se pueden extraer frames equivalentes entre los dos vídeos aplicándola según si se toma como referencia la cámara izquierda o derecha.

4.2. Corrección de imágenes

Las cámaras utilizadas para el sistema de visión estereoscópica son las RLC-511W [1], que no tienen implementado ningún tipo de corrección de la distorsión generada por las lentes, además de que hay que llevar a cabo un proceso de rectificación de estas en caso de que no estén perfectamente alineadas [1]. En Fig. 4.5 se muestra cómo sería este proceso.

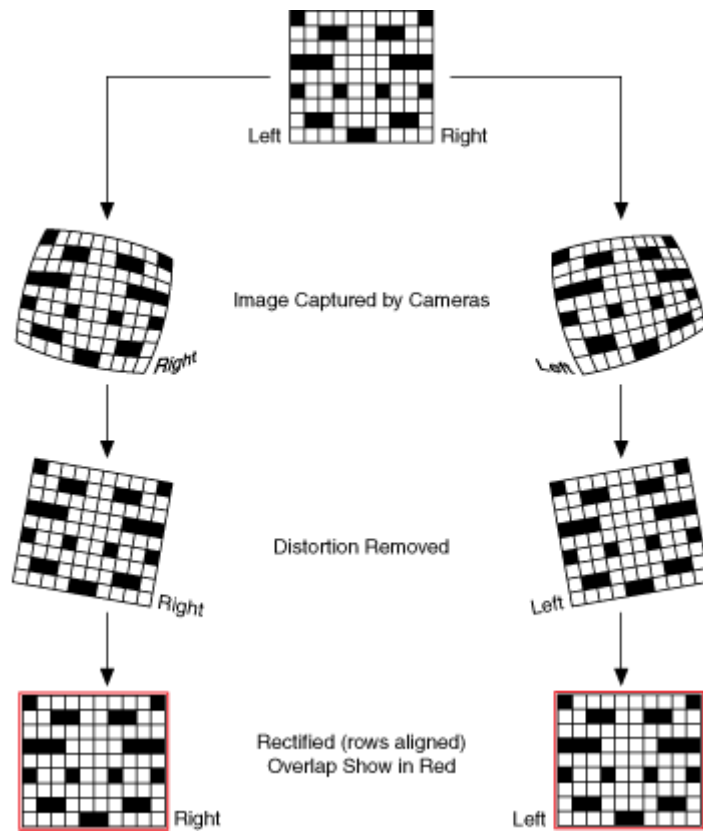


Fig. 4.5. Proceso de dedistorsión y rectificación de las imágenes. Fuente: [1]

Las cámaras estereoscópicas empleadas en el entorno de prueba donde se desarrolló el trabajo de María Castillo tienen los mismos parámetros de calibración que las empleadas para grabar los vídeos del entorno real, por lo que el proceso de corrección de estas se ha llevado a cabo mediante el módulo de corrección de imágenes de este trabajo [1].

En Fig. 4.6 se muestra una comparativa entre una imagen antes y después de la corrección.



a) Antes de la corrección



b) Después de la corrección

Fig. 4.6. Comparativa entre una imagen antes y después de la corrección.

4.3. Aplicación del modelo

Para poder aplicar el modelo SAM se leyó la información contenida en su artículo principal [8] y las instrucciones del modelo presentes en el fichero README.md del repositorio donde está subido el modelo [28].

Puesto que el modelo principal de SAM tiene unas exigencias técnicas muy elevadas que tan solo un ordenador con una gran cantidad de memoria dedicada a GPU podría procesar en un tiempo aceptable, se optó por emplear una versión con menor uso de recursos. Esta versión es MobileSAM, y las diferencias con el modelo original es que fue entrenado con un menor conjunto de datos (100 000 imágenes frente a 1 millón), y que su codificador de imágenes emplea menos parámetros. En la Tabla 1 se muestra la comparativa de los parámetros y velocidad de ambos modelos para el proceso de inferencia completo en una sola GPU.

Tabla 1: Comparativa entre SAM y MobileSAM. Fuente: [29]

Whole Pipeline (Enc+Dec)	Original SAM	MobileSAM
Parameters	615M	9.66M
Speed	456ms	12ms

Los resultados obtenidos con esta versión han sido positivos por lo que es la que ha sido finalmente seleccionada.

El modelo MobileSAM está escrito en el lenguaje de programación *Python*, y presenta un script principal llamado ***amg.py*** [30] que actúa como un generador de máscaras automático. Este script tiene asociado el notebook ***automatic_mask_generator_example.ipynb*** como ejemplo de uso [30]. A partir de este notebook se ha generado un notebook propio (***automatic_mask_generator_turbots.ipynb***) para ser ejecutado en Google Colab y un script para ser ejecutado en local (***automatic_mask_generator_turbots.py***).

El script fue principalmente probado en los ordenadores del laboratorio del Grupo de Aplicaciones Multimedia y Acústica (GAMMA) en el Centro de Investigación en Tecnologías Software y Sistemas Multimedia (CITSEM). Estos ordenadores tenían 6 GB de memoria dedicados a la GPU, y con esto era suficiente para el propósito del proyecto. Aun así, en general el entorno de ejecución en la nube prestado por Google Colab presenta un menor tiempo de ejecución frente a la ejecución local en los ordenadores mencionados, por lo que

las pruebas finales y la recopilación de resultados se han hecho sobre este entorno, aunque el desarrollo se llevase a cabo en su mayor parte en el laboratorio.

CUDA (Compute Unified Device Architecture) es una plataforma de computación paralela y un modelo de programación creado por NVIDIA. Permite a los desarrolladores aprovechar la potencia de procesamiento masivo de las GPU (Unidades de Procesamiento Gráfico) para realizar cálculos complejos de manera mucho más eficiente en comparación con las CPU tradicionales [31]. Las GPUs son particularmente adecuadas para tareas que requieren operaciones matemáticas intensivas y de procesamiento paralelo, como las que se encuentran en la inteligencia artificial y el aprendizaje profundo.

En el contexto del modelo SAM (Segment-Anything-Model), que realiza una segmentación extensiva de objetos dentro de una imagen, el uso de CUDA es crucial. Este modelo necesita procesar grandes cantidades de datos y realizar cálculos intensivos para identificar y delinear objetos con precisión. La capacidad de paralelización de las GPUs, habilitada por CUDA, permite acelerar significativamente estos cálculos, reduciendo el tiempo de procesamiento y mejorando la eficiencia general del modelo. Sin CUDA, el procesamiento de imágenes para segmentar objetos como los rodaballos en un tanque sería mucho más lento y menos eficiente, por lo que se decidió usar para todas las pruebas realizadas con SAM.

4.3.1. Configuración del modelo y obtención de parámetros.

Debido a que SAM segmenta cualquier objeto que tenga unas características que lo puedan delimitar de otro, hay que realizar un filtrado de los objetos que se correspondan con rodaballos y no con otros elementos de la imagen. En Fig. 4.7 se muestran las máscaras de salida del modelo sin realizar ningún tipo de filtrado.



Fig. 4.7. Imagen de un tanque de rodaballos con todas las máscaras procesadas por SAM.
Cada color indica una máscara

Las máscaras son la salida del modelo, y se han impuesto 2 restricciones relativas a sus características para filtrarlas: área y porcentaje de solapamiento. En Fig. 4.7 se puede observar cómo aparece segmentado cualquier objeto que capta el modelo, independientemente de su área o de que su máscara esté solapada con la de otro objeto, lo cual hace necesario este filtrado para que los únicos objetos segmentados se correspondan con rodaballos.

Las máscaras se estructuran en forma de lista de diccionarios de *python*, en el que cada uno de ellos representa una máscara cuyos elementos se muestran a continuación:

- ***segmentation***: es la máscara binaria que representa mediante '1s' los píxeles de la imagen que están contenidos en ella.
- ***area***: el área de la máscara en píxeles.
- ***bbox***: la caja delimitadora de la máscara.
- ***predicted_iou***: predicción del modelo de la calidad de la máscara.
- ***point_coords***: el punto de muestreo que generó la máscara.
- ***stability_score***: medida adicional de la calidad de la máscara.
- ***crop_box***: recorte de la imagen donde se generó la máscara.

De estos elementos, los únicos con los que se ha tratado son *segmentation*, *area* y *bbox*. Los dos primeros para realizar los filtrados comentados anteriormente, y la *bounding box* para poder delimitar la sección que ocupan los peces para su posterior estimación de tamaño.

Mediante un sondeo de los rodaballos de mayor y menor tamaño en la imagen, se estimó que estos no suelen superar el área de los 4500 píxeles, por lo que se seleccionó el valor de 4800 como restricción del área máxima de las máscaras, así como de 500 para el área mínima.

El número de píxeles solapados se calcula en base la operación lógica *and* entre el elemento *segmentation* de cada una de las máscaras. De esta manera se puede contar la cantidad de píxeles coincidentes que están puestos a 1 en ambas máscaras binarias. El porcentaje de solapamiento se calcula según la siguiente expresión:

$$\text{Solapamiento}(\%) = \frac{\text{Nº de píxeles solapados}}{\text{Área total en píxeles}} \cdot 100 \quad (4.1)$$

Puesto que de lo que se trata es de encontrar rodaballos aislados en la imagen, se ha seleccionado un porcentaje de solapamiento del 20% como umbral para descartar las máscaras. De aquellas máscaras que estén solapadas por encima de este umbral, solo se seleccionan aquellas que tengan un área menor, ya que son las que se presupone que se corresponderán con el pez aislado.

El generador de máscaras automático presenta una serie de parámetros ajustables que permiten establecer la cantidad de recursos que se van a emplear para inferir las máscaras, así como la calidad de estas. Estos parámetros son:

- ***points_per_side***: puntos de muestreo por lado de la imagen, especifica el tamaño de la cuadrícula con la que se va a muestrear la imagen.
- ***pred_iou_thresh***: umbral de confianza de la máscara.
- ***stability_score_thresh***: umbral de estabilidad de la máscara
- ***crop_n_layers***: número de capas sobre las que se procesará el modelo. Cada capa define el número de recortes ($2 * n$) de la imagen sobre los que se volverá a procesar el modelo.
- ***crop_n_points_downscale_factor***: factor de escala que determina el número de puntos de muestreo por lado que se reducen en función del número de recorte ($crop_n_points_downscale_factor * n$).
- ***min_mask_region_area***: área mínima de la máscara en píxeles.

Los principales parámetros que determinan el tiempo de ejecución y la precisión del modelo son *points_per_side* y *crop_n_layers*, ya que estos son los que determinan principalmente la cantidad (y el número de veces) que los píxeles serán procesados. Por ello, estos dos parámetros han sido los únicos que han sido modificados. Del resto se han dejado sus valores por defecto.

Por un lado, en Fig. 4.8 se representa el muestreo inicial (capa 0) que se hace sobre la imagen en caso de utilizar un valor de 64 en *points_per_side*, y por el otro, en Fig. 4.9, se representa el muestreo que se realizaría sobre una tercera capa (capa 2) en caso de que se utilice un valor de 2 para *crop_n_layers* y el mismo valor de *points_per_side*.

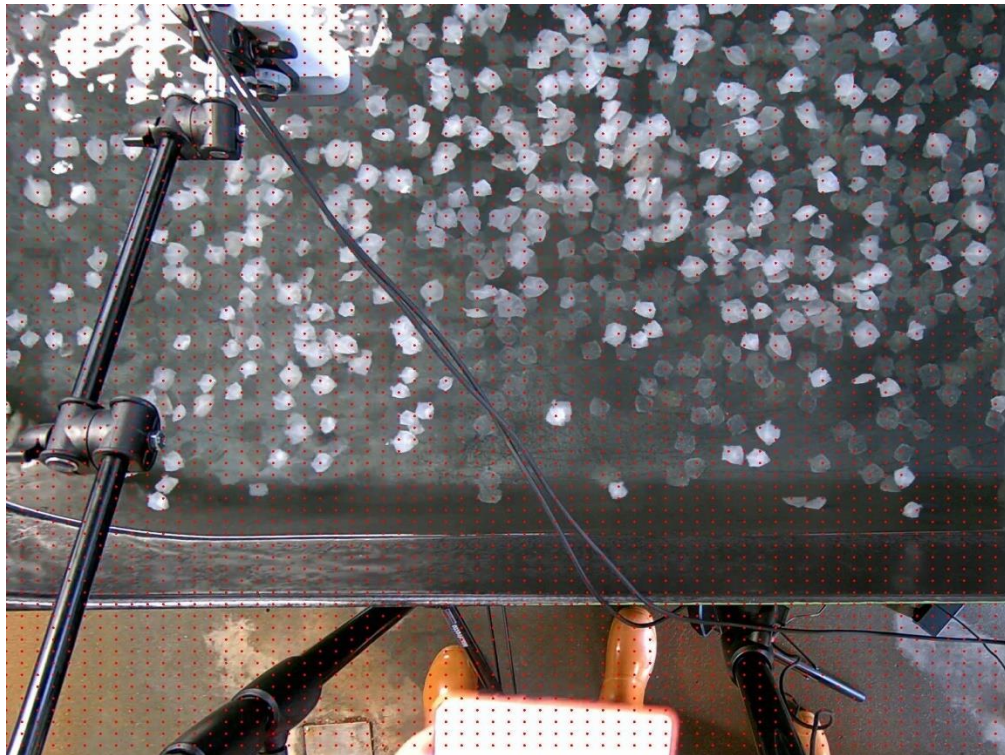


Fig. 4.8. Cuadrícula de puntos de muestreo para *points_per_side* = 64.

