UNIVERSIDAD DE CONCEPCIÓN
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA ELECTRICA

USO DE VISIÓN POR COMPUTADORA PARA LA DETECCIÓN DE OBJETOS NO DESEADOS EN LÍNEAS DE PRODUCCIÓN DE MEJILLONES SIN VALVA

POR

Cristopher Nicolás Silva Zavala

Memoria de Título presentada a la Facultad de Ingeniería de la Universidad de Concepción para optar al título profesional de Ingeniero Civil en Telecomunicaciones

Profesor Guía

Sebastián Godoy Medel

Diciembre, 2023

Concepción (Chile)

“Con amor para mi familia, mi novia y especialmente para mi abuelo José Jara por todo su apoyo.”

Agradecimientos

Cristopher Nicolás Silva Zavala recibe financiamiento desde ANID – Proyecto Anillo ACT210073.

Gracias a mis padres Carlos y Nancy, a mis abuelos Gladys y José y a mi hermana Andrea por permitirme estudiar en concepción, por mantenerme a raya cuando me desvié del camino en los primeros años y por su confianza en que terminaría la carrera, por los altos y bajos durante estos siete años y por estar siempre ahí para mi cuando los necesite, aun con la distancia de por medio.

Gracias a mis tíos Marisol y Oscar por recibirme en su casa estos años y por tener sus puertas abierta siempre para mí, gracias a mis primas Grace y Jael por los buenos momentos, las risas y las celebraciones que compartimos durante todos estos años y por brindarme su apoyo cuando lo necesite, a mis sobrinas Amanda y Violeta que vi crecer mientras estudiaba, por hacerme reír y por recordarme como era ser un niño y recordar de donde salió el yo de hoy en día, Gracias a toda la familia Barra Jara por recibirme y hacer de la casa en penco un segundo hogar.

Gracias a Camila mi polola, por alentarme a seguir adelante y por insistir, aunque yo a veces no quisiera, gracias por estar conmigo desde que nos conocimos y gracias por el amor que me has dado estos años, por estar en los momentos buenos y malos, sin ti este momento no habría llegado tan pronto, gracias por ayudarme en lo que podías y por ser el lugar seguro al que volver si las cosas se torcían, con todo el amor que te tengo te agradezco por todo.

Gracias a mi profesor guía Sebastián Godoy por recibirme y por darme la oportunidad de trabajar con él, gracias por darme el tema con el que llegue a mi memoria de título, y que despertó mi interés por la detección de imágenes y la inteligencia artificial, gracias por escucharme y por darme su tiempo este año y medio.

Gracias a todos los compañeros y personas que conocí durante estos años en la universidad, me permitieron ampliar mi mundo y conocer como es interactuar con personas de distintas carreras y lugares del país, gracias a mis compañeros y compañeras de las generaciones 2017 a 2019 de ingeniería civil en telecomunicaciones con los que compartí gratos momentos, tanto en el aula como fuera de ella, espero que todos consigamos nuestros títulos y logremos nuestras metas.

Finalmente, gracias a todos los profesores y funcionarios por sus enseñanzas, su tiempo y paciencia, que me dieron los conocimientos que me permitieron llegar a donde estoy.

Sumario

La visión artificial y la detección de objetos son de vital importancia en la industria actual, ya que representan los ojos de la automatización. Cada vez más empresas buscan implementar estas tecnologías, por lo que resulta fundamental para un ingeniero aprender a utilizar estos medios. En esta memoria de título, se lleva a cabo la implementación de la detección de objetos mediante Matlab, utilizando el modelo de detector de características de canal agregado y la cámara de un smartphone.

El proceso abarca desde la recopilación de datos hasta el entrenamiento, la optimización y la implementación en imágenes, videos, y la realización de detecciones en tiempo real. El objetivo principal es detectar objetos no deseados en una línea de producción de mejillones, tales como algas, piedras, vidrios y conchas de otras especies marinas.

Se utiliza la aplicación Video Labeler de Matlab, que permite llevar a cabo el etiquetado y la automatización de este proceso mediante un algoritmo de point tracker. Además, se realiza el entrenamiento de un detector específico para mejillones y otros detectores aplicados a la detección de objetos en una imagen. Luego, se realizan optimizaciones en la detección, eliminando etiquetas superpuestas mediante la función "selectStrongestBbox" y removiendo etiquetas por puntaje mínimo en histograma.

A continuación, se lleva a cabo la detección de imágenes en secuencia mediante la detección de videos de la especie de interés. Se emplea la aplicación iVCam para utilizar la cámara de un smartphone como webcam en Matlab, permitiendo así la implementación de la detección de objetos en tiempo real. Posteriormente, se utiliza la función "bboxPrecisionRecall" para obtener métricas de los detectores y optimizarlos con los datos conseguidos de manera individual para cada objeto específico detectado, logrando aumentos de precisión mayores al 60 % en el caso del mejillón.

Finalmente, se aplican los detectores entrenados en la detección de distintas especies en una cinta en movimiento que simula una línea de producción en tiempo real, donde los resultados obtenidos muestran un rendimiento poco favorable, lo que conduce a las conclusiones correspondientes.

Sumary

Artificial vision and object detection are of vital importance in today's industry as they represent the eyes of automation. An increasing number of companies seek to implement these technologies, making it essential for an engineer to learn how to utilize these means. In this thesis, the implementation of object detection is carried out using Matlab, employing the aggregate channel features detector model and the camera of a smartphone.

The process spans from data collection to training, optimization, and implementation in images, videos, and real-time detections. The primary objective is to identify unwanted objects in a mussel production line, such as algae, stones, glass, and shells from other marine species.

The Matlab Video Labeler application is used, allowing for labeling and automation through a point tracker algorithm. Additionally, specific training for mussel detection and other detectors applied to object detection in an image is conducted. Subsequent optimizations in detection involve removing overlapping labels using the "selectStrongestBbox" function and eliminating labels based on minimum scores in the histogram.

Next, image detection in sequence is performed by detecting videos of the target species. The iVCam application is employed to use a smartphone camera as a webcam in Matlab, enabling real-time object detection. Later, the "bboxPrecisionRecall" function is utilized to obtain metrics for the detectors and optimize them with individually obtained data for each specific detected object, achieving precision increases greater than 60% in the case of mussels.

Finally, the trained detectors are applied to detect various species on a moving belt simulating a real-time production line, where the obtained results show less favorable performance, leading to corresponding conclusions.

Índice

1 INTRODUCCIÓN 1

2 REVISIÓN BIBLIOGRÁFICA 3

2.1 INTRODUCCIÓN 3

2.2 TRABAJOS PREVIOS 3

2.3 DISCUSIÓN 7

3 DEFINICIÓN DEL PROBLEMA 9

3.1 INTRODUCCIÓN 9

3.2 OBJETIVOS 9

3.2.1 Objetivo general 9

3.2.2 Objetivos específicos 9

3.3 ALCANCES Y LIMITACIONES 9

3.4 METODOLOGÍA 10

4 MATERIALES Y MÉTODOS 12

4.1 INTRODUCCIÓN 12

4.2 ¿QUÉ ES LA VISIÓN ARTIFICIAL? 13

4.3 CAPTURA DE IMÁGENES EN MOVIMIENTO 13

4.4 SET UP 14

4.5 VISIÓN ARTIFICIAL EN MATLAB 15

4.6 GROUND TRUTH DATA 15

4.7 ENTRENAMIENTO DE UN DETECTOR DE OBJETOS 17

4.8 PROBLEMAS DE UN DETECTOR DE OBJETOS. 18

4.9 DETECTOR DE OBJETOS APLICADO A UN VIDEO 21

4.10 DETECTOR DE MÚLTIPLES OBJETOS APLICADO A UN VIDEO 23

4.11 PROBLEMAS DE ENTRENAMIENTO 24

4.12 DETECCIÓN DE OBJETOS EN TIEMPO REAL 25

4.13 FRAMES PER SECOND. 26

4.14 INSERTAR FECHA Y HORA 28

4.15 EVALUACIÓN DE LOS DETECTORES DE OBJETOS 29

4.16 ESPECIES DISPONIBLES A DETECTAR 32

4.17 RESULTADOS 35

4.18 APLICACIÓN DE DETECTORES CON LAS DEMÁS ESPECIES 37

5 CONCLUSIONES 42

5.1 RESUMEN 42

5.2 RESULTADOS OBTENIDOS 42

5.3 TRABAJO FUTURO 43

6 GLOSARIO 45
7 REFERENCIAS 46
8 ANEXO 52
A. INVESTIGACIÓN PREVIA. 52

Lista de Figuras

Fig. 2.1: Detección y clasificación de pescados que pueden causar alergias utilizando algoritmos de aprendizaje profundo (deep learning) y redes neuronales convolucionales (Convolutional Neural Networks, CNN) [9]. 3

Fig. 2.2: Arquitectura de pirámide de características sensible a la escala llamada SA-FPN para extraer características robustas y abundantes en imágenes submarinas y mejorar el rendimiento en la detección de objetos marinos usada en [17]. 4

Fig. 2.3: Algoritmo de detección basado en Faster R-CNN (Region Convolutional Neural Networks) mejorado para la identificación de mariscos en contextos reales usado en [10]. 5

Fig. 2.4: Diagrama de flujo de las soluciones de clasificación usadas en [18]. 5

Fig. 2.5: Muestras de mejillones con y sin valva usados en [21]. 6

Fig. 4.1: Set up para captura de imágenes 14

Fig. 4.2: Vista aérea del Set Up 14

Fig. 4.3: Aplicación Video Labeler de MATLAB 16

Fig. 4.4: Detección de mejillones con superposición de bboxes. 19

Fig. 4.5: Histograma de puntajes detectados 20

Fig. 4.6: Mejillones detectados correctamente 21

Fig. 4.7: Detección de Piedras 23

Fig. 4.8: Detección de bisos 23

Fig. 4.9: Ajuste de cámara en iVCam 26

Fig. 4.10: Detector de mejillones a otras especies 37

Fig. 4.11: Detector de mejillones a otras especies 2 37

Fig. 4.12: Detección Mejillones (Valva) a especies 37

Fig. 4.13: Detección Mejillones (Valva) a especies 2 37

Fig. 4.14: Detección de Almejas a otras especies 37

Fig. 4.15: Detección de Almejas a otras especies 2 37
Fig. 4.16: Detección de Almejas Interior a especies 38
Fig. 4.17: Detección de Almejas Interior especies 2 38
Fig. 4.18: Detección de Caracol Negro a especies 38
Fig. 4.19: Detección de Caracol Negro a especies 2 38
Fig. 4.20: Detección Caracol Negro Post. a especies 38
Fig. 4.21: Detección Caracol Negro Post. a especies 2 38
Fig. 4.22: Detección de Algas Verdes a otras especies 38
Fig. 4.23: Detección de Algas Verdes a especies 2 38
Fig. 4.24: Detección de Algas Pardas a especies 39
Fig. 4.25: Detección de Algas Pardas a especies 2 39
Fig. 4.26: Detección de Huiros a otras especies 39
Fig. 4.27: Detección de Huiros a otras especies 2 39
Fig. 4.28: Detección de Bisos a otras especies 39
Fig. 4.29: Detección de Bisos a otras especies 2 39
Fig. 4.30: Detección de Vidrios Claros a especies 39
Fig. 4.31: Detección de Vidrios Claros a especies 2 39
Fig. 4.32: Detección de Vidrios Verde a especies 40
Fig. 4.33: Detección de Vidrios Verde a especies 2 40
Fig. 4.34: Detección Piedras Obscuras a especies 40
Fig. 4.35: Detección Piedras Obscuras a especies 2 40
Fig. 4.36: Detección Piedras Claras a otras especies 40
Fig. 4.37: Detección Piedras Claras a otras especies 2 40
Fig. 4.38: Detección Piedras Rojas a otras especies 40
Fig. 4.39: Detección Piedras Rojas a otras especies 2 40

Fig. 8.1: Setup cámara Pika L. 52

Fig. 8.2: Capturas realizadas con cámara Pika L sobre base blanca. 52

Fig. 8.3: Especies disponibles en laboratorio para análisis espectral. 53

Fig. 8.4: Análisis espectral de distintas especies disponibles en laboratorio 53

Fig. 8.5: Banda espectral y espectro visible. 54

1 Introducción

En el ámbito de la ingeniería civil en telecomunicaciones, se ha observado un aumento en la importancia del procesamiento digital de imágenes y el uso de tecnologías de detección de objetos en diversos campos de aplicación. Uno de estos campos es la detección de objetos no deseados en líneas de producción, el cual plantea un desafío relevante en la industria alimentaria.

Tradicionalmente, la detección de objetos no deseados en las líneas de producción se ha hecho manualmente, lo que implica un proceso lento, costoso y susceptible a errores humanos. No obstante, el avance de las tecnologías de procesamiento de imágenes y el aprendizaje automático ha permitido explorar nuevas soluciones automatizadas.

En este proyecto se espera desarrollar o implementar un modelo de aprendizaje supervisado mediante Machine Learning (M.L.) o similar, que sea capaz de detectar objetos no deseados en las líneas de producción de mejillones, tales como piedras, valvas, algas o plásticos. El sistema deberá ser robusto, preciso y rápido, para poder adaptarse a las condiciones reales de la industria. Para ello, se espera utilizar visión artificial, con el fin de extraer características relevantes de las imágenes y clasificarlas según su contenido.

“El procesamiento de imágenes digitales es el conjunto de técnicas aplicadas a las imágenes digitales para mejorar la calidad o facilitar la búsqueda de información” [1]. “Machine learning es una forma de la IA que permite a un sistema aprender de los datos en lugar de aprender mediante la programación explícita” [2]. Ambas disciplinas han experimentado un gran desarrollo en las últimas décadas, gracias al avance de la tecnología y la disponibilidad de grandes cantidades de información.

Hechos importantes en la historia para el desarrollo de estos medos son por ejemplo en 1952, Arthur Samuel creó el primer programa de juego de damas chinas, sentando las bases para el aprendizaje automático en juegos de computadora [3]. En 1957, Frank Rosenblatt diseñó el Perceptrón, uno de los primeros dispositivos para crear redes neuronales y reconocimiento de patrones [3]. En 1967, se introdujo el algoritmo del vecino más cercano (k-NN), un algoritmo fundamental de clasificación en M.L [3]. En 1970, Seppo Linnainmaa desarrolló la diferenciación automática, un conjunto de técnicas utilizadas para evaluar derivadas en cálculos computacionales, fundamental para el entrenamiento de redes neuronales [3]. En 2006, Netflix creó el premio Netflix, incentivando el desarrollo de algoritmos de M.L. más eficaces para mejorar su sistema de recomendación de contenido [3], entre otros hitos.

El uso de estas técnicas en la industria alimentaria y de mariscos tiene una gran importancia, ya que permite mejorar la calidad, la seguridad y la eficiencia de los procesos productivos [4]. Algunas aplicaciones son la clasificación [5], el conteo [6], la inspección y la detección de defectos o anomalías en los productos [7], junto a una basta cantidad de aplicaciones que se le pude dar al uso de M.L. y visión artificial. En particular, la detección de objetos no deseados en las líneas de producción de mejillones es un problema relevante, ya que puede afectar a la salud de los consumidores y al prestigio de las empresas.

