PROYECTO FIN DE GRADO

TÍTULO: Detección y medición automática de rodaballos en tanques de piscifactoría con un sistema de visión estereoscópica.

AUTOR: Jose Ignacio Toro Coronado

TITULACIÓN: Ingeniería Telemática

TUTORA: Juana María Gutiérrez Arriola

DEPARTAMENTO: Ingeniería Telemática y Electrónica (DTE)

VºBº TUTORA

Miembros del Tribunal Calificador:

PRESIDENTE: Rafael Delgado López

TUTORA: Juana María Gutiérrez Arriola

SECRETARIO: Víctor José Osma Ruiz

Fecha de lectura: 23 de Julio de 2024

Calificación:

El Secretario,

Resumen

Este proyecto, titulado “Detección y medición automática de rodaballos en tanques de piscifactoría con un sistema de visión estereoscópica”, se enmarca en el contexto de la acuicultura, en el escenario de la cría de rodaballos en tanques.

En esta industria desarrollada, competitiva y de ámbito global, las compañías tratan de encontrar una ventaja que las separe de la competencia y para ello invierten tiempo y dinero en el desarrollo tecnológico de sus instalaciones y técnicas. Dentro de esta línea, la estimación de la biomasa resulta un tema fundamental, puesto que permite predecir adecuadamente la demanda de alimentación, evitando que los acuicultores sobrealimenten o subalimenten a los peces que están criando.

Situado en este contexto, el objetivo de este proyecto es proporcionar una estimación automática de la longitud de rodaballos que se crían en tanques de acuicultura, definiendo la longitud como su tamaño desde la punta de la cabeza hasta la punta de la cola. Este proyecto es la continuación de otro que se desarrolló en la universidad “Sistema de estimación de la biomasa en piscifactorías mediante imágenes RGB” [1], y que se referenciará en alguna ocasión durante el documento.

El proyecto utiliza un sistema de visión estereoscópica de dos cámaras no sincronizadas entre sí, situadas encima de un tanque del que se quieren medir los peces. De las cámaras se obtienen los vídeos que luego se procesan de forma automática para hallar la longitud de los peces que se encuentran en el tanque. Este proyecto no está pensado para funcionar en tiempo real.

A la conclusión del proyecto se ha obtenido un sistema que obtiene, de forma automática, mediciones de algunos de los rodaballos que han sido captados por el par de cámaras. El resultado se muestra mediante una aplicación de escritorio que toma como entrada un par de imágenes de cámaras previamente calibradas y unos archivos relacionados con esta calibración. La aplicación permite corregir la distorsión de las imágenes y rectificarlas, y después obtener automáticamente la medida de algunos rodaballos, generando información de utilidad para el usuario.

A cerca de las medidas tomadas, resulta complicado cuantificar el error total de los resultados, puesto que no se cuenta con verdad fundamental que tomar como referencia. No obstante, se puede garantizar que el procedimiento propuesto no introduce un error significativo respecto al experimento teórico que se realizó en el anterior trabajo, en el que sí se contaba con verdad fundamental.

Al final se ha obtenido una aplicación de interfaz sencilla y que provee información útil para los trabajadores, que puede servir de utilidad en un ambiente realista a falta de una comprobación de los resultados que los compare con una verdad fundamental.

Abstract

The project, titled “Automatic detection and sizing of turbots in pisciculture tanks using a stereo vision system”, is framed in the context of aquaculture, it is considered for turbot raise in pisciculture tanks.

In a far developed, global and competitive industry such as aquaculture, companies strive to find a competitive edge that separates them from the rest of the competitors. To do so, they invest time and money into technologic development of their facilities and techniques. In this development, biomass estimation is a crucial topic, since it allows to predict food demand of the farmed fishes, thus avoiding overfeeding or underfeeding the animals.

In this context, the aim of the project is to provide an automatic length estimation of turbot larvae being raised in pisciculture tanks, length being defined as their size from the tip of the head to the tip of the tail. This project serves as a continuation to another that was previously developed in the university *“Sistema de estimación de la biomasa en piscifactorías mediante imágenes RGB”* [1], which will be referenced during the document occasionally.

The solution uses a stereo vision system of two non-synchronized cameras, placed on top of the tank whose fish are wanted to size. From the cameras we obtain the videos that are later automatically processed to size the fish on the tank. This project does not contemplate real-time measuring.

At the end of the project, it has been obtained a system that is able to size, automatically, some of the turbots captured on both cameras. The result is shown through a desktop application that takes a pair of images from the cameras (previously calibrated) and files related to the calibration. The app allows to undistort and rectify the given images, and after automatically size some of the turbots, generating useful information for the user, in addition.

About the results on the measurements, it is difficult to quantify the error on the obtained results, since we do not have ground truth to compare with. Nevertheless, we can grant that the proposed procedure does not introduce a significant error comparing to the experiment studied in the prior work, in which they did have ground truth and they managed to obtain positive results.

At the conclusion of the project, it has been obtained a user application with a friendly interface, that provides useful information and could be used by workers in a real-world scenario, pending to prove the results against a ground truth.

ÍNDICE DE FIGURAS

Índice de contenidos

1. Introducción

1.1 Marco y motivación del proyecto

Para el desarrollo de la actividad de la acuicultura, la estimación de la biomasa de los tanques o jaulas es fundamental. Además de ayudar al acuicultor a conocer la cantidad de producto que está produciendo, también ayuda a predecir la demanda de alimentación de los animales y a controlar la densidad de las poblaciones dentro de un área. Para ello, la biomasa se calcula típicamente multiplicando el número de peces en un área concreta por el peso medio de estos. El peso se puede calcular a partir del volumen o el tamaño de los peces, dependiendo de la información de la que se disponga [2]. Así, los sistemas de estimación de la biomasa se centran en calcular la cantidad, el volumen, o el tamaño de los animales en un área delimitada.

De esta forma, con el objetivo de proveer información fiable, automática y en muchas ocasiones a tiempo real, existen soluciones en el mercado que afrontan este problema. Sin embargo, el problema que encuentran estas soluciones es que habitualmente están diseñadas para funcionar en un entorno concreto, y son poco trasladables a otros contextos. Por eso es necesario obtener soluciones para los distintos entornos en los que se practica la acuicultura.

En este proyecto se plantea un sistema para estimar el tamaño de larvas de rodaballo criadas en piscinas de 50 cm de profundidad y en un entorno de interior con buena iluminación.

Con ese objetivo, la solución propuesta en este proyecto utiliza un sistema de visión estereoscópica conformado por dos cámaras IP. En la **Figura 1** [3] se muestra un esquema de un sistema de visión estereoscópica como el que se utiliza.

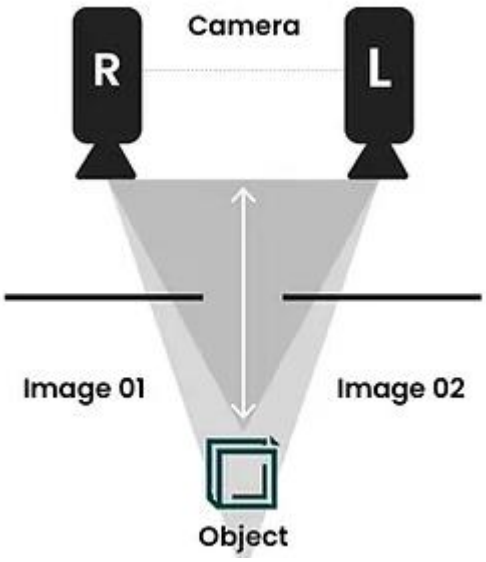

Figura 1: Sistema de visión estéreo de dos cámaras

1.2 Objetivos técnicos y académicos

Los objetivos de este proyecto, desde el punto de vista técnico son:

Objetivo 1: obtención de las medidas de algunas de los alevines de rodaballo contenidos en un tanque de acuicultura.

Objetivo 2: detección y segmentación automática de los rodaballos que se vayan a medir.

Objetivo 3: sincronización automática de dos vídeos provenientes de la granja de rodaballos.

Objetivo 4: desarrollo de una aplicación de usuario fácil de utilizar que presente la solución de forma sencilla y útil para los piscicultores.

Desde el punto de vista académico, durante el desarrollo de este proyecto se han cumplido los siguientes objetivos:

Objetivo Académico 1: Organización y gestión del primer proyecto de ingeniería.

Objetivo Académico 2: Planificación de la solución necesaria para resolver un proyecto de ingeniería.

Objetivo Académico 3: Redacción de un documento técnico que describe un proyecto de ingeniería.

Objetivo Académico 4: Introducción y familiarización con los sistemas de visión estereoscópica y como emplearlos para calcular distancias a partir de imágenes.

Objetivo Académico 5: Introducción y familiarización con el etiquetado de imágenes.

Objetivo Académico 6: Aprendizaje y familiarización con el lenguaje Python y el entorno de desarrollo Visual Studio.

Objetivo Académico 7: Familiarización con la biblioteca OpenCV y con el procesado de imágenes y de vídeo.

Objetivo Académico 8: Acercamiento a la Inteligencia Artificial y al uso de esta por medio de la preparación de un modelo de YOLO.

1.3 Estructura del resto de la memoria

A partir de aquí la memoria se organiza en otros ocho capítulos.

El que sigue es el **Marco Tecnológico**, que es una introducción teórica que servirá de guía el resto del proyecto. En este capítulo se presenta un estado del arte de la estimación de la biomasa en acuicultura, seguido de un resumen teórico de las tecnologías que se han utilizado para desarrollar el proyecto.

Después, en el capítulo de **Especificaciones y Restricciones** se exponen los requisitos y limitaciones que se han tenido presentes desde el comienzo del proyecto.

Más tarde se presenta la **Solución Propuesta**, en la que se explica de forma detallada los pasos que se han seguido para desarrollar el proyecto. La solución se expone bloque a bloque y de forma secuencial, con la finalidad de que resulte más sencilla de leer.

En los **Resultados,** se presentan tanto los resultados finales como el error que introduce cada bloque sobre el resultado final. También se presentan resultados parciales, como los resultados del modelo de Inteligencia Artificial para detectar rodaballos medibles.

Después se presenta el **Presupuesto**. En este capítulo se calcula de forma breve el coste de desarrollo del proyecto, teniendo en cuenta tanto el coste material como el coste de personal.

En el capítulo de **Impacto Social** se reflexiona acerca del impacto de la acuicultura en la sociedad, tomando como referencia los Objetivos de Desarrollo Sostenible aprobados en 2015 por la Asamblea General de las Naciones Unidas. Para ello se utilizan datos proporcionados también por las Naciones Unidas en su informe de 2022.

Le siguen las **Conclusiones**, donde se ponen en contexto los resultados y se reflexiona sobre la viabilidad de la solución en un entorno práctico.

El último capítulo se ha dedicado a las **Referencias** que se han utilizado para desarrollar el proyecto y para escribir esta memoria.

Por último, en los **Anexos** se pueden encontrar algunas tablas y figuras que sirven para apoyar la demostración de los resultados.

2. Marco tecnológico

Tradicionalmente la estimación de la biomasa en acuicultura es un proceso manual, lento y laborioso, que además implica intervenir con la rutina de los peces, y que puede causar daño físico y estrés, importunando el desarrollo de los animales [4].

Con el objetivo de evitar estos inconvenientes, desde hace décadas se desarrollan métodos de estimación automáticos y no intrusivos, como el que se propone en este proyecto.

En general, los sistemas de estimación de la biomasa incluyen los sistemas de contado de ejemplares, sistemas de estimación del tamaño o sistemas de estimación del volumen o de la densidad. A continuación, se introducen algunas de las posibles soluciones que se investigan hoy en día para resolver el problema de estimación de la biomasa.

2.1 Estimación de la biomasa en acuicultura: estado del arte

2.1.1 SISTEMAS DE VISIÓN POR COMPUTADORA.

Una solución común a este problema son los sistemas de visión por computadora, en los que se utiliza la luz como fuente de información para hacer las mediciones.

Este proyecto se encuentra dentro de esta línea de investigación.

Estos sistemas se pueden englobar en pequeños subgrupos, dependiendo de la tecnología que utilicen.

Sistemas estereoscópicos.

Utilizan un par de cámaras para determinar la profundidad a la que se encuentran los peces de los que se quiere obtener el tamaño, comparando la imagen desde las distintas perspectivas que se han tomado, como se hace en este proyecto.

Actualmente existen soluciones en el mercado que aplican esta solución, como por ejemplo BioMass Pro, un producto que ofrece la empresa InnovaSea. Se trata de una cámara estereoscópica subacuática y autónoma que procesa las imágenes utilizando inteligencia artificial y envía los resultados a un centro de control a través de Internet. En la **Figura 2** [5] se muestra un par de cámaras BioMass Pro.

Figura 2: BioMass Pro

Estos sistemas tienen la ventaja de que son fáciles de aplicar en el terreno y son baratos de implementar.

Sistemas de una sola cámara.

En los sistemas de una sola cámara, a diferencia de los sistemas de visión estereoscópica, la distancia a la que se encuentran los objetivos debe ser conocida con anterioridad. Para obtener el tamaño de los peces pasando de píxeles a metros, se puede introducir en el agua un objeto de referencia de tamaño conocido con el que comparar [6].

Sistemas basados en luz infrarroja.

Se pueden dividir en dos grupos:

Escáner infrarrojo

Consisten en situar un escáner infrarrojo en el área en el que se quieran medir los peces para obtener una silueta. Desde uno de los extremos se proyectan rayos de luz infrarroja hacia el otro que los detecta. De esta forma si un objeto pasa por el escáner, se puede obtener su silueta basándonos en los rayos que no han llegado a proyectarse de un lado a otro del escáner. En la siguiente Figura 3 [4] aparece un diseño esquemático de uno de estos sistemas.

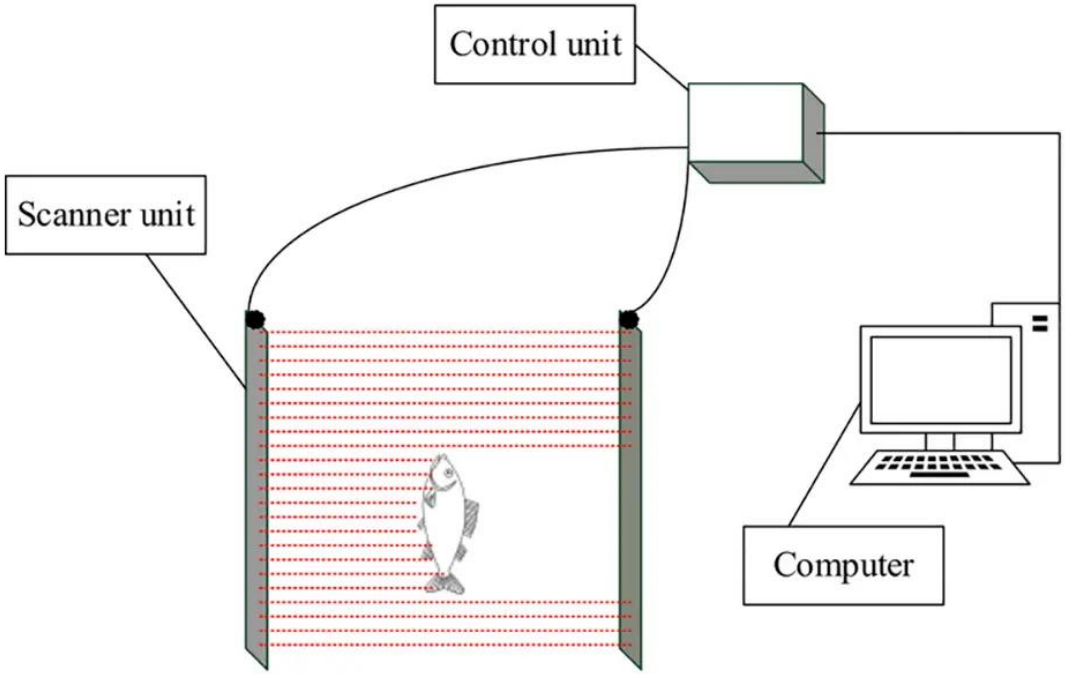

Figura 3: Esquema de escáner infrarrojo para estimación de la biomasa

Tienen la ventaja de que pueden funcionar en entornos con poca visibilidad, pero su rendimiento decae en aguas turbias. Esto se debe a que la luz infrarroja no cuenta con mucha distancia de penetración en el agua [7], un fenómeno que se ve agravado si además el agua está enturbiada. Por eso el escáner debe ser estrecho para que los rayos alcancen el otro extremo con intensidad.

El escáner tampoco puede discriminar entre peces aislados y objetos que no lo sean.

LiDAR

Se puede usar un escáner láser con la tecnología LiDAR para obtener una estimación del volumen de peces de un tanque de acuicultura [8]. Después se puede calcular la cantidad de biomasa multiplicando por la densidad de los peces.

El láser tiene más penetración que el escáner infrarrojo, pero sigue siendo susceptible a la turbiedad del agua.

2.1.2 SISTEMAS ACÚSTICOS.

Utilizan ondas de presión para obtener la información necesaria. A diferencia de la luz, las ondas de sonido pueden viajar largas distancias a través del agua. Pueden funcionar en aguas turbias u oscuras, por lo que pueden ser útiles para acuicultura en alta mar.

Al igual que en los sistemas de visión por computadora, se puede hacer una clasificación de distintos tipos de sistemas acústicos.

Sonda de eco: se emite un pulso de sonido y se obtiene la onda reflejada. No producen imágenes nítidas, pero se pueden utilizar para estimar la densidad de biomasa. [4]

Sonar de imágenes: permite generar imágenes emitiendo pulsos de ultrasonido [9]. En el mercado existen compañías como Optimar que ofrecen esta solución, mediante la comercialización de su producto Aquaculture Biomass Monitor. Este producto preparado para acuicultura en jaulas flotantes es capaz de estimar el tamaño de los peces, entre otras cosas [10].

En la siguiente Figura 4 [10] se muestra una imagen de Aquaculture Biomass Monitor, que es un dispositivo flotante que se coloca encima de la jaula de la que se quiere obtener información.

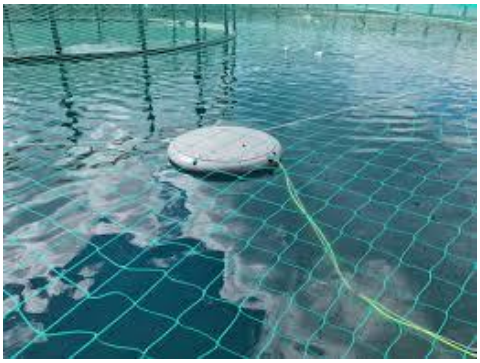

Figura 4: Aquaculture Biomass Monitor

Hidrófonos: se pueden introducir hidrófonos en el agua para tratar de determinar la densidad de peces que hay basándose en el ruido que producen.

2.1.3 OTROS MÉTODOS

Contador de peces utilizando su resistencia: como los peces tienen mayor resistividad que el agua, se ha propuesto medir la cantidad de peces que pasan entre dos electrodos comprobando las variaciones en la intensidad que producen [11].

Métodos basados en el análisis del agua: se ha utilizado el análisis de muestras de agua para probar que existe una relación entre la cantidad de ADN que liberan los organismos en su entorno y la densidad de población que hay en ese entorno [12]. Más tarde se ha probado que también existe relación entre la cantidad de ADN y el tamaño de los peces [13].

Estas son las posibles soluciones que se proponen actualmente para resolver el problema de la estimación de biomasa en acuicultura. En adelante, se introducen las tecnologías que se han utilizado durante la realización de este proyecto.

2.2 Tecnologías utilizadas durante el proyecto

2.2.1 Sistema de visión estereoscópica

A la capacidad de las personas de percibir imágenes en 3D se le llama visión estereoscópica. Esto se consigue gracias al cerebro, que es capaz de interpretar las imágenes en 2D que capta cada uno de nuestros dos ojos como una sola imagen con información de profundidad [14].

Así, los sistemas estereoscópicos son aquellos que tratan de imitar esta capacidad, usando imágenes captadas por dos o más sensores de imagen [15]. La **Figura 5** [16] muestra una cámara estereoscópica Zed 2, de StereoLabs. Como se puede observar, esta cámara cuenta con dos sensores de imagen.

Figura 5: Cámara estereoscópica Zed 2

Gracias a sus múltiples sensores de imagen, que registran una misma imagen desde diferentes perspectivas, los sistemas estereoscópicos pueden calcular la profundidad a la que están los distintos objetos. Esto lo hacen fijándose en la diferencia de distancia horizontal a la que aparece un mismo objeto captado por ambos sensores. A esta diferencia se la conoce como "disparidad". Como se explicará en el apartado de *Solución Propuesta*, la disparidad es inversamente proporcional a la profundidad a la que se encuentra el objeto. La siguiente **Figura 6** [17] muestra como distintas profundidades determinan la disparidad que se registra en un sistema de dos cámaras.