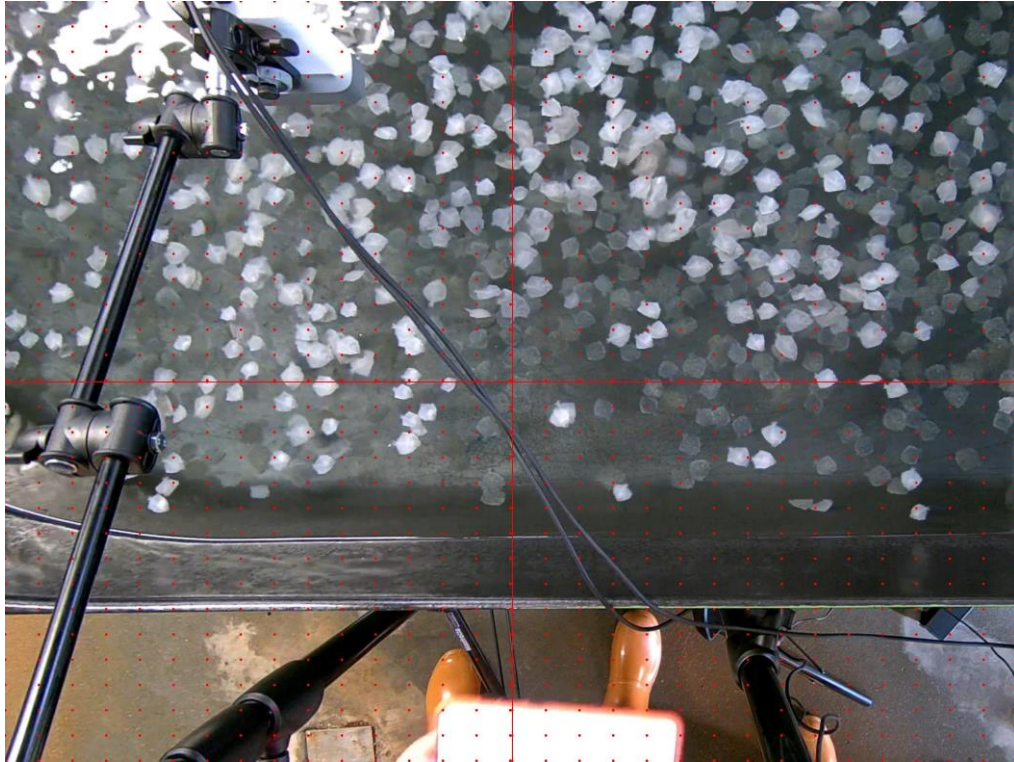


Fig. 4.9. Reprocesamiento de la imagen sobre la segunda capa, para $crop_n_layer = 2$, $crop_n_points_downscale_factor = 2$.

Se muestra en la Fig. 4.10 y en la Fig. 4.11 las máscaras obtenidas, pero con distintos parámetros. En el primer caso se establecieron los puntos de muestreo a 32 sin capas adicionales y en el segundo a 64 con 2 capas adicionales. Si se observa detalladamente la imagen, la definición de las máscaras en el segundo caso es mejor que en el primero, además, se obtuvo un total de 588 máscaras frente a 227.

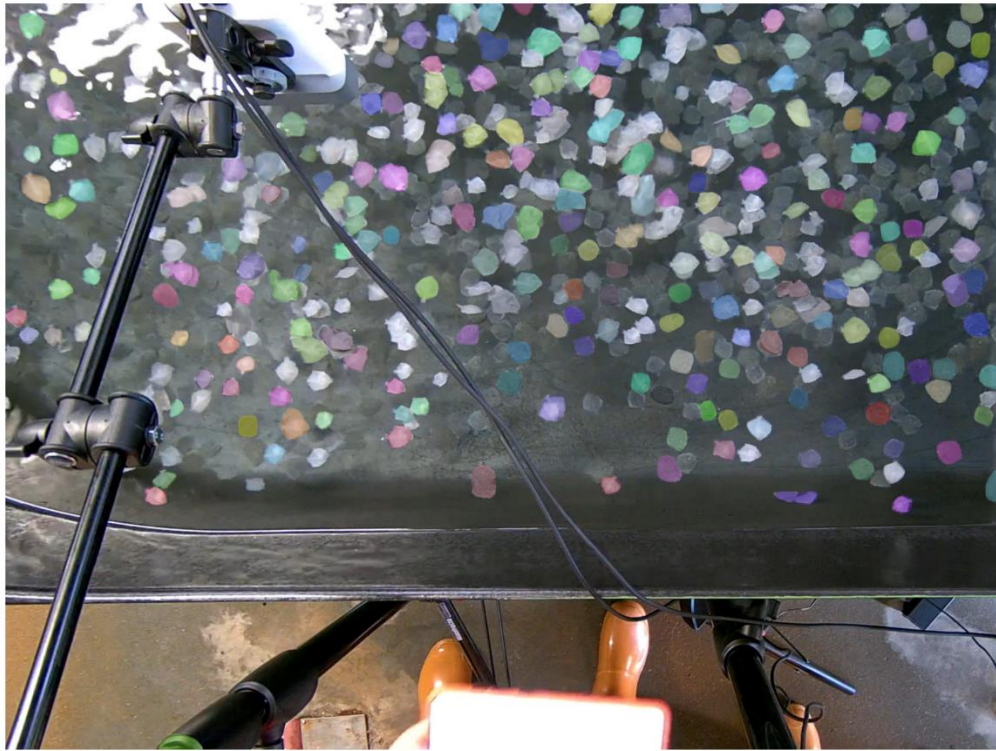


Fig. 4.10. Máscaras de la imagen izquierda con 32 puntos de muestreo y ninguna capa adicional.

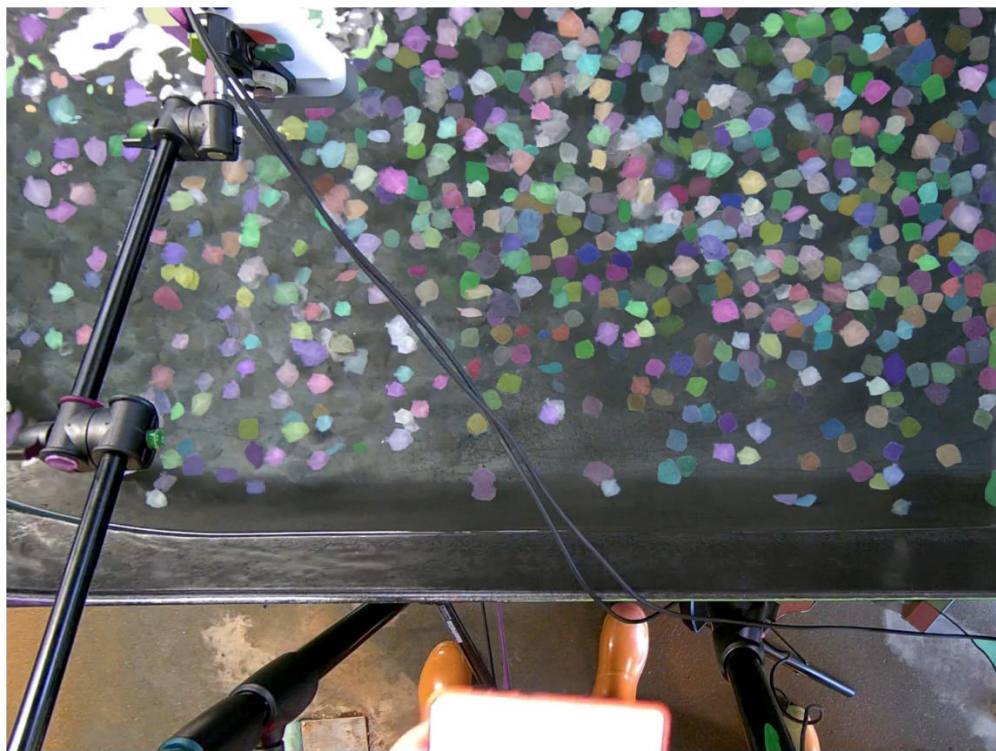


Fig. 4.11. Máscaras de la imagen izquierda con 64 puntos de muestreo y 2 capas adicionales.

Tras varias pruebas configurando los parámetros, se encontró que la configuración de 64 puntos y 2 capas daba buenos resultados, por lo que se estableció esta como configuración

principal. En caso de que se trabajase con imágenes con una menor densidad de peces, se podría reducir la utilización de recursos, así como si se trabaja con imágenes de peor calidad por su iluminación u otros motivos, se podría contrarrestar este efecto aumentando la potencia del modelo. Aun así, aumentar los puntos de muestreo o las capas adicionales más allá de esto, incrementaría considerablemente el coste computacional, lo que dificultaría la integración de este modelo en una aplicación, por lo que no es recomendable.

En la Tabla 2 y 3, se muestran los tiempos de ejecución y el número de máscaras del modelo según la configuración de parámetros realizada.

De los parámetros de las máscaras calculadas por el modelo, el único que se guardó a la salida de este para su posterior uso en otros scripts, fue el de correspondiente con la caja delimitadora, el cual fue almacenado en un archivo JSON estructurado en forma de lista.

El parámetro de *segmentation*, no se empleó finalmente a la salida del script, porque hubiese implicado un tratamiento de datos de cierta complejidad para poder reducir el tamaño de este (téngase en cuenta que cada uno de ellos es una lista de valores binarios de todos los píxeles de la imagen). Además, tener que realizar una segmentación de las imágenes izquierda y derecha por separado (incrementando así el coste computacional), sumado al hecho de que para la posterior estimación de tamaño los puntos del contorno debían ser hallados con alta precisión, hizo que finalmente quedase descartado del sistema.

4.4. Búsqueda de correspondencias entre objetos de distintas imágenes.

Una vez adquiridas las secciones de los objetos de los rodaballos correspondientes a una única imagen, en este caso, la de la cámara estereoscópica superior izquierda, se deben hallar las correspondencias de cada objeto en la imagen derecha. Es decir, se debe asegurar que el rodaballo captado en una imagen sea el mismo en la otra.

Para ello se han tomado las cajas delimitadoras de la salida del generador automático de máscaras. Estas cajas delimitadoras están en formato XYWH, que viene a representar, por un lado, las coordenadas (x,y) de la esquina superior izquierda de la caja mediante X e Y, y por el otro, su alto y ancho mediante los valores W y H. De esta manera, queda perfectamente definida la sección rectangular de la imagen en la que está contenido el objeto. En la Fig. 4.12 se representa una caja delimitadora en este formato.