2 Revisión Bibliográfica

2.1 Introducción

En la bibliografía encontrada se mencionan repetidamente métodos de M.L.[8] y algoritmos de procesamiento de datos e imágenes, se encontraron trabajos relacionados con la acuicultura específicamente en china usando métodos de M.L y deep learning (D.L.) para clasificación de peses para personas alérgicas usando un algoritmo CNN [9] y Clasificación del reconocimiento de mariscos basado en el marco mejorado Faster R-CNN con D.L. [10] , también trabajos en otras áreas como por ejemplo en la agricultura, monitoreando el contenido de agua en las hojas de maíz con datos hiper espectrales [11], Detección de granos de arroz dañados por insectos utilizando una técnica de imagen hiper espectral visible e infrarroja cercana [12], D.L. e Imágenes Hiper espectrales (i.h.e) Basados en Estimación de Firmeza y Contenido de Sólidos Solubles de Tomate [13], Datos de i.h.e y métodos de indexación de bandas de onda para estimar la concentración de nutrientes en cultivaros de lechuga [14] y Comparación entre redes neuronales artificiales y modelos de regresión de mínimos cuadrados parciales para el modelado de dureza durante el proceso de maduración de queso tipo suizo usando perfiles espectrales [15].

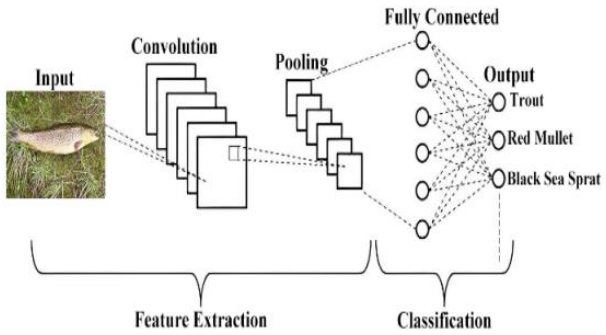


Fig. 2.1: Detección y clasificación de pescados que pueden causar alergias utilizando algoritmos de aprendizaje profundo (deep learning) y redes neuronales convolucionales (Convolutional Neural Networks, CNN) [9].

2.2 Trabajos Previos

La bibliografía publicada como Automatic image pixel clusterin based on mussels wandering optimization [16] muestra un método de agrupación automática pixeles que permite la clasificación de imágenes RGB y en escala de grises, por color y formas, mostrando que no es necesario tener características espectrales para clasificar imágenes y la publicación Scale-aware feature pyramid architecture for marine object detection [17], muestra una algoritmo de Arquitectura piramidal de

funciones con reconocimiento de escala para la detección de objetos marinos, que toma imágenes en tiempo real de distintas especies marinas y es capaz de clasificarlas en el fondo marino.

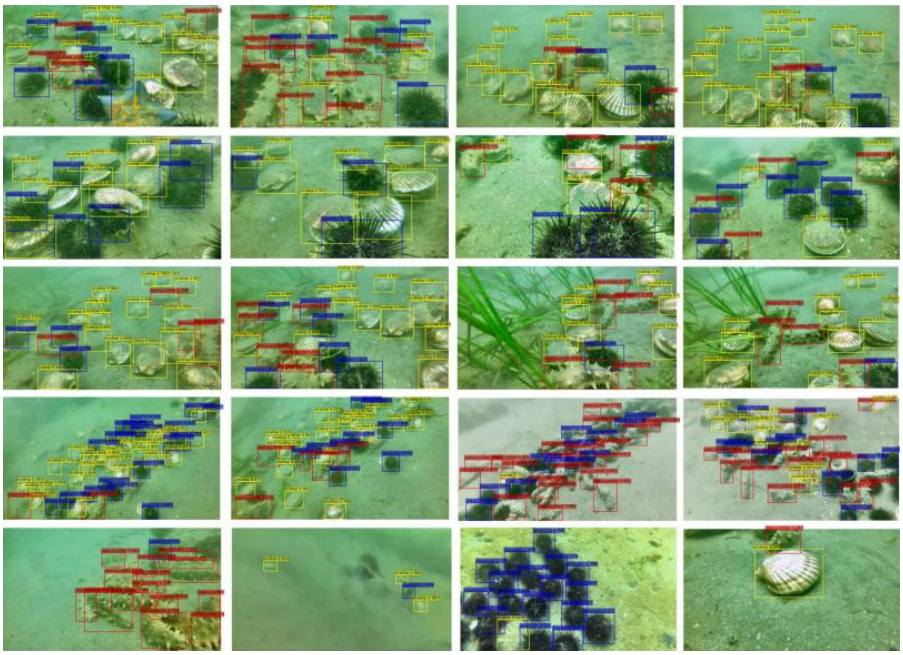


Fig. 2.2: Arquitectura de pirámide de características sensible a la escala llamada SA-FPN para extraer características robustas y abundantes en imágenes submarinas y mejorar el rendimiento en la detección de objetos marinos usada en [17].

Hay cierta cantidad de investigaciones realizadas en el uso de métodos de M.L. y D.L. para la clasificación y detección de especies marinas por ejempló en la antes nombrada [10] se ocupa una combinación entre modelos de D.L. con técnicas de M.L. para clasificar imágenes de 10 especies marinas con bases de datos existentes de D.L. modificadas para cada especie, además se utilizan 4 distintos modelos de M.L. donde el modelo CNN mostro mejor rendimiento con un 99.6% de precisión en la predicción, contra modelos ANN, SVM y KNN que obtuvieron 95.3%, 96.8% y 98.2% de precisión respectivamente. Los investigadores aseguran que este rendimiento no es para todas las especies marinas, solo para las cuales el modelo fue entrenado, mostrando que entrenando un modelo se puede tener una clasificación precisa con baja tasa de errores.

También en [10] se clasifican mariscos con un modelo de D.L Faster R-CNN, este es uno de los estudios más cercanos a lo que se espera conseguir, ya que toman cuatro distintas especies de mariscos y se clasifican en tiempo real con 3 redes de datos de modelos de D.L, ResNet, DenseNet y DenseNet + Soft-NMS, se analizan 4 especies que son Conch, Scallop, Clam y Mussels, esta última son mejillones como los que se quieren analizar en este estudio, el problema es que en [10] se analiza el

Mejillón con valva y en este estudio se espera analizar el mejillón sin valva cullas características difieren bastante en una imagen, el resultado de [10] da como resultado que el uso de DenseNet + Soft-NMS es mejor para la clasificación consiguiendo rendimientos para las cuatro especies de 75.1% 80.3% 88.3% y 84.6% respectivamente sacando casi 4% de diferencia entre la ResNet de peor rendimiento. Muestra de nuevo que se pueden clasificar especies marinas, en este caso con el uso de D.L. en mariscos con valva.

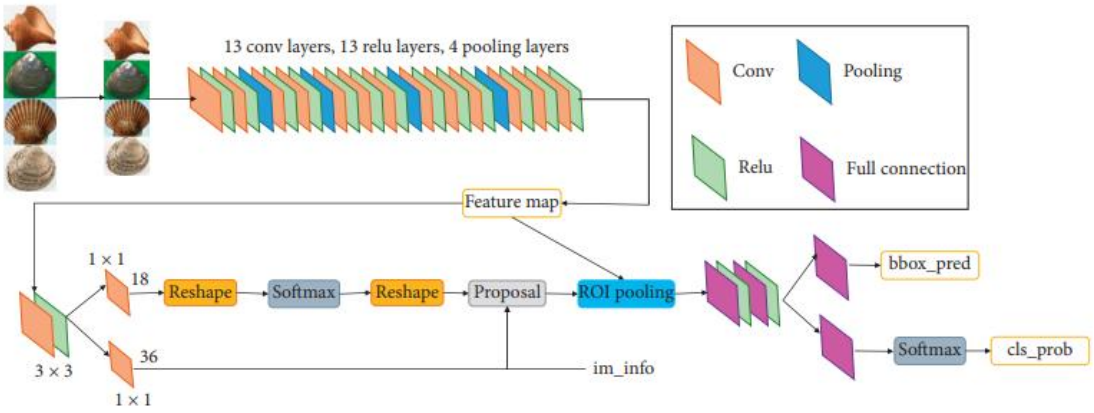


Fig. 2.3: Algoritmo de detección basado en Faster R-CNN (Region Convolutional Neural Networks) mejorado para la identificación de mariscos en contextos reales usado en [10].

Recientemente se han hecho publicaciones como Mussel Classifier System Based on Morphological Characteristics [18], se presenta el desarrollo de un sistema automático que clasifica 5 especies de mejillones con valva provenientes de Chiles en la región del Bío Bío y Los lagos, basándose en características morfológicas de estos para su reconocimiento y clasificación en simultaneo.

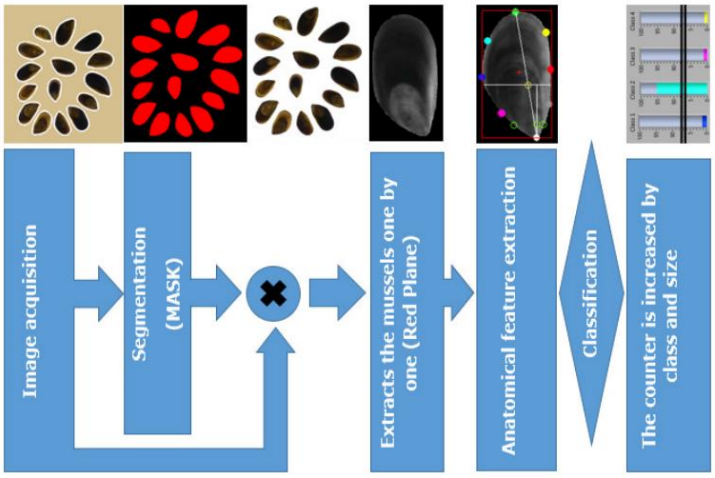


Fig. 2.4: Diagrama de flujo de las soluciones de clasificación usadas en [18].

Otra publicación de interés es acerca la detección rápida de mejillones contaminados por metales pesados mediante espectroscopía de reflectancia en el infrarrojo cercano y una máquina de aprendizaje extremo (ELM) con diferencia restringida [19] que revisa el espectro infrarrojo cercano (NIR) de mejillones no contaminados y contaminados con Zn, Pb, Cd y Cu, y se usa una ELM como clasificador consiguiendo resultados de precisión mayores a 97%.

También se encontró una publicación acerca de la rápida detección de toxinas causantes de intoxicación por mariscos diarreicos en mejillones utilizando espectroscopía NIR y Máquinas de Soporte Vectorial Gemelas Mejoradas (TWSVM) [20], Las toxinas causantes de la intoxicación por mariscos diarreicos o diarrhetic shellfish poisoning (DSP), como describe la publicación, puede causar graves enfermedades intestinales que se quiere evitar, por lo que para la detección en mejillones se usó NIR junto a reconocimiento de patrones para detectar las toxinas DSP en mejillones sanos y contaminados y se usó una TWSVM para la clasificación consiguiendo valores de f1-score = 0.9886, y la precisión de detección alcanzó el 98.83%.

Por último, se tiene una publicación acerca del aprendizaje automático asistido por espectroscopía en infrarrojo cercano y medio (MIR) para la discriminación rápida de mejillones mediterráneos salvajes y de cultivo [21], donde se usa NIR y MIR junto a algoritmos de M.L. para diferenciar entre mejillones salvajes y de cultivo donde se consiguieron valores de precisión mayores a 90%.

Fig. 2.5: Muestras de mejillones con y sin valva usados en [21].

2.3 Discusión

De la bibliografía se puede ver que es posible la clasificación de especies mediante imágenes, una de las posibilidades que se consideran en este estudio es el uso de i.h.e., debido a que se cuenta con una cámara capaz de realizar estas imágenes, pero no es la única opción considerable, los estudios más destacados en la sección anterior usan imágenes RGB para el estudio de las distintas especies que analizaron. Lo necesario para la realización de estas investigaciones, que cuentan con el uso de modelos de M.L. o D.L. para clasificar imágenes son bases de datos. Para entrenar los modelos mencionados antes se usaron una gran cantidad de imágenes, porque esa es la forma en que los modelos aprenden o ya aprendieron en caso de las redes de D.L. ya establecidas.

Para un modelo de M.L. primero hay que tener claro el problema a resolver, a lo que se desea llegar, o lo que se espera que salga del modelo, además de los datos o características que se entregan para que el modelo pueda aprender de estas entradas que le permitan reconocer la solución que debería tener el problema o la solución esperada. “El aprendizaje supervisado utiliza un conjunto de datos de entrenamiento para enseñar a los modelos a generar la salida deseada. Este conjunto de datos de entrenamiento de datos incluye entradas y salidas correctas que, a su vez, permiten que el modelo aprenda con el tiempo. El algoritmo mide su exactitud a través de la función de pérdida, ajustándose hasta que el error se ha minimizado lo suficiente” [22]. por eso se requiere una gran base de datos de muestras con las características que queremos dar al modelo para entrenarlo y conseguir una salida deseada.

Los datos suelen estar distribuidos de distintas maneras y los distintos modelos buscan encontrar la forma de ordenar estos de cierta forma y así asignarles un valor de salida, el problema es que a veces ciertos datos caen en los conjuntos que no deberían y aun así se clasifican con un valor equivocado y esto genera un porcentaje de imprecisión en las predicciones echas por el modelo, por esto es que hay varios modelos que ordenan los datos de distintas maneras para que ciertos datos que se le podrían haber pasado a un modelo no se le pasen a otros, por ejemplo las KNN ordenan los valores con respecto a k vecinos cercanos, ya que se supone que los valores similares deberían representar lo mismo, pero cuando esto no se cumple suele tener errores en la predicción, otros modelo por ejemplo es el lineal que crea una línea entre distintas agrupaciones de datos y a estas le asigna una salida, también puede tener un error, pero será distinto al anterior y así con otros modelos, dependiendo de los datos algunos serán mejores que otros en la clasificación de ciertos datos, pero se busca independiente de la cantidad de datos estos se mantengan con un cierto porcentaje de precisión para

ser considerados buenos modelos ya que si varia demasiado este no sería confiable.

Pero no siempre se quiere entregar datos numéricos como entrada a un modelo a veces se desea entregar imágenes, voz o texto, en estos casos se puede usar un tipo especifico técnica de M.L. que es el Deep Learning, que usa redes neuronales para extraer características y realizar predicciones, anteriormente se nombró el uso de estas técnicas para clasificar imágenes, la gran diferencia con los modelos de M.L. tradicionales es que estas redes neuronales que se mencionan ya fueron entrenadas previamente por algún otro ente investigador y permiten el uso de bases de datos predeterminados para implementar la detección, lo bueno de estas es que aceptan imágenes, y no debería ser necesario extraer características manualmente como en los modelos entrenados normalmente, el problema es que no siempre la salida que se espera conseguir está en las bases de datos predeterminadas, y es necesario elegir alguna base de datos que si tenga lo que se necesita o en caso de no tenerlo usar igualmente otras técnicas para adaptar los modelos.

Otro aspecto a tener en cuenta en los trabajos previos es el uso del NIR o el MIR, que ya consiguió resultado en la detección de metales pesados, especies distintas y especies contaminadas, está comprobado que las distintas especies tienen diferencias espectrales que facilitan su clasificación, además las publicaciones encontradas muestran que es posible discriminar especies con distintos algoritmos como el TWSVM y ELM, y además se consiguen puntajes de precisión mayores a 90% lo que es considerablemente bueno para implementar en la industria. El inconveniente del NIR y MIR es el tamaño de los datos que son mayores a una imagen RGB, lo que aumentaría el tiempo de procesamiento de datos en tiempo real o requeriría un hardware de mayor calidad y por lo tanto aumentaría los costos de implementación.

3 Definición del Problema

3.1 Introducción

En este trabajo se espera lograr determinar un método capaz de procesar los datos obtenidos por la cámara en tiempo real y que identifique los objetos no deseados en la captura. Entre los objetos no deseados se considera cualquier objeto que no sea un mejillón sin valva.