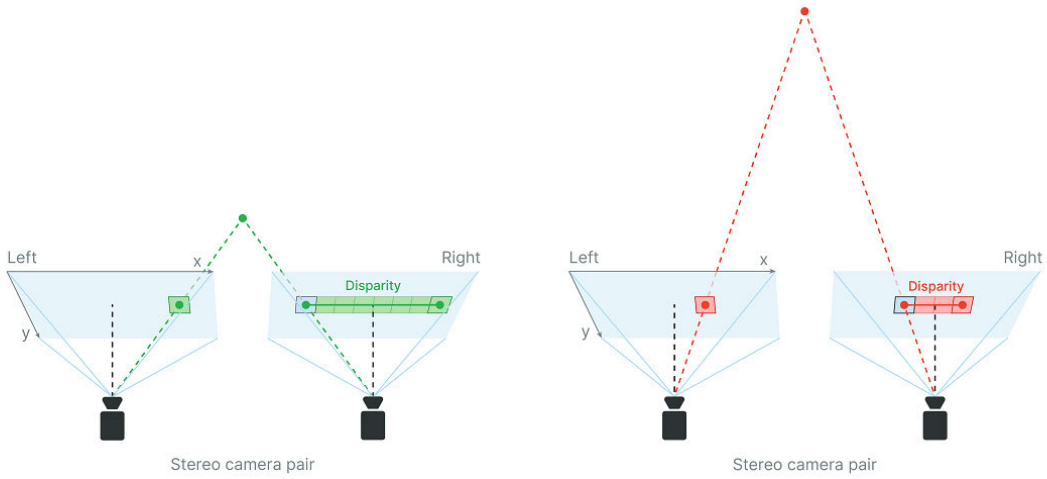

Figura 6: Disparidad en visión estereoscópica

A la distancia entre sensores se la conoce como “línea de base”. La elección de una mayor o menor distancia de línea de base determina la capacidad del sistema. Según la publicación [18] : “una línea de base más pequeña aumenta la precisión de los objetos más cercanos, pero también limita el rango de estimación de profundidad, ya que los objetos más allá de cierta distancia tendrán una disparidad cero o negativa, lo que los hace indistinguibles del fondo”.

Para facilitar el procesado de las imágenes captadas por el sistema estereoscópico, es fundamental realizar un preprocesado de las imágenes que se han captado. El preprocesado se encarga de **eliminar la distorsión** que introduce cada una de las lentes y después de **rectificar** el par de imágenes.

Ambos procesos requieren de una calibración previa que no se ha realizado durante este proyecto, sino que se ha tomado del anterior [1]. Sin embargo, es necesario presentarla para poder entender la totalidad del proyecto.

Corrección de la distorsión

Las lentes de las cámaras introducen dos tipos de distorsión: radial y tangencial. Para corregirla se capturan imágenes de un objeto del que se conocen sus proporciones auténticas, y después se comparan con las proporciones que ha capturado la cámara. El objetivo es obtener una matriz con los parámetros de distorsión de la cámara, y otra con los “parámetros intrínsecos” de la cámara (esta matriz contiene la distancia focal y el centro óptico). A este proceso se le conoce como “calibración” [18]. La **Figura 7** [19] expone los distintos tipos de distorsión.

Figura 7: Efectos de los distintos tipos de distorsión

Típicamente se utiliza como referencia una plantilla de ajedrez para realizar la calibración. El proceso es el siguiente:

1. Se toman imágenes de la plantilla, en distintas inclinaciones y con distinto ángulo de captura desde la cámara. Cuantas más mejor, y en [19]se indica que no deben ser menos que 10.
2. Se buscan los puntos de las imágenes para comparar con la referencia. En el caso del tablero de ajedrez se consideran las esquinas de cada casilla del tablero.
3. Se calculan las matrices requeridas comparando los puntos obtenidos con la referencia. Para hacerlo se deben resolver unas ecuaciones que no se presentan en este documento, pero que se pueden consultar en [19].

Una vez se tienen las cámaras calibradas se puede corregir la distorsión de las imágenes utilizando la información obtenida durante el proceso de calibración (es decir, la matriz de parámetros intrínsecos y la matriz de distorsión). Todos los pasos de este proceso se pueden realizar utilizando funciones de la biblioteca *OpenCV*. En la Figura, la imagen de abajo se ha corregido la distorsión radial de la imagen de arriba. En la imagen de arriba se puede observar como las líneas de la cancha de baloncesto se ven curvas, cuando no lo son.

Rectificación

El proceso de rectificación está directamente relacionado con el sistema estereoscópico, a diferencia de la corrección de la distorsión que se ha introducido anteriormente, que es característico de cada cámara.

La rectificación de las imágenes consiste en proyectarlas en un plano común, paralelo a la línea de base del sistema estereoscópico. El objetivo de este proceso es que los puntos comunes

de los objetos captados por ambas cámaras aparezcan en las mismas filas de píxeles (o dicho de otra forma, a la misma altura en ambas imágenes).

Para rectificar las imágenes, es necesario calibrar el sistema estereoscópico. Esta vez, el resultado de la calibración será una matriz de rotación "R" y otra de traslación "T". Estas matrices contienen los "parámetros extrínsecos" del par de cámara, que representan como de rotada y desplazada está una cámara respecto de la otra.

El proceso de calibración es similar al anterior, con la diferencia de que ahora ambas cámaras deben tomar imágenes de la plantilla al mismo tiempo. De nuevo este proceso se puede realizar utiliazndo la biblioteca de funciones *OpenCV.* En la **Figura 8** se muestra uno de los pares de imágenes que se ha usado para calibrar el sistema estereoscópico.

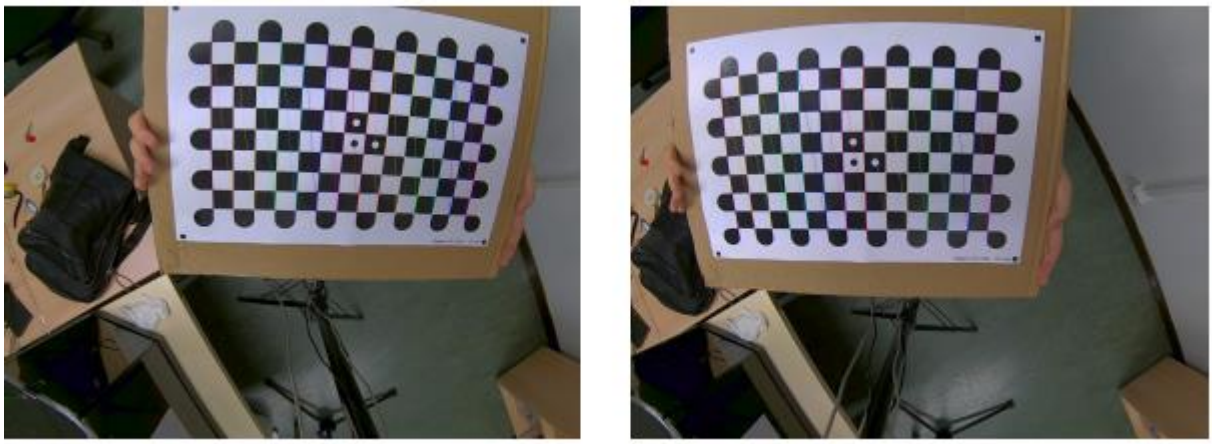

Figura 8: Imágenes tomadas para la calibración del sistema de visión estéreo

Tras obtener las matrices se puede rectificar la imagen. La **Figura 9** [20] muestra un par de imágenes sin rectificar (arriba) y rectificadas(abajo). En las imágenes rectificadas se puede comprobar que los puntos comunes se encuentran a la misma altura en ambas imágenes.

Figura 9: Ejemplo de imagen rectificada.

2.2.2 Algoritmos de comparación de imágenes

2.2.2.1 SIFT

Scale-Invariant Feature Transform (SIFT) es un algoritmo de que permite detectar puntos comunes entre dos imágenes. Se puede utilizar para hacer seguimiento de objetos, registro de imágenes, o para buscar un objeto plantilla en una imagen. SIFT permite encontrar objetos rotados y cambiados de escala.

SIFT fue propuesto en el año 2004 por David G. Lowe en su trabajo *Distinctive Image Features from Scale-Invariant Keypoints* [21].

El algoritmo de SIFT se puede explicar de forma resumida en 4 pasos [21]:

1. **Búsqueda de posibles puntos clave:** se localizan máximos y mínimos locales presentes en distintas escalas de desenfoque Gaussiano y a distintos tamaños de imagen. Así se puede asegurar que los puntos clave son invariables ante el cambio de escala, en caso de que vayamos a buscar por ejemplo una plantilla en una imagen en la que ha visto reducido u tamaño.
2. **Depuración de puntos clave**: se seleccionan los puntos clave más estables de los que se habían elegido.

3. **Asignación de orientación:** a cada punto clave se le asigna una orientación. Esto se hace calculando el gradiente de la imagen en ese punto. El gradiente se define como un cambio direccional en la intensidad o el color de la imagen. Obtener los gradientes permite conocer la rotación que ha sufrido un punto clave desde la imagen plantilla hasta la imagen objetivo, y se puede comparar con los gradientes de otros puntos clave para comprobar si realmente son puntos del mismo objeto (que habrá sufrido la misma rotación en todos sus puntos).

4. **Descriptor de los puntos clave:** finalmente se computa un descriptor de punto para cada uno de los puntos clave, que contiene información acerca de su gradiente. Los gradientes se escriben con una representación que es tolerante con cierto nivel de distorsión local y cambios en la iluminación.

Más tarde se pueden comparar los descriptores para comprobar los puntos clave que se pueden reconocer en la imagen objetivo.

La utilización de SIFT está soportada por la biblioteca *OpenCV* de Python. En la **Figura 10** [22], vemos un ejemplo de utilización de SIFT para encontrar puntos en común entre dos imágenes.

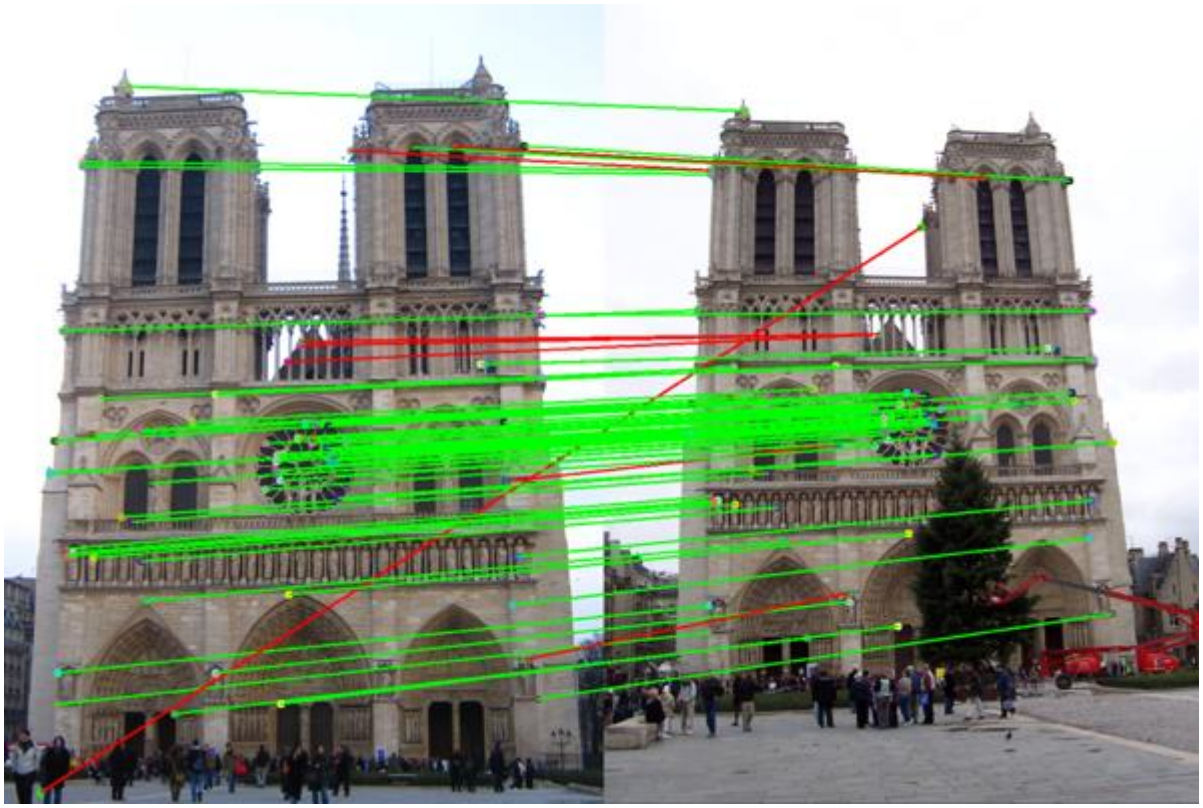

Figura 10: Búsqueda de puntos en común con SIFT

2.2.2.2 Método de la correlación cruzada

La correlación cruzada es una operación matemática que se puede utilizar para saber como de parecidas son dos series de datos. En el procesado de imágenes, se utiliza habitualmente para encontrar una imagen plantilla en una imagen objetivo, pero fuera de él tiene otros usos como por ejemplo para comparar tendencias de mercado [23].

La correlación cruzada entre *h* y *F* se expresa:

$$G = h \otimes F$$

Ecuación 1: Símbolo de la correlación cruzada

Y viene descrita por la siguiente fórmula matemática:

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} h[u,v]F[i+u, j+v]$$

Ecuación 2: Expresión de la correlación cruzada

En procesamiento de imágenes se puede interpretar que la imagen plantilla se va desplazando por cada uno de los puntos de la imagen objetivo (para cada píxel), obteniendo un coeficiente de correlación. En el ejemplo la imagen plantilla sería h y la imagen objetivo, *F*. Al final, el resultado de mayor parecido se corresponde con aquel que ha devuelto el mayor coeficiente de correlación.

2.2.3 YOLO

You Only Look Once (YOLO) es la tecnología que se ha utilizado para detectar los rodaballos aislados en la imagen de referencia. La empresa Ultralytics distribuye la última versión, YOLOv8, que es el algoritmo de estado del arte para detección y clasificación de objetos. Cualquier distribución de YOLOv8 viene incluida con la licencia AGPL-3.0, que permite la utilización de esta versión para proyectos de investigación. Las principales ventajas de YOLOv8 son:

Rapidez: YOLOv8 puede procesar desde 40 hasta 145 imágenes por segundo, por lo que puede funcionar en tiempo real en caso de que sea necesario [24].

Precisión: YOLOv8 es capaz de reconocer los objetos independientemente de la orientación que tengan y de su tamaño. Si bien, la precisión puede caer para objetos que aparezcan muy pequeños en la imagen [25].

Soporte: YOLOv8 está perfectamente integrado con Python gracias a la biblioteca *ultralytics*. Adicionalmente, cuenta con una gran comunidad de usuarios, y es fácil encontrar cursos de utilización y foros de consulta.

En cuanto a la parte técnica, YOLO es una red neuronal convolucional para detección de objetos que fue desarrollada en 2015 durante el proyecto *You Only Look Once: Unified, Real-Time Object Detection* [26]. Para poder entender mejor su funcionamiento, vamos a hacer una pequeña introducción previa a las redes neuronales.

Las redes neuronales son un tipo de algoritmo de software para Machine Learning que recibe unos datos en su capa de entrada, los procesa en sus capas intermedias (o capas ocultas) y devuelve la salida más probable en su capa de salida. Cada una de estas capas está formada por "neuronas" que son nodos que las conectan entre sí. Reciben este nombre porque su funcionamiento está basado en las neuronas que componen el cerebro. Cada neurona recibe una entrada a la que responde con un nivel de activación sobre sí misma. En una red neuronal, el nivel de activación de las neuronas de una capa determina el nivel de activación de cada una de las neuronas de la siguiente capa. Para eso, cada neurona cuenta con un "peso" sobre la activación de cada neurona concreta de la siguiente capa, que se hace efectivo solo si es mayor que un "umbral" específico de cada neurona. Las redes neuronales "entrenan" para mejorar sus resultados recibiendo datos de entrada y el resultado que se debería obtener a la salida del sistema. Este proceso de entrenamiento consiste en averiguar todos los pesos y umbrales para cada una de las neuronas de las capas intermedias que conforman la red, de forma que optimicen el resultado a la salida. En la **Figura 11** [27] se puede ver un esquema de la composición de las redes neuronales.

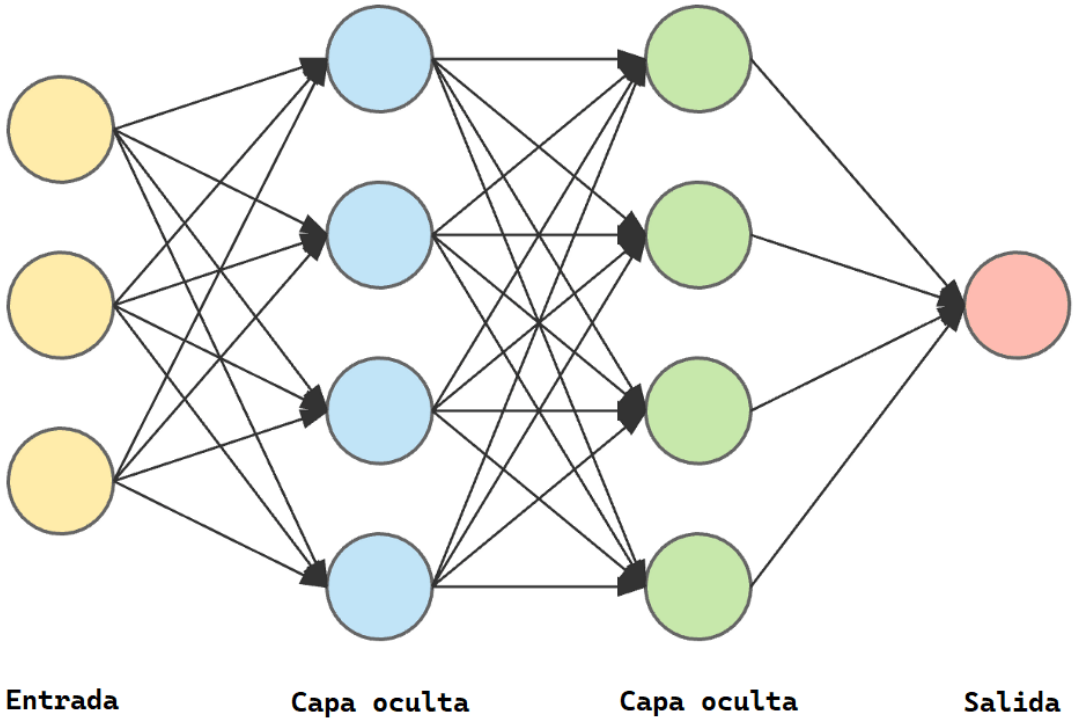

Figura 11: Esquema de red neuronal

Las redes neuronales convolucionales (CNN) son un tipo de red neuronal especialista en el procesado de imagen, audio o habla [28]. Estas redes son la mejor solución para los problemas de clasificación y detección de objetos en imágenes. Las CNN están conformadas por tres tipos de capas distintas: las capas convolucionales, las capas de agrupamiento y capas totalmente conectadas. La arquitectura puede intercalar capas convolucionales y de agrupamiento, pero al final siempre se sitúan las capas directamente conectadas. Las **capas convolucionales** aplican filtros de convolución sobre la entrada que reciben y devuelven un mapa de características que resalta la imagen del filtro sobre la entrada que reciben. Estos mapas de características luego sirven de entrada a las siguientes capas convolucionales. Aunque en un principio los filtros sólo son capaces de detectar características sencillas (como diagonales, bordes o texturas) a medida que se aplican en las siguientes capas sobre los mapas de características obtenidos por las capas anteriores, los filtros empiezan a reconocer patrones más complejos. Por otro lado, las **capas de agrupamiento** resumen la información más relevante de los mapas de características y favorecen la abstracción de la información que aprende la red. Por último, las **capas directamente conectadas** se encargan de resolver el problema de clasificación empleando la información que aportan las capas anteriores. Durante el proceso de entrenamiento de estas redes se calculan los coeficientes y el umbral de cada uno de los filtros, así como los pesos y umbrales de las neuronas de la capa totalmente conectada.

En la **Figura 12** [26], está representada la arquitectura de Yolo. La imagen está obtenida del documento de 2015 que presentó la red por primera vez.

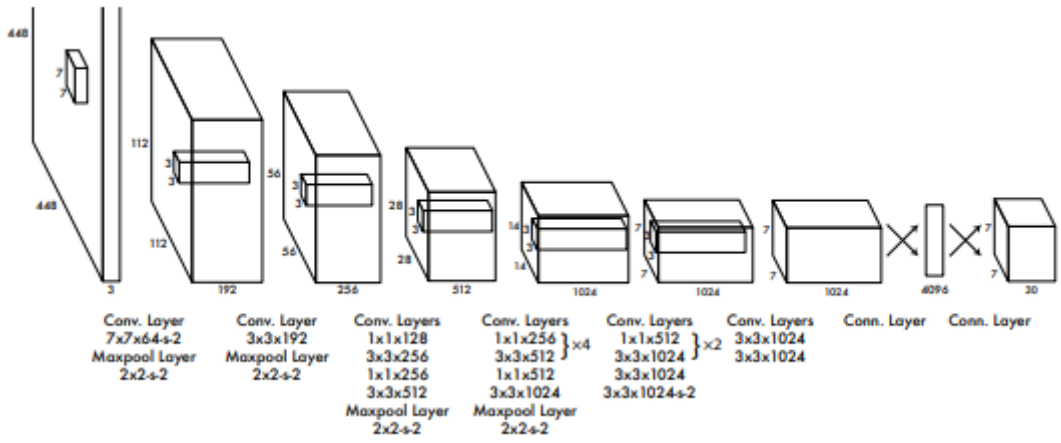

Figura 12: Arquitectura de la red YOLO

En [26] se explica que Yolo cuenta con 24 capas convolucionales (algunas de ellas intercalan capas de agrupamiento para reducir el tamaño de los mapas), seguidas de 2 capas totalmente conectadas.