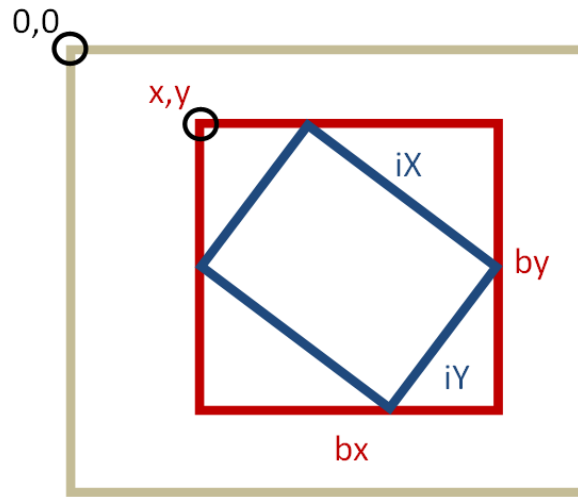


Fig. 4.12. Caja delimitadora en formato XYWH. W y H han sido sustituidos por bx y by respectivamente. [32]

Para hallar las correspondencias entre el objeto delimitado por la *bounding box* y su homólogo en la otra imagen, se ha empleado la función correlación cruzada, la cual da una medida de la similitud entre dos señales [33]. En (4.2) aparece reflejada su expresión para señales discretas, que son con las que se está trabajando.

$$r_{xy}(l) = \sum_{n=-\infty}^{\infty} x(n)y(n-l) \quad l = 0 \mp 1 \mp 2, \dots \quad (4.2)$$

En la expresión de la correlación cruzada el índice l hace referencia al desplazamiento de la secuencia $y(n-l)$ a lo largo del espacio de muestras.

Trasladar esto a imágenes, implica el mismo concepto, pero en vez de considerar señales unidimensionales, se están considerando señales bidimensionales discretas. Estas señales no son más que las matrices que representan los valores de los píxeles en una imagen. De lo que se trata es de tener una imagen de referencia o plantilla, que se vaya desplazando a lo largo de otra imagen de tal manera que se halle su correlación cruzada para cada píxel considerando los píxeles vecinos que conformen una imagen del mismo tamaño que el de la plantilla y que esté alineada con esta. En la Fig. 4.13 se ilustra esto, donde M2 sería la plantilla que se va desplazando sobre la imagen M1.

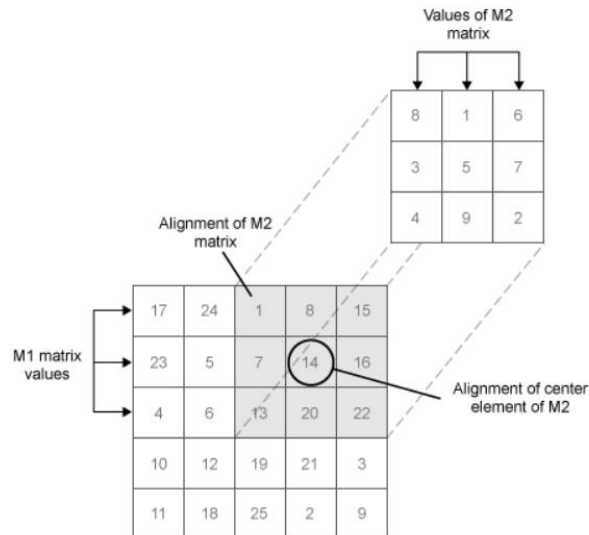


Fig. 4.13. Correlación cruzada bidimensional. Fuente: [34]

Para implementar esto en *Python*, se ha empleado la función *cv.matchTemplate()* de la librería *OpenCV* [35]. Esta función toma dos parámetros a su entrada, una imagen y una plantilla. Lo que hace es desplazar la plantilla a lo largo de la imagen de entrada comparando la plantilla y una porción de la imagen equivalente a esta [35]. Hay varios métodos de comparación, pero en este caso se ha empleado *cv.TM_CCORR_NORMED* [35], que se corresponde con la correlación cruzada normalizada.

Por lo tanto, el proceso consiste en extraer recortes de la imagen izquierda correspondientes a las delimitaciones de las cajas delimitadoras y utilizarlos como plantillas en la función *cv.matchTemplate()*. La imagen de entrada es la imagen derecha, pero en lugar de utilizar la imagen completa, se selecciona una región determinada por la posición de la caja delimitadora de la imagen izquierda, trasladada a la imagen derecha, con un margen adicional de 400 píxeles a cada lado de la caja. Esto se hace porque se espera que el objeto no se encuentre fuera de esta región, dado que el desalineamiento de las cámaras no es tan grande. Esta estrategia permite reducir considerablemente el coste computacional.

En la Fig. 4.14 se muestra un recorte de un rodaballo, y en la Fig. 4.15 la región de búsqueda correspondiente a este. En esta última, se señala mediante un punto rojo la localización de la máxima correlación obtenida. Este punto está ubicado en el píxel donde se encuadra la esquina superior izquierda de la plantilla sobre la porción de imagen con la que se compara. De esta forma, se obtiene fácilmente la caja delimitadora del segundo objeto trasladando la del primero a este punto. En la Fig. 4.16 se observa el objeto de la segunda imagen delimitado por su *bounding box*.

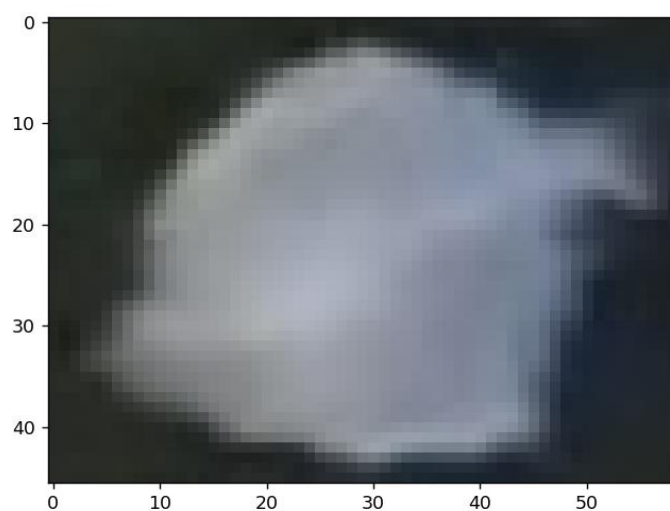


Fig. 4.14. Recorte de un rodaballo

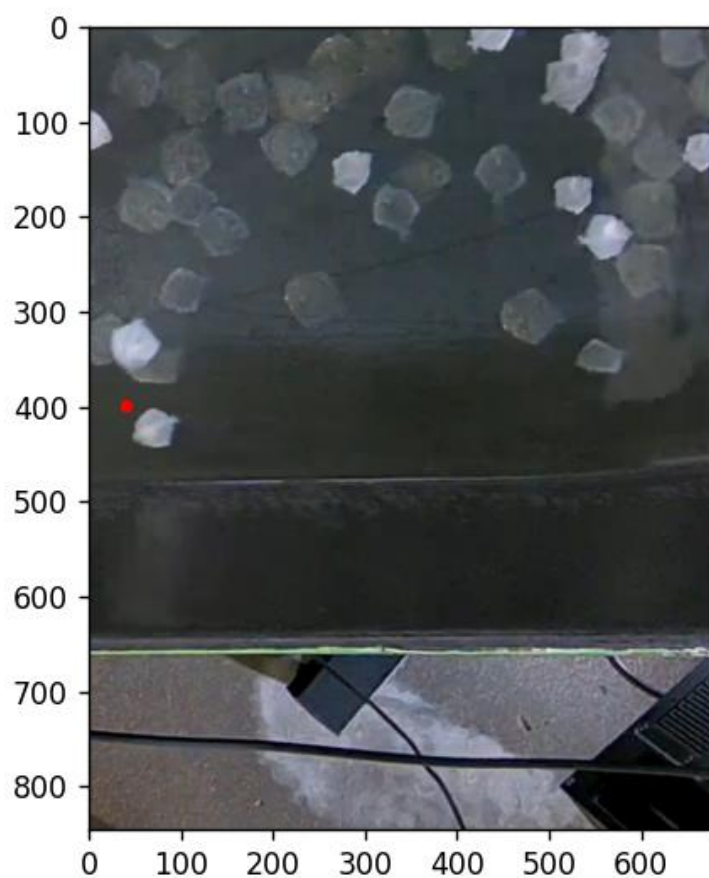


Fig. 4.15. Región de búsqueda y máxima correlación señalada mediante el punto rojo

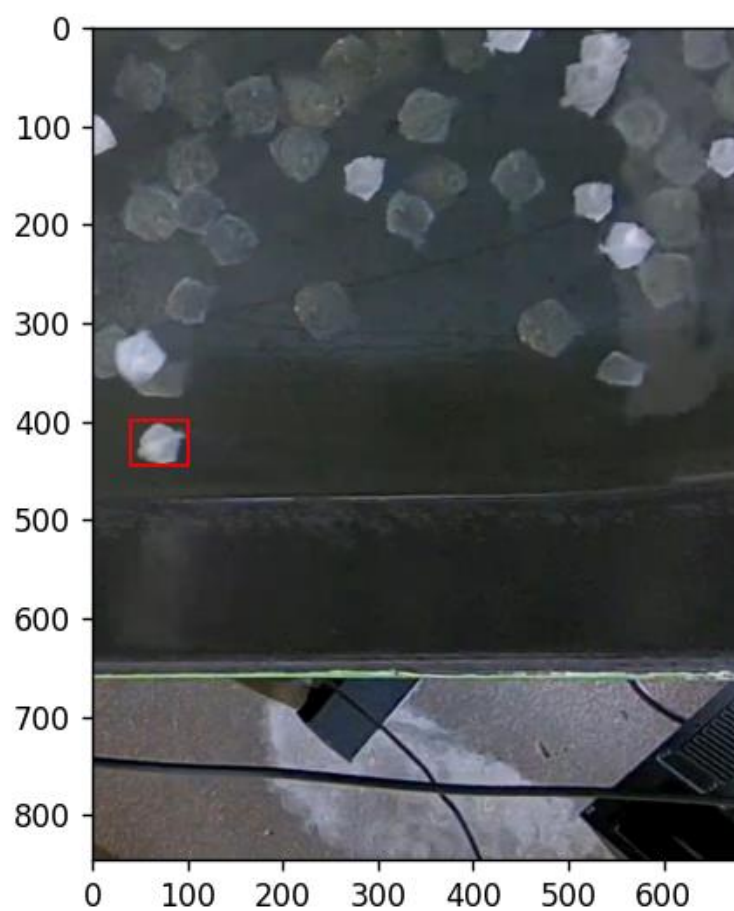


Fig. 4.16. Región de búsqueda y caja delimitadora del objeto hallado.

Este proceso se repite para cada uno de los recortes extraídos. El resultado se muestra en la Fig. 4.17.

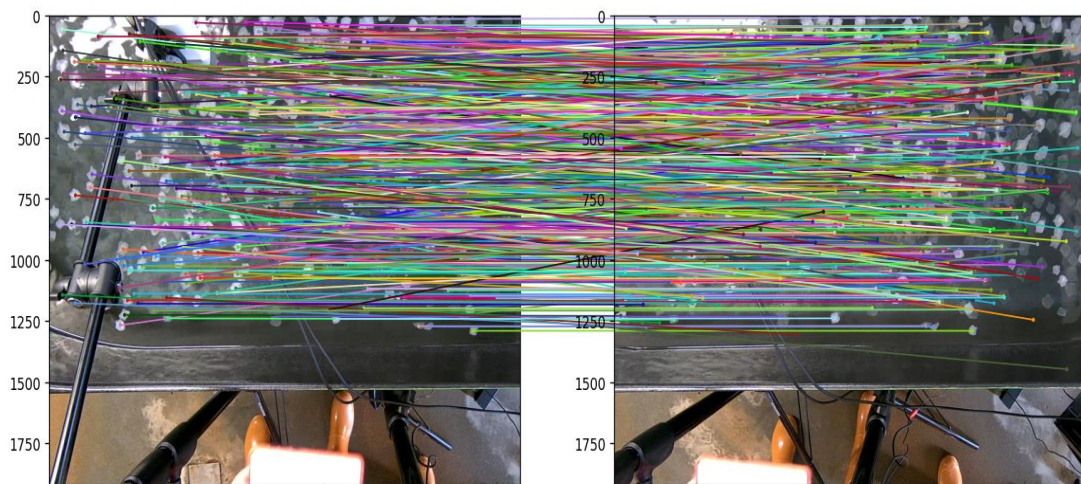


Fig. 4.17. Correspondencias entre la imagen izquierda y derecha.

Ya que no todas las correspondencias obtenidas son correctas, se realiza un filtrado de estas teniendo en cuenta que la posición relativa de los objetos debe ser similar para todos debido a la ubicación invariable de las cámaras durante la captura de los vídeos.

La técnica empleada consiste en calcular el ángulo que conforma el vector que va desde un punto central del objeto izquierdo al punto central del objeto derecho con respecto al eje horizontal de la imagen.