3.2 Objetivos

3.2.1 Objetivo general

El objetivo es entrenar un detector de objetos, que permita identificar los objetos no deseados en una línea de producción de mejillones, se espera identificar distintos objetos que se consideren contaminantes o no deseados en el contexto planteado y así detectar tanto mejillones como no mejillones en una imagen en tiempo real.

3.2.2 Objetivos específicos

1. Grabación de mejillones y objetos no deseados en el contexto de una línea de producción o similar.
2. Desarrollar competencia en la detección de objetos utilizando MATLAB.
3. Desarrollar e implementar detectores de objetos para una variedad de categorías en MATLAB.
4. Utilizar los detectores de objetos implementados para realizar la detección en tiempo real de video o imágenes utilizando MATLAB.

3.3 Alcances y Limitaciones

Alcances:

- Detección de mejillones en imágenes o video: El proyecto incluirá el entrenamiento de un detector de objetos que pueda identificar mejillones en imágenes o secuencias de video.
- Detección de objetos no deseados: El proyecto avanzará para identificar otros objetos que se consideren contaminantes o no deseados en un contexto específico.

- Modelo integral: Se espera desarrollar un modelo completo capaz de detectar tanto mejillones como objetos no deseados en una imagen.
- Pruebas en tiempo real: Se trabajará en la implementación de este modelo para que funcione en la captura en tiempo real, lo que puede implicar la integración con hardware de captura de imágenes en tiempo real, como cámaras.

Limitaciones:

- Precisión de detección: La precisión de detección dependerá del tipo de objetos no deseados, la calidad de las imágenes y la cantidad de datos de entrenamiento disponibles. Además, la eficacia del modelo puede variar según la diversidad de objetos no deseados en el contexto específico.
- Escalabilidad: Puede ser desafiante hacer que el modelo sea escalable para adaptarse a una amplia gama de situaciones y tipos de objetos no deseados.
- Hardware y recursos computacionales: La implementación en tiempo real podría requerir hardware adecuado, y la limitación de recursos computacionales podría ser un factor restrictivo. El hardware que tiene la computadora es una CPU Intel CORE i7-10750H, una GPU NVIDIA GeForce GTX 1660 Ti Max-Q (6 GB) y 16 GB de RAM DDR4.
- Entrenamiento de datos: La disponibilidad de datos de calidad puede ser una limitación, especialmente si los objetos no deseados son variados y poco comunes a la hora de entrenar los detectores de objetos, además la efectividad del modelo puede verse afectada por las condiciones ambientales, como iluminación y fondo.

3.4 Metodología

Objetivo: Grabación de mejillones y objetos no deseados en el contexto de una línea de producción o similar.

Se llevará a cabo una metodología que implica la configuración de una cámara en el entorno de producción y la grabación de videos que contengan tanto mejillones como objetos no deseados en movimiento. Estos videos servirán como base de datos para el entrenamiento de detectores de objetos. Se utilizarán herramientas de edición de video para extracción de fotogramas y etiquetado de objetos

de interés y no deseados. A continuación, se desarrollarán y entrenarán detectores de objetos en MATLAB utilizando esta base de datos.

Objetivo: Desarrollar competencia en la detección de objetos utilizando MATLAB.

Se llevará a cabo la adquisición de conocimientos fundamentales sobre el procesamiento de imágenes y la detección de objetos. Se aprovecharán recursos de aprendizaje en línea, tutoriales y documentación de MATLAB. Además, se realizarán ejercicios prácticos para aplicar los conceptos adquiridos y se ajustarán parámetros de los detectores de objetos en ejemplos de prueba.

Objetivo: Desarrollar e implementar detectores de objetos para una variedad de categorías en MATLAB.

Se seguirá una metodología que implica la selección de categorías de objetos de interés y la obtención de conjuntos de datos adecuados para cada categoría. Se procederá a etiquetar los objetos en las imágenes, y se entrenarán detectores específicos para cada categoría utilizando técnicas de aprendizaje automático y algoritmos de visión por computadora en MATLAB. Finalmente, se evaluará el rendimiento de los detectores en conjuntos de datos de prueba.

Objetivo: Utilizar los detectores de objetos implementados para realizar la detección en tiempo real de video o imágenes utilizando MATLAB.

Se integrarán los detectores previamente entrenados en un entorno de procesamiento de imágenes en tiempo real. Se programarán scripts y algoritmos en MATLAB que permitan la adquisición de video en vivo o el procesamiento de imágenes en tiempo real a partir de una cámara. Los detectores se utilizarán para identificar y marcar objetos de interés en tiempo real, y se mostrarán los resultados en una interfaz gráfica o mediante visualización en pantalla.

4 Materiales y Métodos

4.1 Introducción

La visión artificial es increíblemente útil en diversas áreas, y en particular, es de gran utilidad para la detección de objetos. Por esta razón, será de gran valor para el propósito de este proyecto. Se utilizará MATLAB para desarrollar un detector que pueda identificar mejillones y otros objetos considerados no deseados en imágenes. Cuando se detecten estos objetos no deseados, se emitirá una advertencia para facilitar su eliminación en el contexto de imágenes de una línea de producción de mejillones. Además, se espera que el detector funcione en tiempo real en última instancia y que pueda reconocer varios objetos que no deberían estar en la imagen. Para lograr esto, se utilizarán detectores de imágenes personalizados diseñados específicamente para el proyecto, con el objetivo de lograr la máxima precisión posible.

Al inicio de este proyecto, antes de convertirse en una Memoria de título, se comprobó que era posible detectar mejillones mediante una cámara que captura imágenes en RGB. Esto se logró mediante la adquisición de imágenes hiperespectral (i.h.e) que capturaban esta y otras especies marinas. Con estas imágenes, se estudió el espectro de las distintas especies en el rango de 400 a 1000 nanómetros, centrándose en el espectro del mejillón en comparación con otras especies. Se encontraron características espectrales clave en el rango de 410 nm a 500 nm, donde diferentes mejillones presentaban un espectro similar con una distancia reducida entre puntos en comparación con otras secciones del espectro. Esta diferencia fue suficiente para descartar algunas especies que no compartían una característica similar en esa sección del espectro. Sin embargo, dado que algunas especies cumplían con estas características, se buscó otro distintivo en la especie del mejillón. Se encontró una cualidad en el rango de 530 nm a 580 nm que mostraba una curva creciente en los gráficos del mejillón que otras especies no tenían. Esto permitió crear 2 filtros que lograron una clasificación espectral del mejillón y lo diferenciaron de otras especies marinas que fueron fotografiadas.

Además, el estudio concluyó que las principales características del mejillón se encontraban dentro del espectro visible para el ojo humano, lo que sugirió que era viable utilizar una cámara de menor costo y especificaciones en lugar de una cámara hiperespectral más cara para continuar la investigación.

Posteriormente, se realizaron experimentos con imágenes de mejillones sin valva obtenidas con la cámara de un teléfono Samsung A22 como base de datos. Las imágenes se convirtieron a escala de

grises para reducir su tamaño y se extrajeron cuatro características de las distintas imágenes. Se creó una tabla de características para entrenar los diferentes modelos disponibles en MATLAB a través de la aplicación "classification learner". Se utilizaron 34 modelos para la clasificación y se obtuvieron resultados en las pruebas de 34 modelos entrenados, con un rendimiento que varió entre un 95.36% y un 98.05% de precisión. Estos resultados no se obtuvieron en un entorno de línea de producción, por lo que no garantizan una efectividad similar en ese contexto, pero indican un avance para seguir experimentando en un entorno industrial.

De esta manera, se determinó que era posible detectar imágenes con una cámara de bajo costo, como la de un teléfono, y es con esta que se continuara la investigación en este informe. Se tienen expectativas de obtener resultados satisfactorios basados en los experimentos anteriores y se confía en que la visión artificial cumplirá con su cometido.

4.2 ¿Qué es la visión artificial?

Según IBM “la visión artificial es un campo de la IA que permite que las computadoras y los sistemas obtengan información significativa de imágenes digitales, videos y otras entradas visuales, y tomen acciones o hagan recomendaciones basadas en esa información” [23].

La visión artificial permite a las máquinas ver y comprender su entorno. Aunque su funcionamiento es similar al de la visión humana, los humanos tienen la ventaja de la experiencia y el contexto. La visión artificial capacita a las máquinas para realizar estas tareas de manera más eficiente, utilizando cámaras, datos y algoritmos. Esto permite que un sistema de visión artificial pueda analizar rápidamente miles de elementos o procesos, superando las capacidades humanas al detectar defectos o problemas imperceptibles.

4.3 Captura de imágenes en movimiento

El trabajo realizado anteriormente tuvo como objetivo comprobar si era posible detectar un mejillón mediante algoritmos de ML. Los resultados demostraron que el software es capaz de detectar tanto la especie como de diferenciarla de otras, utilizando características de interés presentes en las imágenes. Los modelos lograron hacer predicciones con gran exactitud.

El objetivo de esta investigación es detectar mejillones en una línea de producción en movimiento. Esto significa que el objeto de interés se encuentra en constante movimiento. Además, no se busca únicamente detectar mejillones en una cinta en movimiento, sino también identificar objetos no deseados en este contexto. Para lograr este objetivo, se avanzará en el siguiente punto.

Se simuló una cinta transportadora que lleva mejillones y otros objetos para realizar pruebas de detección. Se configuró un montaje donde se ubicó la cámara sobre la superficie de la cinta, que consiste en una plancha de cartón recubierta de papel blanco y se protegió con film plástico para evitar el contacto con el agua o la humedad proveniente de los mejillones. La plancha se desplaza a una velocidad constante, y los objetos de interés se colocan en ella. La cámara, ubicada en una posición elevada, abarca toda la anchura de la plancha, lo que permite capturar cualquier movimiento similar al de una cinta transportadora.

4.4 Set up

Para la captura de imágenes y videos de mejillones en una cinta, se armó un set up que consta de un soporte de madera fijado al techo. Este se construyó para ubicar la cámara a una altura de 0.40 metros sobre la mesa que sostendrá al cartón cubierto con papel blanco y film plástico que simula la cinta transportadora donde se colocaron los mejillones y objetos no deseados. El soporte cuenta con una ampolleta de luz blanca en su extremo con fines de iluminación. Además, se instaló una lámpara colgante en el techo con una ampolleta de luz blanca también para iluminar la escena. Se instalaron dos ampolletas con el fin de iluminar la entrada y la salida de objetos en la escena. Además, se utiliza el flash de la cámara para iluminar la zona central de la imagen.

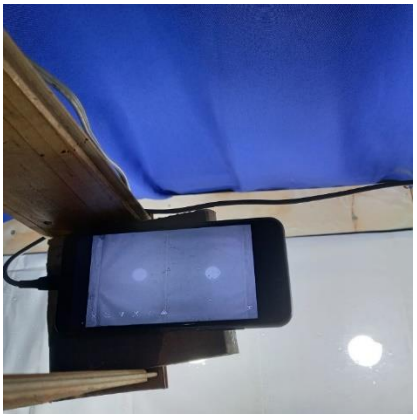


Fig. 4.1: Set up para captura de imágenes

Fig. 4.2: Vista aérea del Set Up

4.5 Visión artificial en Matlab

La visión artificial permite a la computadora detectar patrones de interés en diversos objetos presentes en una imagen. Estas imágenes suelen corresponder a los frames de un video que capturan el objeto que se desea analizar. Según la RAE, un frame o fotograma se define como "cada una de las imágenes que se suceden en una película cinematográfica" [24]. Otra definición que podría ser más acertada para esta investigación es que el frame es “una imagen concreta dentro de una sucesión de imágenes en movimiento (video o animación)” [25].

Para trabajar con videos en Matlab, existe la función VideoReader que permite crear una variable que “lee” y almacena archivos que contienen datos de video. Esta variable contiene información del archivo de video, como el tamaño de la imagen y la duración del video, entre otras propiedades del archivo [26]. Además, facilita la importación de los fotogramas de forma individual para su posterior procesamiento, lo que simplifica el trabajo posterior.

Después de crear un VideoReader, se pueden extraer los fotogramas de la imagen mediante la función “readFrame”, la cual lee el siguiente fotograma de video disponible del archivo de video v [27]. Esta función se utiliza para analizar los fotogramas como imágenes independientes. Sin embargo, dado que es necesario detectar mejillones en una cinta en movimiento y seguir su cambio de ubicación de un frame a otro, es esencial contar con una base de datos que contenga las características de los objetos de interés. Para este propósito, es posible utilizar modelos que identifiquen estas características a partir de una secuencia de imágenes o de un video. En Matlab, esto se logra mediante una aplicación llamada Video Labeler, que permite crear un modelo que representa cómo se ve el objeto.

4.6 Ground Truth Data

Se debe utilizar un modelo que identifique características. Son necesarios datos para obtener más información, y los datos obtenidos dependerán del modelo que se utilice. En este contexto, generalmente, más datos son preferibles. No se requieren solo imágenes, sino también las ubicaciones y las etiquetas de los objetivos que se desean detectar con el algoritmo.

La base de datos que se utiliza para entrenar un detector mediante M.L. o D.L. se conoce como "Ground Truth Data" (Datos de Verdad o GTD). Esto crea un modelo de cómo los humanos pueden describir "cómo se ve el objeto". Un buen modelo se sustenta en un sólido GTD. Para construirlo, se pueden emplear videos de los objetos de interés en el contexto relevante para la investigación.

Para definir un GTD se puede utilizar la aplicación "Video Labeler" de Matlab, que permite importar un video para etiquetar los objetos de interés que se observan en él. La aplicación permite asignar nombres a las etiquetas y utilizar diferentes geometrías para seleccionar y etiquetar objetos, como rectángulos o conjuntos de puntos que forman figuras. La aplicación mostrará los fotogramas del video seleccionado, con el primer fotograma como predeterminado. En los fotogramas posteriores, se pueden seleccionar áreas que se deseen etiquetar. Esto se puede hacer manualmente, frame a frame, o de manera automatizada mediante varios algoritmos que ofrece la aplicación [28].

Para esta MDT, se simula una línea de producción que transporta una gran cantidad de objetos similares o de la misma especie desde el punto A al punto B, con la línea moviéndose en una dirección constante. El algoritmo utilizado para automatizar el etiquetado se llama "Point Tracker", que rastrea características de un fotograma a otro y puede manejar varios objetos diferentes que no se desplazan en línea recta, etiquetándolos simultáneamente. Dado el número de objetos que se planea detectar, se utilizará este algoritmo para definir el Ground Truth Data.

Al iniciar la automatización del etiquetado, el algoritmo agrega las etiquetas creadas en un fotograma específico del video al resto de los fotogramas. Luego, se puede revisar el trabajo del algoritmo y corregir posibles errores en la detección antes de guardar los cambios. Una vez completado el proceso de etiquetado, se pueden guardar los resultados como archivo o en el espacio de trabajo, lo que proporcionará una variable groundTruth (gT) para continuar con el código.