De las 24 capas convolucionales, 20 están entrenadas previamente usando imágenes de ImageNet, y las últimas 4 son las que se ajustan según el entrenamiento de especialización que se les quiera dar.

La información que necesita la red YOLO para entrenar consiste en: un archivo YAML de configuración y una colección de imágenes etiquetadas en formato YOLO [29].

El archivo de configuración YAML debe indicar la ruta al directorio que contiene los datos, y las rutas específicas a los directorios con los datos de entrenamiento, validación y test. También debe contener una asociación de los números de las clases con los nombres que se les quiera dar.

Respecto las imágenes etiquetadas, a la hora de entrenar la red se debe contar con los archivos imágenes que se han etiquetado y con las etiquetas (en formato YOLO), almacenadas en archivos .txt. En total, se debe contar con un archivo de etiquetas por cada archivo de imagen.

Las imágenes etiquetadas se pueden obtener de bases de datos populares, como por ejemplo Common Objects in Context (COCO) u OpenImagesV7. Si bien, si se decide obtener las imágenes etiquetadas de una fuente externa, hay que asegurarse de modificar el formato de las etiquetas en caso de que sea necesario [29].

En caso de que se considere necesario también se pueden etiquetar las imágenes a mano, en un proceso que resulta bastante laborioso. Para ello se pueden utilizar aplicaciones como *labelme*, *labellmg* o *CVAT*. Estas aplicaciones reciben como entrada las imágenes que se han elegido para etiquetar, las muestran por pantalla y requieren que se seleccione sobre ellas la "caja delimitadora" que encierra el objeto de la clase concreta que se ha detectado. La caja delimitadora es el rectángulo de menor tamaño que contiene el objeto en la imagen.

En la **Figura 13** [29] se muestra un ejemplo de una imagen etiquetada, con su correspondiente archivo de etiquetas.

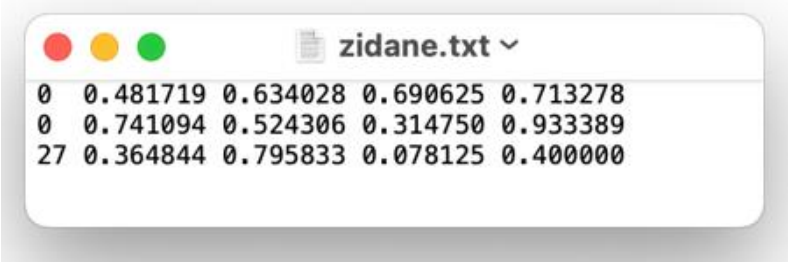

Figura 13: Ejemplo de imagen etiquetada según la web de Ultralytics

Este ejemplo puede servir para explicar el formato YOLO. Si nos fijamos en la **Figura 13** observamos que el archivo cuenta con tres filas, donde cada fila corresponde a una clase que se ha detectado y etiquetado. Cada fila cuenta con 5 columnas, donde cada una representa, de izquierda a derecha:

La clase que se ha etiquetado: representada por un número.

La coordenada abcisa del centro de la caja delimitadora: expresada con un número entre 0 y 1, relativo a la anchura total de la imagen.

La coordenada ordenada del centro de la caja delimitadora: expresada con un número entre 0 y 1, relativo a la altura total de la imagen.

La anchura de la caja delimitadora: expresada con un número entre 0 y 1, relativo a la anchura total de la imagen.

La altura de la caja delimitadora: expresada con un número entre 0 y 1, relativo a la altura total de la imagen.

Esta es la información que deben contener los archivos .txt en formato YOLO. En este caso el archivo .txt las dos primeras filas representan objetos de la clase 0:“Persona” y la tercera representa un objeto de la clase 27: “Corbata”.

3. Especificaciones y restricciones de diseño

El proyecto debe cumplir con las siguientes especificaciones y restricciones:

Especificación 1: El sistema estereoscópico que se utilice debe ser el mismo que se propuso en el proyecto anterior.

Especificación 2: El sistema toma como entrada dos grabaciones de los peces, vistos desde arriba.

Especificación 3: La sincronización de los vídeos (emparejamiento de las imágenes correspondientes al mismo instante) debe hacerse automáticamente.

Especificación 4: El procesado de los vídeos debe hacerse de forma automática usando Python.

Especificación 5: Se deben detectar y escoger automáticamente los rodaballos que se van a medir.

Especificación 6: Se debe obtener la medida de tantos rodaballos como resulte posible, aunque no es necesario medirlos todos.

Especificación 7: Las medidas deben tener el mínimo error posible.

Restricción 1: No es posible recalibrar las cámaras durante la realización del proyecto.

Restricción 2: El número de grabaciones es limitado.

Restricción 3: El entorno y la fuente de las grabaciones no se pueden adaptar.

4. Descripción de la solución propuesta

El sistema que se ha utilizado para resolver el proyecto consta de dos cámaras separadas 16 cm entre sí, conformando un sistema estereoscópico. Las cámaras se han situado encima de un tanque de acuicultura, en un ángulo perpendicular a la superficie de la piscina, y a una altura suficiente para poder grabar una buena porción de ésta. Este montaje se puede observar en la **Figura 14**.

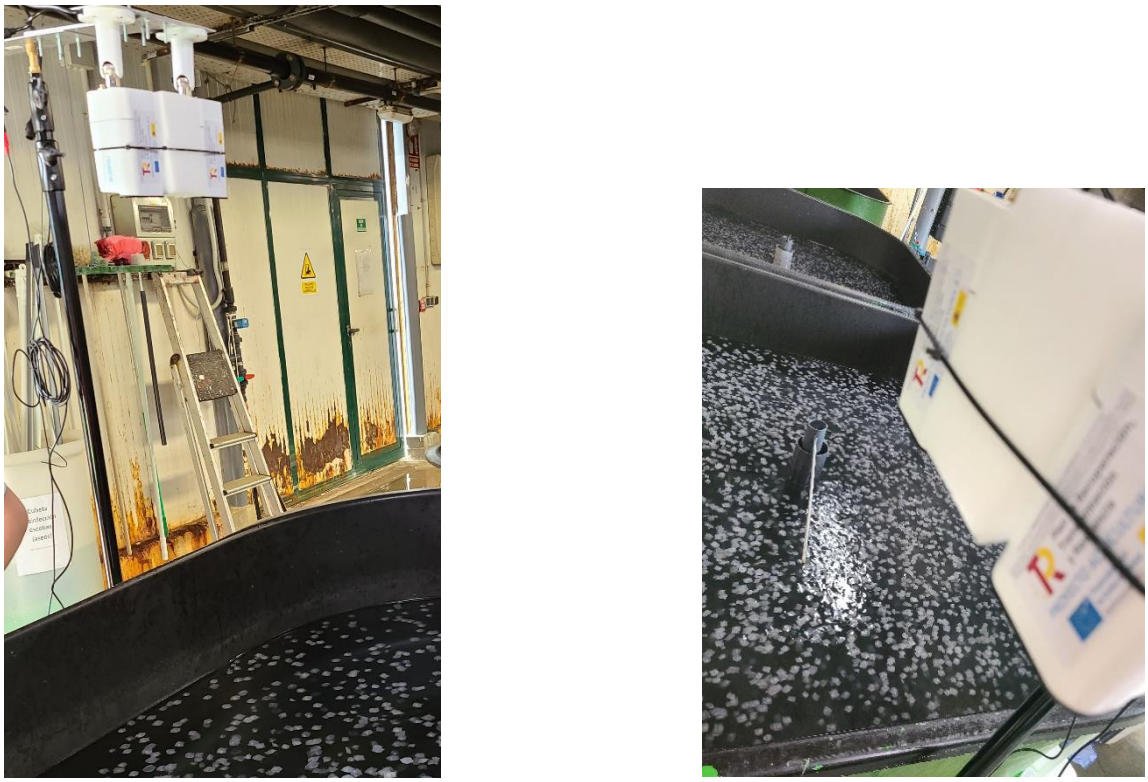

Figura 14: Cámaras del proyecto situadas en la piscifactoría de Vigo

Como se ha comentado anteriormente, utilizar el sistema estereoscópico para resolver el proyecto formaba parte de los prerrequisitos. Durante la realización de este proyecto no se ha modificado el sistema ni su configuración, y tampoco existía la posibilidad de hacerlo.

La necesidad de utilizar un sistema estereoscópico para resolver el problema que se propone radica en la obligatoriedad de conocer la distancia a la que están los rodaballos para poder estimar su tamaño real a partir de los vídeos que se han obtenido.

Las cámaras que se han utilizado para tomar las imágenes son del modelo Reolink 511 W.

Este proyecto utiliza un sistema que requiere una solución en distintas fases. Para facilitar la comprensión de este, podemos organizar el sistema en el siguiente diagrama de bloques (**Figura 15**).

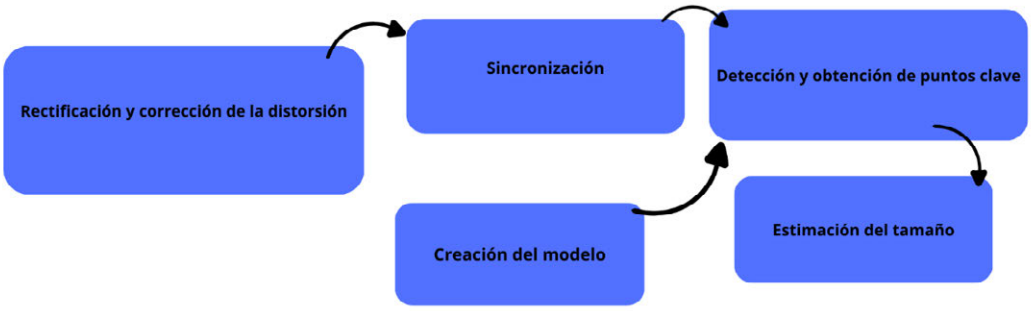

Figura 15: Diagrama de bloques de la solución

En el que la funcionalidad de cada uno de los bloques se puede resumir brevemente de la siguiente manera:

· **Rectificación y corrección de la distorsión**: se elimina la distorsión radial (efecto "ojo de pez") que introduce cada una de las cámaras a partir de unas matrices de calibración de las cámaras. Además, se rectifican las imágenes que obtiene nuestro sistema de dos cámaras a partir de otras matrices de calibración del sistema estereoscópico. Se obtienen y se guardan algunos parámetros del sistema de cámaras.

· **Sincronización:** Se sincroniza el vídeo estereoscópico para obtener al menos un par de capturas correspondientes al mismo instante de tiempo. Cada par contendrá una captura de la cámara izquierda, y otra captura de la cámara derecha.

· **Creación del modelo**: Preparación de un modelo capaz de reconocer y segmentar rodaballos que se puedan medir. Se han considerado como medibles los rodaballos que no se encuentran parcialmente ocultados por otros objetos o rodaballos y que resultan sencillos de segmentar de otros objetos o rodaballos que les rodeen.

· **Detección y obtención de puntos clave**: se detectan los rodaballos medibles (aislados) de las imágenes ya rectificadas, usando el modelo de reconocimiento de objetos que se ha preparado. Se obtienen los puntos clave para medir la longitud de los peces seleccionados (cabeza, cola y centro) y se genera un archivo con los datos.

· **Estimación de tamaño**: se miden todos los peces seleccionados a partir de los datos obtenidos y se genera información relevante acerca de su tamaño.

Por último, para presentar la solución se ha creado una aplicación de usuario con interfaz para escritorio.

Esta aplicación permite cargar un par de imágenes, rectificarlas y obtener medidas de algunos peces, comprobar la validez de los resultados y borrar los que se consideren oportunos, generar información útil para el usuario y descargarla.

La aplicación se ha creado en Python haciendo uso de la biblioteca *CustomTkinter.* Se ha diseñado utilizando una arquitectura de capas, facilitando comentarios que pueden ayudar a la lectura del código por si en algún momento se desea ampliarla o realizar alguna modificación.

4.1 Sincronización

Este bloque tiene la función de sincronizar los vídeos de ambas cámaras para obtener imágenes estereoscópicas de las que obtener medidas. Debido a que los peces están en constante movimiento, es esencial sincronizar los vídeos con la máxima precisión para obtener medidas precisas y que el movimiento del objeto no afecte al cálculo de disparidad.

Para hacerlo, se ha probado a utilizar una comparativa de imágenes, bajo la hipótesis de que el parecido será mayor cuanto más cerca temporalmente estén entre sí las capturas de una cámara y otra. El algoritmo que se ha utilizado para hacer esta comparativa es SIFT y la forma de medir el parecido es contar la cantidad de puntos en común que se detecten entre dos imágenes. Se ha escogido SIFT en lugar de otros algoritmos de comparativa de imágenes porque es resistente a la rotación de los objetos a la hora de calcular los puntos clave. En el momento de escogerlo, se pensó que esta característica ayudaría a lidiar con el cambio de perspectiva que se produce de una cámara a otra.

Para hacer la comparativa, primero se toma una imagen como referencia en la cámara izquierda, y después se compara con imágenes próximas (según las marcas de tiempo) en la cámara derecha. Finalmente se considera como contemporánea la imagen que tiene el mayor número de puntos en común con la referencia. En la **Figura 16** se están comparando dos imágenes tomadas por cada una de las dos cámaras.

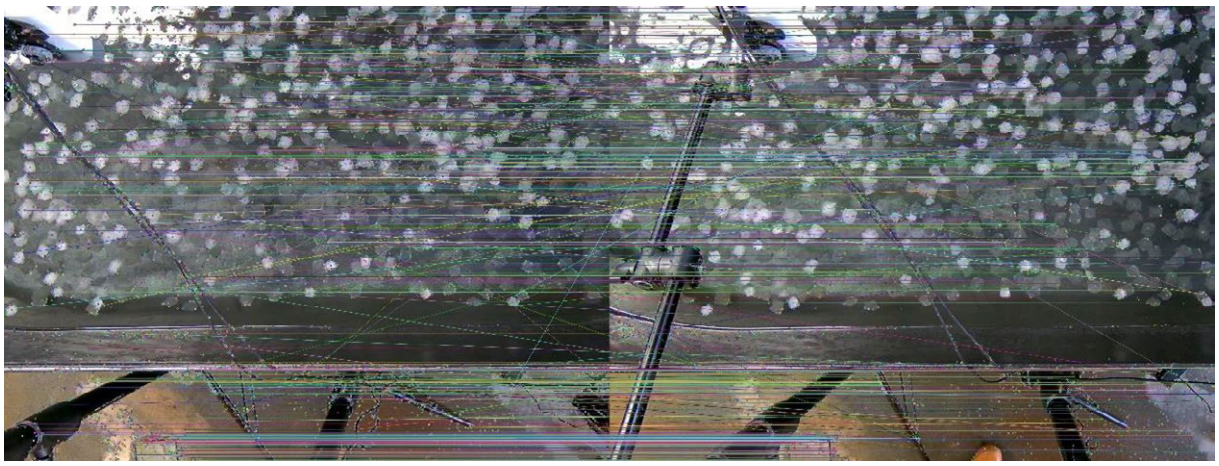

Figura 16: Comparativa de imágenes con SIFT

Sin embargo, este sistema no ha dado buen resultado ya que resulta demasiado rígido para comparar una imagen con otra con un cambio de perspectiva, y con la distorsión en las distintas distancias que se capturan por efecto de la disparidad. Se ha demostrado que no hay una relación directa entre el número de puntos en común y la cercanía temporal entre dos imágenes captadas cada una por una de las cámaras. En el capítulo *Resultados* se proporciona más información detallada del resultado de la utilización de este algoritmo.

A partir de aquí, y para poder aportar una solución al resto del problema que se plantea, se considera como punto de partida una imagen estereoscópica que se ha obtenido sincronizando los vídeos manualmente en base a su parecido.

4.2 Rectificación y corrección de la distorsión

El bloque de rectificación y corrección de la distorsión tiene la función de preparar las imágenes ya sincronizadas para facilitar el cálculo de la profundidad y de evitar introducir error en el resultado final por culpa de la distorsión de las cámaras.

Este bloque se ha obtenido del trabajo anterior a este, y se ha modificado como se considera oportuno para que funcione en el contexto en el que se está utilizando.

Para llevar a cabo estas funciones se parte de las imágenes de calibración que se obtuvieron en el momento de la visita a la granja de piscifactoría. Tras eso se han obtenido los parámetros intrínsecos y extrínsecos de las cámaras utilizando un script del proyecto anterior.

A partir de la obtención de los parámetros, el proceso consta de tres fases:

1. **Obtener la transformada de rectificación para el sistema de las cámaras**: se obtiene la transformada necesaria haciendo uso de la función *stereoRectify()* de OpenCV[https://amroamroamro.github.io/mexopencv/matlab/]. Esta fase calcula la distancia focal del sistema estereoscópico y su línea de base. Estos son datos importantes que luego se tendrán en cuenta a la hora de calcular la profundidad de los objetos captados por el sistema estereoscópico.

2. **Obtener el mapa de corrección de la distorión y rectificación para cada una de las dos imágenes**: se obtiene un mapa de transformación para cada una de las dos imágenes captadas, haciendo uso de la función *initUndistortRectifyMap()* de OpenCV[], que toman como parámetro de entrada la transformada obtenida anteriormente, entre otros.

3. **Corregir las imágenes haciendo uso de los mapas obtenidos:** se usa cada uno de los mapas obtenidos sobre su respectiva imagen y se obtienen las imágenes rectificadas. Para ello se utiliza la función *remap()* de OpenCV[].

La **Figura 17** es un ejemplo del resultado de realizar este proceso sobre una imagen previamente sincronizada.

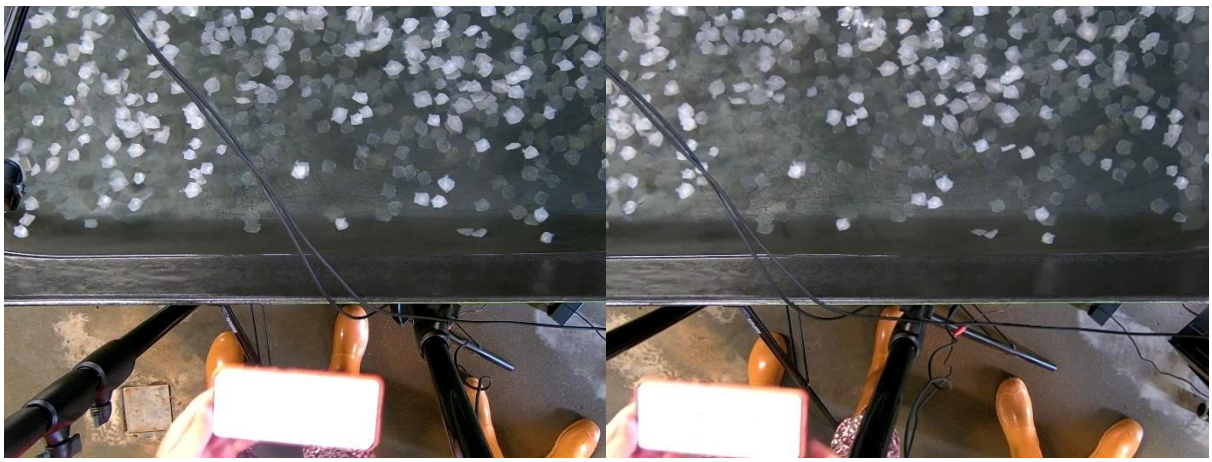

Figura 17: Imagen estéreo de los tanques rectificada

Además, tras la ejecución de este bloque se genera un archivo JSON (**Figura 18**) que resume información necesaria para calcular la longitud de los rodaballos más adelante, como por ejemplo la distancia focal calculada y la línea de base calculada, ambas del sistema estereoscópico.


```
{
    "focal_length": 2087.9831334685546,
    "baseline": 0.1575925481276722,
    "opt_cnt_x": 1676.0670318603516,
    "opt_cnt_y": 1118.675064086914
}
```


Figura 18: Parámetros del sistema estéreo

4.3 Creación del modelo.

Una vez obtenida una pareja de capturas correspondientes al mismo instante (una imagen estereoscópica), y tras haber corregido la distorsión introducida por el sistema de cámaras, el siguiente paso es detectar los peces que se vayan a medir en la imagen.

Para hacerlo se va a recurrir a Yolo, una red neuronal convolucional que se especializa en el reconocimiento de objetos en imágenes. Se ha utilizado una distribución de la red proporcionada por la empresa Ultralytics, y que cuenta con una licencia para estudiantes. Se ha empleado la versión YOLOv8, que es la última versión ofrecida por Ultralytics.

Si bien esta versión de Yolo ofrece una serie de clases (objetos) que se pueden detectar, para hacer viable su aplicación en nuestro contexto la red se debe reentrenar para que pueda detectar los rodaballos que se desean medir.

El proceso de reentrenamiento se puede resumir en dos fases, obtención de los datos de entrenamiento y entrenamiento del modelo.

4.3.1 Obtención de los datos de entrenamiento

Esta fase consiste en obtener los datos con los que entrenará la red, en el formato necesario.

Ante la falta de colecciones de datos aplicables debido a la especificidad de nuestra aplicación, ha sido necesario generar una colección de datos propia, etiquetando imágenes de los estanques llenos de peces.

Como herramienta de etiquetado se ha utilizado una aplicación de usuario de Matlab que fue desarrollada por la tutora con anterioridad para otros proyectos en la misma línea de investigación.