Se obtiene el ángulo más común calculado mediante la función *Counter()* de *Python* [36], y se aplica un margen de 20 grados con respecto a este ángulo para filtrar aquellos que no estén dentro de esta horquilla. El resultado de este filtrado se muestra en la Fig. 4.18, donde se puede apreciar que todas las líneas trazadas tienen una pendiente muy similar, a diferencia de lo que ocurría en la Fig. 4.17. Además, estas líneas reflejan que las cámaras están alineadas en el eje Y, lo cual concuerda con el tipo de alineamiento que deben tener las cámaras de visión estereoscópica [1].

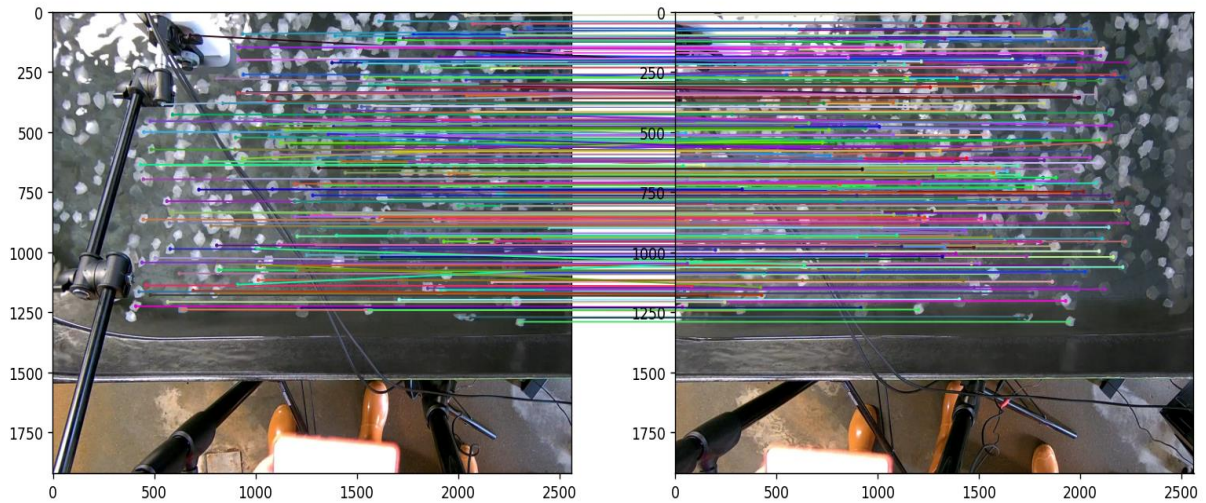


Fig. 4.18. Correspondencias después del filtrado por ángulo.

Finalmente se muestran las 5 mejores correspondencias, esto es, las 5 correspondencias con mayor correlación para que se aprecie como efectivamente se está encontrando el mismo objeto. Esto aparece representado en la Fig. 4.19.

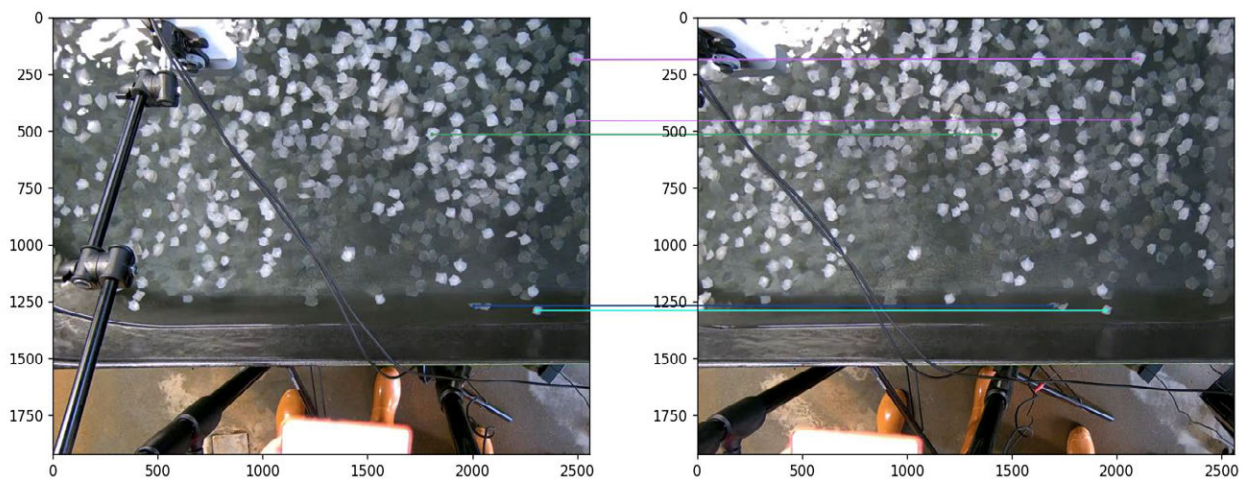


Fig. 4.19. Imágenes izquierda y derecha con las 5 mejores correspondencias

Se vuelven a guardar las cajas delimitadoras de los rodaballos, en este caso las de cada imagen, teniendo en cuenta el proceso de filtrado que se ha llevado a cabo. Los archivos son

masks_filt_corr_l.json para las máscaras de la imagen izquierda y *masks_filt_corr_r.json* para las máscaras de la imagen derecha.

4.5. Tratamiento aislado de recortes de rodaballos.

Para asegurar que los objetos identificados en las imágenes corresponden a rodaballos aislados y no a grupos de rodaballos u otros objetos, se ha diseñado un procedimiento basado en las características morfológicas del pez.

Procedimiento de Detección de contornos

El proceso de identificación de contornos se divide en los dos siguientes pasos:

- 1) Umbralización del objeto presente en el recorte.
- 2) Detección de los contornos del objeto umbralizado.

La función de umbralización asigna el valor de 255 a los píxeles por encima de un umbral determinado y 0 a los demás. La expresión de esta función es la siguiente:

$$f(x) = \begin{cases} 0, & x < Umbral \\ 255, & x \geq Umbral \end{cases} \quad (4.4)$$

Puesto que las distintas secciones de la imagen no presentan una iluminación homogénea, se optó inicialmente por que el umbral estuviese determinado por el brillo medio del recorte de la imagen. El problema de este método es que, los rodaballos que se encuentran en un nivel más profundo del tanque son muy oscuros, por los que no hay prácticamente diferencia entre los píxeles contenidos en el pez y el fondo. Por ello, se decidió emplear el método de umbralización dinámica de Otsu. Este método lo que hace es minimizar la varianza de los conjuntos de píxeles con brillo similar, y maximizar la varianza entre estos conjuntos [37]. La expresión de la varianza entre conjuntos está dada en (4.5).

$$\sigma_B^2 = \omega_0(t) \cdot \omega_1(t) \cdot [\mu_0(t) - \mu_1(t)]^2 \quad (4.5)$$

Donde, σ_B^2 es la varianza entre los dos conjuntos y el propósito del método es maximizarla, $\omega_0(t)$ y $\omega_1(t)$ son las probabilidades acumuladas de los dos conjuntos (4.6) y, $\mu_0(t)$ y $\mu_1(t)$, son las medias acumuladas de los dos conjuntos que representan la fracción de píxeles en cada uno (4.7).

$$\omega(t) = \sum_{i=0}^t p(i) \quad (4.6)$$

$$\mu(t) = \frac{\sum_{i=0}^t i \cdot p(i)}{\omega(t)} \quad (4.7)$$

El método de Otsu facilita la diferenciación del rodaballo al buscar la zona del histograma que está entre los dos picos representados por cada conjunto. Así, el umbral se adapta según el histograma del recorte en cuestión.

En la Fig. 4.20 y Fig. 4.21 se muestra la umbralización de un rodaballo por el método de Otsu y el umbral hallado respectivamente.



a) Recorte original



b) Recorte tras umbralización por Otsu

Fig. 4.20. Antes y después de la umbralización por Otsu en una imagen de baja iluminación.

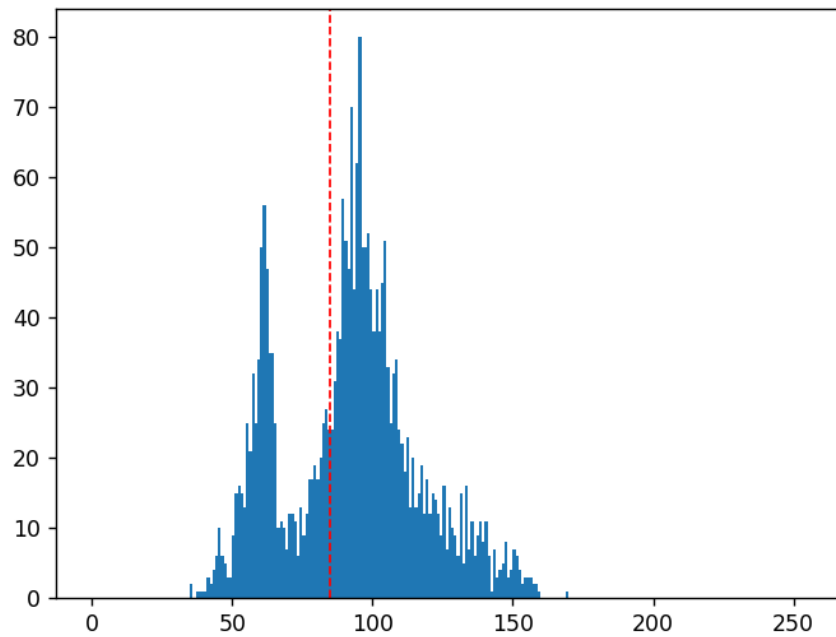


Fig. 4.21. Histograma de la imagen de Fig. 4. 20 con umbral determinado por Otsu marcado en rojo.

Los contornos se detectan en función de los bordes del objeto, es decir, los cambios en los niveles de gris en la imagen que delimitan un objeto de otro. El método empleado para encontrar contornos está dado por la función *findContours()* [38] de la librería OpenCV. Para facilitar esta detección, se ha aplicado la umbralización por Otsu a cada recorte de la imagen. De los contornos encontrados, se selecciona el contorno que encierra la mayor área, que generalmente se corresponde con el objeto principal.

Determinación de la Circularidad

Una vez determinado el contorno del objeto, se calcula la circularidad, definida como la medida en que la forma de un objeto se aproxima a un círculo perfecto. La expresión de la circularidad es la siguiente:

$$C = \frac{4 \pi \cdot \text{Área}}{\text{Perímetro}^2} \quad (4.8)$$

Un valor de 1 indica un círculo perfecto, y valores menores indican desviaciones de la forma circular. Dado que el rodaballo no es un círculo perfecto, no se puede seleccionar un valor de 1 para realizar la discriminación.

Los estudios sobre la forma del rodaballo indican que, aunque no es perfectamente circular, su forma es suficientemente redondeada como para que un umbral determinado capture su

geometría sin excluir casos genuinos. Esto se apoya en la literatura del campo de la acuicultura que describe la forma de los rodaballos como plana y casi circular [39].

El método para hallar el valor de filtrado consiste en ajustar el parámetro de la circularidad observando que los rodaballos aislados son procesados correctamente y los objetos no pertenecientes a este grupo son descartados.

Se presentan ejemplos de imágenes procesadas correctamente y descartadas. En la figura 4.22 se muestra un recorte procesado correctamente, mientras que en la figura 4.23 se muestra una imagen descartada debido a un contorno no circular. La imagen descartada presenta baja circularidad debido a la proximidad de otro rodaballo, resultando en un contorno irregular.



Fig. 4.22. Recorte procesado correctamente.



Fig. 4.23. Imagen descartada por contorno no circular.

Tras varias pruebas realizadas, se seleccionó un valor de 0.6. Este valor discrimina eficazmente entre rodaballos aislados y otros objetos o grupos de rodaballos. Casos con circularidad inferior a este umbral suelen deberse a la presencia de múltiples rodaballos o zonas donde la iluminación no permite distinguir correctamente el fondo del objeto, que resultan en contornos irregulares.

El uso de la umbralización dinámica de Otsu combinado con la medida de circularidad proporciona una metodología robusta para la identificación de rodaballos aislados en

imágenes. Este enfoque considera variaciones en la iluminación y la forma circular de los rodaballos, resultando en una detección precisa y fiable.

4.6. Obtención de contornos

La obtención de contornos representa la última fase del proceso de delineación de los objetos. Esta fase se conecta estrechamente con el tratamiento aislado, ya que solo los contornos procesados correctamente se guardan en el archivo *contours.json* para la posterior obtención de los puntos extremos del objeto.