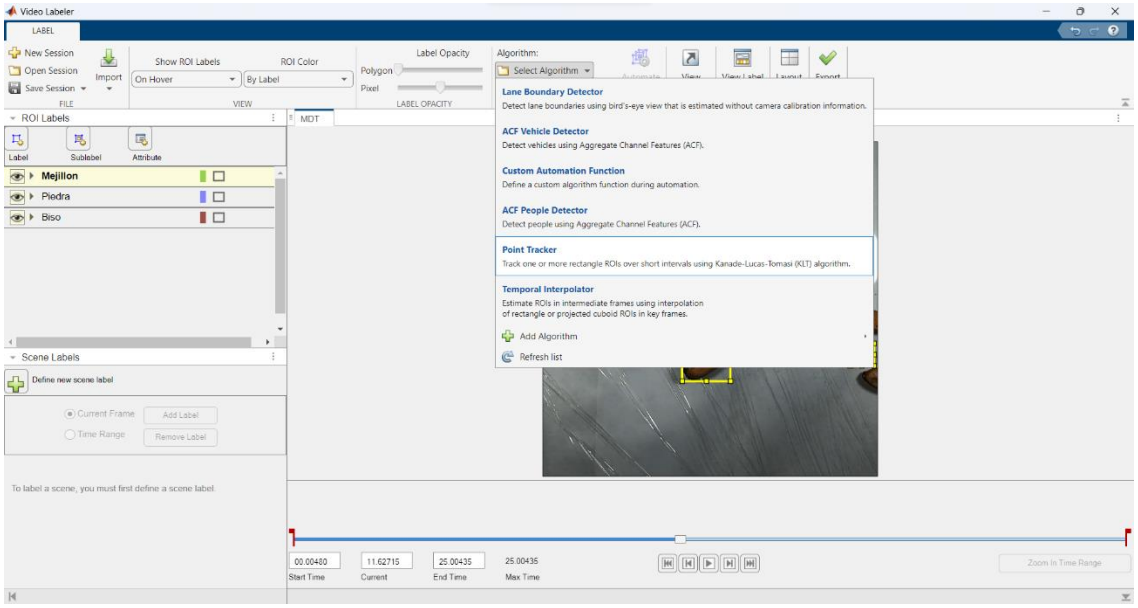


Fig. 4.3: Aplicación Video Labeler de MATLAB

4.7 Entrenamiento de un detector de objetos.

Al tener la variable gT, esta se guarda para su posterior uso. Se utiliza para crear un "objectDetectorTrainingData", que toma como entrada la variable gT y produce dos variables de salida: "imds" y "blds". Estas corresponden a una lista de imágenes y a las etiquetas de cajas delimitadoras para las imágenes. Estas etiquetas se crearon previamente en el GTD y ahora servirán para definir las imágenes que se utilizarán como base de datos [29].

La función "objectDetectorTrainingData" permite elegir conjuntos de imágenes para el entrenamiento. Por ejemplo, se pueden seleccionar imágenes del video para el entrenamiento del detector cada diez fotogramas en los que hay etiquetas. También, si el intervalo tiene un valor de uno, se pueden usar todos los fotogramas etiquetados. Además, la función ofrece diversas configuraciones que afectan la precisión del detector.

Se crea una variable que servirá como la base de datos para entrenar al detector. Esto se logra utilizando la función "combine" para unir la lista de imágenes con sus respectivas etiquetas [30]. Esta base de datos se usará como entrada para el detector.

Una vez que se tiene el conjunto de imágenes de entrenamiento, se puede entrenar al detector. En este caso, se utiliza la función "trainACFObjectDetector" para entrenar un detector. Esta función de MATLAB se emplea para entrenar un detector de objetos basado en ACF (canal de características de aprendizaje automático). ACF es una técnica popular para la detección de objetos que utiliza un conjunto de características extraídas de una imagen para identificar y delimitar objetos de interés.

La función "trainACFObjectDetector" permite entrenar un detector personalizado para una clase de objetos específica utilizando un conjunto de imágenes de entrenamiento etiquetadas. Además, ofrece una configuración detallada de parámetros que afectan la calidad del detector [31]. Después de entrenar el detector, este se puede guardar para su uso posterior cuando sea necesario.

```
%Carga de gTruth
load GT.mat


%Extrae todas las imágenes y las etiquetas de gTruth
[imList, boxLabels] = objectDetectorTrainingData(gTruth,...
    "SamplingFactor",1,...     %Factor de muestreo 1 usa todos los frames
    "NamePrefix","Mejillones",...    %Se da nombre a las imágenes
    "WriteLocation","D:\Grabaciones A22-5G\VIDEOS PROYECTO\MDT2\ImgTraining");
      %Ubicación donde se guardarán las imágenes
```

```
%Combina imágenes con sus etiquetas en una variable
imWithBoxLabels = combine(imList,boxLabels);

%Se define el detector y se entrena al compilar
detector = trainACFObjectDetector(imWithBoxLabels,"ObjectTrainingSize","auto")
```

4.8 Problemas de un detector de objetos.

Ahora que el detector está listo para funcionar, es el momento de ponerlo a prueba. Para ello, se puede utilizar el mismo video con el que se entrenó al detector o cualquier otro en el que aparezcan mejillones, que es lo que se debería detectar.

Primero, se carga el video utilizando la función "VideoReader". Luego, para aplicar el detector, se utiliza la función "detect", que toma como entradas el detector que se desea utilizar y una imagen, que en este caso sería un fotograma del video. Como resultado, se obtienen dos variables: "bboxes" (cuadros delimitadores) y "scores" (puntuaciones de confianza) de los cuadros delimitadores. También, puede proporcionar una variable "labels" (etiquetas) de los cuadros delimitadores [32].

Con esta función, el detector detectará lo que considere que es un mejillón y le asignará un cuadro delimitador, lo cual es el resultado deseado. Sin embargo, no está exento de problemas, como que un mejillón se detecte más de una vez en una imagen. Esto podría ocurrir si se entrenó al detector con un "point tracker". Por lo tanto, si el detector encuentra más de un punto en un mejillón que podría ser considerado un mejillón, lo marcará como tal. Esto se puede observar en la figura 3, donde hay una superposición de cuadros delimitadores.

```
load imList.mat
load boxLabels.mat
load detector.mat
%Carga el frame n almacenado en imlist a una imagen
n = 150;
f = imList.Files{n};
im = imread(f);
%Usa el detector en la imagen cargada
[bbox,score] = detect(detector,im);


%Inserta los bboxes en la imagen detectada
frameLabeled = insertObjectAnnotation(im,"rectangle",bbox,score);
%Muestra el número de bboxes
numBoxes = size(bbox,1)
```

```
numBoxes = 13
```

```
%Crea un string para mostrar el número de mejillones detectados
str = numBoxes + " mejillon(es) detected"
```

```
str = "13 mejillon(es) detected"
```

```
%Agrega el texto a la imagen
frameLabeled = insertText(frameLabeled,[50 50], str,"FontSize",60);
%Imagen con problema de superposición
imshow(frameLabeled)
```

Fig. 4.4: Detección de mejillones con superposición de bboxes.

Una solución al problema de la superposición es utilizar la función "selectStrongestBbox", que toma como entradas las variables "bbox" y "score". La función selecciona los cuadros delimitadores "más fuertes" de grupos superpuestos mediante la supresión no máxima (NMS) y devuelve cuadros delimitadores seleccionados que tienen una alta puntuación de confianza. La función emplea la NMS para eliminar los cuadros delimitadores superpuestos que provienen de la entrada. Además, se puede especificar un "OverlapThreshold" (Umbral de superposición) que permite a la función eliminar cuadros delimitadores alrededor del cuadro de referencia. Este umbral puede ser un valor entre cero y uno. Reducir este valor disminuirá la cantidad de cuadros delimitadores seleccionados. No obstante, si se reduce demasiado la relación de superposición, es posible que se eliminen cuadros que representen objetos cercanos entre sí en la imagen [33].

Además, se puede aplicar una limitación a las puntuaciones que proporciona el detector, por ejemplo, no considerar puntuaciones mayores a 30. Para determinar el valor, se puede examinar el histograma de las puntuaciones entregadas por el detector y, en función de los resultados, eliminar los resultados

que se consideren menos confiables. Aplicando las soluciones mencionadas anteriormente, en la figura 3 se puede eliminar la superposición de cuadros delimitadores en la imagen, como se puede apreciar en la figura 4 y el código que se muestra a continuación.

```
%Histograma de detecciones
histogram(score,10);
title("Puntajes Detectados")
xlabel("Puntaje")
ylabel("N° de Detecciones")
```

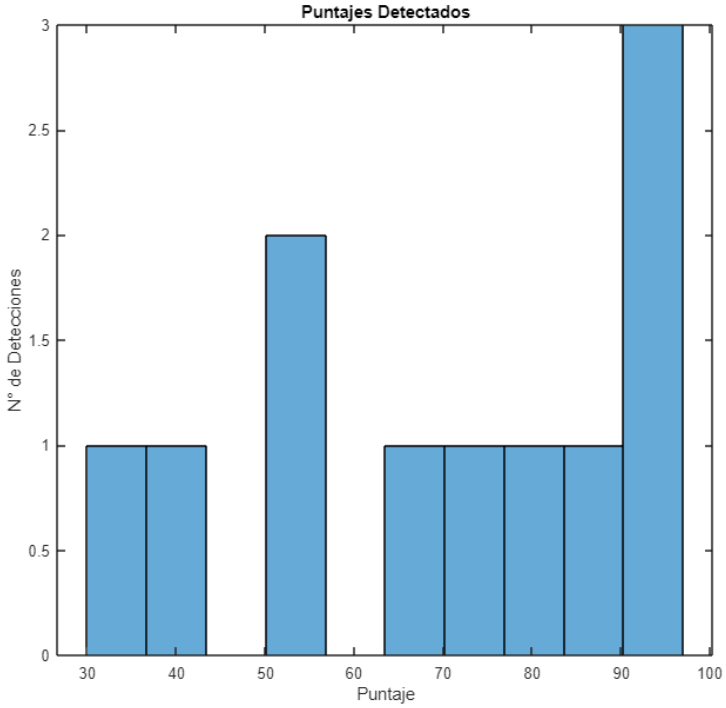


Fig. 4.5: Histograma de puntajes detectados

```
%Eliminación de bboxes con score menor a 30
bbox = bbox(score>30,:);
score = score(score>30);
%Selección de bboxes con mayor score
[selectedBbox,selectedScore] = selectStrongestBbox(bbox,score,...
    "OverlapThreshold",0.1);
%Inserta los bboxes con mayor score en la imagen detectada
img2 = insertObjectAnnotation(im,"rectangle",...
    selectedBbox,selectedScore);
%Muestra el número de bboxes
numBoxes = size(selectedBbox,1)
```

```
numBoxes = 8
```

```
%Crea un string para mostrar el número de mejillones detectados
str = numBoxes + " mejillon(es) detected"
```

```
str = "8 mejillon(es) detected"
```

```
%Agrega el texto a la imagen
imgCounted = insertText(img2,[50 50], str,"FontSize",60);
%Muestra la imagen con las bboxes y el número de detecciones correcta
imshow(imgCounted)
```

Fig. 4.6: Mejillones detectados correctamente

4.9 Detector de objetos aplicado a un video

Ahora que el problema de la superposición debería estar resuelto, se puede aplicar el detector a un video que contenga mejillones. La imagen utilizada en las figuras 3 y 4 proviene de un fotograma extraído del video que se usó para entrenar al detector. El código mostrado en la sección anterior es el utilizado para resolver la superposición.

Para aplicar la detección a los distintos fotogramas del video, se puede utilizar un ciclo "while" que lea todos los fotogramas del video. Esto se logra utilizando la función "hasframe", que devuelve un valor lógico "true" (1) si hay un fotograma disponible para leer y "false" (0) en caso contrario. De esta manera, el ciclo "while" se ejecutará hasta que no queden más fotogramas por leer [34].

Luego, se agrega una condición "if" con la función "isempty" que verifica si un arreglo está vacío, devolviendo "true" (1) en caso afirmativo y "false" (0) en caso contrario [35]. Dentro del "if", se utiliza

un "break" para salir del bucle cuando el fotograma sea nulo. Posteriormente, se incorpora el detector y la limitación del puntaje.

Después de esta condición, se añade otra "if", esta vez utilizando "isempty" con negación, para comprobar si "bbox" no es nulo. Si no lo es, se continúa la ejecución del código y se prosigue con la detección de los siguientes fotogramas. Dentro de esta condición, se realiza la selección de los cuadros delimitadores "más fuertes" y se aplica el resto del código utilizado en la sección anterior, que muestra la imagen con los mejillones detectados y cuenta la cantidad en cada fotograma.

Finalmente, se añade la función "drawnow", que actualiza la figura y permite procesar el siguiente fotograma del bucle, mostrando las actualizaciones de un objeto gráfico de inmediato en pantalla [36]. De esta manera, al concluir la ejecución del código, se generará una animación en formato de archivo de video MP4 que mostrará los fotogramas procesados por el detector, con los cuadros delimitadores sobre los mejillones detectados.

```
v = VideoReader("MDT.mp4");
% Crea una figura para mostrar el video
figure;
while hasFrame(v)
    frame = readFrame(v);


    if isempty(frame)
        % El cuadro es nulo, salir del bucle
        break;
    end


    [bbox, score] = detect(detector, frame);
    bbox = bbox(score > 30, :);
    score = score(score > 30);


    if ~isempty(bbox)
        [selectedBbox, selectedScore] = selectStrongestBbox(bbox, score, ...
            "OverlapThreshold", 0.1);


        numBoxes = size(selectedBbox, 1);
        str = numBoxes + " Mejillon(es) detected";
        img = insertObjectAnnotation(frame, "rectangle", ...
            selectedBbox, "Mejillon: " + selectedScore, "FontSize", 20);
        img = insertText(img, [50 50], str, "FontSize", 50, "TextColor",);
```

```
        % Muestra el cuadro actual con las anotaciones
        imshow(img);
        drawnow;
    end
end
```

4.10 Detector de múltiples objetos aplicado a un video

Para la detección de múltiples objetos se seguirá la misma metodología que para la detección de un objeto la diferencia está en que se entrenara un detector individualmente para cada objeto que se va a detectar, por ejemplo en el video que se usó anteriormente, además de mejillones había cuatro piedras y 2 bisos (tipo de alga hilachenta), por lo tanto se entrenara 2 detectores más que corresponderán a bisos y piedras respectivamente, y para hacerlos funcionar en simultaneo se aplicaran a la imagen antes de ser mostrada, y luego se continuara la detección en los siguientes frames como se hizo en la detección de un solo objeto.

Como se puede ver en la figura 4.6 y 4.7 el detector funciona de forma similar a como lo hacía el detector de mejillones solo que en este caso etiqueta las piedras y Bisos presentes en el video.

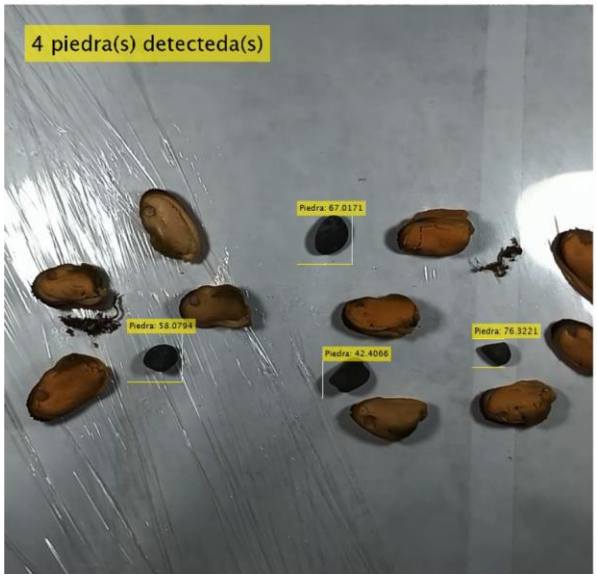


Fig. 4.7: Detección de Piedras

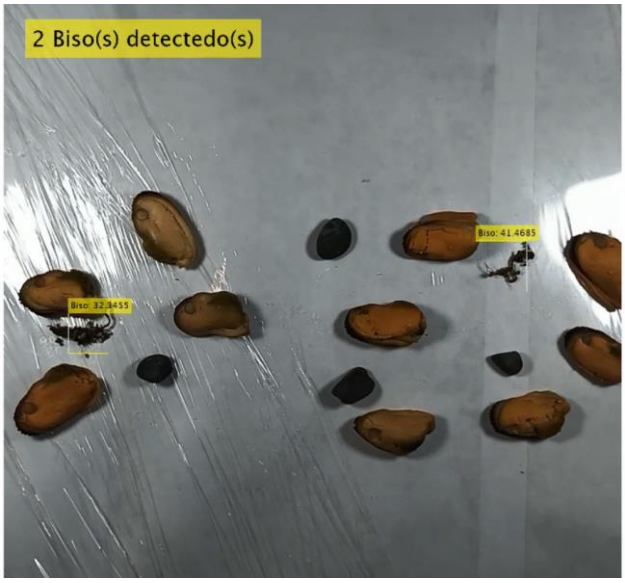


Fig. 4.8: Detección de bisos

4.11 Problemas de entrenamiento

A la hora de entrenar el detector, una de las cosas más importantes es la base de datos, como se mencionó anteriormente. Para entrenar el detector de un objeto mediante imágenes, se necesita una gran cantidad de imágenes del objeto en cuestión. Por lo tanto, es necesario contar con muchas muestras. Sin embargo, surge un problema cuando el objeto que se desea detectar puede variar en gran cantidad de características entre una muestra y otra. Un claro ejemplo de esto son las piedras, las cuales existen en diversos tamaños y colores.