Las razones de su uso por encima de otras herramientas de etiquetado comunes son varias. Primero, la aplicación está preparada para facilitar el etiquetado objetos difíciles de reconocer manualmente, como lo son en nuestro caso los rodaballos. Para ello la aplicación muestra un zum del objeto que se vaya a etiquetar, y además ofrece la posibilidad de visualizar las capturas posteriores y anteriores en caso de que se considere oportuno. Por otro lado, la aplicación permite clasificar el objeto con diferentes etiquetas que pueden resultar de utilidad para una variedad de aplicaciones. Por esto, el etiquetado ha sido realizado por diferentes investigadores (estudiantes o no) con diferentes propósitos. Esto ha permitido generar más datos que si se hubiese realizado de forma individual.

Cuando terminamos de etiquetar la imagen se genera un archivo .mat. Los datos de mayor interés de estos archivos son:

· BS: contiene un vector por cada uno de los objetos que se ha segmentado que representa su contorno.

· mrProps: contiene información acerca de las características morfológicas del objeto segmentado, el número de objeto y una etiqueta que indica como se ha clasificado el objeto. Esa última información resulta clave para nuestra aplicación. Hay definidas 9 etiquetas distintas, donde cada una indica lo siguiente dependiendo de si el objeto es un rodaballo, varios o ninguno y el grado de certeza que se tiene:

- 1: No se sabe si es o no es un rodaballo o varios.
- 2: No es un rodaballo o varios.
- 3: Se cree, sin seguridad, que no es un rodaballo o varios.
- 4: Es un rodaballo aislado seguro.
- 5: Es un grupo de rodaballos seguro.
- 6: Seguro que hay algún rodaballo, pero no se sabe con seguridad si hay uno o varios.

- 7: Se cree, sin seguridad, que es un rodaballo aislado.
- 8: Se cree, sin seguridad, que es un grupo de rodaballos.
- 9: No se sabe si hay algún rodaballo. En el caso de que lo hubiera, no se sabe si habría uno o varios.

La **Figura 19** es uno de los archivos .mat generados tras el etiquetado, el nombre contiene información útil como el vídeo y el número de captura que se ha etiquetado.

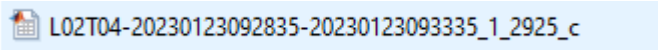

Figura 19: Formato del nombre de los archivos .mat

Para esta aplicación se han utilizado únicamente los objetos etiquetados con la etiqueta número 4, "Seguro que es un rodaballo aislado". Se podrían haber incluido también los objetos marcados con la etiqueta 7, pero se ha hecho así porque se ha considerado prioritaria la precisión frente a la cantidad.

Recordemos que para poder entrenar la red Yolo se necesitan archivos .txt en el formato adecuado (además de las imágenes correspondientes en archivos .jpg o .png). (Ver **Figura 13**).

Por eso, para transformar la información de los archivos .mat en los archivos .txt requeridos se ha preparado un script de Matlab. El script lee los archivos .mat de un directorio específico y escribe en otro directorio, para cada uno de ellos, un .txt en el formato necesario.

Para hacerlo se han obtenido las cajas delimitadoras de todos los objetos con etiqueta 4 a partir de sus contornos. Después, se ha obtenido el centro de cada una de estas cajas delimitadoras a partir de sus coordenadas. Se han obtenido los tamaños relativos dividiendo entre el alto o ancho de la imagen, según se requiera.

Aquí hay un ejemplo (**Figura 20**) de un par de líneas en formato Yolo, pertenecientes a uno de los archivos .txt que se han obtenido después de la ejecución del script.

```
0 0.616406 0.477865 0.023438 0.025521
0 0.436914 0.265885 0.024609 0.024479
```

Figura 20: Líneas en formato YOLO generadas automáticamente

En total se han etiquetado 27 imágenes, de las que se han generado 27 archivos .txt con cientos de objetos marcados con etiqueta 4 cada uno. Con esto ya tenemos las etiquetas necesarias para entrenar.

Las imágenes se han obtenido a partir de los vídeos de los que se dispone, utilizando un script de Python que lee los vídeos y números de captura que se han etiquetado de los nombres de

los archivos .mat, para después recortar las capturas a partir de los vídeos, contenidos en un directorio especificado.

Los datos etiquetados se han repartido en tres bloques distintos, uno para entrenamiento (train), otro para que el modelo valide los resultados que obtiene durante el entrenamiento (validación) y uno último de prueba (test), para poder probar el modelo una vez hemos terminado de entrenarlo.

El reparto de datos para cada bloque se ha hecho según la recomendación típica, que sugiere un 60-80% de los datos para entrenamiento, otro 10-20 % para validación y otro 10- 20% para pruebas. En este caso se han usado 20 (74%) imágenes etiquetadas para entrenamiento, 5 (18%) para validación y 2 para pruebas (8%).

Una vez obtenidos todos los datos podemos continuar con el entrenamiento del modelo.

4.3.2 Entrenamiento del modelo

Una vez obtenidos los datos, entrenar la red YOLOv8 de Ultralytics resulta sencillo. Lo primero es descargarse la biblioteca de Python “ultralytics”. Después de importarla tan sólo hay que hacer dos cosas. Se carga el modelo previamente entrenado que nos ofrece Ultralytics, y se entrena con el método train() [30].

El método train() recibe como entrada un archivo .yaml que indica la colección de datos con la que se debe entrenar, y además permite configurar varios parámetros para ajustar el entrenamiento de nuestro modelo. Para esta fase se han creado distintos modelos utilizando distintos parámetros y variando ligeramente la colección de datos de entrenamiento. Aquí se presenta el modelo que mejor resultado ha ofrecido.

Los parámetros más relevantes que se han utilizado son:

· **epochs:** ese el número total de épocas de entrenamiento. Cada época representa una pasada completa por los datos. Este parámetro afecta en gran medida al entrenamiento. Un mayor número de épocas puede mejorar el rendimiento, pero también es posible que nuestro modelo pierda capacidad de abstracción por especializarse excesivamente en los datos con los que está entrenando. Por lo general, a partir de un número se estabilizan los resultados que ofrece el modelo. Este modelo se ha entrenado con 800 épocas.

· **imgsz:** es el tamaño de imagen con el que entrena Yolo. Si no se especifica ancho y alto, cortará las imágenes en un cuadrado de lado el número de pixeles que se indique. Este modelo se ha e ntrenado con “imgsz” = 800.

· **batch:** se traduce como “tamaño de lote” en castellano. Hace referencia a la cantidad de datos que se utilizan para entrenar de una sola vez en una pasada por la red. Determina, en parte, tanto el rendimiento del modelo como la carga computacional que supone entrenarlo, (un valor más alto necesitará más espacio en memoria, pero

el entrenamiento llevará menos tiempo). El tamaño óptimo depende de la tarea específica que se quiera resolver, y la forma de averiguarlo es experimentar con él [31]. Este modelo ha utilizado un valor ("batch" = -1) que propone Ultralytics, y que se corresponde con un uso del 60% de la GPU.

La **Figura 21** muestra un ejemplo práctico de uso de la función train().

```
model = YOLO("yolov8n.pt")
model.train(data='datos.yaml', epochs=800,imgsz=800, device='cpu', batch=-1, cache=False)
```

Figura 21: Entrenamiento con *train()*

El contenido del archivo .yaml se puede ver en la siguiente **Figura 22.**

```
train: entrenarBoxes\dataset\images\train
val: entrenarBoxes\dataset\images\val
test: entrenarBoxes\dataset\images\test
nc: 1
names: ["Rodaballo"]
```

Figura 22: Archivo de configuración YAML que se ha usado para realizar el entrenamiento

En este archivo, "train", "val" y "test" indican las rutas a los directorios con las imágenes para entrenar, validar y hacer pruebas, respectivamente. Después "nc" indica el número de clases que se van a reconocer y "names" el nombre que van a recibir estas clases.

Las etiquetas se deben situar en un directorio "labels", que debe colgar del mismo directorio que "images", ambos deben tener esos nombres, explícitamente. Además, las imágenes y sus etiquetas correspondientes deben estar guardadas en archivos con el mismo nombre para que Yolo las pueda relacionar.

Tras el entrenamiento se obtiene un modelo .pt, además de numerosas estadísticas que indican los resultados del entrenamiento. Para probar el modelo basta con utilizar la función predict() tras cargar el nuevo modelo. La **Figura 23** muestra un ejemplo de este uso.

```
model = YOLO("mi_modelo.pt")
model.predict("imagen.jpg", device='cpu', show_labels=False, line_width=1, save=True, imgsz=640)
```

Figura 23: Uso de la función predict()

Tras ejecutar estas líneas se obtiene este resultado de detección sobre "imagen.jpg", mostrado en la **Figura 24.**

Figura 24: Detección de rodaballos aislados con YOLO

Donde la clase reconocida se ha encuadrado en una caja delimitadora roja. En este caso se han encuadrado todos los “rodaballos aislados”.

Una vez se está satisfecho con los resultados del modelo, podemos dar por terminada la fase de creación del modelo.

4.4 Obtención de puntos clave

La función del bloque de obtención de puntos clave es obtener los puntos necesarios de los peces, a partir de un par de imágenes rectificadas, para poder medirlos. Debemos obtener estos puntos tanto en la imagen izquierda como en la imagen derecha.

Para ello se utiliza como referencia la imagen izquierda, de donde se obtienen los peces que se quieren medir, y después se encuentran sus pares en la imagen derecha.

Para facilitar la comprensión, se puede organizar la operación de este bloque en el siguiente diagrama secuencial. El paso 1 y 2 se ejecutan sobre la imagen izquierda y el 3 y 4 sobre la imagen derecha.

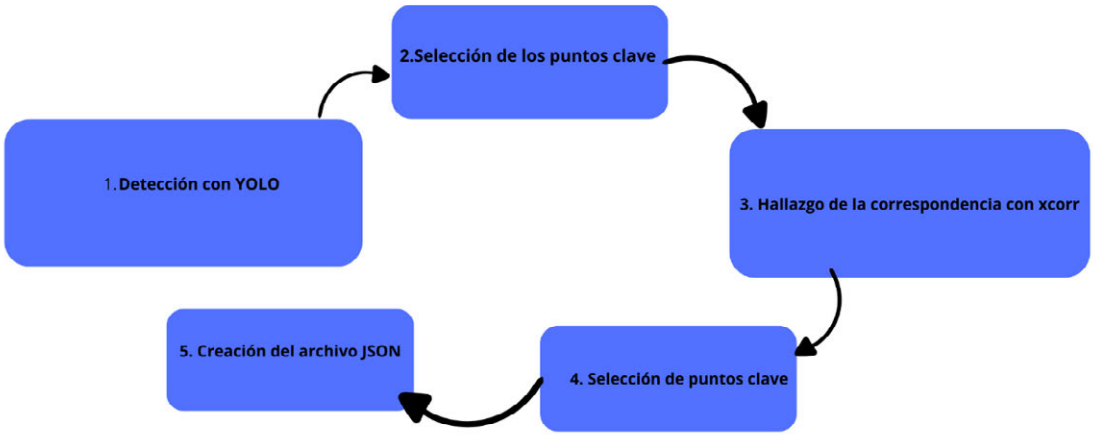

Figura 25: Diagrama de secuencia para obtener los puntos clave del rodaballo captado por las dos cámaras

Primero se detectan los peces medibles en la imagen izquierda utilizando el modelo de Yolo que se ha entrenado con anterioridad, obteniendo las cajas delimitadoras que encierran a cada uno de los peces que se ha seleccionado.

Una vez obtenidas las cajas delimitadoras los puntos clave de los peces seleccionados en la imagen izquierda se calculan de la siguiente forma.

· Cabeza y cola: se consideran como cabeza y cola el par de puntos más alejados que pertenecen al contorno del pez.

· Centro: se escoge como centro el centroide del contorno del pez.

Por lo que para obtener los puntos claves es imprescindible conocer el contorno. Con el objetivo de obtener el contorno, se segmenta el pez del resto del entorno que encierra la caja delimitadora. La segmentación que se ha implementado utiliza el modo OTSU de la biblioteca OpenCV.

Para obtener el contorno de un objeto segmentado, solamente hay que recurrir a la función *findContours()* de OpenCV, que devuelve todos los contornos de las máscaras que recibió como parámetro, de dónde escogemos tan sólo el de mayor tamaño.

Obtenido el contorno, se puede aplicar un algoritmo que calcule los puntos más alejados de este conjunto de coordenadas. Para ello se ha utilizado una función codificada por María Castillo en su trabajo [1]. La función itera sobre todos los pares de puntos y acaba devolviendo los puntos más alejados entre sí. Después se obtiene el centro utilizando otra función desarrollada durante el mismo proyecto, y que hace uso de la biblioteca OpenCV. Tras este

proceso se obtiene un rodaballo con sus puntos clave señalizados, al que denominaremos como “marcado”.

Figura 26: Ejemplos de la obtención de un contorno (izquierda) y un rodaballo marcado (derecha)

El siguiente procedimiento que se debe llevar a cabo es emparejar el rodaballo marcado con su imagen captada en la cámara derecha.

Para esto, se ha recurrido a un sistema de búsqueda de imagen tradicional, empleando la correspondencia por correlación cruzada mediante el uso de la biblioteca OpenCV [32].

Para obtener un resultado válido es imprescindible reducir el área de búsqueda a un rectángulo del menor tamaño posible. En el caso de nuestra aplicación, esta área supone un rectángulo de 411x100 píxeles. En este momento esta área tiene un valor estático, que debería ser recalculado cada vez que se sitúen las cámaras sobre un nuevo tanque para poder garantizar resultados. Otra posibilidad sería ser más riguroso con la altura a la que se colocan las cámaras sobre el tanque. La justificación de estas medidas es la siguiente:

Altura= 100: como se han rectificado las imágenes, la altura de las correspondencias es la misma. Se han dejado 100 píxeles de margen para asegurar que no se corta el rodaballo por si la rectificación no ha salido perfecta.

Anchura = 411: La anchura se ha escogido en base a la mayor disparidad que puede experimentar un pez de una imagen a la otra. El cálculo necesario para conocer este valor el siguiente:

$$D_{max} = fB/Z_{mín}$$

Ecuación 3: Cálculo de la disparidad máxima que puede experimentar un objeto

Donde *f, B* y *Z* son la distancia focal del sistema estereoscópico, la línea de base y la profundidad mínima a la que puede estar el objeto, respectivamente.

Esta expresión se obtiene a partir de la fórmula para el cálculo de profundidades en un sistema estereoscópico y se desarrollará más adelante, en el apartado de *Estimación del tamaño.* Sí es necesario explicar la elección de la profundidad para este cálculo, que es la profundidad que introduce el máximo desplazamiento horizontal para un pez de una imagen a otra. Esta profundidad coincide con la profundidad mínima a la que se puede grabar un pez. Esto es, la distancia a la que se encuentra la superficie de la piscina desde las cámaras, (puesto que un pez no puede nadar más cerca de las cámaras que por la superficie).

Sin embargo, esta distancia no se midió *in situ* y no se conoce a la fecha de la realización de este proyecto. Por eso se ha tenido que obtener manualmente. Para ello se ha utilizado una aplicación que se desarrolló durante el transcurso del proyecto, y que permite obtener la profundidad (y el tamaño) de un rodaballo marcando sus puntos clave manualmente. Así se midió la profundidad a la que están los rodaballos que aparentan estar más cercanos a la superficie, (porque reflejan más la luz y porque se ven más grandes) y se obtuvo una distancia aproximada de un metro.

Esta distancia se ha corroborado tras obtener muchas más medidas de profundidad una vez acabado el proyecto. Aun así, para el cálculo de este desplazamiento horizontal máximo se ha introducido un margen del 20% sobre el resultado, y finalmente se ha considerado 80 cm como la mínima distancia a la que se puede encontrar un pez de las cámaras para calcular la disparidad máxima que puede experimentar su imagen.

Este desplazamiento horizontal máximo (y hacia la izquierda) que sufre un rodaballo captado por la cámara izquierda en su imagen tomada por la cámara derecha provoca también que se hayan descartado como medibles los rodaballos que se encuentran demasiado cerca del margen izquierdo en la cámara izquierda.

Finalmente, el área de búsqueda se sitúa sobre la región en la que se encontró el pez en la imagen izquierda, y tras ejecutar la correlación se obtiene una caja del mismo tamaño que la plantilla que encierra el pez encontrado en la imagen derecha. En la **Figura 27** se muestra a la izquierda una plantilla escogida como referencia en la imagen izquierda, y a la derecha, el resultado de correlación que se obtuvo al buscar en el área de interés de la imagen captada por la cámara derecha.

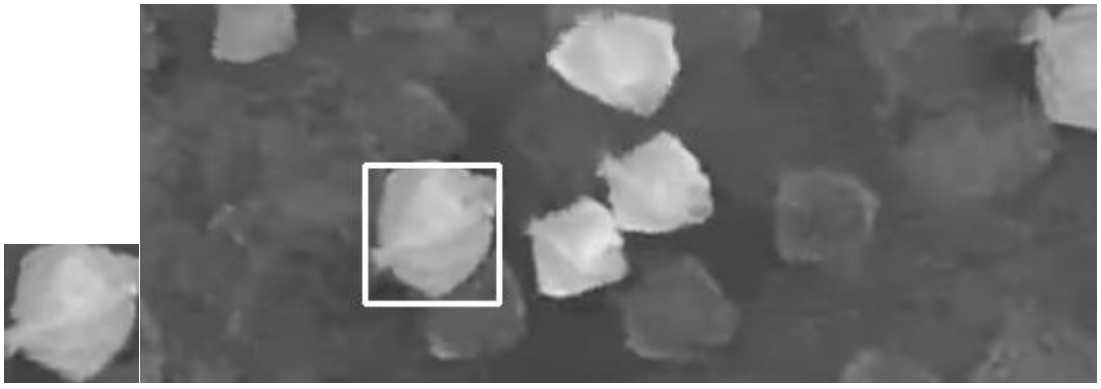

Figura 27: Ejemplo de correspondencia utilizando la correlación cruzada

Para encontrar los puntos clave en el pez captado por la imagen derecha, simplemente se ha supuesto que los puntos se encuentran en las mismas coordenadas en las que estaban en la imagen izquierda, relativas a la pequeña caja que los encierran. Si bien no es una solución perfecta, se ha escogido por simplicidad, y porque introduce un error mínimo en la aplicación que será comentado en el capítulo de resultados. Si se observa con detenimiento la **Ilustración 2** del capítulo *Anexo* se puede apreciar que algunos de los puntos clave seleccionados en la

imagen captada por la cámara derecha no se encuentran sobre el rodaballo, si no ligeramente fuera de su imagen.

Con esto, se han obtenido los puntos clave en la imagen izquierda y derecha de uno de los rodaballos que se seleccionó como medible. Una vez se hayan obtenido todos, se genera un archivo JSON (**Figura 28**) que contiene la información de los puntos clave (sus coordenadas en la imagen izquierda y derecha) de cada uno de los peces seleccionados.

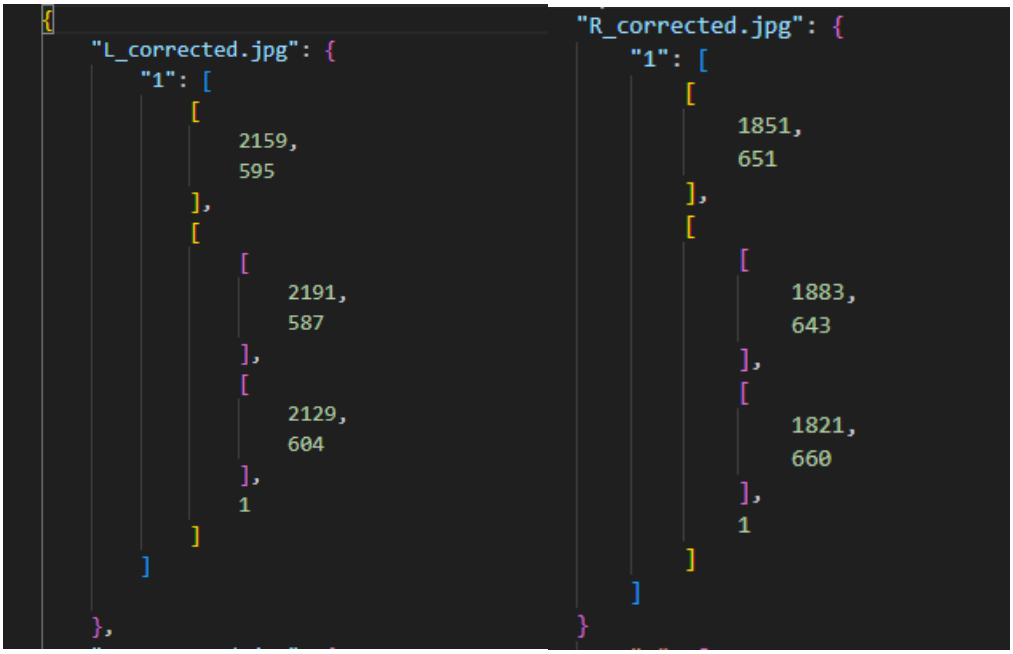

Figura 28: Ejemplo de un rodaballo marcado en JSON

Este archivo JSON se le facilitará al último de los bloques, encargado de obtener las longitudes de los rodaballos y otras informaciones útiles para el usuario.