En este JSON se almacenan conjuntamente los contornos de ambas imágenes, etiquetados como 'l' o 'r' según la cámara con la que fueron captados. Además, se agrupan con su correspondiente contorno de la otra imagen, lo que permite al sistema de visión estereoscópica funcionar de manera eficiente.

4.7. Obtención de puntos extremos

Para obtener los puntos extremos del objeto, esto es, el par de puntos del contorno con una mayor distancia entre sí se parte del módulo de segmentación desarrollado por María Castillo [1]. Este módulo se implementa mediante un algoritmo que sigue los siguientes pasos:

- Extrae los contornos de cada objeto 'l' y 'r'
- Recorre todos los puntos de cada contorno en un bucle
- Para cada punto, calcula distancia que tiene con el resto de los puntos del contorno
- Actualiza la distancia máxima en cada iteración

El problema de este algoritmo es que los puntos de máxima distancia del contorno del objeto 'l' y el objeto 'r' no tienen por qué estar necesariamente en el mismo eje, ya que, aunque se trate del mismo objeto, puede presentar distinta forma debido a factores como las imperfecciones derivadas del proceso de rectificación de imágenes, la perspectiva con la que las cámaras captan al objeto, etc. Esto tiene como consecuencia que la posterior estimación de tamaño no se haga correctamente debido a la incoherencia geométrica que existe.

Es por esto por lo que se ha tenido que modificar este módulo para que los resultados sean más apropiados, y así las líneas que intersectan los dos puntos de máxima distancia de cada objeto estén en un eje similar.

La solución adoptada consiste en que, conociendo el vector desplazamiento $\vec{d}$ entre ambos objetos, dado por la diferencia entre las coordenadas (x, y) de sus cajas delimitadoras (4.6), se traslade el par de puntos del contorno 'l' o 'r' (4.8) que mayor distancia presente (4.7).

$$\vec{d} = (x_{bbox2} - x_{bbox1})\vec{i} + (y_{bbox2} - y_{bbox1})\vec{j} \quad (4.6)$$

$$P(x, y) = \max [P(x, y)_l, P(x, y)_r] \quad (4.7)$$

$$P'(x', y') = P(x, y) + \vec{d} = (x + dx, y + dy) \quad (4.8)$$

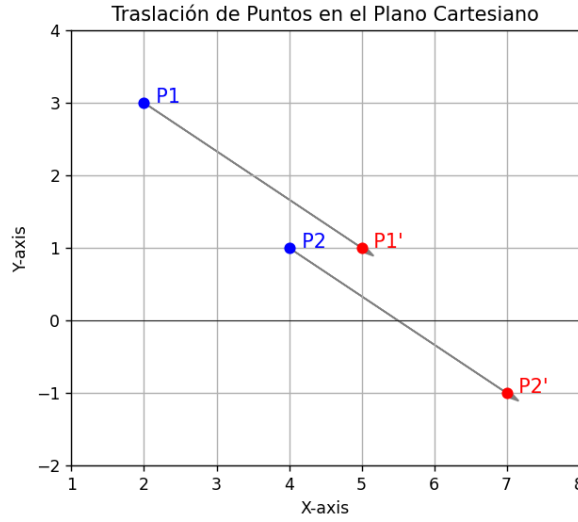


Fig. 4.24. Traslación de un par de puntos P1 y P2 a P1' y P2'.

Después de hallar los puntos trasladados, para asegurar que estos se corresponden con el objeto en cuestión, y no con otros fuera del contorno, se halla la distancia euclídea mínima de los puntos trasladados a los del contorno. La expresión de la distancia euclídea es la siguiente:

$$d_k(P, Q) = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2} \quad (4.9)$$

Donde (x_i, y_i) se corresponderían con las coordenadas del contorno, y (x_j, y_j) con las coordenadas de los puntos trasladados. Se comparan cada una de las distancias euclídeas d_k en un bucle, y aquellos puntos del contorno que minimicen esta distancia, son los puntos extremos finales de este contorno. En caso de que la distancia entre puntos sea mayor a 5 píxeles, se descartan estos contornos, puesto que se considera que los puntos están demasiado alejados del objeto.

En Fig. 4.25 se muestra cómo sería este cálculo de distancias en un plano 2D, que es con el que se trata en imágenes.

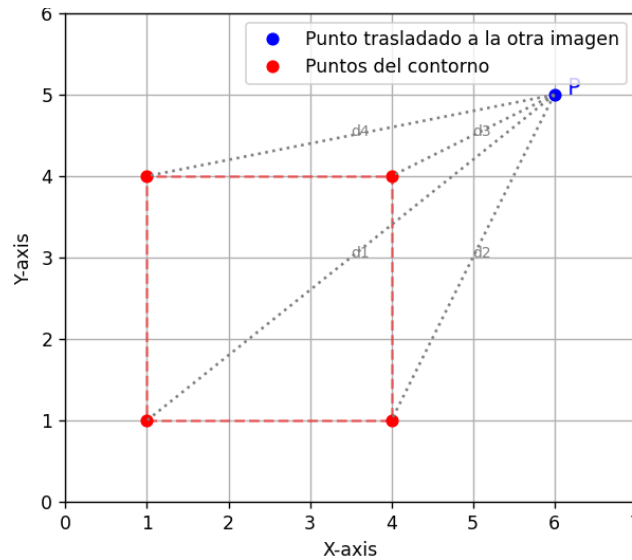


Fig. 4.25. Representación gráfica de un punto trasladado y las distancias euclídeas al otro contorno.

4.8. Estimación de tamaño

La estimación de tamaño se basa en el módulo desarrollado por María Castillo en su TFM, detallado en el script ***depth_estimation.py*** [1]. Este módulo integra la estimación de profundidad, inclinación y tamaño para proporcionar una medida precisa del tamaño de los rodaballos en el tanque.

El proceso se fundamenta en el uso del sistema de visión estereoscópica, el cual permite, a partir de los puntos extremos y el centro de un objeto, la distancia en píxeles de estos puntos entre las dos imágenes del sistema, junto con los parámetros de calibración de las cámaras y su posición relativa, estimar la profundidad y el tamaño de los peces. Esta técnica aprovecha las relaciones entre estos datos para generar estimaciones precisas.

4.9. Sistema final

Cada módulo del sistema se corresponde con un script escrito en el lenguaje de programación *Python*, cuyo desarrollo se ha llevado a cabo en un entorno virtual creado en el IDE *Visual Studio Code*. Además, se ha llevado a cabo un control de versiones mediante *GitHub* donde todo el material empleado para el proyecto está guardado en un repositorio privado [40], a excepción de los vídeos, que, debido a su tamaño, se almacenan en una carpeta local.

En Fig. 4.26 se presenta un diagrama de flujo con cada uno de los procesos presentes en el del diseño final del sistema.

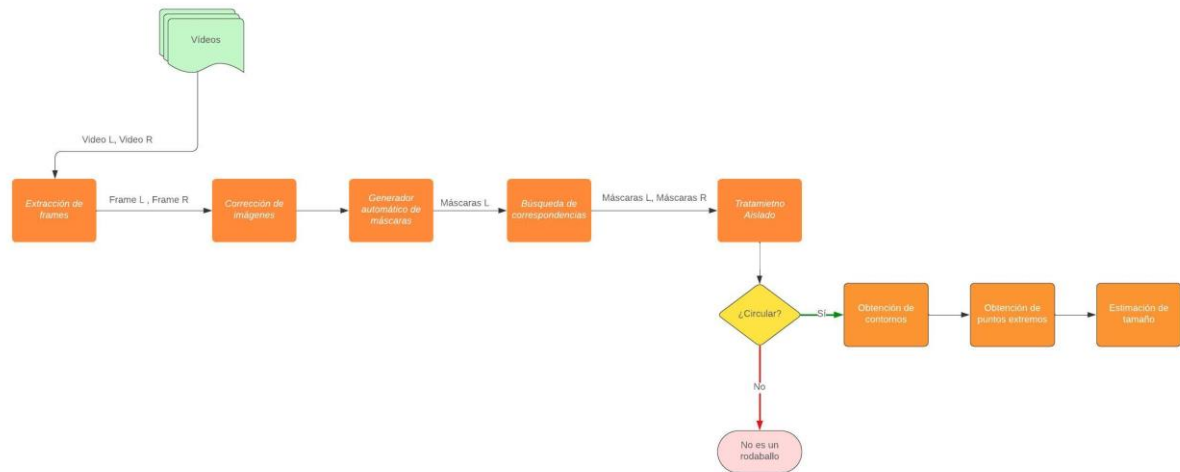


Fig. 4.26. Diagrama de flujo del sistema final

Los scripts asociados a cada uno de los módulos son los siguientes:

- ***extraccion_frames.py***: lleva a cabo la extracción de dos frames idénticos, uno de ellos correspondientes al vídeo capturado por la cámara superior izquierda y el otro por la cámara superior derecha.
- ***img_correction.py***: corrige las imágenes que se le pasan como entrada.
- ***automatic_mask_generator_turbots.py***: genera máscaras de forma automática para el frame de la cámara superior izquierda.
- ***compare_images.py***: busca las correspondencias entre los objetos del frame izquierdo y derecho.
- ***tratamiento_aislado.py***: procesa cada uno de los recortes asociados a la caja delimitadora de cada objeto y decide si el contorno es lo suficientemente circular como para ser considerado un rodaballo.
- ***contours.py***: obtiene los contornos de cada uno de los recortes filtrados en el módulo anterior.
- ***segmentation.py***: obtiene los puntos extremos del contorno, así como su centro.
- ***depth_estimation.py***: estima el tamaño de los objetos definidos por sus puntos extremos y su centro.

5. Resultados

5.1. Análisis Global

Además, se realiza un conteo del número total de objetos medidos y un cálculo del tamaño promedio obtenido.

En la Fig. 5.1 y Fig. 5.2 se muestran dos imágenes con distinta distribución de rodaballos, ilustrando los resultados de la estimación de tamaño. En la Fig. 5.1, la iluminación es bastante buena, permitiendo la detección de 96 objetos y estimando un tamaño promedio de 2,67 cm, como se indica en el recuadro negro en la esquina inferior derecha de la imagen.

En la Fig. 5.2 se observa una imagen con más rodaballos visibles en la profundidad del tanque. El tamaño promedio en esta imagen es de 2,95 cm, aproximadamente 0,3 cm mayor que en el caso anterior. Esto se debe a la detección de un mayor número de rodaballos en aguas profundas. A medida que los rodaballos envejecen, tienden a sumergirse en aguas más profundas, lo que explica el mayor tamaño promedio en este caso. Sin embargo, la detección de rodaballos en estas áreas se complica por dos factores:

- Escasa separación entre rodaballos.
- Baja iluminación.

Estos factores deben considerarse al valorar el tamaño general de los rodaballos en el tanque, ya que generalmente se detectan más rodaballos en la superficie que en el fondo.

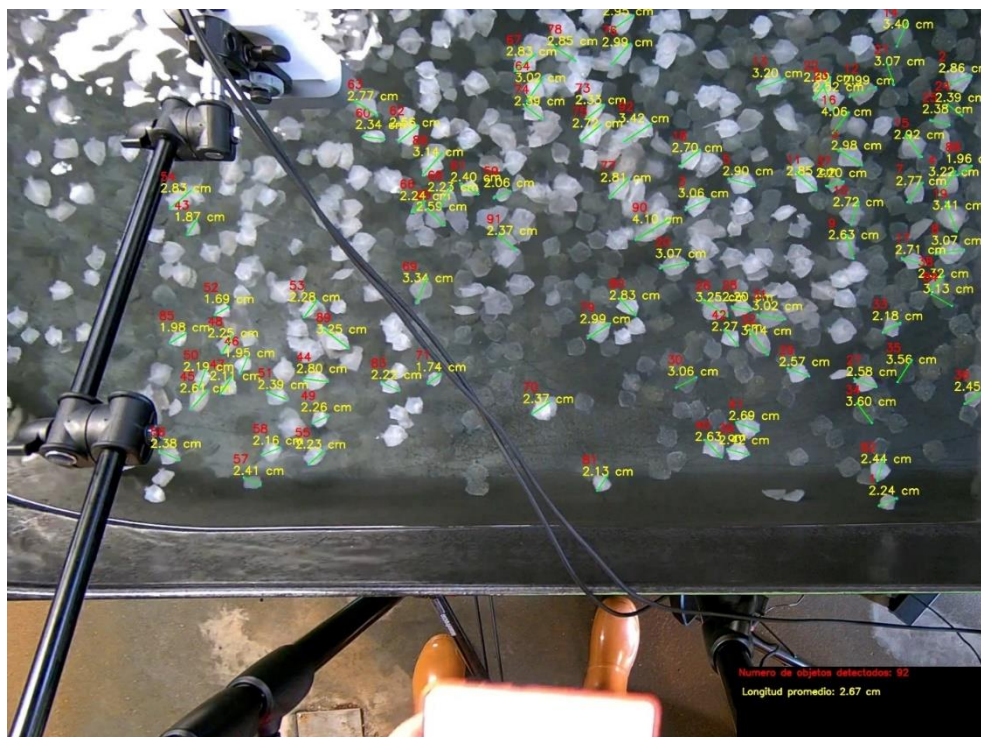


Fig. 5.1. Resultado de la estimación con imagen bien iluminada.