Otro inconveniente se presenta cuando un objeto puede dividirse en partes más pequeñas y aun así seguir siendo el objeto que se quiere detectar. Un ejemplo de esto es el alga, que incluso al cortarse en trozos sigue siendo considerada un alga. En este caso, las algas pueden cambiar de forma, arrugarse, comprimirse o extenderse, y seguirán siendo algas. Este hecho representa un gran desafío para el investigador, ya que requiere tomar medidas especiales para abordar estos distintos objetos polimorfos.

En el caso de este proyecto, se obtuvieron muestras de playa negra ubicada en Penco región del Bío Bío, incluyendo piedras de playa, algas, valvas de diversos moluscos y otros contaminantes como vidrios o plásticos arrojados por la marea. Entre estos elementos, las algas, vidrios, piedras y plásticos presentaron complicaciones en su detección. En el caso de las piedras, el problema se inició en el etiquetado. Debido a que ciertas piedras eran muy pequeñas, no eran consideradas por el algoritmo de etiquetado automático de la aplicación Video Labeler, por lo que tuvieron que etiquetarse manualmente. Lo mismo ocurrió en algunos casos con los plásticos y vidrios.

Otro problema se presentó con respecto al color. Al entrenar un detector de piedras con colores cercanos al blanco o colores claros, el detector no se entrenaba correctamente debido al fondo blanco de la cinta. Por lo tanto, fue necesario cambiar a un fondo negro, pero la detección de piedras claras aún presentaba problemas.

Además, se enfrentaron desafíos relacionados con el tamaño y la forma de las muestras de una misma especie, como en el caso de las algas. Dado que las algas son tan finas, se rompen fácilmente, y las muestras obtenidas tenían tamaños diferentes. Aunque no hubo problemas en el etiquetado, ya que el tamaño en general era más grande que los mejillones que se detectaban perfectamente surgieron problemas con la forma durante la detección. Algunas algas eran grandes y angostas, mientras que otras eran anchas y pequeñas. El detector reconocía las algas, pero a veces solo parte de ellas, otras

veces la etiqueta era más grande que el alga detectada y, en algunas ocasiones, no se detectaban porque estaban plegadas sobre sí mismas y se veían más oscuras que cuando estaban estiradas parcial o completamente. Este desafío hace que el entrenamiento del detector sea más difícil y requiera medidas como utilizar varios detectores para cada uno de los posibles colores o tamaños de un objeto, lo cual, al final, ralentiza el código para lograr una detección más precisa.

4.12 Detección de objetos en tiempo real

Para llevar a cabo la detección de objetos en tiempo real, es necesario conectar una cámara web a Matlab. Dado que la cámara disponible es la de un smartphone, se requiere el uso de un segundo programa que permita utilizar la cámara como webcam. Para este propósito, hay varias aplicaciones disponibles en la tienda de Android, pero la elegida fue la aplicación "iVCam". Esta aplicación permite seleccionar diversas resoluciones para la captura, así como controlar el brillo, ISO, contraste, enfoque y zoom de la imagen. Para activar la cámara del smartphone como webcam se entra en modo desarrollador, se activa la depuración USB, luego, se abre la aplicación, se desconecta el wifi del teléfono para evitar la conexión por este medio y se conecta por cable USB al ordenador, donde se vincula como webcam. Es importante destacar que la conexión se realiza por USB en lugar de wifi para lograr una transmisión óptima e independiente de la calidad de la red a la que se conecten ambos equipos.

La configuración utilizada en iVCam para realizar las capturas de videos se muestra en la imagen de la Fig. 4.9. Además, se utiliza una resolución de 640 por 480 píxeles, ya que es la mayor resolución que soporta Matlab durante la aplicación de la detección en tiempo real. Debido a las limitaciones de hardware del ordenador, esta resolución se elige, ya que resoluciones mayores provocan pérdida de fotogramas y retraso en la captura en tiempo real. Se puede afirmar que, a mayor resolución, hay una mayor carga en el hardware y un menor rendimiento del detector.

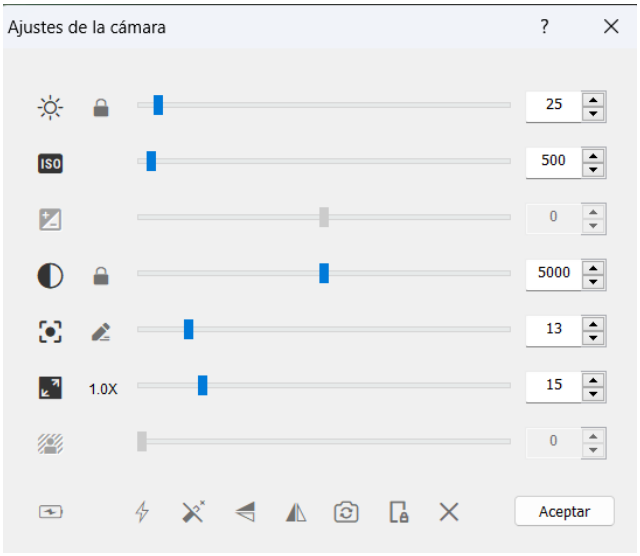


Fig. 4.9: Ajuste de cámara en iVCam

La implementación de la detección de objetos en tiempo real en Matlab se realiza mediante un bucle "while", en el cual se ejecuta la función "snapshot", que adquiere una única imagen desde la webcam y la asigna a una variable. Esta imagen es el fotograma actual, y llamar a la función en un bucle proporciona un nuevo fotograma en cada iteración. Además, la imagen siempre será una imagen RGB con la resolución de la cámara (640 por 480 píxeles en este caso) [37]. Al fotograma capturado se le aplican los detectores, así como las etiquetas de advertencia y los contadores de cada especie detectada en la imagen. Esta imagen se actualiza con cada nuevo fotograma hasta que finaliza la ejecución del programa o hasta que no se encuentran más fotogramas. Para esto, se utiliza la función "ishandle" junto con el bucle "while", la cual devuelve un arreglo de valores 1 si hay un objeto gráfico y un arreglo de valores 0 si no los hay [38]. Así, esta función valida si hay objetos gráficos dentro del gráfico.

4.13 Frames per second.

La utilidad de conocer los frames per second (fps) o cuadros por segundo radica en observar cómo afecta la configuración utilizada por la cámara web llamada por Matlab para realizar la detección. Esto permite identificar la variación en la fluidez de la captura y evaluar las repercusiones de aplicar detectores en tiempo real. Se pudo apreciar que, al agregar detectores, se pierden fps, lo que conlleva a una disminución en la cantidad de datos presentes en la detección. Las razones de esta reducción se deben a que cada detector utiliza una cierta cantidad de recursos del hardware, y al aumentar la carga,

el equipo no puede procesar tantos datos a la misma velocidad que con un menor número de detectores.

Las soluciones para el problema de la disminución de fps pueden incluir mejoras en el hardware, pero esto puede generar costos monetarios mayores a los esperados. Otra solución posible, aplicada en el código del detector, es la disminución de la resolución mediante la función "imresize", que permite ajustar la resolución modificando la escala, por ejemplo: "frame = imresize(frame,0.5)", entre otras formas de cambiar la resolución. Otras opciones para disminuir fps incluyen la optimización del código y la paralelización, entre otras alternativas.

Para mostrar los fps, se utiliza el siguiente código:

```
% Carga los detectores...
% Crear un objeto de cámara
cam = webcam("e2eSoft iVCam");

while ishandle(h)

    %inicia contador tic
    tic

    %captura imagen de la webcam
    frame = snapshot(cam);
    frame = imresize(frame,0.5);

    %Resto del código...

    %Inserta los FPS
    fps = .9*fps + .1*(1/toc);
    %fps = 1/toc;
    avgfps = [avgfps, fps];

    if fps<=30
        frameLabeled  = insertText(frameLabeled , [1, 25],  sprintf('FPS %2.2f',
fps), 'FontSize', 12, 'BoxColor', 'r');
    elseif fps>30 && fps<60
        frameLabeled  = insertText(frameLabeled , [1, 25],  sprintf('FPS %2.2f',
fps), 'FontSize', 12, 'BoxColor', 'y');
    else
        frameLabeled  = insertText(frameLabeled , [1, 25],  sprintf('FPS %2.2f',
fps), 'FontSize', 12, 'BoxColor', 'g');
    end
    %muestra la imagen en tiempo real
    imshow(frameLabeled);
end
```

El contador de FPS (cuadros por segundo) se calcula utilizando las funciones tic y toc en MATLAB. La función tic se llama al inicio del bucle para marcar el tiempo de inicio, mientras que la función toc se llama después de capturar y procesar cada cuadro para medir el tiempo transcurrido desde la última llamada a tic. La función toc devuelve el tiempo transcurrido en segundos desde la última llamada a tic[39][40].

La ecuación 1 actualiza el valor del contador de FPS (fps) utilizando un filtro de media móvil ponderada. Esto ayuda a suavizar las variaciones y proporciona una estimación más estable del rendimiento actual. Opcionalmente, se puede utilizar la ecuación 2, que entrega el valor capturado en cada iteración, pero este método puede resultar menos estable en comparación con la fórmula del filtro.

$$FPS = 0.9 \times FPS \; + \; 0.1 \times (1/toc) \qquad (1)$$

$$\mathrm{FPS} = 1/\mathrm{toc} \qquad (2)$$

Se inserta el contador de FPS en la imagen, y dependiendo del valor del contador de FPS, se añade un texto en la imagen para mostrar los FPS. Si los FPS son menores o iguales a 30, se utiliza un cuadro rojo ('BoxColor', 'r'). Si los FPS están entre 30 y 60, se utiliza un cuadro amarillo ('BoxColor', 'y'). Si los FPS son mayores a 60, se utiliza un cuadro verde ('BoxColor', 'g').

Este proceso se repite en cada iteración del bucle principal, proporcionando una estimación en tiempo real de los FPS en función del tiempo transcurrido entre cuadros consecutivos. La suavización con el filtro de media móvil ponderada ayuda a que la visualización de los FPS sea más estable.

4.14 Insertar fecha y hora

En MATLAB la fecha y la hora se obtienen y se representan utilizando la función de fecha y hora “datetime” como se muestra en la sección de código siguiente:

```
% Inserta la fecha y hora
timestamp = datetime('now', 'Format', 'd-M-y HH:mm:ss.SSS');


frameLabeled = insertText(frameLabeled, [10 10], char(timestamp), ...
    'FontSize', 12, 'TextColor', 'white', 'TextBoxColor', 'black');
```

datetime('now', 'Format', 'd-M-y HH:mm:ss.SSS'): Esta línea crea un objeto de fecha y hora actual.

La línea datetime('now', 'Format', 'd-M-y HH:mm:ss.SSS') crea un objeto de fecha y hora actual. Aquí, 'now' especifica que se utilice la fecha y hora actuales, y 'Format', 'd-M-y HH:mm:ss.SSS' define el formato en el que se desea representar la fecha y la hora. En este caso, se utiliza el formato día-mes-año (d-M-y) seguido por la hora, los minutos, los segundos y los milisegundos [41].

Posteriormente, char(timestamp) convierte el objeto de fecha y hora en una cadena de caracteres para que pueda ser insertado en la imagen como texto. Este proceso se realiza utilizando la función insertText, que agrega el texto en la posición especificada ([10 10] en este caso) sobre la imagen frameLabeled. Se configuran parámetros adicionales, como el tamaño de fuente, el color del texto y el color del cuadro del texto.

Este texto, que contiene la fecha y hora actual, se actualiza en cada iteración del bucle while, proporcionando así una marca de tiempo dinámica en la imagen, como se aprecia en la figura.

4.15 Evaluación de los Detectores de Objetos

Para evaluar los detectores de objetos se implementó un código en Matlab capaz de usar las imágenes de entrenamiento del detector para comparar las etiquetas del gorundTruth data con las etiquetas que genera la detección de la imagen. Para hacer la evaluación se adquieren tres métricas, las cuales son "Precisión", "Recall o Sensibilidad", las cuales son entregadas por la función de Matlab "bboxPrecisionRecall" y de estas se calcula "F1-score"[42][43].

La precisión se calcula mediante la función "bboxPrecisionRecall", que compara las cajas delimitadoras detectadas con las cajas delimitadoras manuales (ground truth), mide la proporción de las detecciones positivas realizadas por el modelo que son realmente correctas. En el contexto del detector de objetos, la precisión se calcula como la proporción de verdaderos positivos (detecciones correctas) respecto a la suma de verdaderos positivos y falsos positivos (detecciones incorrectas).

$$Precisión = \frac{TP}{TP + FP} \quad (3)$$

Una alta precisión indica que las detecciones positivas realizadas por el modelo son confiables y tienen menos probabilidades de ser falsas alarmas.

Recall, también conocido como sensibilidad, se calcula de manera similar a través de la función "bboxPrecisionRecall", mide la capacidad del modelo para encontrar todos los casos positivos. En el

contexto del detector de objetos, recall se calcula como la proporción de verdaderos positivos respecto a la suma de verdaderos positivos y falsos negativos (casos positivos no detectados).

$$\text{Recall} = \frac{TP}{TP + FN} \quad (4)$$

Un alto recall indica que el modelo es efectivo para identificar la mayoría de los casos positivos presentes en los datos.