4.5 Estimación del tamaño

El bloque de estimación del tamaño es el último de la solución, y se encarga de obtener medidas y otros resultados de interés (como la media o la desviación típica de la longitud de los rodaballos medidos) a partir del archivo JSON que contiene la información sobre los puntos clave de todos los peces de interés.

Este bloque está tomado en su mayoría del anterior trabajo [1]. Solamente se ha tenido que adaptar la implementación para que funcione en este contexto, tanto para funcionar en la aplicación como fuera de ella.

Para calcular la longitud del pez lo primero es calcular la profundidad a la que están sus puntos clave dadas las coordenadas que se han obtenido anteriormente. El cálculo de profundidad se puede deducir del siguiente esquema (**Figura 29**).

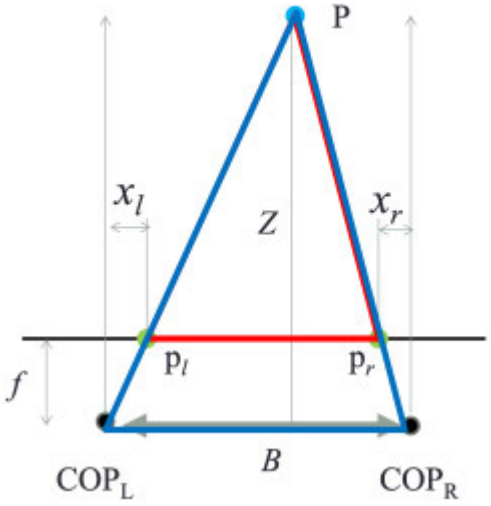

Figura 29: Esquema para cálculo de profundidad

Dónde:

Z: es la incógnita que se desea conocer. Representa la profundidad (coordenada z) a la que está el punto P desde el plano horizontal que definen las dos cámaras.

f: representa la distancia focal de las cámaras.

X_l, Xr: el número de píxeles desde el centro óptico de cada cámara hasta la representación del mismo punto de la realidad en cada una de las imágenes que toman.

B: la línea de base del sistema estereoscópico, que representa la distancia entre las cámaras.

Se puede obtener Z definiendo dos triángulos semejantes, uno de base p_r-p_ly altura Z-f y el otro de base B y altura Z[1]. Con esto:

$$\frac{Z}{B} = \frac{Z - f}{B - Xl - Xr}$$

Ecuación 4

Y además:

$$X_l + X_r = (p_l - c_x) + (p_r - c_x) = p_l - p_r$$

Ecuación 5

Entonces:

$$Z = \frac{fB}{(p_l - p_r)}$$

Ecuación 6: Expresión para el cálculo de profundidad de un objeto

Donde p_r y p_l corresponden con la coordenada horizontal del punto para la cámara derecha e izquierda. Esta última es la expresión que utilizamos para calcular profundidades.

Para obtener la longitud de un objeto en una unidad real, como los metros, conocida la profundidad a la que está y su longitud en píxeles puede resultar de utilidad el siguiente diagrama [1] (**Figura 30**):

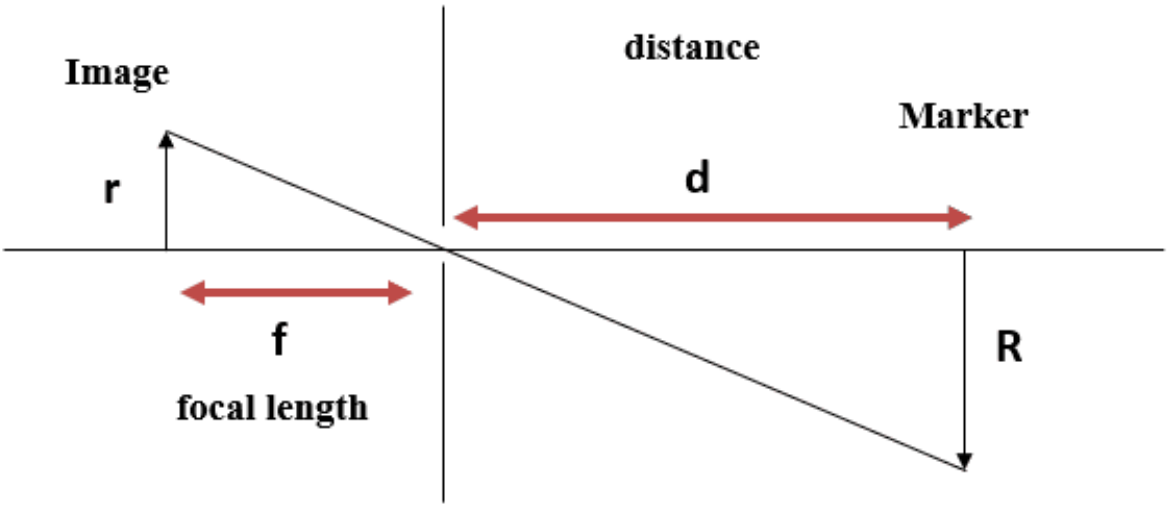

Figura 30: Esquema para el cálculo de unidades reales

Y de nuevo por triángulos semejantes:

$$\frac{R}{d} = \frac{r}{f}$$

$$R = d * \frac{r}{f}$$

Ecuación 7

Y R, d son las medidas reales del objeto (es decir la profundidad y su tamaño) y r y f son la proyección que capta la cámara y la distancia focal, respectivamente. Esta expresión es la que utilizamos para pasar de la medida en píxeles a metros, y d es la profundidad a la que se encuentra el centro del rodaballo.

Por último, observando este diagrama (**Figura 31**) y utilizando las dos expresiones que se han mostrado anteriormente se puede calcular la longitud real del pez:

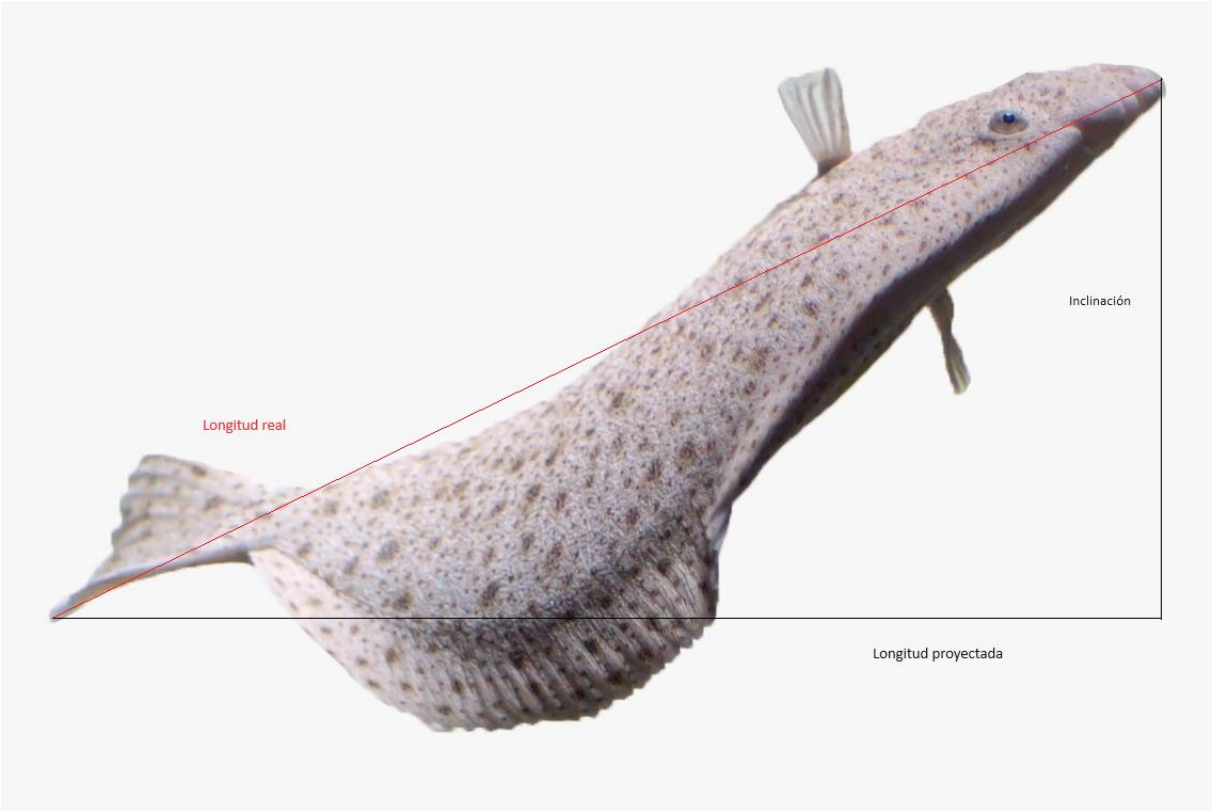

Figura 31: Diagrama que describe las distintas longitudes que se miden en el cálculo

Por el Teorema de Pitágoras:

$$Longitud\ real = sqrt(Inclinación^2 + Longitud\ proyectada^2)$$

Ecuación 8

Donde la longitud proyectada es la distancia euclídea entre la cabeza y la cola en la imagen izquierda y la inclinación es la diferencia entre la profundidad de la cabeza y la cola del pez.

Finalmente, el proceso total para calcular la longitud de un pez sería:

1. Calcular la profundidad de los puntos clave con la ecuación de profundidad (**Ecuación 6**).

2. Calcular la longitud proyectada y su valor en metros con la ecuación para obtener las unidades reales y la inclinación, obteniendo la diferencia de profundidad entre la cabeza y la cola del pez (**Ecuación 7**).

3. Aplicar el Teorema de Pitágoras para calcular la longitud final estimada del rodaballo (**Ecuación 8**).

En la siguiente **Figura 32** se muestra el resultado de la aplicación del procedimiento descrito en este capítulo.

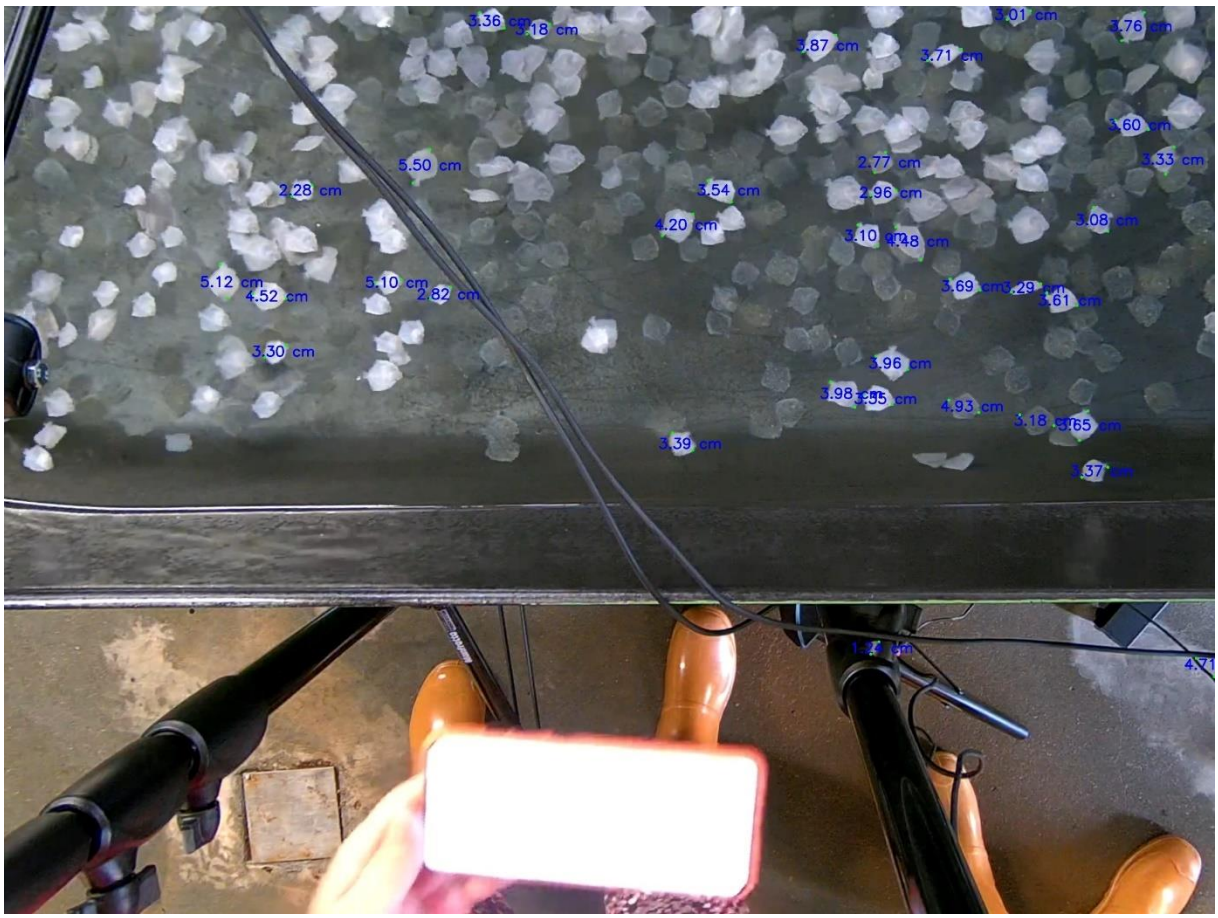

Figura 32: Resultado final tras la implementación de la solución descrita

En la figura aparecen 35 objetos que han sido detectados como rodaballos aislados. Después se indica sobre los objetos los puntos que se han escogido como la cabeza y la cola (en verde claro) y la longitud de un punto a otro (en azul marino). Los resultados muestran algunas anomalías que se comentarán (y solucionarán en caso de que sea posible) en el capítulo de resultados.

Con esto queda finalizada la explicación del diagrama de bloques que se presentó y de la solución que se ha utilizado en este proyecto.

Sin embargo, hay una última cosa que se debe tener en cuenta en esta implementación. Como se ha asumido previamente que la cabeza, la cola y el centro están en las mismas coordenadas del rectángulo que encierra el rodaballo en la cámara izquierda y derecha, al realizar el cálculo de la profundidad de estos puntos clave del pez, el resultado es el mismo para todos los puntos clave.

Esto es así porque la disparidad de los puntos clave, que resulta fundamental para calcular la profundidad, es igual para cada uno de ellos. El valor de disparidad es el mismo para la cola, la cabeza y el centro del pez.

La disparidad sería:

$$D_t = D_c + D_{pc}$$

$$D_{pc} = 0$$

$$D_t = D_c$$

Ecuación 9

D_t = disparidad total horizontal de un punto clave del rodaballo de una cámara a la otra.

D_c= disparidad de la caja delimitadora que encierra al rodaballo de una cámara a otra.

D_{pc}= desplazamiento de las coordenadas de la cabeza, centro o cola dentro de la caja delimitadora (coordenadas relativas a la caja). Esta variable tiene un valor nulo por las razones que se han comentado.

La distancia de la inclinación es la siguiente:

$$Inclinación = Zcab - Zcol$$

Y al final la inclinación es nula para cualquier rodaballo que se esté midiendo, puesto que, en nuestro sistema, todos los puntos clave de un mismo objeto están a la misma profundidad. Por esto, en esta implementación:

$$Longitud\ real = Longitud\ proyectada$$

Ecuación 10

Que es una conclusión que introduce un error pequeño que se comentará en más profundidad en la sección de resultados.

5. Resultados

5.1 Resultado Final

El resultado final mostrado en la **Figura 33** es el resultado de realizar un filtrado sobre los resultados mostrados en la **Figura 32**, basado en la profundidad de cada objeto. Se han descartado los objetos que estaban a menos de 0.99 m de distancia o a más de 1.51 m de distancia.

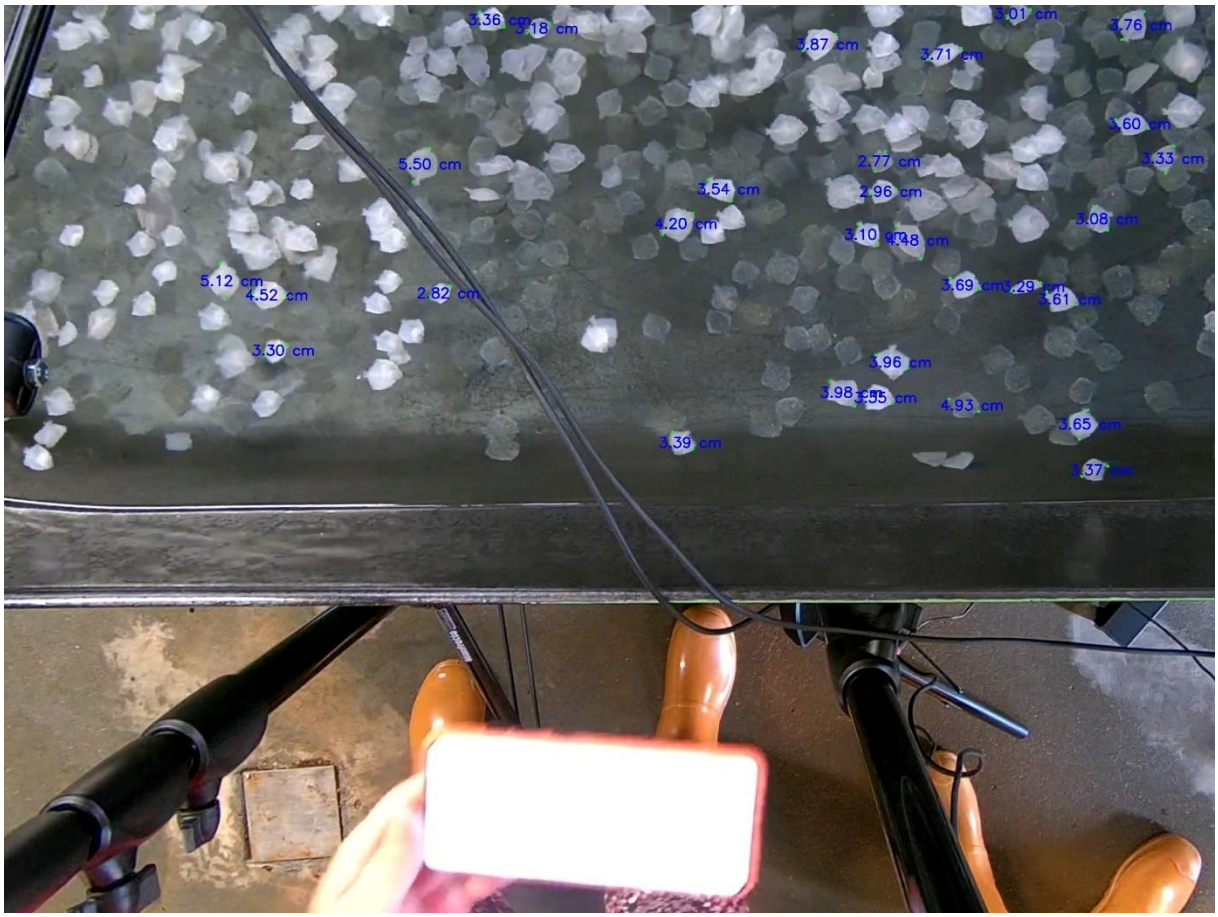

Figura 33: Resultado definitivo

Este filtrado nos ha permitido deshacernos de cinco valores que eran erróneos (ver objetos número 6, 20, 28, 33 y 34 en la figura 2 del *Anexo*), tres de ellos porque la correspondencia entre imágenes no se había resuelto correctamente y dos porque habían sido marcados como rodaballos cuando no lo eran. Al final el resultado obtenido automáticamente no contiene ningún valor que contenga un error de cálculo severo.

Si bien los resultados obtenidos están dentro del rango de valores esperados, no es posible conocer con exactitud el error cometido debido a la falta de verdad fundamental. Sin embargo, se pueden comentar los resultados de cada paso que se ha seguido (aquellos que introducen algo de error o pueden provocar errores de cálculo) para hacernos una idea de la

precisión que tiene la solución descrita. Además, en este capítulo también se van a presentar de forma resumida los resultados de entrenamiento del modelo YOLO que se ha creado para detectar rodaballos aislados. Por último, se exponen los resultados de sincronización, aunque el bloque no forme parte del sistema final.

Los resultados que se muestran corresponden a una imagen que se ha sincronizado manualmente.

5.2 Resultados Parciales

5.2.1 Modelo de Yolo

El modelo de Yolo que se ha utilizado ha servido bien para la aplicación que se le quería dar, con el apunte de que podría haber marcado alguno más de los rodaballos como medible.

Durante el proyecto se han entrenado distintos modelos, variando los parámetros de entrenamiento como las épocas, la distribución de los datos para entrenamiento, validación y test y el tamaño de imagen con el que ha entrenado la red. Adelante se presentan algunas métricas que nos pueden dar una idea de los resultados que ofrece el mejor de los modelos que se ha creado.

Precisión: la precisión indica cuántas de las predicciones del modelo han sido realmente correctas. La siguiente ecuación es la fórmula para calcularlo.

$$P = TP/(TP + FP)$$

Ecuación 11: Expresión de la Precisión

El resultado de precisión del modelo es P= 591/ (591+226) = 0,72.

Recuperación: la recuperación (“recall” en inglés) indica cuantos de los objetos reales han sido detectados por el modelo.

$$R = TP/(TP + FN)$$

Ecuación 12: Expresión de la Recuperación

El resultado del modelo es R= 591/ (591+354) = 0,62.