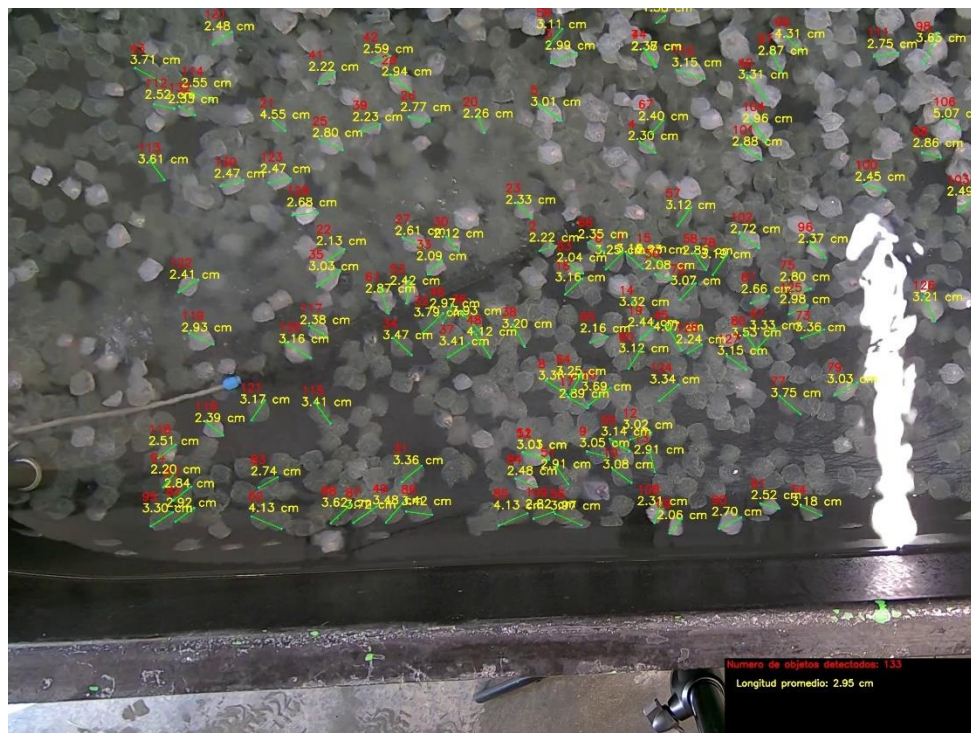


Fig. 5.2. Resultado de la estimación con alta densidad de rodaballos situados en el fondo.

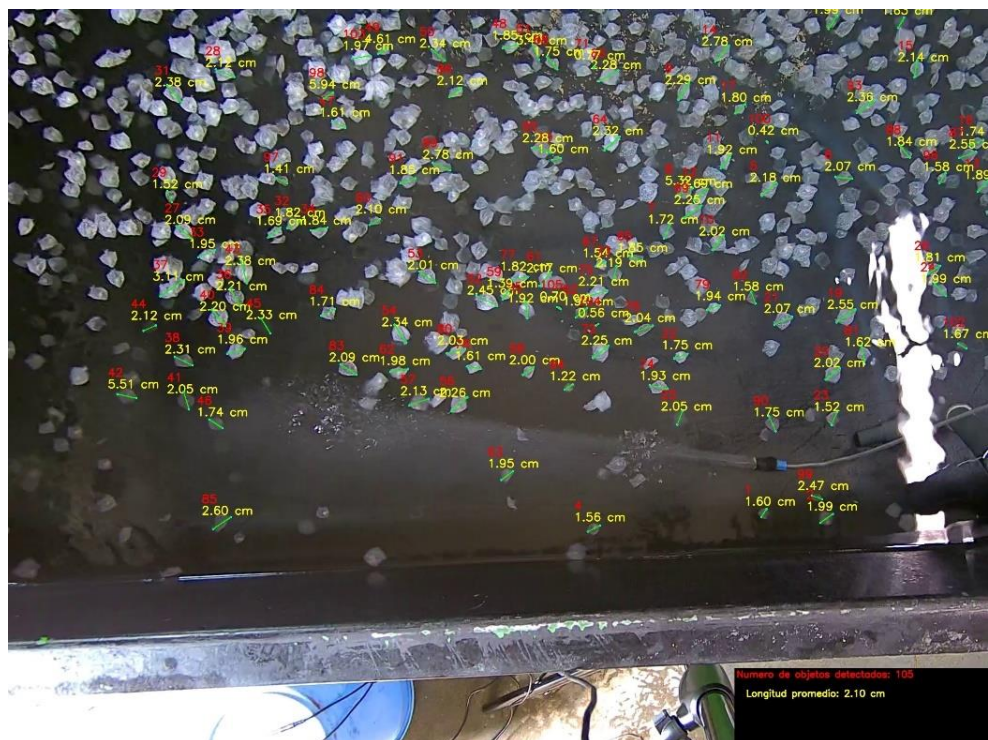


Fig. 5.3. Resultado con imagen de buena calidad.

En Fig.5.4 tan solo se llegan a detectar 36 rodaballos. Esto se debe a la mala iluminación que presenta la imagen, haciendo que esta se emborrona por la refracción de la luz en el agua.



Fig. 5.4. Resultado de la estimación con mala iluminación

En general, los resultados son bastante buenos y tras varias pruebas, se ha encontrado que se suelen llegar a detectar entre 50 y 200 rodaballos, dependiendo del tanque presentado en la imagen y la distribución de los rodaballos en este. Por otro lado, teniendo en cuenta que el tamaño de un rodaballo que está pasando por el periodo de metamorfosis, esto es, la transición entre estado larvario y alevinaje es de entre 2,2 y 4,5 cm [2], la estimación de tamaño se encuentra dentro de un rango bastante razonable, ya que se obtienen promedios que están dentro de este rango y no se observa apenas ningún pez que supere los 4 cm.

A pesar de esto, hay imágenes muy concretas, como es el caso de Fig. 5.4, cuya calidad no es la más apropiada para procesarlas en este sistema, principalmente debido a su iluminación. Puesto que, para compensar la mala iluminación, habría que aumentar considerablemente la potencia del generador automático de máscaras, la solución más adecuada es que desde un inicio se capten las imágenes con una buena calidad para facilitar los posteriores procesos de detección y segmentación.

5.2. Validación de resultados

5.2.1. Detección y segmentación de rodaballos

Ya que el modelo MobileSAM no proporciona una fase de validación, puesto que tampoco ha sido ni puede ser entrenado con un conjunto de datos específico, la única manera de validar una correcta detección y segmentación es recurrir al análisis cualitativo de la imagen.

La detección en la mayoría de los casos es correcta, son casos muy específicos en los que se puede ver que se haya detectado un objeto que no es un rodaballo aislado. Los casos en los que se observa una mala detección suelen deberse a grupos de rodaballos muy unidos entre sí, que generan una forma lo suficientemente circular como para ser considerado un rodaballo. En Fig. 5.5 se ofrece una comparativa de dos recortes de la imagen 5.1.



a) Rodaballo aislado b) Grupo de rodaballos

Fig. 5.5. Comparativa entre detección de un rodaballo aislado y un grupo de rodaballos

A pesar de esto, en casi todos los casos se detectan rodaballos completamente aislados, y tan sólo puntualmente se observa una mala detección, por lo que se puede considerar que el sistema de detección es bastante robusto.

En cuanto a la segmentación, se observan principalmente dos situaciones para la consideración de puntos extremos de un contorno:

- Cabeza-Cola
- Lateral-Lateral

Las dos situaciones dependen del eje sobre el que se estén detectando los puntos, en el segundo este puede variar según la forma del objeto, y es perfectamente posible que según el rodaballo sus puntos extremos se correspondan con una u otra situación, aunque por la morfología del rodaballo es más común que se dé la primera y así sucede en los resultados. En Fig. 5.6. se ofrece una comparativa de dos recortes de Fig. 5.6 presentando ambas situaciones.

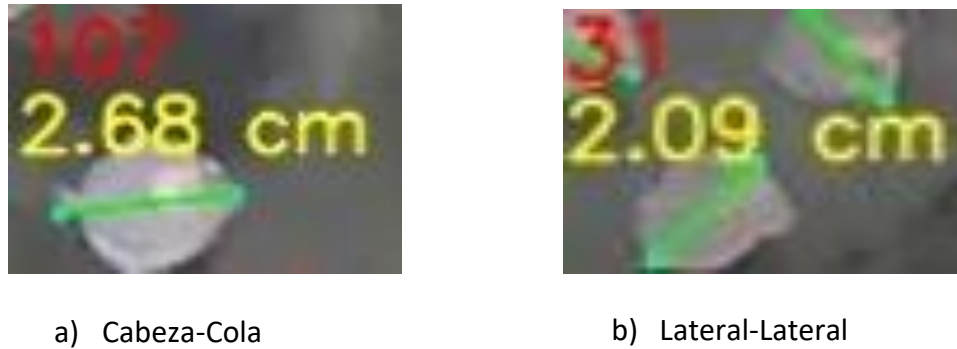


Fig. 5.6. Comparativa entre segmentación cabeza-cola y lateral-lateral

Generalmente la segmentación es correcta en la mayoría de los casos, lo cual se corrobora por el hecho de que los tamaños estimados están dentro de un rango razonable para el tipo de tanques de las imágenes, que albergan rodaballos que están en periodo de transición entre larva y alevín.

Sin embargo, puede haber segmentaciones imprecisas, especialmente si se observan rodaballos situados en aguas muy profundas del tanque, ya que estos están muy sintetizados con el fondo, lo que da lugar a que la segmentación no sea plenamente validable a nivel subjetivo. Un ejemplo de esto se puede ver en Fig. 5.7 donde se observa un recorte de Fig. 5.1, en el que no se puede valorar a nivel visual si el rodaballo está correctamente segmentado.



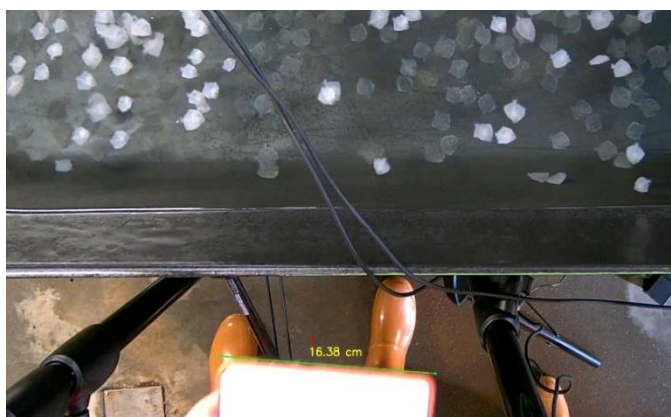
Fig. 5.7. Segmentación de un rodaballo en agua profunda.

De todas formas, aquí solo entran en juego apreciaciones subjetivas, por lo que es complicado valorar si la segmentación es correcta o no. Además, sí que se puede intuir que, incluso si la segmentación no es completamente precisa, sí que se aproxima a lo que se espera. Por lo tanto, se considera que el sistema de segmentación da buenos resultados en la mayoría de los casos, aunque algunos de estos no se puedan validar subjetivamente.

5.2.2. Estimación de tamaño

Para validar la estimación de tamaño, se han tomado objetos reales que se capturan durante los vídeos cuyo tamaño es conocido. Mediante la herramienta online *PixSpy*, se pueden determinar las coordenadas de los píxeles de una manera interactiva. El procedimiento consiste en seleccionar con el cursor de *PixSpy* los puntos extremos del objeto y el centro que los une, anotar sus coordenadas, y escribirlas en el archivo *head_tail_pts.json* que toma como entrada el módulo de estimación de tamaño para que este las procese.