El F1-score se calcula utilizando las precisiones y recalls obtenidas anteriormente. Esta métrica proporciona un equilibrio entre precisión y recall, se utiliza cuando hay un desequilibrio entre las clases (por ejemplo, más negativos que positivos o viceversa). Un F1-score alto indica un buen equilibrio entre precisión y recall.

$$F1 - Score = 2 \times \frac{Precisión \times Recall}{Precisión + Recall} \quad (5)$$

El F1-score es especialmente útil cuando se busca un rendimiento balanceado en la identificación de positivos y la minimización de falsos positivos y falsos negativos.

```
close all; clear; clc;
% Cargar detector y datos de prueba
load('detector_mejillones.mat');
load('imList_Mejillones.mat');
load('boxLabels_Mejillones.mat');
load('imgBoxLabelsMejillones.mat');


% Inicializar variables para almacenar las precisiones
precisiones = zeros(1, numel(imList.Files));
recalls = zeros(1, numel(imList.Files));
f1_scores = zeros(1, numel(imList.Files));
% Iterar sobre todas las imágenes
for i = 1:numel(imList.Files)
    % Obtener la imagen actual
    im = imWithBoxLabelsMejillones.UnderlyingDatastores{1}.Files{i};
    img = imread(im);

    % Obtener las etiquetas manuales para la imagen actual
    label = imWithBoxLabelsMejillones.UnderlyingDatastores{2}.LabelData{i};

    % Detectar objetos en la imagen actual
    [bb, sc] = detect(detector_Mejillones, img);
```

```
    % Seleccionar cuadros delimitadores para cada tipo de objeto
    [sbb, ssc] = selectStrongestBbox(bb, sc, "OverlapThreshold", 0.1);

    % Calcular precision, recall y f1-score
    [precision, recall] = bboxPrecisionRecall(sbb, label);

    % Manejar división por cero
    if (precision + recall) > 0
        f1_score = 2 * (precision * recall) / (precision + recall);
    else
        f1_score = 0;
    end
    precisiones(i) = precision;
    recalls(i) = recall;
    f1_scores(i) = f1_score;
end
% Calcular promedios
PrecisionPromedio = mean(precisiones)
RecallProemdio = mean(recalls)
F1_scorePromedio = mean(f1_scores)
```

Los resultados obtenidos por el código anterior entregan valores numéricos entre 0 y 1, donde cero indica un mal desempeño en las métricas y uno indica un desempeño perfecto. Las métricas pueden cambiar debido a distintas variables, las cuales comienzan incluso antes de entrenar al detector. Estas son la cantidad de muestras, la calidad de estas y la calidad en la que se capturan las imágenes. Además, posteriormente, está la calidad del etiquetado, el cual puede no ser óptimo al automatizar el proceso o puede contener errores en el proceso de etiquetado manual.

Las variables por considerar que pueden afectar las métricas son las propias variables de la función "trainACFObjectDetector", las cuales son "ObjectTrainingSize", "NumStages", "NegativeSampleFactor" y "MaxWeakLearners". Estas variables, con cambios leves, aumentan o disminuyen el desempeño de los distintos detectores, como se puede observar en las comparaciones de las tablas de resultados.

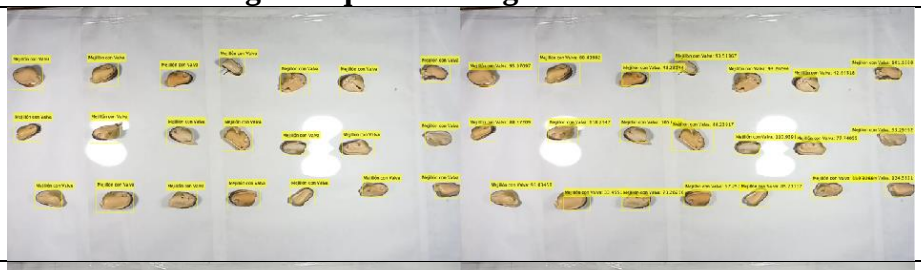
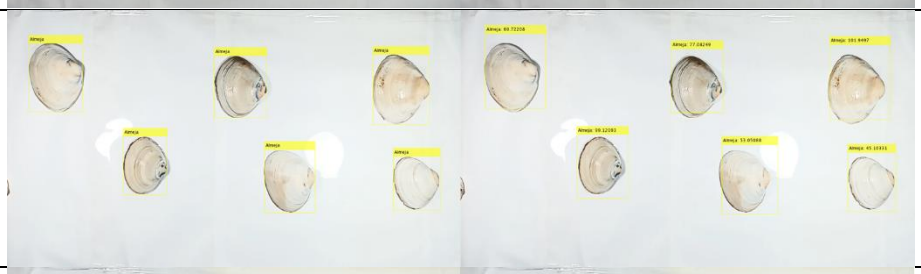
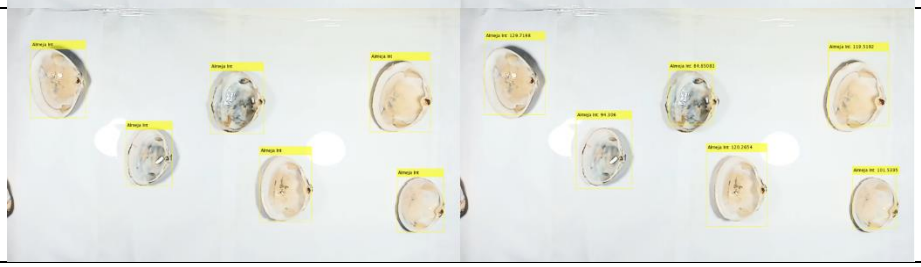
Las últimas variables que afectan al desempeño del detector son el "OverlapThreshold" en la función "selectStrongestBbox" y, finalmente, la última variable es el filtro aplicado al puntaje de cada detección, el cual establece un límite al puntaje mínimo aceptado en la detección.

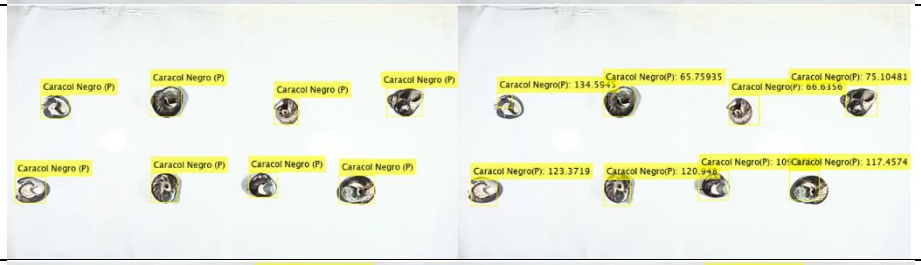
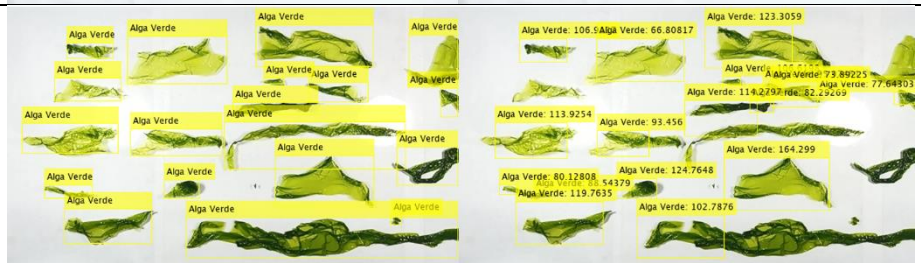
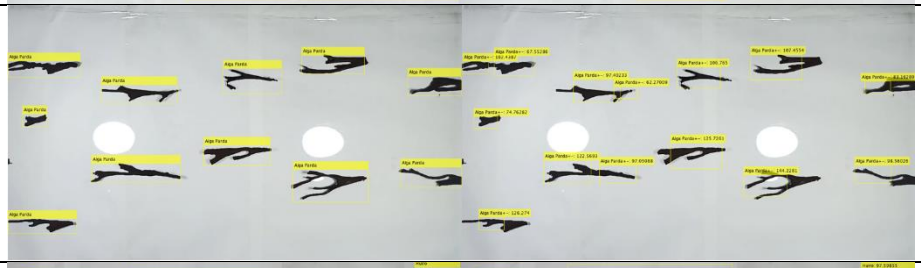
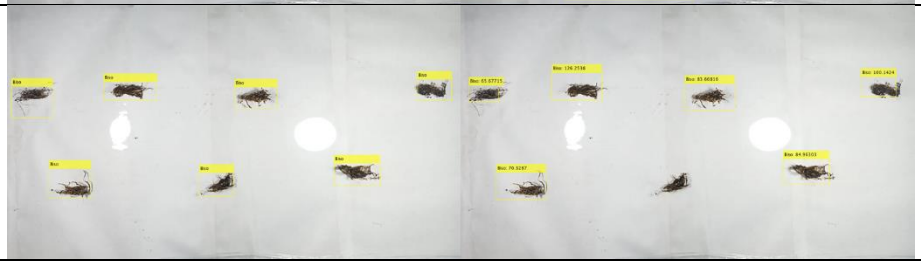
4.16 Especies disponibles a detectar

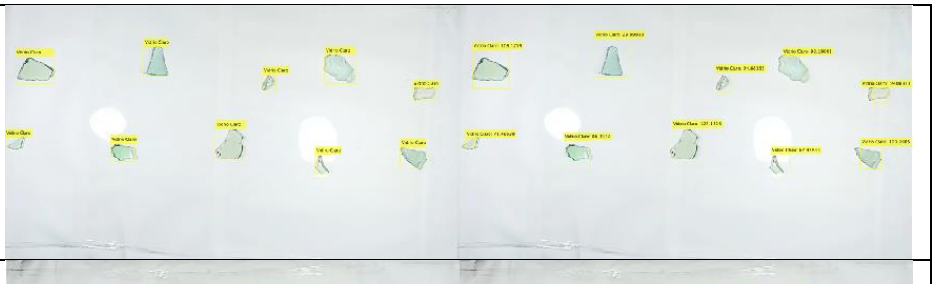
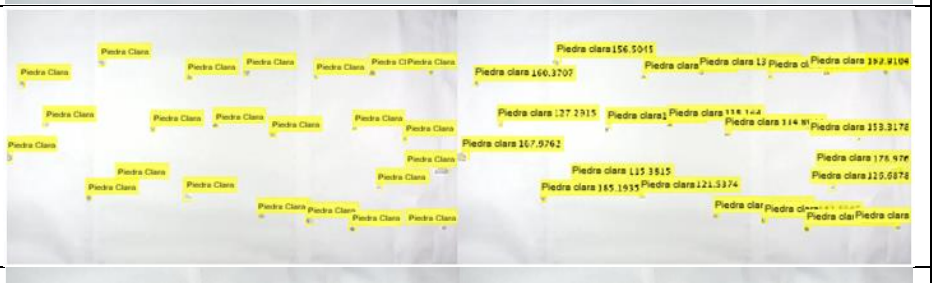
Para llevar a cabo los experimentos de detección, se obtuvieron 12 especies distintas, de las cuales tres fueron capturadas desde dos perspectivas diferentes, generando así un total de 15 objetos a detectar. Estos objetos fueron grabados, etiquetados, entrenados y detectados siguiendo los pasos previamente mencionados. Posteriormente, se llevó a cabo una optimización individual de cada detector, utilizando métricas como precisión, recall y F1-score. Además, se implementó un threshold y un filtro que se describirá en la siguiente sección.

En la Tabla 1 se presentan las especies junto con la imagen de su etiquetado y la detección optimizada de manera individual.

Tabla 4.1: Especies Detectadas

N°1	Especie	Imagen etiquetada: original – detectada
1	Mejillón	
2	Mejillón (con Valva)	
3	Almeja (Valva)	
4	Almeja (Valva Interior)	

5	**Caracol Negro**	
6	**Caracol Negro (Posterior)**	
7	**Alga Verde**	
8	**Alga Parda**	
9	**Huiro (Alga)**	
10	**Bisos**	

11	**Vidrio Claro**	
12	**Vidrio Verde**	
13	**Piedra Obscura**	
14	**Piedra Clara**	
15	**Piedra Roja**	

Se observa que hay variabilidad en la calidad de detección entre las especies, y algunos objetos presentan múltiples detecciones mientras que otros no son detectados en absoluto. Estas observaciones serán explicadas más detalladamente en la siguiente sección. No obstante, se evidencia la posibilidad de detectar más objetos que únicamente el mejillón. Por ejemplo, los objetos más pequeños como las piedras presentan dificultades en la detección, y aquellos con características similares son detectados con mayor precisión.

4.17 Resultados

Los resultados obtenidos fueron diversos en las diferentes métricas de cada detector. Para obtener estas mediciones, primero se llevó a cabo el entrenamiento utilizando la función "trainACFObjectDetector" con sus variables en valores por defecto. Además, se aplicó el cálculo de métricas mediante el código sin filtrar el puntaje mínimo ni utilizar "selectStrongestBbox". Esto resultó en una detección sin ninguna optimización, más allá de la realizada previo al entrenamiento, cuyos valores se presentan en la tabla 1.

Tabla 4.2: Tabla 4.2: Porcentajes de Precisión, Recall, F1-Score de detector entrenado a parámetros estándar

Objeto	Object Training Size	Precisión [%]	Recall [%]	F1-Score [%]
Mejillón	[9 14]	24.86	88.01	38.56
Mejillón (Con Valva)	[8 16]	47.00	79.21	58.68
Almeja	[10 8]	16.49	63.43	26.24
Almeja (Interior)	[27 22]	93.62	94.69	93.75
Caracol Negro	[9 10]	21.95	66.17	32.87
Caracol Negro (Posterior)	[10 10]	28.57	66.39	39.74
Alga Verde	[8 15]	20.50	36.05	28.37
Alga Parda	[8 24]	23.30	73.43	35.20
Huiro (Alga)	[20 24]	71.84	86.46	77.98
Biso (Alga)	[10 18]	49.6	76.67	60.62
Vidrio Claro	[12 11]	22.96	74.05	34.93
Vidrio Verde	[22 26]	54.50	94.11	68.85
Piedra Obscura	[8 9]	16.22	71.32	26.38
Piedra Clara	[8 9]	17.65	77.89	28.76
Piedra Roja	[9 8]	11.51	45.95	18.38

Posteriormente, después de realizar la detección con parámetros estándar en cada uno de los objetos de muestra, se llevó a cabo la optimización individual de cada detector. Esto se hizo con la intención de aumentar las métricas mediante ajustes en el entrenador de objetos, variando el valor de "ObjectTrainingSize". Después de alcanzar valores máximos posibles encontrados manualmente, se aplicó un filtro de puntajes mínimos y se realizaron modificaciones en "selectStrongestBbox" para lograr una mayor optimización del detector.

Al cambiar el valor de las distintas variables, los valores de las métricas fueron variando hasta alcanzar un punto máximo en cada una de ellas. Durante el entrenamiento de los detectores, se observó que los máximos de la precisión y la sensibilidad del detector se logran a expensas del otro valor. Por lo tanto,

se buscó un equilibrio entre precisión y sensibilidad, logrando valores máximos en la métrica F1-Score.Tener una mayor precisión implica que es más seguro que los objetos detectados sean correctos, mientras que tener una mayor sensibilidad indica cuántos objetos podrá detectar el detector. Lograr una alta sensibilidad y precisión es esencial en los detectores de objetos, ya que no se quieren detectar falsos positivos por falta de precisión ni se quieren omitir objetos por falta de sensibilidad en el detector.

Los resultados obtenidos al optimizar los detectores se pueden apreciar en la tabla 2, donde queda claro que tanto la precisión como la sensibilidad de los detectores variaron al cambiar los valores de "ObjectTrainingSize", además de la adición del "OverlapThreshold" y el filtro por puntaje mínimo, donde se considera como optimo al mayor porcentaje de F1-score obtenido con diferentes configuraciones, lo que implica un mayor equilibrio en el detector.

Tabla 4.3: Detectores optimizados en F1-Score mediante ObjectTrainingSize, OverlapThreshold y Puntaje mínimo.