De estos dos resultados, en este caso, la precisión es el más importante de los dos. Esto es así porque para la aplicación que se le va a dar al modelo, se considera prioritaria la calidad de los objetos detectados frente a la cantidad. Si bien ambos resultados son bastante bajos, no son malos, puesto que el modelo ha conseguido evitar que se detecten grupos de rodaballos como rodaballos aislados o porciones del fondo o de fuera del tanque como rodaballos (0 casos en la **Figura 24** y 2 casos en la **Figura 32)**. Se puede decir que el bajo valor de precisión no se corresponde con detección de rodaballos de los que no se puede obtener una medida u objetos que directamente no son rodaballos.

Estos bajos resultados se pueden explicar por el distinto criterio de etiquetado de cada uno de los etiquetadores. También por la complejidad natural del problema, donde al modelo se le pide que interprete de distinta forma, objetos que en realidad son iguales (rodaballo) dependiendo del contexto en el que estén (en grupo o en solitario), y además los rodaballos aislados son objetos muy pequeños con relación al tamaño total de la imagen. Las cajas delimitantes tienen de valor más típico un 0,01% del área total de la imagen (1% de ancho y 1% de alto). En la siguiente figura (**Figura 34**) se muestra un mapa de densidad de los valores de ancho y alto de las cajas delimitadoras.

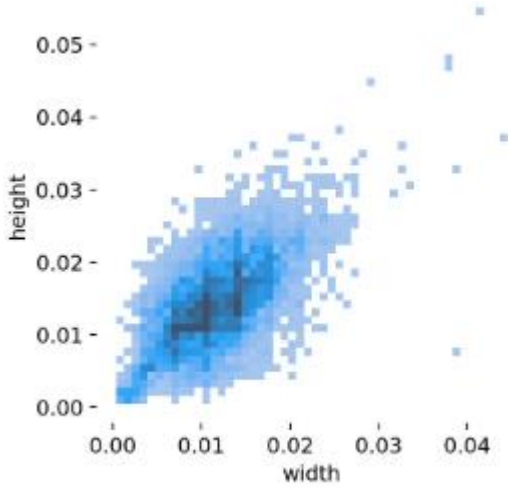

Figura 34: Mapa de densidad del tamaño de las cajas delimitadoras

5.2.2 Resultados de sincronización

Como se ha comentado en el capítulo de solución, el bloque de sincronización no ha podido ser resuelto durante este proyecto para que cumpla su objetivo. Tras probar el método de sincronización midiendo el parecido entre dos imágenes captadas por distintas cámaras con SIFT, se ha llegado a la conclusión de que no existe una relación suficientemente clara entre la cantidad de puntos en común encontrados y la diferencia temporal a la que se han capturado dos imágenes por diferentes cámaras.

En la siguiente **Figura 35** se muestra el resultado de comparar una imagen con su equivalente sincronizada captada por la otra cámara.

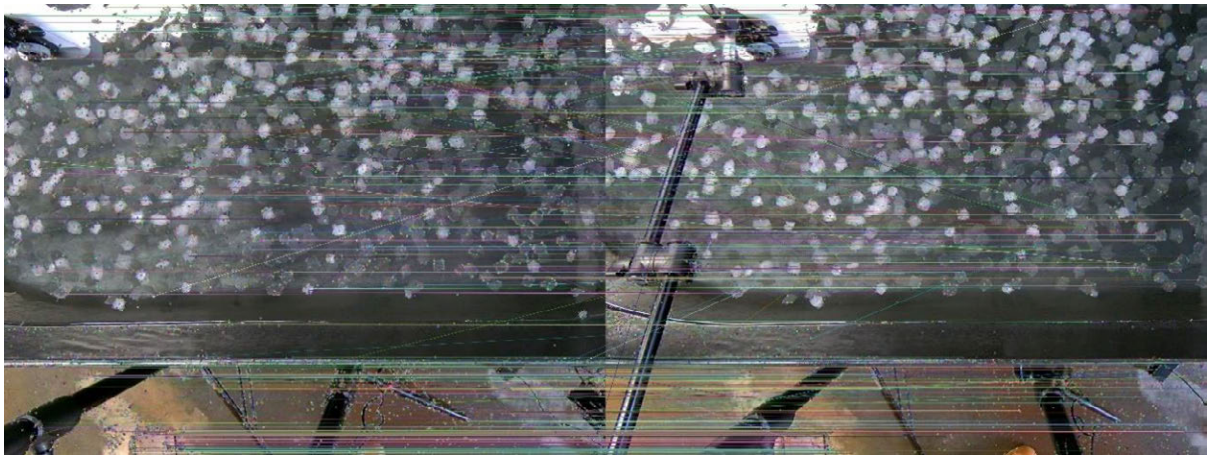

Figura 35: Comparativa de imágenes perfectamente sincronizadas

En la **Figura 36** podemos encontrar los puntos en común capturados entre la misma imagen de referencia de la anterior figura, y otra desplazada 10 capturas por delante en la otra cámara.

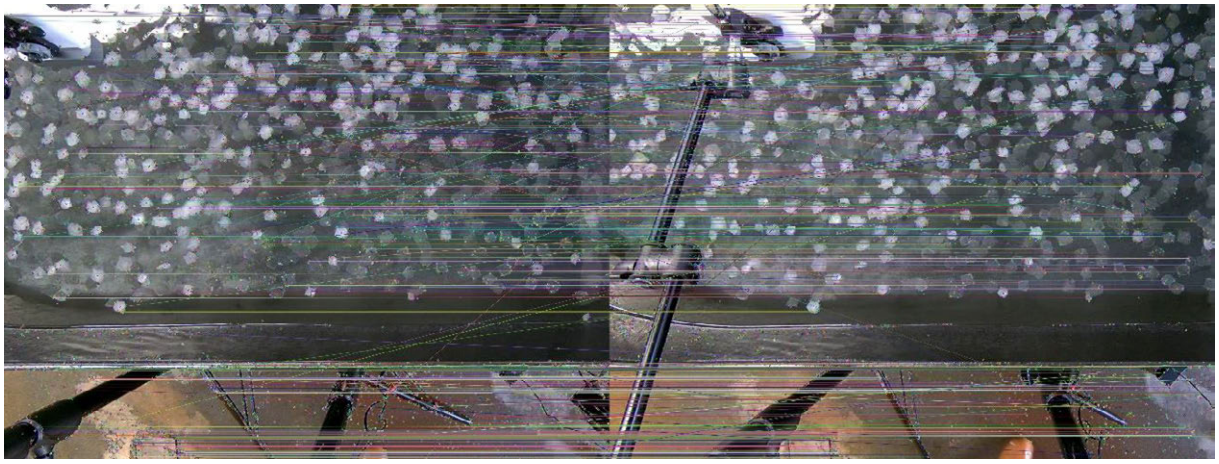

Figura 36: Comparativa de imágenes desfasadas 10 capturas (0,3 segundos aproximadamente)

Aunque resulta imposible verlo a simple vista, en la imagen desplazada 10 capturas por delante, se han detectado 47 puntos más en común que en la que se ha sincronizado a mano.

En las Tabla 5 del capítulo *Anexo*, se pueden encontrar los datos que expresan la relación entre puntos en común encontrados y cercanía temporal entre dos capturas. Si bien en el segundo resultado se cumple que la captura sincronizada es la que cuenta con más puntos en común, en el primero queda demostrado que esta relación es demasiado endeble como para sincronizar el vídeo contando solo con esta información. Sí se puede afirmar que en general se encuentran más puntos en común en regiones cercanas al sincronismo, sin embargo.

5.2.3 Cálculo de puntos clave

El cálculo de puntos clave (cola, centro y cabeza) es el paso que más error introduce sobre los resultados. Se producen errores tanto en la elección de los puntos de referencia sobre la imagen de la cámara izquierda, como al encontrar su correspondencia en la derecha.

Selección de puntos clave en la cámara izquierda (referencia)

Como se ha comentado durante el anterior capítulo, el algoritmo que escoge los puntos clave de referencia calcula la máxima distancia entre dos puntos en el rodaballo visto por la cámara. Sin embargo, esta distancia no siempre se corresponde con la longitud de cabeza a cola. En ocasiones, visto desde la perspectiva que capta la cámara, la mayor distancia se corresponde con otros puntos del contorno del rodaballo, típicamente con la longitud desde el vértice de la aleta dorsal hasta el vértice de la aleta anal. El resultado de este error es fácilmente detectable si nos fijamos en la imagen resultado, aunque no es posible corregirlo a menos que se escoja otro algoritmo.

En la **Figura 37** se ve un ejemplo de estos casos al que se le ha hecho un zum.

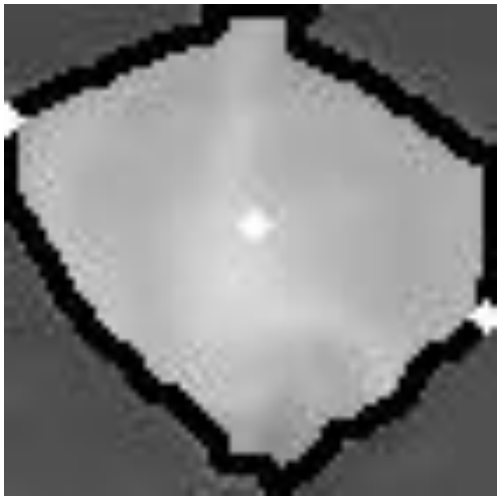

Figura 37: Error en la elección de cabeza y cola

En otras ocasiones, este fenómeno ha pasado porque la detección automática ha cortado una pequeña porción de la cola o la cabeza del rodaballo.

Si bien estos errores se pueden considerar outliers y eliminarlos de los resultados, se puede estimar el efecto que tienen sobre el resultado general en caso de no ser descartados (habría que descartarlos manualmente si se quisiera hacer).

Tabla 1: Error introducido por la mala selección de puntos en la imagen referencia

Número de rodaballo	Profundidad	Long (p)	Long(m)	Long auto(m)	Error absoluto
12	1,08597552	62,60990337	3,256387528	3,08	0,176387528
14	1,119219668	70	3,752203527	3,96	0,207796473
23	1,179392769	53,15072906	3,002207465	2,77	0,232207465
26	1,237032265	57,30619513	3,395123802	3,39	0,005123802
27	1,170998514	59,30430001	3,325948667	3,33	0,004051333

Y el error relativo medio es de 3,79% en 5 de cada 32 resultados. El error relativo medio que introduciría sobre la media del tamaño sería 5/32 * 3,79= 0,59%.

Para el resto de los objetos, aunque la elección no ha sido óptima en algunos casos, se considera que los puntos escogidos están suficientemente cerca de la cabeza y la cola para que se evalúen como buenas selecciones.

Selección de puntos clave en la cámara derecha (correspondencia)

Este proceso introduce errores en dos pasos distintos. Algunos errores de resultado se producen al realizar la correspondencia por correlación cruzada de una imagen a otra, y otros, al seleccionar los puntos clave tras haber realizado la correspondencia.

Correspondencia por correlación cruzada.

La correspondencia por correlación ha dado muy buen resultado para buscar un pequeño objeto captado por la cámara izquierda en la imagen captada por la cámara derecha. El hecho de que los rodaballos contengan poca información de profundidad (se encuentra todo el cuerpo más o menos a la misma altura, y aunque estuvieran en vertical son muy pequeños) hace que se produzca poca distorsión por disparidad de una imagen a otra. Además, como están grabados casi del mismo ángulo, la diferente perspectiva tampoco afecta mucho a la comparativa de imágenes.

Para la primera imagen estereoscópica, se han realizado 32/35 correlaciones correctamente, y aquellas en las que el resultado no fue bueno es porque el objeto ha sido ocultado parcialmente por otro, por efecto de la disparidad (los objetos más cercanos se desplazan más que los que están más lejos, y por eso existe la posibilidad de que el objeto que queremos encontrar haya sido ocultado en la segunda imagen). Además, los tres fallos han sido eliminados automáticamente tras realizar el filtro de resultados por profundidad (los objetos estaban aparentemente fuera del tanque). En la **Figura 38** se muestran dos casos donde el método de correlación ha fallado, porque el objeto se ha visto oculto en la siguiente imagen. En cualquier caso, tampoco se podrían obtener medidas de esos rodaballos. En el cuadrado blanco se encuentra el resultado obtenido, mientras que el rojo muestra el rodaballo que se estaba buscando.

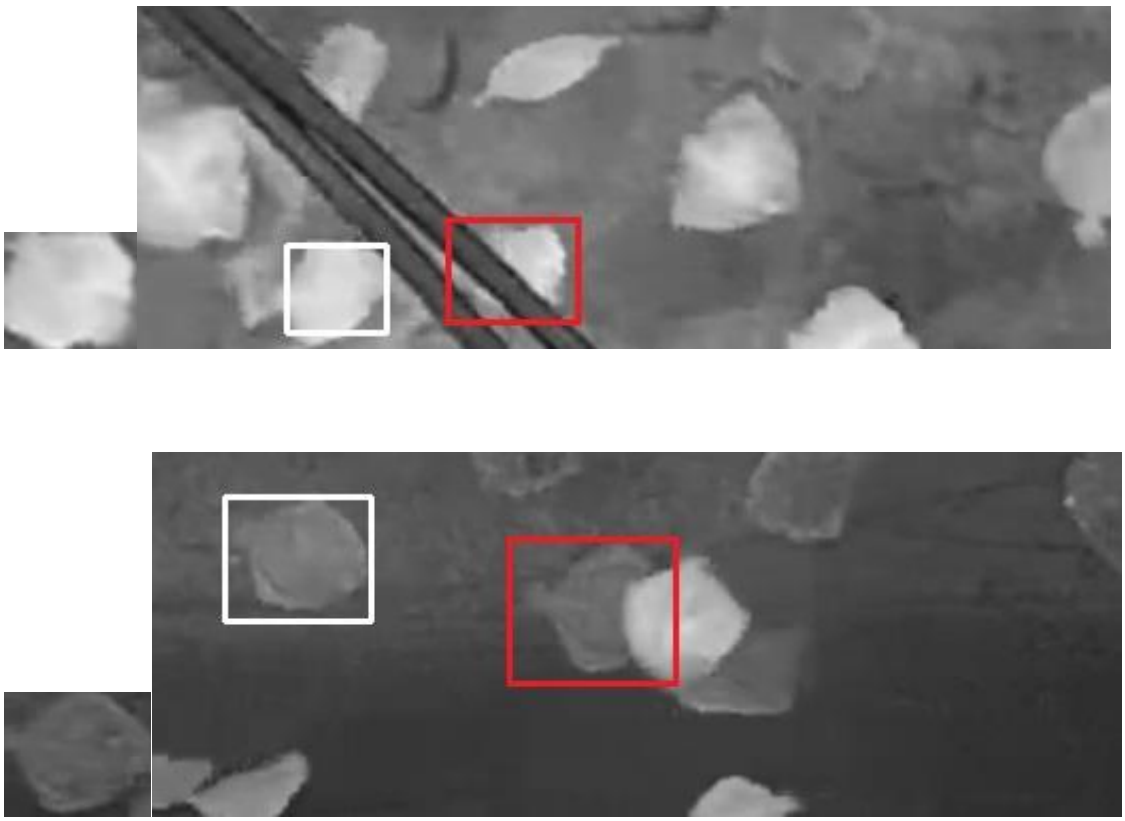

Figura 38: Ejemplos de correspondencias fallidas

En el caso de que finalmente se calcule la longitud de un objeto del que su correspondencia está mal elegida, el efecto sobre el resultado final es impredecible (pueden salir resultados muy sorprendentes), pero pueden ser eliminados habitualmente si se añade otro filtro que elimine resultados atípicos.

Elección de los puntos clave tras realizar la correlación

En el capítulo de solución se explicó que por el método de selección de puntos clave en la imagen derecha, se descarta la información relativa a la inclinación del rodaballo, afectando al resultado final. Si bien este algoritmo no es óptimo, introduce un error pequeño (para todos los peces que se miden) como se va a presentar a continuación. Este resultado es mucho mejor del que se obtiene, por ejemplo, si se vuelve a ejecutar el algoritmo de máxima distancia en el rodaballo captado por la cámara derecha.

Se ha analizado el efecto de eliminar la inclinación del pez del resultado total, comparando los resultados con los que se obtienen midiendo la inclinación a mano. Para este experimento se han elegido manualmente los puntos clave de la imagen derecha, tratando de que coincidan con aquellos que se han elegido en la imagen izquierda. Se ha medido la inclinación de todos los objetos evaluados como buenas selecciones (todos los de la **Figura 33** excepto los cuatro del apartado anterior).

En el subcapítulo *Tablas* del *Anexo* se pueden ver los datos que se han utilizado para hacer los cálculos. La tabla contiene los puntos clave escogidos a mano de la imagen derecha, as

En conclusión, se ha obtenido una distancia de inclinación media de 0,016 cm. Además, se ha obtenido que la omisión de la inclinación en el cálculo ha introducido un error absoluto medio de 0,00008 cm, y el error relativo no ha alcanzado el 0,01% en ninguno de los casos.

5.3 Precisión del sistema de visión estereoscópica.

Por último, ya que no se cuenta con verdad fundamental, se puede obtener la precisión del sistema de visión estereoscópica por separado del resto de bloques, teniendo en cuenta los resultados de [1]. Los resultados exponen que se obtuvo un error relativo medio del 4,67 %, explicando que el error crece a mayor profundidad debido a que los rayos de luz se difractan al entrar en contacto con el agua, haciendo que los objetos se vean más grandes de lo que en realidad son. Esto no se tuvo en cuenta en [1] ni se ha tenido en cuenta en este proyecto.

Por lo tanto, podemos hacer una estimación final del error medio de la solución propuesta sumando el error descrito en este capítulo (Apartado 5.2.3), con el error caracterizado en [1], y que se estima que está siendo el mismo en este proyecto. El error relativo total de la media de tamaño de los rodaballos sería del 5,26 %.

Reflexión sobre los resultados.

Los resultados obtenidos están dentro del rango de valores esperados, y aunque no ha sido posible cuantificar el error de forma definitiva por la falta de verdad fundamental, se ha demostrado que la solución propuesta introduce poco error sobre el experimento de laboratorio que se desarrolló en [1], que si contaba con verdad fundamental. Además, se ha logrado medir un buen número de rodaballos y se ha podido comprobar que todos los resultados en los que no se ha producido un error de cálculo se encontraban en el rango de profundidades esperado. El error estimado final es de media de un 5,26 % sobre todas las medidas tomadas.

Sin embargo, el hecho de no haber conseguido sincronizar el vídeo estereoscópico de forma automática hace imposible implementar el sistema de forma totalmente automática tal y como está pensado ahora, hasta que se consiga. En la sección Pasos Futuros del capítulo *Conclusiones*, se proporcionan algunas ideas (que no ha dado tiempo a probar) sobre cómo se puede tratar de resolver este problema, además de algunas soluciones alternativas que implicarían alterar el diseño del sistema tal y como se plantea en este proyecto.

6. Presupuesto

Los costes de presupuesto que ha necesitado el proyecto se pueden descomponer en:

Costes del sistema estereoscópico: el sistema estereoscópico que se utiliza requiere de dos cámaras Reolink 511 W, lo que supone un total aproximado de 200 €.

Presupuesto de mano de obra: el desarrollo de este proyecto ha requerido en torno a 300 h de trabajo. El sueldo por hora aproximado de un ingeniero recién graduado es de 12,82 € [33]. Por lo tanto, la mano de obra de este proyecto ha costado alrededor de: 12,82 x 300= 3846 €.

En total, el coste de desarrollo de este proyecto ha sido alrededor de 4046 €.

*En caso de que se quisiera comercializar el proyecto o hacer un uso empresarial del mismo, habría que añadir a este presupuesto el precio de la licencia para uso empresarial de Ultralytics. El precio de esta licencia lo calcula Ultralytics en función de los servicios que se requieran, y es necesario rellenar un cuestionario para iniciar el trámite.

Además, para que el sistema funcione se requieren cables Ethernet, un rúter y un ordenador.

7. Impacto del proyecto

Este proyecto no solo pretende resultar de ayuda para la compañía asociada con el CITSEM, sino que además puede contribuir con un impacto positivo sobre la sociedad aportando al desarrollo de la acuicultura, una industria clave para el desarrollo sostenible a nivel global y la competitividad económica del país. En este apartado se presenta el impacto positivo que puede tener el desarrollo de las técnicas de acuicultura para la sociedad.

En las últimas décadas, la acuicultura ha presentado una tendencia de crecimiento al alza, tanto económicamente como en volumen de producción. Según [34]durante las últimas tres décadas, la acuicultura ha crecido un 6,7% de media al año, y es habitualmente referida como el sector de producción alimentaria que más crecimiento está experimentando. En cuanto a volumen de producción, en 2021 la acuicultura produjo 90,9 millones de toneladas de producto (pescado y marisco), cuando el valor en 1990 era de 21,8 Mt [35]

Estos datos son especialmente significativos si los comparamos con los resultados de la pesca en el mismo periodo, que se ha mantenido estable (88,9 Mt en 1990 y 90,3 Mt en 2020, con tendencia negativa en los últimos dos años) [35]. La **Figura 39** expone estas tendencias.