En Fig. 5.8 se puede observar la estimación de tamaño de un móvil cuyo largo mide 16 cm. El resultado de la estimación es de 16,38 cm, lo que supone un error de 0,38 cm.



a) Imagen izquierda

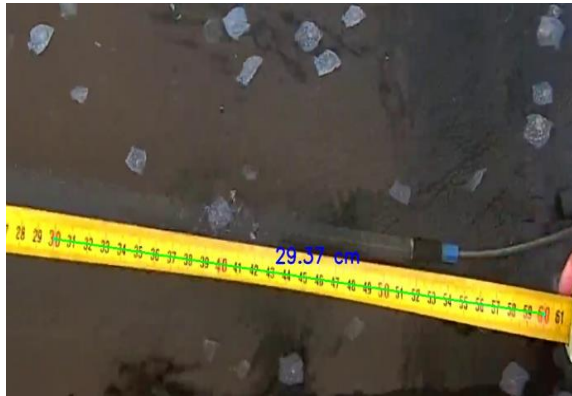


b) Imagen derecha

Fig. 5.8. Estimación del tamaño del largo de un móvil.

Teniendo en cuenta que los puntos están tomados de manera manual, y que el módulo de estimación de tamaño presenta un MAE de 1, 86mm y un MAPE de 4, 67% [1], se puede considerar un resultado bastante aceptable.

En Fig. 5.9 se ha realizado el mismo procedimiento, pero en este caso para medir los 30 cm que van desde el centímetro 30 al centímetro 60 de un metro empleado durante la grabación del vídeo correspondiente a la imagen de Fig.5. 2. En este caso se obtienen 29,37 cm, lo cual representaría un error de 0,63 cm, que son 0,25 cm más que en el caso del móvil. Aparte de los factores que se comentaron anteriormente, para el metro habría que tener en cuenta también que este no tiene por qué estar completamente extendido y firme, lo cual derivaría en obtener una distancia menor de la real.



a) Imagen izquierda



b) Imagen derecha

Fig. 5.9. Estimación de 30 cm a lo largo de un metro.

5.2.3. Estimación de profundidad

La estimación de profundidad también parece ser correcta, puesto que se encuentran rodaballos situados a mayores profundidades que, con la misma extensión que uno que se encuentra en la superficie, presentan mayor tamaño. Esto se debe a que la distancia a la que estén situados afecta a la perspectiva, ya que, para un mismo tamaño real, aquellos que estén situados más próximos a la cámara parecerán más grandes y viceversa.

Para corroborar esto se han realizado dos recortes de Fig. 5.1 de los objetos 29 y 30, los cuales se presentan en Fig.10.a) y Fig.10.b) respectivamente. Se puede ver como el primero está en la superficie del agua al tener una mayor iluminación y el segundo se encuentra más profundo al ser más oscuro.



a) Rodaballo situado en la superficie



b) Rodaballo situado en zona profunda

Fig. 5.10. Comparativa del tamaño de dos rodaballos con extensión similar pero situados a distintas profundidades.

A pesar de esto, habría que tener en cuenta que el sistema de estimación de profundidad desarrollado por María Castillo puede incrementar el error de estimación de tamaño [1]. Esto se debe a que en este sistema no se tienen en cuenta factores como la refracción de la luz sobre el agua.

En Fig. 5.11 se presenta la variación del error en función de la profundidad a la que estén situados los peces.

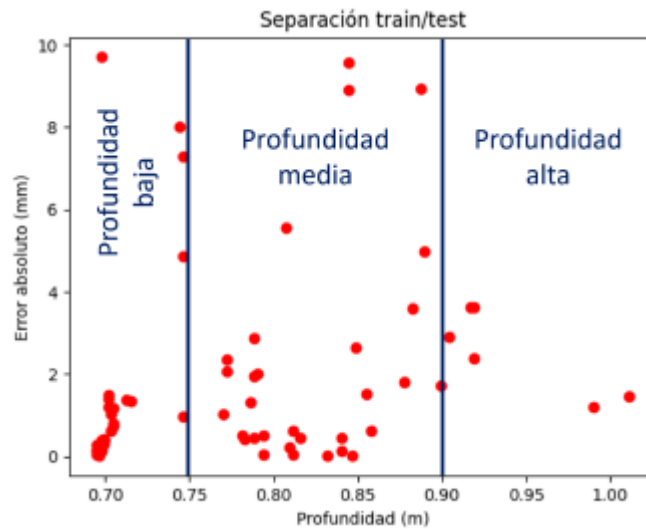


Fig. 5.11. Variación del error en función de la profundidad. Fuente: [1]

Por lo tanto, no existe certeza de que los rodaballos situados a una profundidad alta tengan una estimación de tamaño correcta. A pesar de esto, el error medio global es un MAE de 1,86 mm [1], por lo que no se considera que la profundidad sea un factor que invalide el sistema.

5.3. Representación de resultados.

Para facilitar la visualización de los resultados de una forma sintetizada, se ha añadido a la salida del módulo de estimación de tamaño, una representación en forma de histograma del tamaño de los peces en intervalos de 0,5 cm y su frecuencia relativa de aparición.

En Fig. 5.12 se representa el histograma correspondiente a los resultados de Fig.5.1, donde había predominancia de rodaballos situados en la superficie. Por otro lado, en Fig. 5.13 se representa el histograma correspondiente a los resultados de Fig.5.2, donde había predominancia de rodaballos situados en el fondo.

Si se comparan ambos histogramas, se puede observar que en el primero las frecuencias de aparición aparecen concentradas en los rangos que van de los 2 a los 3 cm, mientras que, en el segundo, aparte de en estos rangos, hay bastantes apariciones en el rango de 3 a 3,5 cm, lo cual concuerda con que sea una imagen con más rodaballos situados en el fondo que en la anterior.

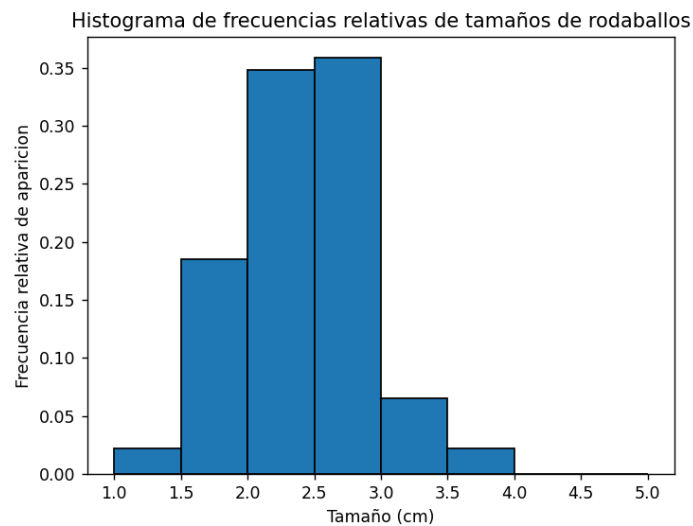


Fig. 5.12. Histograma de frecuencias relativas de los tamaños de rodaballos en intervalos de 0,5 cm correspondientes a una imagen con predominancia de rodaballos situados en la superficie.

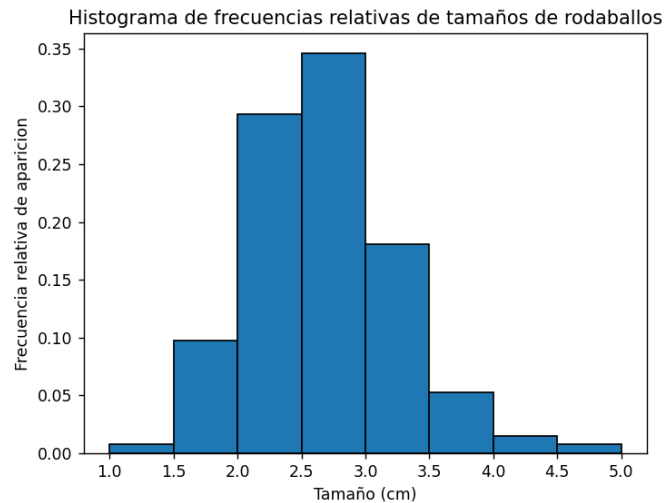


Fig. 5.13. Histograma de frecuencias relativas de los tamaños de rodaballos en intervalos de 0,5 cm correspondientes a una imagen con predominancia de rodaballos situados en la profundidad del agua.

Este histograma es replicable a cualquier imagen que se quiera procesar. De esta forma, se puede observar cuál es el rango de tamaño más común en el tanque, y cuál es la variación de este frente a los otros, lo cual concuerda precisamente con el objetivo de este proyecto, facilitar la determinación del calibre del pienso que hay que suministrar a un tanque en función de la homogeneidad del tamaño de los rodaballos presentes en este.

6. Presupuesto

Se han calculado los costes de producción del proyecto teniendo en cuenta que este se realizase en España

Tabla 2: Presupuesto del proyecto

Concepto	Descripción	Costo unitario (Eur)	Unidades	Total (Eur)
Ordenador HP Pavilion Gaming	Ordenador personal Gaming	1200	1	1200
100 unidades de computación de Google Colab	Recursos de procesamiento en la nube	11,19	1	11,19
Recién Graduado	Salario de recién graduado	14,6/h	360 horas	5256

- Costo unitario del ordenador HP Pavilion Gaming: 1200 euros. [41]
- Costo unitario de las 100 unidades de computación de Google Colab: 11,19 euros [42]
- Costo unitario por hora de un recién graduado: [43]

El presupuesto total sería de 6467,19 euros.

7. Impacto del proyecto

En este capítulo se evaluarán las implicaciones sociales, de salud y seguridad, ambientales, económicas, tecnológicas e industriales del proyecto, así como su posible contribución a los Objetivos de Desarrollo Sostenible (ODS) de las Naciones Unidas.

El proyecto tiene el potencial de impactar positivamente en la industria acuícola al optimizar la alimentación de los peces, lo que puede conducir a una producción más eficiente y sostenible. Esto puede beneficiar a las comunidades locales mediante la mejora de la economía regional.

El uso de tecnologías automatizadas para la estimación del tamaño de los peces puede reducir la necesidad de intervención humana directa, minimizando así el riesgo de accidentes y mejorando las condiciones de trabajo en las piscifactorías.

La optimización en la alimentación de los peces puede reducir el desperdicio de alimentos y la contaminación del agua, contribuyendo a una práctica más sostenible y respetuosa con el medio ambiente.

La automatización del proceso de estimación del tamaño de los peces puede reducir costos operativos y aumentar la eficiencia, lo que puede tener un impacto positivo en la rentabilidad de las piscifactorías.

El proyecto demuestra el uso efectivo de la visión por computador y el aprendizaje automático en un entorno industrial, lo que puede fomentar la adopción de estas tecnologías en otras áreas de la acuicultura y más allá.

El proyecto contribuye directamente a varios ODS [44], incluyendo:

- ODS 2: Hambre cero, al mejorar la eficiencia de la producción de alimentos.
- ODS 8: Trabajo decente y crecimiento económico, al mejorar la productividad.
- ODS 9: Industria, innovación e infraestructura, al promover el uso de tecnologías avanzadas.
- ODS 12: Producción y consumo responsables, al reducir el desperdicio de alimentos y la contaminación.

8. Conclusiones

En el último apartado de la memoria se ha de incluir una visión general del trabajo realizado: problema propuesto, solución planteada y resultados obtenidos. Junto con esta descripción, también hay que especificar qué conocimiento nuevo se puede derivar de toda la información expuesta en el informe: validez o no del sistema, nuevos aspectos del problema detectados en el trabajo, etc. Finalmente, si el proyecto no deja totalmente cerrado y resuelto el problema tecnológico abordado, el proyectista debe aprovechar su experiencia para proponer líneas futuras de trabajo.

8.1. Conclusiones

La estimación precisa del tamaño de los rodaballos en imágenes de tanques acuícolas es crucial en el campo de la acuicultura para determinar adecuadamente el calibre del pienso a suministrar. En este proyecto, se ha logrado este objetivo principal mediante el desarrollo de un sistema automatizado que estima el tamaño de una cantidad significativa de rodaballos por imagen, oscilando entre 50 y 200 individuos, según el tanque presentado en la imagen y la distribución de los rodaballos en este.