Objeto	Object Training Size	Precisión [%]	Recall [%]	F1-Score [%]	Overlap Tresh Hold	Puntaje mínimo
Mejillón	[8 15]	92.97	89.53	91.16	0.001	30
Mejillón (con Valva)	[9 27]	78.11	63.64	69.60	0.01	0
Almeja (Valva)	[27 22]	98.42	90.16	93.82	0.01	10
Almeja (Valva Interior)	[27 22]	99.84	94.55	96.92	0.001	0
Caracol Negro	[8 8]	60.43	54.08	56.70	0.001	30
Caracol Negro (Posterior)	[8 8]	76.56	72.26	73.94	27	0.01
Alga Verde	[13 26]	63.62	49.16	54.91	0.1	60
Alga Parda	[8 25]	66.62	63.08	64.63	0.0001	30
Huiro (Alga)	[22 26]	96.58	88.05	91.89	0.1	20
Bisos	[10 18]	69.03	68.78	68.80	0.01	30
Vidrio Claro	[14 13]	79.12	77.75	78.16	0.001	20
Vidrio Verde	[21 27]	89.16	85.55	87.20	0.01	35
Piedra Obscura	[8 8]	37.42	38.12	37.75	0.0001	70
Piedra Clara	[8 8]	59.94	56.95	58.36	100	0.00001
Piedra Roja	[8 9]	52.12	52.61	52.35	0.001	50

4.18 Aplicación de detectores con las demás especies

Con los resultados de la sección anterior, se comprueba la precisión y sensibilidad de los detectores aplicados en el contexto en el que se encuentran entre otros objetos de su misma especie. Ahora, mediante la detección en tiempo real, se observa cómo se comportan los detectores en un contexto de coexistencia con las demás especies.

Al aplicar los detectores optimizados individualmente a las distintas especies, se visualiza el funcionamiento del detector en tiempo real con la cinta en movimiento. Como se aprecia en las siguientes imágenes, el comportamiento del detector con otras especies es totalmente diferente a su comportamiento en solitario.

Fig. 4.10: Detector de mejillones a otras especies

Fig. 4.11: Detector de mejillones a otras especies 2

Fig. 4.12: Detección Mejillones (Valva) a especies

Fig. 4.13: Detección Mejillones (Valva) a especies 2

Fig. 4.14: Detección de Almejas a otras especies

Fig. 4.15: Detección de Almejas a otras especies 2

Fig. 4.16: Detección de Almejas Interior a especies

Fig. 4.17: Detección de Almejas Interior especies 2

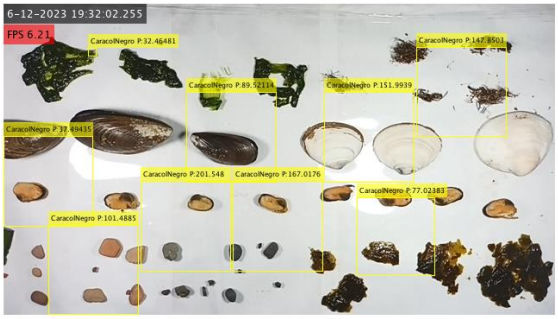


Fig. 4.18: Detección de Caracol Negro a especies

Fig. 4.19: Detección de Caracol Negro a especies 2

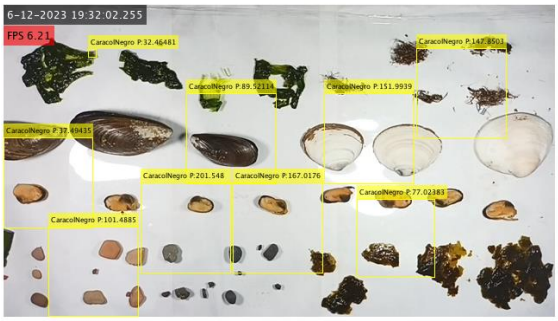


Fig. 4.20: Detección Caracol Negro Post. a especies

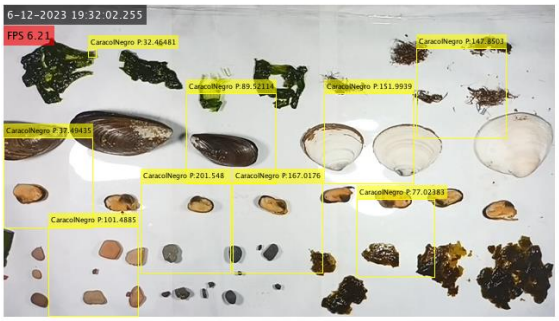


Fig. 4.21: Detección Caracol Negro Post. a especies 2

Fig. 4.22: Detección de Algas Verdes a otras especies

Fig. 4.23: Detección de Algas Verdes a especies 2

Fig. 4.24: Detección de Algas Pardas a especies

Fig. 4.25: Detección de Algas Pardas a especies 2

Fig. 4.26: Detección de Huiros a otras especies

Fig. 4.27: Detección de Huiros a otras especies 2

Fig. 4.28: Detección de Bisos a otras especies

Fig. 4.29: Detección de Bisos a otras especies 2

Fig. 4.30: Detección de Vidrios Claros a especies

Fig. 4.31: Detección de Vidrios Claros a especies 2

Fig. 4.32: Detección de Vidrios Verde a especies

Fig. 4.33: Detección de Vidrios Verde a especies 2

Fig. 4.34: Detección Piedras Obscuras a especies

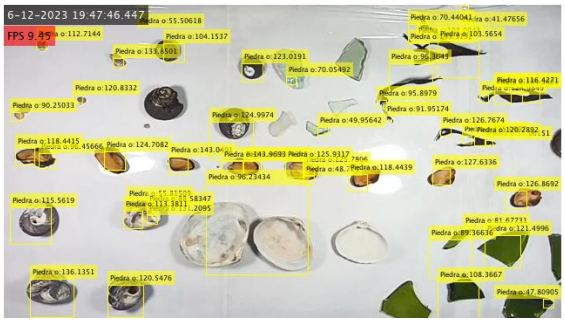


Fig. 4.35: Detección Piedras Obscuras a especies 2

Fig. 4.36: Detección Piedras Claras a otras especies

Fig. 4.37: Detección Piedras Claras a otras especies 2

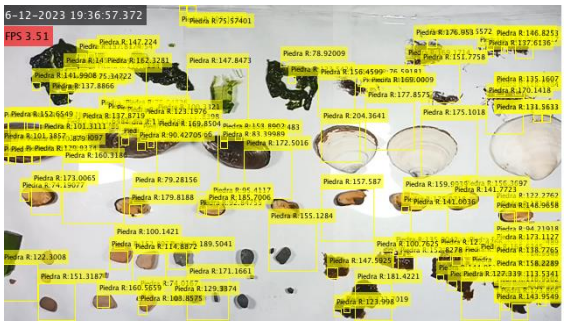


Fig. 4.38: Detección Piedras Rojas a otras especies

Fig. 4.39: Detección Piedras Rojas a otras especies 2

Como se aprecia en las imágenes que muestran la aplicación de los distintos detectores a todas las especies detectadas, se evidencia que los detectores presentan deficiencias al aumentar el número de especies en la imagen. Se observa un gran número de falsos positivos, y la precisión y sensibilidad alcanzadas con la detección de un solo objeto disminuyen considerablemente.

Ciertamente, es posible reducir la cantidad de falsos positivos en algunos casos mediante el aumento del puntaje mínimo para detectar cada objeto. Sin embargo, este ajuste solo puede realizarse hasta el menor puntaje establecido para el objeto específico para el cual el detector está optimizado. Por lo tanto, si los objetos detectados superan este puntaje, seguirán siendo considerados falsos positivos, lo que compromete la eficacia del detector de objetos y lo hace deficiente, por lo tanto, no se recomienda su implementación.

Estos problemas se presentan de manera recurrente en la mayoría de los detectores analizados. Por lo tanto, podría considerarse cambiar la forma de entrenar los detectores o sustituir el modelo ACF por otro.

5 Conclusiones

5.1 Resumen

Para entrenar un detector de objetos, es necesario contar con una gran y variada cantidad de imágenes, preferiblemente en secuencia o video. Se requiere etiquetar los objetos manual o automáticamente para realizar el entrenamiento del detector. El detector ACF funciona notablemente bien con objetos de tamaño, color y forma similares entre muestra, sin embargo, muestra deficiencias al detectar objetos que varíen en estas características otro problema se da al detectar objetos demasiado pequeños en la imagen. Por lo tanto, es recomendable realizar tomas cercanas del objeto de menor tamaño para evitar este problema. Además, se ha concluido que es posible optimizar un detector mediante diversas variables para mejorar su rendimiento individual. También es seguro afirmar que la calidad de un detector depende de su precisión y sensibilidad y finalmente como se pudo comprobar el detector muestra su peor desempeño cuando se detectan varias especies.

Se ha verificado que es posible aplicar más de un detector a una imagen simultáneamente. Además, se ha demostrado que es posible la detección en tiempo real de diversos objetos, pero que el rendimiento de la detección cae enormemente, por esto mismo es que la detección de objetos no deseados en una línea de producción no es viable con los resultados obtenidos. Se puede apreciar por los resultados que las detecciones realizadas no son infalibles y presentan un margen de error tanto en precisión como en sensibilidad del detector y no todos los objetos se detectan correctamente, además puede haber falsas detecciones debido a esto, sin embargo, estas imprecisiones se pueden reducir mediante optimización y filtros.

5.2 Resultados obtenidos

Los resultados obtenidos al aplicar los detectores de objetos entrenados a la totalidad de las especies permiten concluir que la detección de objetos no deseados en líneas de producción de mejillones no es posible mediante el método aplicado. No obstante, algunos detectores muestran la capacidad de funcionar correctamente con una sola especie. Además, algunos de ellos exhiben una mejor precisión en la detección de objetos, como es el caso del detector de mejillones, que mediante filtros es capaz de reducir la cantidad de falsos positivos y destacar entre otros.

En términos generales, los detectores funcionan de manera deficiente si no se les realiza una optimización. Todos ellos poseen la capacidad de mejora mediante las configuraciones adecuadas, especialmente en la detección de un objeto en particular. Sin embargo, la detección de una mayor cantidad de objetos no muestra un rendimiento aceptable para implementar los detectores en la industria debido a su reducida capacidad de detección en objetos específicos presentes en una amplia variedad de especies. A pesar de ello, se logró mejorar numéricamente la precisión y sensibilidad de los detectores.

Es acertado afirmar que se requiere una nueva configuración para llevar a cabo la detección. Esto podría lograrse mediante una mayor cantidad de muestras, un etiquetado más específico, el aumento de las regiones de interés antes del entrenamiento o cambiando el método de detección de objetos.

Es relevante destacar que se logró realizar detecciones en imágenes, videos y, posteriormente, en tiempo real. Esta última instancia fue donde se llevaron a cabo las últimas pruebas. A pesar del desempeño desfavorable de los detectores en las pruebas con múltiples especies, se logró la detección en tiempo real, uno de los objetivos del proyecto.

En general, se lograron alcanzar los objetivos más básicos del proyecto, como la grabación y etiquetado de distintas especies para su entrenamiento. También se logró el desarrollo de competencias en la detección de objetos, su aplicación, la adquisición de parámetros mediante ejemplos de prueba, la implementación de detectores de distintas especies para su posterior optimización y puesta en marcha, y, por último, la transición de la detección de objetos en imágenes fijas a videos y, finalmente, en tiempo real, a pesar de los resultados ya comentados.

5.3 Trabajo Futuro

Para futuras investigaciones, sería beneficioso explorar otros detectores de objetos proporcionados por Matlab, como CNN, K-NN o YOLO, entre otros. Asimismo, se podría considerar experimentar con lenguajes de programación adicionales, como Python, profundizando en su implementación.

En el caso de Matlab, sería necesario realizar pruebas utilizando el etiquetado en la aplicación Video Labeler, que permite etiquetar figuras de manera más específica que la herramienta utilizada anteriormente. Además, se sugiere aumentar la captura de muestras para analizar los efectos generados en el entrenamiento. También se podría mejorar el rendimiento de los detectores mediante el

entrenamiento en equipos con mayor capacidad de hardware, ya que las limitaciones de procesador y tarjeta RAM se hacen evidentes durante la detección de objetos.

Una tarea adicional sería investigar y desarrollar estrategias que permitan a un detector reducir de manera significativa la cantidad de falsos positivos en presencia de varias especies, abordando este desafío específico para mejorar la eficacia del sistema de detección.

6 Glosario

Mayúsculas

M.L. : Machine Learning

D.L. : Deep Learning.

IA : Inteligencia Artificial.

NMS : Supresión No Máxima.

IR : Infrarrojo.

NIR : Infrarrojo cercano.

MIR : Infrarrojo medio.

TP : True Positive.

FP : False Positive.

FN : False Negative.

CNN : Convolutional Neural Network.

RNN : Recurrent Neural Networks.

ANN : Artificial Neural Networks.

SVM : Support Vector Machine.

KNN : K Nearest Neighbors.

ACF : Aggregate Channel Fures.

TWSVM : Soporte Vectorial Gemelas Mejoradas.

DSP : Diarrhetic Shellfish Poisoning.

ELM : Máquina de Aprendizaje Extremo.

Minúsculas

i.h.e : imágenes hiper espectrales.

fps : frame per second.

7 Referencias

[1] colaboradores de Wikipedia, “Procesamiento digital de imágenes”, https://es.wikipedia.org/w/index.php?title=Procesamiento_digital_de_im%C3%A1genes&oldid=149291818.

[2] J. Hurwitz y D. Kirsch, *Machine Learning IBM Limited Edition*. 2018. [En línea]. Disponible en: http://www.wiley.com/go/permissions.

[3] “Historia de la Inteligencia Artificial, el Machine Learning y el Deep Learning”. Accedido: 24 de diciembre de 2023. [En línea]. Disponible en: https://www.algotive.ai/es-mx/blog/historia-de-la-inteligencia-artificial-el-machine-learning-y-el-deep-learning#ancla_1

[4] “IA en la industria alimentaria: Tendencias, estrategias y casos de éxito”. Accedido: 24 de diciembre de 2023. [En línea]. Disponible en: https://thefoodtech.com/tendencias-de-consumo/la-inteligencia-artificial-invade-a-la-industria-alimentaria/

[5] “Aplicaciones de visión artificial en el sector alimentación”. Accedido: 24 de diciembre de 2023. [En línea]. Disponible en: https://infaimon.com/blog/aplicaciones-vision-artificial-sector-alimentacion/

[6] A. Felipe, L. Quintana, D. Andrés, M. Herrera, L. Eduardo, y M. Salazar, “Sistema de visión artificial para conteo de objetos en movimiento”. Accedido: 24 de diciembre de 2023. [En línea]. Disponible en: https://www.redalyc.org/articulo.oa?id=47826850010

[7] “Cómo los sistemas de visión artificial están mejorando la eficiencia en la producción de alimentos”. Accedido: 24 de diciembre de 2023. [En línea]. Disponible en: https://e2mcouth.com/2023/06/08/vision-artificial-en-la-industria-alimentaria/

[8] P. A. Flach, “Machine Learning The Art and Science of Algorithms that Make Sense of Data”. Accedido: 8 de junio de 2023. [En línea]. Disponible en: http://www.cs.put.poznan.pl/tpawlak/files/ZMIO/W02.pdf

[9] P. S. V. Reddy, M. V. Krishna, R. Aishwarya, R. Yogitha, y K. A. Kumar, “Fish Species Classifier for Allergic People using CNN Algorithm”, en *2023 7th International Conference on Computing Methodologies and Communication (ICCMC)*, IEEE, feb. 2023, pp. 489–494. doi: 10.1109/ICCMC56507.2023.10084124.

[10] Y. Feng, X. Tao, y E.-J. Lee, “Classification of Shellfish Recognition Based on Improved Faster R-CNN Framework of Deep Learning”, *Math Probl Eng*, vol. 2021, pp. 1–10, jun. 2021, doi: 10.1155/2021/1966848.

[11] L. G. T. Crusiol *et al.*, “In-Season Monitoring of Maize Leaf Water Content Using Ground-Based and UAV-Based Hyperspectral Data”, *Sustainability*, vol. 14, nº 15, p. 9039, jul. 2022, doi: 10.3390/su14159039.