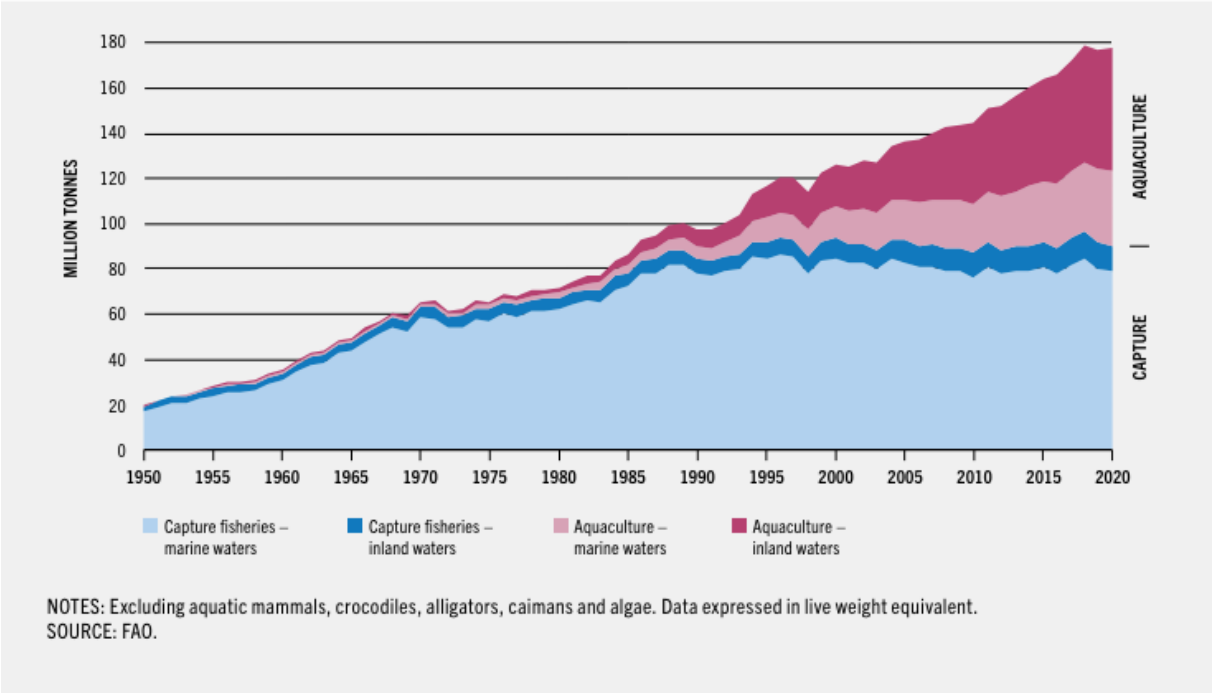

Figura 39: Tendencias de crecimiento de la producción en la acuicultura y la pesca en las últimas décadas

Sin embargo, no se debe caer en el error de tratar la acuicultura como un problema para la pesca, pues no lo es, sino todo lo contrario. La acuicultura permite aliviar la presión que pone la pesca sobre un recurso imprescindible no sólo para la economía sino para la vida en el planeta: los océanos; que ya se encuentran amenazados además por otras razones como la acidificación de las aguas o la contaminación provocada por deshechos humanos. Con eso, no

solo es importante proteger los ecosistemas oceánicos por asegurar una enorme fuente de alimento. La biodiversidad marina es un agente crucial en la investigación y desarrollo de medicamentos [36]. También, por ejemplo, los océanos suponen la fuente de ingresos principal de los "pequeños estados insulares en desarrollo" (PEID) como las Seychelles, que a su vez controlan el 30% de los mares y océanos [37]. La **Figura 40** representa el porcentaje de especies que se sobrepescan, se explotan al límite o se podrían explotar más.

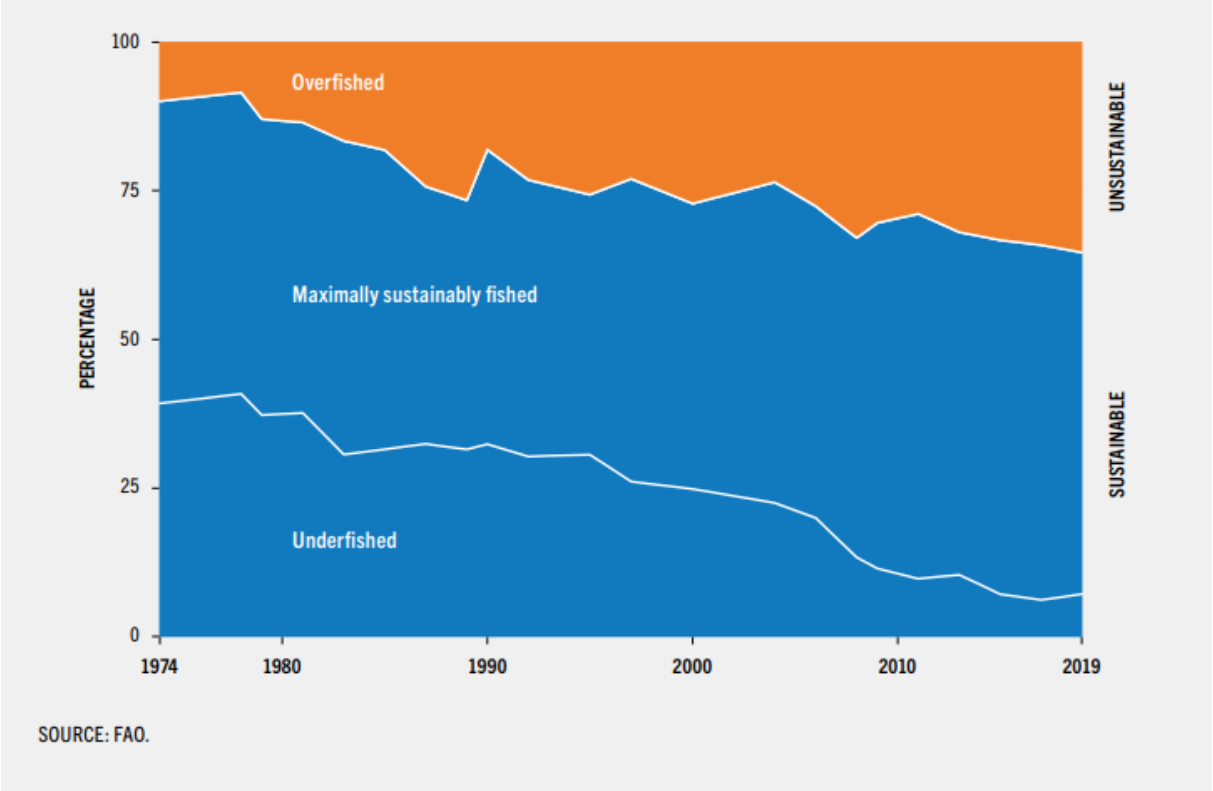

Figura 40: Porcentaje de especies explotadas según la presión a la que estén sometidas

En este contexto crítico, la acuicultura cubre la necesidad de obtener pescado y permite que los recursos del océano se regeneren de forma natural, evitando la sobrepesca y favoreciendo un comercio sostenible, que reportará mayor beneficio económico a largo plazo. Por esta razón, el desarrollo responsable de la industria de la acuicultura puede ayudar a cumplir con el Objetivo de Desarrollo Sostenible 14 (ODS 14).

A nivel nacional, la acuicultura resulta un sector clave para la economía de España, que representa (en 2022) un valor estimado del 25,2 % de producción del total de la UE, siendo el segundo productor acuícola de Europa solamente por detrás de Noruega [38]. Por eso, la inversión en desarrollo es imprescindible para mantener la competitividad del sector a nivel nacional, con el objetivo de poder seguir generando trabajo y riqueza. La **Figura 41** representa los países europeos según su nivel de producción acuícola.

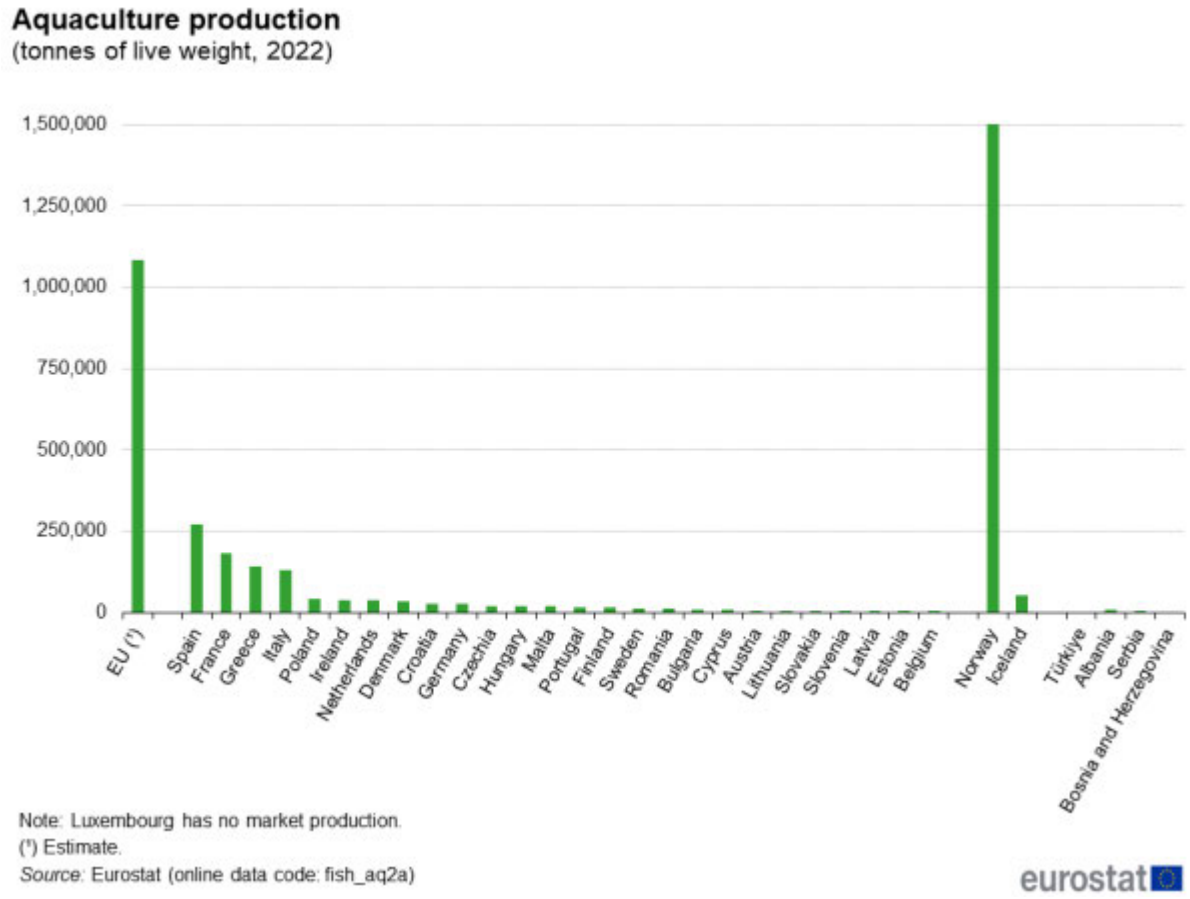

Figura 41: Gráfico de la producción de los distintos países europeos

En otro ámbito, la acuicultura también resulta una alternativa más ecológica (basándonos en el tamaño de la huella de carbono) que la gran mayoría del resto de fuentes de proteína animal, tal y como expone la **Figura 42.**

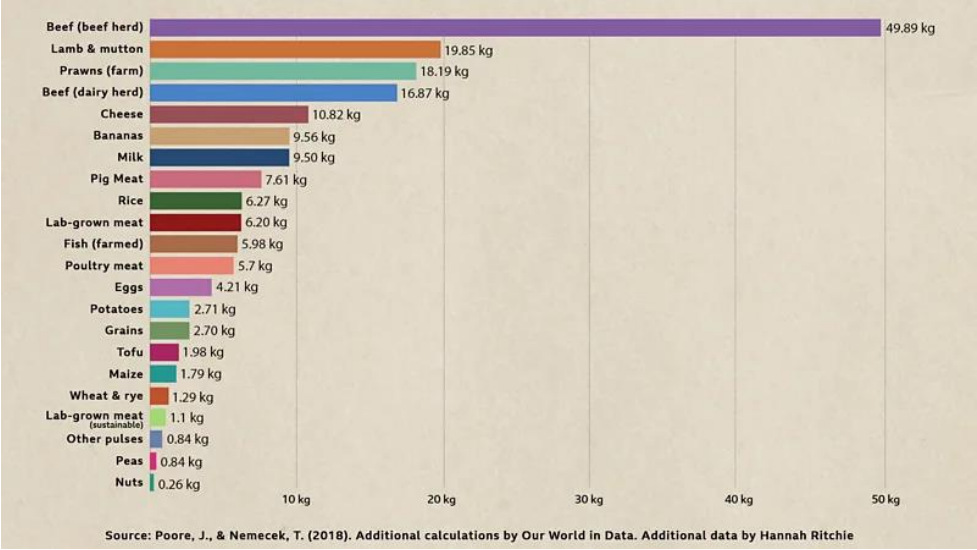

Figura 42: Huella de carbono por 100g de proteína proveniente de distintas fuentes

De donde el 50-60% de la huella de carbono proviene de la producción y transporte del alimento de las dietas basadas en pellets [39]. De esta forma, la elección del consumo de carne de pescado de cultivo por encima de otras fuentes de proteína animal ayuda a reducir las emisiones de gases de efecto invernadero que provocan la acidificación de los océanos y el cambio climático. Por esto, se puede concluir que la inversión y el desarrollo de la industria de la acuicultura, por encima de otras como la ganadera, puede favorecer el cumplimiento del ODS 13. En comparación con otras fuentes de proteína como la carne de cerdo o el pollo, el pescado de cultivo también resulta más eficiente en cuanto a la conversión de alimento en proteína [40].

La última de las cuestiones a abordar en este apartado es como el desarrollo de la producción acuícola puede ayudar a abordar el problema de alimentar a una población mundial en crecimiento, (se prevé alcanzar los 9700 millones de habitantes en 2050 según la ONU), en una situación donde la pesca está estancada a nivel de producción y el pescado es la principal fuente de proteína animal que se consume en el mundo [41]. Además, entre 1961 y 2019 el consumo de pescado por cápita creció en un 3% de media anual, casi el doble que la tasa de crecimiento de la población para el mismo periodo (1,6%) [35]. En este contexto el continuo desarrollo de la industria acuícola es fundamental para poder satisfacer una necesidad de mercado creciente, al tiempo que se aporta seguridad alimentaria a la población. Según [35], para favorecer la producción acuícola sostenible: “las áreas de prioridad para desarrollar prácticas acuícolas innovadoras son la alimentación, digitalización, y promover las prácticas eficientes y ecológicas”. Así, la inversión en proyectos de estimación de la biomasa puede facilitar la digitalización de los resultados del proceso acuícola y optimizar la alimentación evitando el desperdicio y maximizando la eficiencia de las prácticas acuícolas. Por todo esto podemos concluir que el proyecto es también beneficioso para los ODS 2, ODS 8 y ODS 9.

8. Conclusiones

En este capítulo se exponen las conclusiones tras finalizar el proyecto y se proponen futuros pasos con el fin de mejorar la solución que se aporta.

8.1 Conclusiones

Tras la finalización del proyecto, se puede concluir que se han cumplido muchas de las especificaciones y los objetivos que se fijaron al comienzo, aunque no ha resultado posible cumplir con todos.

En el aspecto positivo, se han logrado los objetivos: Objetivo 1, Objetivo 2 y Objetivo 4. En lo negativo, no ha sido posible cumplir con el Objetivo 3 de sincronización. Durante el desarrollo, se han respetado todas las restricciones con las que se contaba.

El sistema que se ha obtenido puede obtener de forma precisa la longitud de los rodaballos tal y como se demostró en el capítulo de resultados, pero no se puede implementar de forma totalmente automática puesto que para ello es necesario sincronizar los vídeos de las cámaras de forma automática.

Se puede concluir que el sistema que se propone puede servir para calcular la longitud de larvas de rodaballo con precisión en este entorno de forma automática. Si bien no ha sido posible realizar la sincronización de forma automática en las condiciones dadas hasta la fecha, eso no quiere decir que no pueda hacerse, y quedan muchas posibilidades de algoritmos que se pueden probar en el futuro para resolver este problema.

8.2 Trabajos futuros

Para mejorar los resultados del proyecto, se propone continuarlo de las siguientes formas, si se considerase oportuno.

Mejoras en la aplicación: el desarrollo de la aplicación tiene el potencial de permitir la corrección de errores manualmente, obteniendo un resultado más fiable. Por ejemplo, se podría añadir la opción de borrar la medida de un pez obtenido, por ejemplo, porque se detecte que el objeto que se mide no es un rodaballo. También se podría señalar los objetos que han sacado un coeficiente de correlación más bajo de lo normal (aún después del filtro por profundidades), para que se pueda comprobar si, en efecto, el resultado de la correlación es erróneo, para después borrar ese objeto. También se podría proporcionar una opción para guiar a los trabajadores en la calibración de las cámaras, o crear una aplicación nueva para realizar ese proceso. Se propone también el desarrollo de una capa de persistencia que guarde los datos que se van obteniendo, con el objetivo de poder proporcionar información acerca de las tendencias de crecimiento de los rodaballos. La aplicación está desarrollada de forma que sea fácilmente escalable (e interpretable por un programador), de modo que se podrían implementar estos cambios sin tener que modificar nada de lo hecho hasta ahora.

Sincronización del vídeo: es el problema principal que se debe trabajar en el futuro para obtener un resultado mucho más práctico para los trabajadores del que se tiene a la conclusión de este proyecto. Para lograr una buena sincronización se proponen varias alternativas. Como opciones que implicarían cambiar de diseño, se pueden plantear las siguientes.

La primera, sustituir el par de cámaras IP por una cámara estereoscópica. Sería la opción que facilitaría más la implementación del resto de la solución, y proporcionaría una sincronización perfecta. Zed 2 [16] por ejemplo proporciona una biblioteca que aporta funcionalidades desarrolladas con IA y que se puede integrar en proyectos de Python o C++, por ejemplo. Otra opción sería sustituir las cámaras IP por cámaras de disparador ("trigger") extraíble, de forma que se puedan sincronizar las capturas de las cámaras usando electrónica. Otra opción, por ejemplo, es grabar el audio de los vídeos y sincronizar el vídeo haciendo uso de los audios, utilizando un software externo como Syncaila [42] o simplemente utilizando la correlación cruzada en las señales de audio.

En las condiciones dadas, se proponen nuevos algoritmos para resolver el problema. Se puede probar a hacer una comparativa de imágenes utilizando correlación en un área de búsqueda pequeña y de interés, tal y como se hace para realizar la correspondencia de objetos entre cámaras. Un argumento a favor puede ser que la diferencia entre imágenes captadas por dos cámaras distintas es menor para regiones pequeñas, especialmente si no hay muchos objetos distintos a distintas profundidades. El resultado será aquel que maximice el resultado de coeficiente de correlación. Otro algoritmo que además resultaría rápido de probar es SSIM, implementado por la biblioteca *scikit* de Python. Este algoritmo devuelve un valor de parecido entre -1 (imágenes totalmente distintas) y 1 (exactamente iguales) [43].

Otra opción es utilizar algoritmos menos rígidos que sean capaces de captar el parecido entre dos imágenes tomadas desde distinta perspectiva, se propone el uso de redes neuronales como Inception o redes siamesas con este fin [43]. También hay que tener en cuenta que para sincronizar se puede considerar más información que el parecido entre dos imágenes. Se puede utilizar otra información como la cantidad de luminosidad o la cantidad de movimiento (si se consideran capturas consecutivas en lugar de tomar una sola captura como referencia). Una red neuronal podría hacer uso de esa información para sincronizar los vídeos.

Para aportarle robustez al algoritmo una vez se tenga un resultado probable se propone obtener el vector de movimiento entre capturas de un objeto, tras encontrar su correspondencia en la otra imagen. Después de realizar una transformación al vector teniendo en cuenta el cambio de sistema de referencia de una cámara a otra (se conoce la traslación y rotación de una cámara a otra), se pueden comparar estos vectores. Si son suficientemente similares, el resultado será correcto. También se podría comparar el parecido entre imágenes separadas un número igual de capturas desde una cámara y la otra, y comprobar que demuestran la máxima similitud entre ellas también.

Cambio del algoritmo de selección de puntos clave: el algoritmo de selección basado en la máxima distancia de dos puntos del contorno del pez ha introducido un error (aunque no muy grande y no muy frecuente) con el que es complicado de tratar, porque no se puede detectar cuando sucede. Se propone un cambio a un algoritmo más sofisticado, por ejemplo, un algoritmo de estimación de pose utilizando YOLOpose[yolopose]. Se podría aplicar el mismo algoritmo en la cámara derecha tras calcular la correspondencia para obtener los mismos puntos exactamente en ambas cámaras (y calcular la inclinación del pez), aunque eso añadiría carga de procesamiento para resolver un problema que se ha demostrado que apenas introduce error.

Comparar los resultados con una verdad fundamental: para cuantificar el error de forma definitiva, sería muy conveniente comparar los resultados con una verdad fundamental. Idealmente se debería comparar con datos proporcionados directamente del entorno real, pero otra posibilidad puede ser realizar un experimento con peces de verdad de los que se conozca el tamaño.

9. Referencias

[1] M. Castillo Moral, «Sistema de estimación de la biomasa en piscifactorías mediante imágenes RGB». 2023.

[2] M. Valodia, S. Nicolaev, G. Radu, y I. Staicu, «ESTIMATION OF GROWING PARAMETERS FOR THE MAIN DEMERSAL FISH SPECIES IN THE ROMANIAN MARINE AREA».