La validación de las estimaciones realizadas se ha llevado a cabo contrastándolas con los tamaños reales de los rodaballos presentes en las imágenes, los cuales varían entre 2,2 y 4,5 cm. Los resultados obtenidos se encuentran dentro de estos rangos esperados, lo que corrobora la precisión del sistema desarrollado. Además, se proporciona la frecuencia de aparición del tamaño de los rodaballos en intervalos de 0,5 cm, facilitando así al personal encargado del suministro de pienso en los tanques la adecuada estimación del calibre necesario.

La validación del sistema también se ha realizado mediante la comparación con medidas de objetos reales presentes en las imágenes, evaluando el error en la estimación de estos objetos en relación con el sistema desarrollado por María Castillo en su Trabajo Fin de Máster [1].

Las principales limitaciones del sistema están relacionadas con la validación subjetiva de las estimaciones, aunque los resultados obtenidos se encuentran dentro de un rango razonable. Se considera, por tanto, que el sistema es adecuado para estimar el tamaño de rodaballos en tanques acuícolas, siempre y cuando las imágenes no presenten una iluminación de mala calidad.

Es importante destacar que el sistema muestra ciertas imprecisiones al estimar el tamaño de rodaballos situados en el fondo del tanque, debido a la complejidad de segmentar correctamente objetos muy oscurecidos. Sin embargo, en general, el sistema demuestra robustez frente a variaciones en la iluminación y en la distribución de los rodaballos en el tanque, lo que sugiere su aplicabilidad en diversas situaciones

8.2. Trabajos futuros

Se identifican principalmente tres líneas de trabajo futuro para ampliar y mejorar el sistema desarrollado.

- **Sincronización Automática de Vídeos:** Esta línea de trabajo se enfocaría en la sincronización automática de vídeos, aprovechando los parámetros que se puedan extraer del módulo de búsqueda de correspondencias desarrollado en este proyecto. Es importante señalar que, aun empleando estos parámetros, no se garantiza la extracción de dos frames idénticos.
- **Integración en Aplicaciones de Gestión:** Otra línea de trabajo se centraría en la integración del sistema en una aplicación que facilite o incluso automatice la decisión en la administración del calibre del pienso. Esta integración permitiría una gestión más eficiente y precisa del suministro de alimento en los tanques acuícolas.
- **Validación del Sistema mediante Aprendizaje Supervisado:** Se propone la validación del sistema mediante un modelo de aprendizaje supervisado. En este enfoque, los rodaballos segmentados se utilizarían como conjunto de datos de entrenamiento para mejorar la precisión y fiabilidad del sistema.

Adicionalmente, se sugiere trabajar en las siguientes áreas para mejorar el desempeño del sistema:

- **Mejora de la Detección y Segmentación en Profundidad:** Implementar mejoras en la detección y segmentación de rodaballos situados en las profundidades del tanque, donde la iluminación es más deficiente y los objetos pueden estar oscurecidos.
- **Mejorar la detección de rodaballos aislados:** Aunque el sistema reconozca rodaballos aislados correctamente conociendo el área y la circularidad en la mayoría de los casos, se podrían usar métodos estadísticos en los que se incluyan más parámetros morfológicos del pez para aumentar la robustez del sistema.
- **Optimización de la Iluminación:** Mejorar la iluminación del entorno para que las cámaras puedan captar imágenes de mayor calidad, especialmente en las zonas más profundas del tanque.

Estas líneas de trabajo futuro no solo ampliarán la aplicabilidad del sistema, sino que también mejorarán su precisión y robustez en diversas condiciones operativas.

9. Referencias

- [1] M. Castillo, «Sistema de estimación de la biomasa en piscifactorías mediante imágenes RGB,» Facultad de Ciencias Matemáticas, Universidad Complutense de Madrid, Escuela Técnica Superior de Ingenieros de Telecomunicación, Universidad Politécnica de Madrid, Madrid, 2023.
- [2] M. R. Alvariño y O. D. Martínez, «Producción de alevines de rodaballo,» *Hojas Divulgadoras, Ministerio de Agricultura Pesca y Alimentación*, nº 3, pp. 10-22, 1990.
- [3] H. A. O. Reyes y C. A. P. Ramírez, «Visión por Computador, su historia y algunos principales hitos,» [En línea]. Available: https://revistaingenieriaaldia.ucentral.cl/rev_10/hernan_olmi.pdf.
- [4] «Stemmer Imaging,» [En línea]. Available: <https://es.stemmer-imaging.com/blog/vision-2d-3d/historia-evolucion-vision-artificial/>. [Último acceso: 8 Junio 2024].
- [5] «IBM,» [En línea]. Available: <https://www.ibm.com/es-es/topics/deep-learning>. [Último acceso: 8 Junio 2024].
- [6] «IBM,» [En línea]. Available: <https://www.ibm.com/es-es/topics/image-segmentation#:~:text=La%20segmentaci%C3%B3n%20de%20im%C3%A1genes%20es,objetos%20y%20otras%20tareass%20relacionadas>. [Último acceso: 2024 Junio 9].
- [7] Diego, «Medium,» [En línea]. Available: <https://medium.com/@di3cruz/detecci%C3%B3n-de-objetos-en-tiempo-real-con-python-y-opencv-e760fdaad58e>. [Último acceso: 9 Junio 2024].
- [8] F. Meta AI Research, «Segment Anything,» Meta AI, 2023.
- [9] «Telefónica Tech,» Telefónica, [En línea]. Available: <https://aiofthings.telefonicatech.com/recursos/datapedia/aprendizaje-auto-supervisado>. [Último acceso: 11 Junio 2024].
- [10] «IBM,» IBM, [En línea]. Available: <https://www.ibm.com/es-es/topics/supervised-learning>. [Último acceso: 14 Junio 2024].
- [11] Z. Kelta, «Datacamp,» Enero 2024. [En línea]. Available: <https://www.datacamp.com/es/blog/yolo-object-detection-explained>. [Último acceso: 15 Junio 2024].

- [12] J. Redmon, S. Divvala, R. Girshick y A. Farhadi, «You Only Look Once: Unified, Real-Time Object Detection,» 2016.
- [13] J. D. H. S. J. K. S. S. S. M. Z. H. A. K. A. K. M. B. A. C. B. a. L. F.-F. O. Russakovsky, «ImageNet Large Scale Visual Recognition Challenge,» *International Journal of Computer Vision (IJCV)*, 2015.
- [14] D. Bergmann, «IBM,» IBM, 12 Enero 2023. [En línea]. Available: <https://www.ibm.com/es-es/topics/fine-tuning>.
- [15] N. A. Dobernack, «Biblioteca de la Universidad de Sevilla,» [En línea]. Available: https://biblus.us.es/bibing/proyectos/abreproy/12112/fichero/Documento_por_capitulos%252F3_Cap%C3%ADtulo_3.pdf. [Último acceso: 16 Junio 2024].
- [16] C. D. Martín, «Análisis de imágenes-Segmentación,» Madrid.
- [17] D. Lowe, «Distinctive Image Features from Scale-Invariant Keypoints,» *International Journal of Computer Vision*, 2004.
- [18] H. T. T. & V. G. L. Bay, «"SURF: Speeded Up Robust Features",» de *European Conference on Computer Vision*, 2006.
- [19] E. R. V. K. K. & B. G. Rublee, «"ORB: An efficient alternative to SIFT or SURF",» de *International Conference on Computer Vision 2011*.
- [20] N. & T. B. Dalal, «"Histograms of Oriented Gradients for Human Detection",» de *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2005.
- [21] M. K. Hu, «"Visual Pattern Recognition by Moment Invariants",» de *IRE Transactions on Information Theory*, 1962.
- [22] K. & Z. A. Simonyan, «"Very Deep Convolutional Networks for Large-Scale Image Recognition",» arXiv preprint arXiv:1409.1556, 2014.
- [23] J. G. I. L. Y. S. E. & S. R. Bromley, «"Signature Verification using a 'Siamese' Time Delay Neural Network",» de *Advances in Neural Information Processing Systems*, 1994.
- [24] F. K. D. & P. J. Schroff, « "FaceNet: A unified embedding for face recognition and clustering",» de *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.
- [25] I. P.-A. J. M. M. X. B. W.-F. D. O. S. .. & B. Y. Goodfellow, «Generative Adversarial Net,» de *Advances in Neural Information Processing Systems*, 2014.
- [26] J. P. Lewis, «Fast Template Matching,» de *Vision Interface*, 1995.

- [27] R. Szeliski, *Computer Vision: Algorithms and Applications.*, Springer, 2010.
- [28] N. Ravi, «GitHub,» Microsoft, 2023. [En línea]. Available: <https://github.com/facebookresearch/segment-anything/blob/main/README.md>. [Último acceso: 26 Abril 2024].
- [29] G. J. Chaoning Zhang, «Ultralytics,» 10 Junio 2024. [En línea]. Available: <https://docs.ultralytics.com/models/mobile-sam/#box-prompt>. [Último acceso: 18 Junio 2024].
- [30] C. Zhang, «GitHub,» Microsoft, 20 Diciembre 2023. [En línea]. Available: <https://github.com/ChaoningZhang/MobileSAM>. [Último acceso: 7 Mayo 2024].
- [31] N. Developer, «CUDA Toolkit Documentation,» [En línea]. Available: <https://docs.nvidia.com/cuda/>.
- [32] Campbeln, «Stack Overflow,» 1 Mayo 2012. [En línea]. Available: <https://stackoverflow.com/questions/10392658/calculate-the-bounding-boxes-x-y-height-and-width-of-a-rotated-element-via-jav>. [Último acceso: 18 Junio 2024].
- [33] «Wikipedia,» 6 Mayo 2021. [En línea]. Available: https://es.wikipedia.org/wiki/Correlaci%C3%B3n_cruzada. [Último acceso: 12 Junio 2024].
- [34] «MathWorks,» [En línea]. Available: <https://es.mathworks.com/help/signal/ref/xcrr2.html>. [Último acceso: 14 Junio 2024].
- [35] «OpenCV - Open Computer Vision Library,» [En línea]. Available: https://docs.opencv.org/4.x/d4/dc6/tutorial_py_template_matching.html. [Último acceso: 25 Mayo 2024].
- [36] L. P. Ramos, «Real Python,» [En línea]. Available: <https://realpython.com/python-counter/>. [Último acceso: 27 Mayo 2024].
- [37] S. A. C.-C. B. M.-D. D. G.-I. L. L. C.-A. Carlos Vicente Niño-Rondón, «Comparativa entre la técnica de umbralización binaria y el método de Otsu para la detección de personas,» *Revista UIS ingenierías*, vol. 20, nº 2, pp. 65-74, 16 Julio 2020.
- [38] «OpenCV,» [En línea]. Available: https://docs.opencv.org/4.x/d4/d73/tutorial_py_contours_begin.html. [Último acceso: 19 Mayo 2024].

- [39] «Ministerio de Agricultura Pesca y Alimentación,» [En línea]. Available: https://www.mapa.gob.es/es/ministerio/servicios/informacion/rodaballo_tcm30-102478.pdf. [Último acceso: 28 Mayo 2024].
- [40] A. A. C. Ortiz, «GitHub,» Microsoft, 30 Junio 2024. [En línea]. Available: <https://github.com/andresorc/Proyecto>.
- [41] «HP,» [En línea]. Available: <https://www.hp.com/co-es/shop/portatil-hp-pavilion-gaming-15-ec1025la-162d6la.html>. [Último acceso: 1 Julio 2024].
- [42] «Google Colab,» [En línea]. Available: <https://colab.research.google.com/signup>.
- [43] «jooble,» [En línea]. Available: <https://es.jooble.org/salary/reci%C3%A9n-graduado/Espa%C3%B1a#hourly>.
- [44] «Naciones Unidas,» [En línea]. Available: <https://www.un.org/sustainabledevelopment/es/objetivos-de-desarrollo-sostenible/>. [Último acceso: 22 Junio 2024].

Anexos

Tabla 3: Tiempo de ejecución de la inferencia de MobileSAM según los parámetros de *points_per_side* y *crop_n_layers*

Puntos de muestreo	Capas adicionales	Tiempo de ejecución(s)
32	0	72
32	1	162
32	2	223
64	0	287
64	1	314
64	2	373

Tabla 4: Número de máscaras a la salida del modelo MobileSAM según los parámetros de *points_per_side* y *crop_n_layers*

Puntos de muestreo	Capas adicionales	Nº de máscaras
32	0	227
32	1	348
32	2	426
64	0	485
64	1	523
64	2	588