[12] S. Srivastava y H. N. Mishra, “Detection of insect damaged rice grains using visible and near infrared hyperspectral imaging technique”, *Chemometrics and Intelligent Laboratory Systems*, vol. 221, p. 104489, feb. 2022, doi: 10.1016/j.chemolab.2021.104489.

[13] Y. Xiang *et al.*, “Deep Learning and Hyperspectral Images Based Tomato Soluble Solids Content and Firmness Estimation”, *Front Plant Sci*, vol. 13, may 2022, doi: 10.3389/fpls.2022.860656.

[14] S. Eshkabilov, J. Stenger, E. N. Knutson, E. Küçüktopcu, H. Simsek, y C. W. Lee, “Hyperspectral Image Data and Waveband Indexing Methods to Estimate Nutrient Concentration on Lettuce (Lactuca sativa L.) Cultivars.”, *Sensors (Basel)*, vol. 22, nº 21, p. 8158, oct. 2022, doi: 10.3390/s22218158.

[15] N. Vásquez, C. Magán, J. Oblitas, T. Chuquizuta, H. Avila-George, y W. Castro, “Comparison between artificial neural network and partial least squares regression models for hardness modeling during the ripening process of Swiss-type cheese using spectral profiles”, *J Food Eng*, vol. 219, pp. 8–15, feb. 2018, doi: 10.1016/j.jfoodeng.2017.09.008.

[16] X. Zhong, F. Y. Shih, y X. Guo, “Automatic Image Pixel Clustering based on Mussels Wandering Optimization”. Accedido: 8 de junio de 2023. [En línea]. Disponible en: https://arxiv.org/ftp/arxiv/papers/1909/1909.03380.pdf

[17] F. Xu, H. Wang, J. Peng, y X. Fu, “Scale-aware feature pyramid architecture for marine object detection”, *Neural Comput Appl*, vol. 33, nº 8, pp. 3637–3653, abr. 2021, doi: 10.1007/s00521-020-05217-7.

[18] P. A. Coelho-Caro, C. E. Saavedra-Rubilar, J. P. Staforelli, M. J. Gallardo-Nelson, V. Guaquin, y E. Tarifeño, “Mussel Classifier System Based on Morphological Characteristics”, *IEEE Access*, vol. 6, pp. 76935–76941, 2018, doi: 10.1109/ACCESS.2018.2884394.

[19] Y. Liu, L. Xu, S. Zeng, F. Qiao, W. Jiang, y Z. Xu, “Rapid detection of mussels contaminated by heavy metals using near-infrared reflectance spectroscopy and a constrained difference extreme learning machine”, *Spectrochim Acta A Mol Biomol Spectrosc*, vol. 269, mar. 2022, doi: 10.1016/j.saa.2021.120776.

[20] Y. Liu, F. Qiao, L. Xu, R. Wang, W. Jiang, y Z. Xu, “Fast Detection of Diarrhetic Shellfish Poisoning Toxins in Mussels Using NIR Spectroscopy and Improved Twin Support Vector Machines”, *Front Mar Sci*, vol. 9, jun. 2022, doi: 10.3389/fmars.2022.907378.

[21] H. Ayvaz *et al.*, “Machine Learning-Assisted Near- and Mid-Infrared spectroscopy for rapid discrimination of wild and farmed Mediterranean mussels (Mytilus galloprovincialis)”, *Microchemical Journal*, vol. 196, p. 109669, ene. 2024, doi: 10.1016/j.microc.2023.109669.

[22] “¿Qué es el aprendizaje supervisado? | IBM”. Accedido: 24 de diciembre de 2023. [En línea]. Disponible en: https://www.ibm.com/es-es/topics/supervised-learning

[23] IBM, “¿Qué es la visión artificial?” Accedido: 17 de octubre de 2023. [En línea]. Disponible en: https://www.ibm.com/mx-es/topics/computer-vision

[24] “Fotograma”. Accedido: 17 de octubre de 2023. [En línea]. Disponible en: https://dle.rae.es/fotograma

[25] “Significado y definición de Frame”. Accedido: 17 de octubre de 2023. [En línea]. Disponible en: https://sistemas.com/frame.php

[26] “Create object to read video files - MATLAB - MathWorks América Latina”. Accedido: 17 de octubre de 2023. [En línea]. Disponible en: https://la.mathworks.com/help/matlab/ref/videoreader.html?searchHighlight=video%20reader&s_tid=srchtitle_support_results_1_video%20reader

[27] “Read next video frame - MATLAB readFrame - MathWorks América Latina”. Accedido: 17 de octubre de 2023. [En línea]. Disponible en: https://la.mathworks.com/help/matlab/ref/videoreader.readframe.html?s_tid=doc_ta

[28] “Ground truth label data - MATLAB - MathWorks América Latina”. Accedido: 19 de octubre de 2023. [En línea]. Disponible en: https://la.mathworks.com/help/vision/ref/groundtruth.html?s_tid=doc_ta

[29] “Create training data for an object detector - MATLAB objectDetectorTrainingData - MathWorks América Latina”. Accedido: 19 de octubre de 2023. [En línea]. Disponible en: https://la.mathworks.com/help/vision/ref/objectdetectortrainingdata.html?s_tid=srchtitle_site_search_1_objectDetectorTrainingData%20

[30] “Combine data from multiple datastores - MATLAB combine - MathWorks América Latina”. Accedido: 19 de octubre de 2023. [En línea]. Disponible en: https://la.mathworks.com/help/matlab/ref/matlab.io.datastore.combine.html?s_tid=doc_ta

[31] “Train ACF object detector - MATLAB trainACFObjectDetector - MathWorks América Latina”. Accedido: 19 de octubre de 2023. [En línea]. Disponible en: https://la.mathworks.com/help/vision/ref/trainacfobjectdetector.html?searchHighlight=trainACFObjectDetector%20&s_tid=srchtitle_support_results_1_trainACFObjectDetector%20

[32] “Detect objects using YOLO v3 object detector - MATLAB detect - MathWorks América Latina”. Accedido: 19 de octubre de 2023. [En línea]. Disponible en: https://la.mathworks.com/help/vision/ref/yolov3objectdetector.detect.html?searchHighlight=detect&s_tid=srchtitle_support_results_3_detect

[33] “Select strongest bounding boxes from overlapping clusters using nonmaximal suppression (NMS) - MATLAB selectStrongestBbox - MathWorks América Latina”. Accedido: 19 de octubre de 2023. [En línea]. Disponible en: https://la.mathworks.com/help/vision/ref/selectstrongestbbox.html?s_tid=doc_ta

[34] “Determine if video frame is available to read - MATLAB hasFrame - MathWorks América Latina”. Accedido: 19 de octubre de 2023. [En línea]. Disponible en: https://la.mathworks.com/help/matlab/ref/videoreader.hasframe.html?s_tid=doc_ta

[35] “Determinar si un arreglo está vacío - MATLAB isempty - MathWorks América Latina”. Accedido: 19 de octubre de 2023. [En línea]. Disponible en: https://la.mathworks.com/help/matlab/ref/double.isempty.html?searchHighlight=isempty&s_tid=srchtitle_support_results_1_isempty

[36] “Update figures and process callbacks - MATLAB drawnow - MathWorks América Latina”. Accedido: 19 de octubre de 2023. [En línea]. Disponible en: https://la.mathworks.com/help/matlab/ref/drawnow.html?s_tid=doc_ta

[37] “Acquire single image frame from a webcam - MATLAB snapshot - MathWorks América Latina”. Accedido: 21 de noviembre de 2023. [En línea]. Disponible en: https://la.mathworks.com/help/supportpkg/usbwebcams/ug/webcam.snapshot.html?s_tid=srchtitle_site_search_1_snapshot

[38] “Test for valid graphics or Java object handle - MATLAB ishandle - MathWorks América Latina”. Accedido: 21 de noviembre de 2023. [En línea]. Disponible en: https://la.mathworks.com/help/matlab/ref/ishandle.html?s_tid=srchtitle_site_search_1_ishandle

[39] “Iniciar el cronómetro temporizador - MATLAB tic - MathWorks América Latina”. Accedido: 6 de diciembre de 2023. [En línea]. Disponible en: https://la.mathworks.com/help/matlab/ref/tic.html?searchHighlight=tic&s_tid=srchtitle_support_results_1_tic

[40] “Leer el tiempo transcurrido de un cronómetro - MATLAB toc - MathWorks América Latina”. Accedido: 6 de diciembre de 2023. [En línea]. Disponible en: https://la.mathworks.com/help/matlab/ref/toc.html?s_tid=srchtitle_site_search_1_toc

[41] “Arrays that represent points in time - MATLAB - MathWorks América Latina”. Accedido: 6 de diciembre de 2023. [En línea]. Disponible en: https://la.mathworks.com/help/matlab/ref/datetime.html?s_tid=doc_ta

[42] “Compute bounding box precision and recall against ground truth - MATLAB bboxPrecisionRecall - MathWorks América Latina”. Accedido: 6 de diciembre de 2023. [En línea]. Disponible en: https://la.mathworks.com/help/vision/ref/bboxprecisionrecall.html?searchHighlight=bboxPrecisionRecall&s_tid=srchtitle_support_results_1_bboxPrecisionRecall

[43] “Puntuación F1 en aprendizaje automático: introducción y cálculo”. Accedido: 6 de diciembre de 2023. [En línea]. Disponible en: https://www.v7labs.com/blog/f1-score-guide

8 Anexo

A. Investigación previa.

Para llegar decidir que era posible usar imágenes RGB para el proyecto se hicieron mediciones con la cámara espectral Pika L antes mencionada, y se sacaron algunas conclusiones y muestras no usadas para la elaboración del informe, de esta información se tiene como relevancia el set up utilizado que se muestra en la figura 6.1 y las capturas realizadas en la figura 6.2.

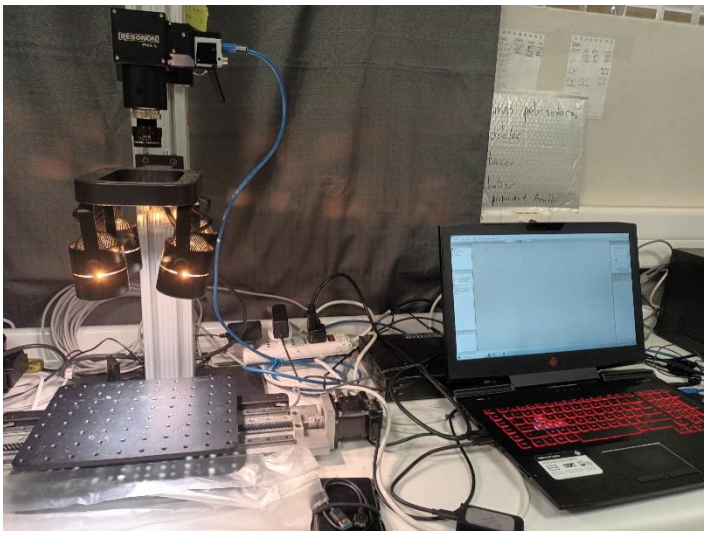


Fig. 8.1: Setup cámara Pika L.

Fig. 8.2: Capturas realizadas con cámara Pika L sobre base blanca.

Analizando algunas de las especies en el laboratorio se encontraron previamente a este informe los espectros del mejillón sin valva y junto a otras especies que se muestran en las figuras 6.3 y 6.4.

Fig. 8.3: Especies disponibles en laboratorio para análisis espectral.

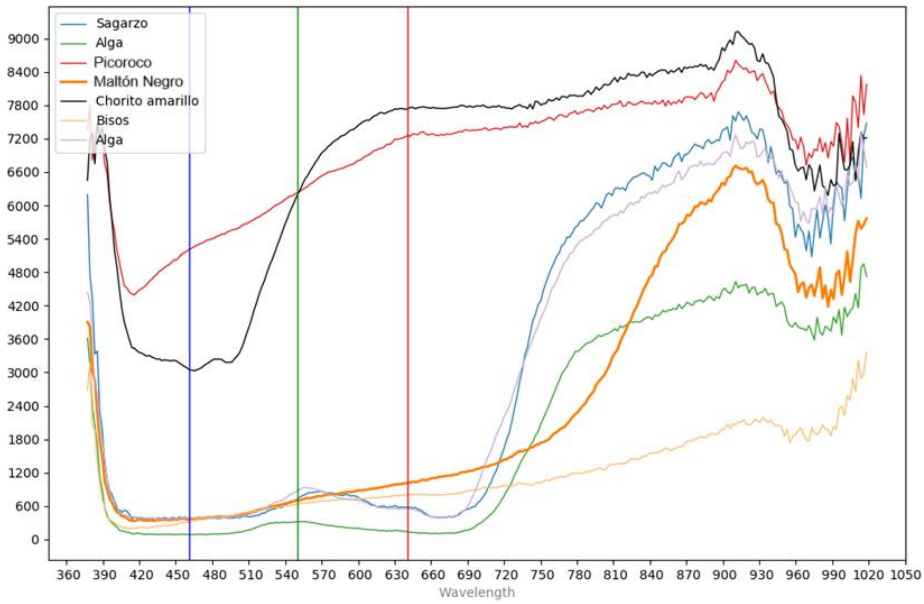


Fig. 8.4: Análisis espectral de distintas especies disponibles en laboratorio

Con estas imágenes se hizo el análisis espectral del mejillón y otras especies y la primera clasificación de la especie donde se encontró que las características espectrales del mejillón están en el espectro visible, figura 8.5.

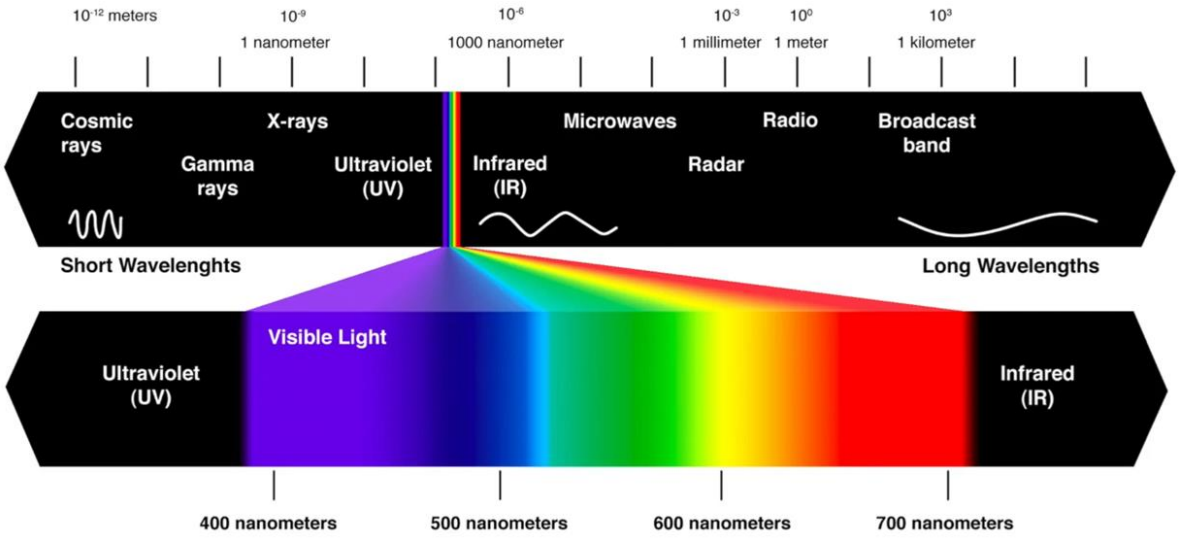


Fig. 8.5: Banda espectral y espectro visible.