[3] V. Imaging, «ToF vs Stereo Vision: Which is Preferable? | Vadzo», Vadzo Imaging. Accedido: 9 de julio de 2024. [En línea]. Disponible en: https://www.vadzoimaging.com/post/time-of-flight-vs-stereo-vision

[4] «Nonintrusive methods for fish biomass estimation in aquaculture, part 1 - Responsible Seafood Advocate», Global Seafood Alliance. Accedido: 9 de julio de 2024. [En línea]. Disponible en: https://www.globalseafood.org/advocate/nonintrusive-methods-for-fish-biomass-estimation-in-aquaculture-part-1/

[5] «Biomass Estimation for Aquaculture», Innovasea. Accedido: 9 de julio de 2024. [En línea]. Disponible en: https://www.innovasea.com/aquaculture-intelligence/biomass-estimation/

[6] G. Monkman, K. Hyder, M. Kaiser, F. Vidal, G. Monkman, y K. Kaiser, «Accurate estimation of fish length in single camera photogrammetry with a fiducial marker», *ICES J. Mar. Sci.*, vol. 77, feb. 2019, doi: 10.1093/icesjms/fsz030.

[7] «Patterns of vertical penetration of light between ultraviolet and... | Download Scientific Diagram». Accedido: 9 de julio de 2024. [En línea]. Disponible en: https://www.researchgate.net/figure/Patterns-of-vertical-penetration-of-light-between-ultraviolet-and-infrared-into-the-water_fig3_347945401/actions#reference

[8] C. Almansa, L. Reig, y J. Oca, «The laser scanner is a reliable method to estimate the biomass of a Senegalese sole (*Solea senegalensis*) population in a tank», *Aquac. Eng.*, vol. 69, pp. 78-83, nov. 2015, doi: 10.1016/j.aquaeng.2015.10.003.

[9] «Imaging Sonar - an overview | ScienceDirect Topics». Accedido: 9 de julio de 2024. [En línea]. Disponible en: https://www.sciencedirect.com/topics/engineering/imaging-sonar

[10] «Biometrics». Accedido: 9 de julio de 2024. [En línea]. Disponible en: https://optimar.no/biometrics

[11] D. Li *et al.*, «Automatic counting methods in aquaculture: A review», *J. World Aquac. Soc.*, vol. 52, n.º 2, pp. 269-283, abr. 2021, doi: 10.1111/jwas.12745.

[12] T. Takahara, T. Minamoto, H. Yamanaka, H. Doi, y Z. Kawabata, «Estimation of Fish Biomass Using Environmental DNA», *PLOS ONE*, vol. 7, n.º 4, p. e35868, abr. 2012, doi: 10.1371/journal.pone.0035868.

[13] H. Mizumoto, H. Urabe, T. Kanbe, M. Fukushima, y H. Araki, «Establishing an environmental DNA method to detect and estimate the biomass of Sakhalin taimen, a critically endangered Asian salmonid», *Limnology*, vol. 19, n.º 2, pp. 219-227, abr. 2018, doi: 10.1007/s10201-017-0535-x.

[14] «How Does Stereoscopic (3D) Vision Work? | The Optometry Center for Vision Therapy». Accedido: 9 de julio de 2024. [En línea]. Disponible en: https://ocvt.info/how-does-stereoscopic-3d-vision-work

[15] «What is a stereo vision camera? - e-con Systems». Accedido: 9 de julio de 2024. [En línea]. Disponible en: https://www.e-consystems.com/blog/camera/technology/what-is-a-stereo-vision-camera-2/

[16] «ZED 2 Stereo Camera», Stereolabs Store. Accedido: 9 de julio de 2024. [En línea]. Disponible en: https://store.stereolabs.com/en-es/products/zed-2

[17] «Configuring Stereo Depth». Accedido: 9 de julio de 2024. [En línea]. Disponible en: https://docs.luxonis.com/hardware/platform/depth/configuring-stereo-depth

[18] «¿Cómo elegir la línea de base y la resolución óptimas para una cámara de visión estéreo?» Accedido: 23 de junio de 2024. [En línea]. Disponible en: https://www.linkedin.com/advice/1/how-do-you-choose-optimal-baseline-resolution-stereo

[19] «OpenCV: Camera Calibration». Accedido: 9 de julio de 2024. [En línea]. Disponible en: https://docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html

[20] «Rectification example. Top: a pair of original images. Bottom: the... | Download Scientific Diagram». Accedido: 10 de julio de 2024. [En línea]. Disponible en: https://www.researchgate.net/figure/Rectification-example-Top-a-pair-of-original-images-Bottom-the-rectified-images-with_fig5_265107440

[21] D. G. Lowe, «Distinctive Image Features from Scale-Invariant Keypoints», *Int. J. Comput. Vis.*, vol. 60, n.º 2, pp. 91-110, nov. 2004, doi: 10.1023/B:VISI.0000029664.99615.94.

[22] «Project 2: Local Feature Matching». Accedido: 10 de julio de 2024. [En línea]. Disponible en: https://faculty.cc.gatech.edu/~hays/compvision2018/proj2/

[23] «What Is Cross-Correlation? Definition, How It's Used, and Example», Investopedia. Accedido: 10 de julio de 2024. [En línea]. Disponible en: https://www.investopedia.com/terms/c/crosscorrelation.asp

[24] «YOLOv8 vs SSD: Choosing the Right Object Detection Model», Keylabs: latest news and updates. Accedido: 10 de julio de 2024. [En línea]. Disponible en: https://keylabs.ai/blog/yolov8-vs-ssd-choosing-the-right-object-detection-model/

[25] M. A. Nadeem, «Unraveling YOLO v8: Benefits and Challenges in Object Detection», Medium. Accedido: 10 de julio de 2024. [En línea]. Disponible en: https://medium.com/@masadnadeem23/unraveling-yolo-v8-benefits-and-challenges-in-object-detection-1ec762debefd

[26] J. Redmon, S. Divvala, R. Girshick, y A. Farhadi, «You Only Look Once: Unified, Real-Time Object Detection», en *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA: IEEE, jun. 2016, pp. 779-788. doi: 10.1109/CVPR.2016.91.

[27] «Líderes en formación tecnológica, reskilling y upskilling | OpenWebinars», OpenWebinars.net. Accedido: 10 de julio de 2024. [En línea]. Disponible en: https://openwebinars.net/blog/que-son-las-redes-neuronales-y-sus-aplicaciones/

[28] «What are Convolutional Neural Networks? | IBM». Accedido: 10 de julio de 2024. [En línea]. Disponible en: https://www.ibm.com/topics/convolutional-neural-networks

[29] Ultralytics, «Object Detection Datasets Overview». Accedido: 18 de junio de 2024. [En línea]. Disponible en: https://docs.ultralytics.com/datasets/detect

[30] Ultralytics, «Tren». Accedido: 10 de julio de 2024. [En línea]. Disponible en: https://docs.ultralytics.com/es/modes/train

[31] «¿Cuál es el mejor tamaño de lote para optimizar los modelos de aprendizaje profundo?» Accedido: 10 de julio de 2024. [En línea]. Disponible en: https://www.linkedin.com/advice/0/what-best-batch-size-optimizing-deep-learning-kw4if

[32] «Stereo Vision: Depth Estimation between object and camera | by Apar Garg | Analytics Vidhya | Medium». Accedido: 10 de julio de 2024. [En línea]. Disponible en: https://medium.com/analytics-vidhya/distance-estimation-cf2f2fd709d8

[33] «Salario para Ingeniero Junior en España - Salario Medio». Accedido: 10 de julio de 2024. [En línea]. Disponible en: https://es.talent.com/salary?job=ingeniero+junior

[34] «A decadal outlook for global aquaculture - Mair - 2023 - Journal of the World Aquaculture Society - Wiley Online Library». Accedido: 10 de julio de 2024. [En línea]. Disponible en: https://onlinelibrary.wiley.com/doi/full/10.1111/jwas.12977

[35] *In Brief to The State of World Fisheries and Aquaculture 2022*. FAO, 2022. doi: 10.4060/cc0463en.

[36] «Do medicines come from the sea? : Ocean Exploration Facts: NOAA Office of Ocean Exploration and Research». Accedido: 10 de julio de 2024. [En línea]. Disponible en: https://oceanexplorer.noaa.gov/facts/medicinesfromsea.html

[37] U. Nations, «Sustainable blue economy vital for small countries and coastal populations», United Nations. Accedido: 10 de julio de 2024. [En línea]. Disponible en: https://www.un.org/en/desa/sustainable-blue-economy-vital-small-countries-and-coastal-populations

[38] «Home - Eurostat». Accedido: 10 de julio de 2024. [En línea]. Disponible en: https://ec.europa.eu/eurostat

[39] «Assessing the carbon footprint of aquaculture - Responsible Seafood Advocate», Global Seafood Alliance. Accedido: 10 de julio de 2024. [En línea]. Disponible en: https://www.globalseafood.org/advocate/assessing-carbon-footprint-of-aquaculture/

[40] M. R. Hasan, M. Halwart, y Food and Agriculture Organization of the United Nations, Eds., *Fish as feed inputs for aquaculture: practices, sustainability and implications*. en FAO fisheries and aquaculture technical paper, no. 518. Rome: Food and Agriculture Organization of the United Nations, 2009.

[41] «Consumo mundial de proteína animal por tipo en 2022», Statista. Accedido: 10 de julio de 2024. [En línea]. Disponible en: https://es.statista.com/estadisticas/1314427/consumo-mundial-de-proteina-animal-por-tipo/

[42] «Syncaila - software de sincronización de audio y vídeo multicámara». Accedido: 10 de julio de 2024. [En línea]. Disponible en: https://syncaila.com/es

[43] V. Reddy, «Exploring Image Similarity Approaches in Python», ScrapeHero. Accedido: 23 de junio de 2024. [En línea]. Disponible en: https://medium.com/scrapehero/exploring-image-similarity-approaches-in-python-b8ca0a3ed5a3

Anexo

A.1 Imágenes Auxiliares

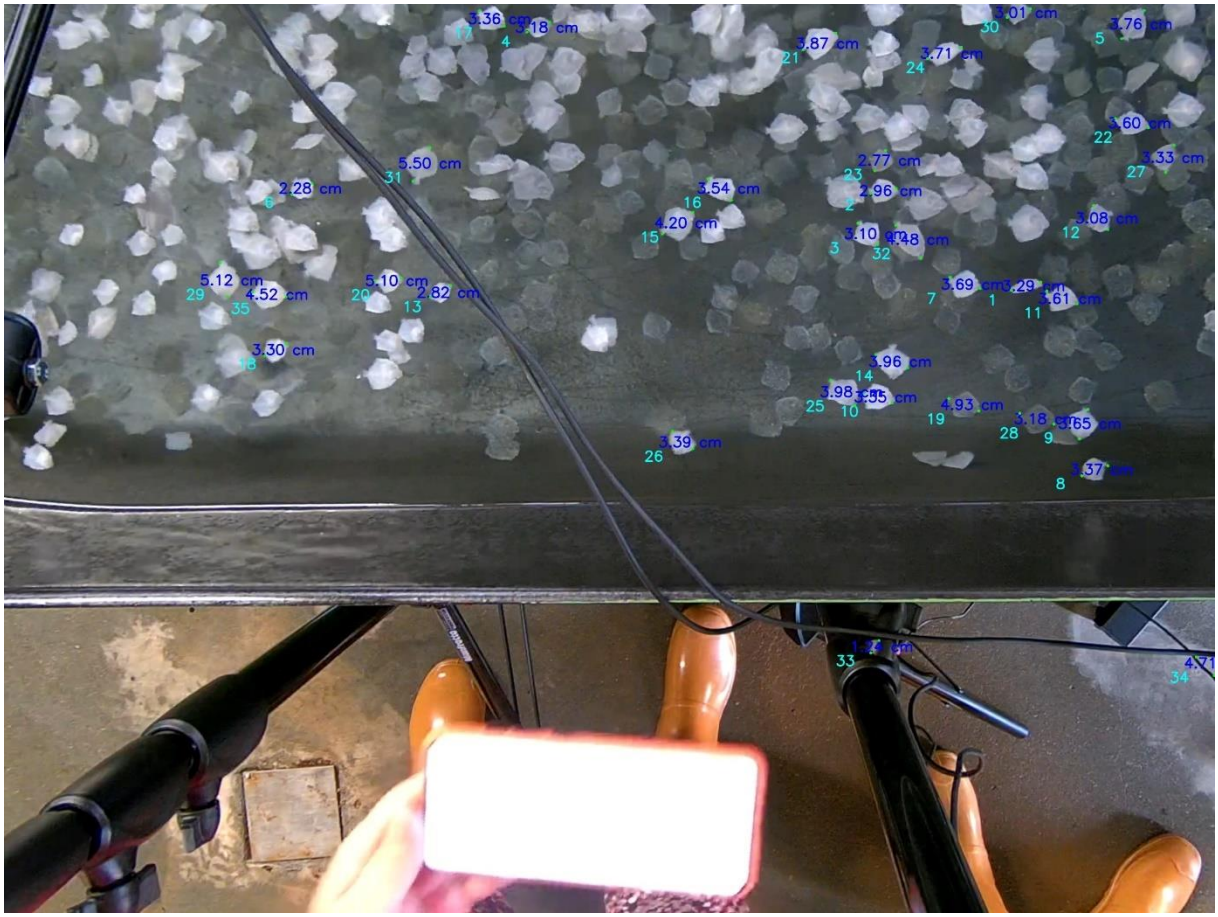

Ilustración 1: Imagen Izquierda con medidas y números de objeto

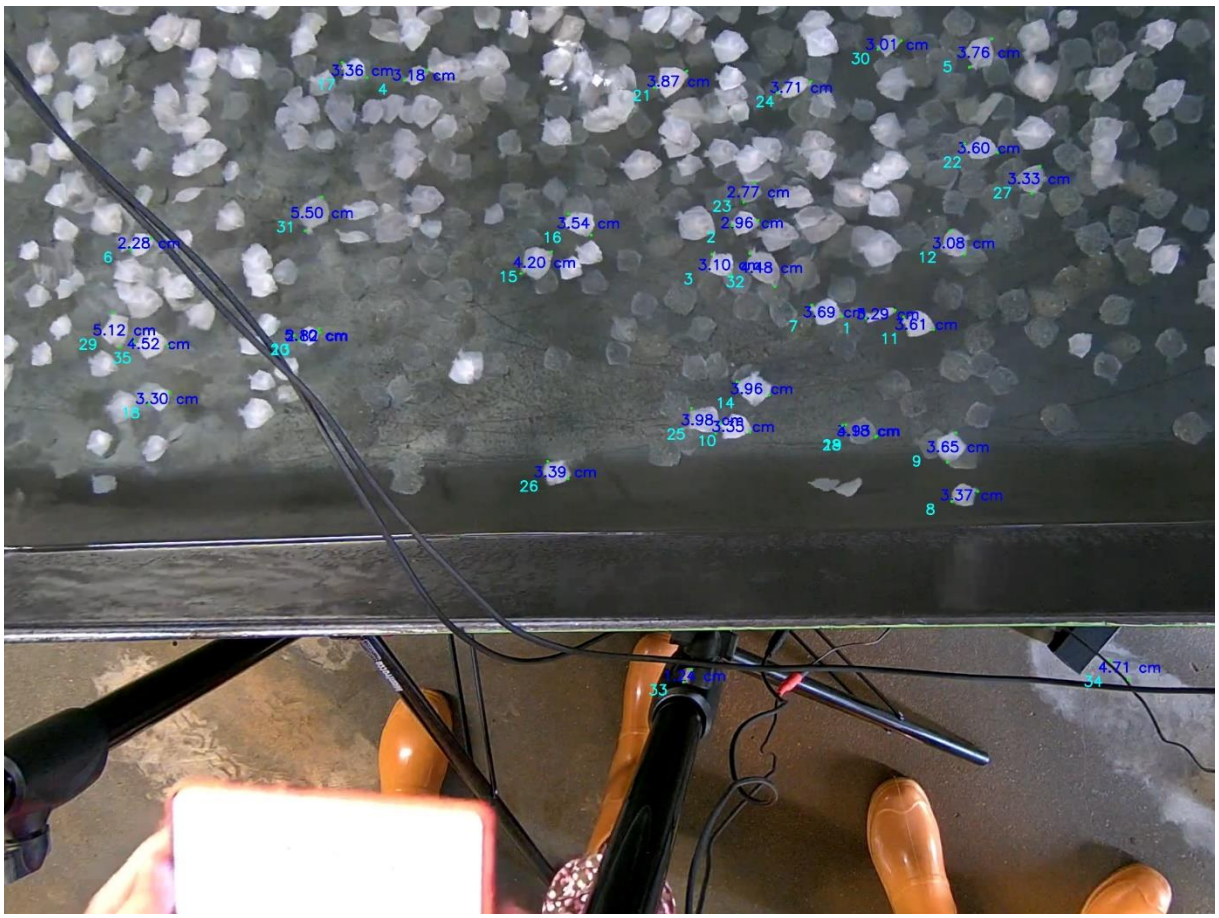

Ilustración 2: Imagen Derecha con medidas y números de objeto

A.2 Tablas Auxiliares

Tabla 2: Datos para medir el error de 5.2.3

Número de rodaballo	Centro x (L)	Centro x (R)	Cabeza x (L)	Cabeza y (L)	Cola x(L)	Cola y (L)	Cabez x ®	Cabeza y R	Cola x R	Cola y R
12	2313	2010	1983	516	2039	488	2259	466	2342	440
14	1880	1586	1878	791	1878	721	1587	845	1587	775
23	1855	1576	1881	332	1828	328	1602	395	1556	391
26	1431	1165	1419	950	1447	900	1154	1012	1180	962
27	2452	2171	2424	325	2483	319	2141	365	2203	364

Tabla 3: Datos para medir la inclinación obtenidos a mano

Número de rodaballo	Cabeza coord x	Cabeza coord y	Cola coord x	Cola coord y
1	1881	638	1821	660
2	1593	453	1539	464
3	1537	569	1496	524
4	893	136	846	159
5	2085	68	2042	129
7	1771	653	1706	635
8	2003	1047	2058	1026
9	1994	964	2012	902
10	1576	901	1515	882
11	1965	684	1901	657
13	664	668	624	708
15	1152	517	1097	564
16	1234	486	1198	438
17	763	152	715	120
18	344	815	308	844
19	1843	910	1778	888
21	1440	141	1380	175
22	2101	308	2032	293
24	1705	159	1634	183
25	1508	900	1451	857
29	251	757	226	658
30	1890	72	1847	91
31	673	402	638	478
32	1629	593	1576	523
35	340	715	280	713

Tabla 4: Datos de inclinación

Número de rodaballo	Profundidad cabeza	Profundidad cola	Distancia de Inclinación	Longitud Auto	Longitud con inclinación
1	1,061453492	1,068346047	0,006892555	3,29	3,29000722
2	1,123039531	1,123039531	0	2,96	2,96
3	1,064888616	1,064888616	0	3,1	3,1
4	1,265579163	1,270465569	0,004886406	3,18	3,180003754
5	1,012463331	1,02828307	0,01581974	3,76	3,76003328
7	1,126885556	1,123039531	0,003846026	3,69	3,690002004
8	1,196547573	1,196547573	0	3,37	3,37
9	1,179392769	1,179392769	0	3,65	3,65
10	1,093191304	1,093191304	0	3,35	3,35
11	1,08597552	1,08597552	0	3,61	3,61
13	1,175180652	1,18790824	0,012727588	2,82	2,820028722
15	1,082403232	1,119219668	0,036816436	4,2	4,20016136
16	1,08597552	1,134657181	0,048681661	3,54	3,540334716
17	1,119219668	1,13858333	0,019363662	3,36	3,360055796
18	1,30575628	1,354117623	0,048361344	3,3	3,300354348
19	1,516362131	1,502514075	0,013848056	4,93	4,930019449
21	1,034750259	1,064888616	0,030138357	3,87	3,870117352
22	1,034750259	1,038014456	0,003264196	3,6	3,60000148
24	1,034750259	1,034750259	0	3,71	3,71
25	1,134657181	1,111657373	0,022999808	3,98	3,980066456
29	1,488916663	1,430654706	0,058261956	5,12	5,120331479
30	1,179392769	1,205313489	0,02592072	3,01	3,010111607
31	1,449562037	1,455976029	0,006413991	5,5	5,50000374
32	1,064888616	1,064888616	0	4,48	4,48
35	1,285353838	1,343063602	0,057709764	4,52	4,520368394

Tabla 5: Datos sobre puntos en común a diferentes distancias temporales

Capturas de diferencia	-200	-30	-10	-5	-1 Sincronizadas	1	5	10	30	200
Puntos en común detectados	233	328	316	313	313	550	328	338	315	318	240

Capturas de diferencia	-200	-30	-10	-5	-1 Sincronizadas	1	5	10	30	200
Puntos en común detectados	254	342	371	342	379	376	393	429	423	356	320

Manual de usuario

A.3 Aplicación de usuario

La aplicación de usuario es sencilla e intuitiva de usar y no necesita manual de usuario. El código fuente utiliza una estructura de capas que separa la interfaz gráfica de la capa lógica de negocio, que contiene distintas clases para obtener los puntos procesando las imágenes, medir o rectificarlas. Si se fuera a modificar se recomienda crear clases nuevas para integrarlas por separado con la interfaz de usuario o la capa de negocio, y en caso de que fuese necesario, añadir nuevos métodos a las clases que ya están creadas.