Universidad Politécnica del Estado de Morelos

Sistema de rastreo de posición y tamaño de objetos acústicamente levitados

TESIS

Que para obtener el título de:

INGENIERO EN ELECTRÓNICA Y COMUNICACIONES

Presenta

Yesenia Bahena Vicencio

Directores de tesis

Dr. Juan Paulo Sánchez Hernández

Dr. Víctor Ulises Lev Contreras Loera

Jiutepec, Morelos. Agosto de 2021

Contenido

Agradecimientos

Agradezco a la **Universidad Politécnica del Estado de Morelos (UPEMOR)**, por ser mi segunda casa de estudio durante 3 años, por brindarme las herramientas necesarias y los conocimientos para mi formación profesional.

Agradezco al **Instituto de Ciencias Físicas, UNAM** por permitirme realizar mi proyecto de tesis.

Agradezco al **Programa de Apoyo a Proyectos de Investigación e Innovación Tecnológica (PAPIIT) IN109221 de DGAPA-UNAM** por la beca otorgada y así poder cubrir gastos del transporte al **Instituto de Ciencias Físicas, UNAM**.

Agradezco especialmente al **Dr. Víctor Ulises Lev Contreras Loera** y al **Dr. Juan Paulo Sánchez Hernández** por su tiempo, apoyo y paciencia que me brindaron, sobre todo agradezco la disposición que tuvieron al resolver cada una de mis dudas.

Agradezco al **M.C Daniel Rojas Sandoval** por sus consejos, por el ánimo que nos daba, por la paciencia en explicar cada una de mis dudas y lo más importante por la confianza que brinda.

Agradezco al **M.C Miguel Ángel Velasco Castillo** por el ánimo que en cada una de las clases nos dio, por su tiempo en resolver alguna duda sobre una evidencia, por compartirnos experiencias de trabajo, por los consejos para no rendidos en el camino, por sus chistes que nos hacía despertar a mitad de clase, por ser un buen tutor, por su comprensión y sobre todo por preocuparse por el grupo.

Agradezco a cada uno de mis profesores de la universidad por sus palabras de motivación, por el tiempo dedicado a la resolución de dudas de algún tema y por su empeño para enseñar.

Agradezco y dedico con todo mi amor y cariño:

A mis padres **Rubén Bahena Reyes** y **Romelia Vicencio Hernández** por confiar en mí de principio a fin, por cada una de sus palabras de motivación, por cada

consejo que me ayudó a afrontar la vida, por su comprensión, confianza, amor e impulso para lograr cada una de mis metas, por su tiempo y apoyo brindado durante la pandemia, y sobre todo por ser un gran pilar en mi vida.

A mis hermanos **Alfredo Bahena Vicencio** y **Anayely Bahena Vicencio** por su apoyo incondicional, sus consejos, sus palabras motivadoras y por confiar en mí.

A **Moisés Gamaliel Estrada Vergara** por ser un gran pilar en mi vida, por su apoyo, por alegrar mis días y por los buenos momentos compartidos.

A mis mejores amigas **Elienai B.** y **Cristal Andrea M.** por alegrar mis días, por su apoyo, sus consejos y por los bonitos recuerdos que han dejado en mi vida.

A mis amigos de preparatoria **Cristina, Paulina, Abraham, Jonathan, Mauricio, Cristian, Ricardo** y **Eliel** por su apoyo y los buenos momentos compartidos.

A mis amigos de la universidad **Alan J.**, **Diana**, **Adrián**, **Santiago**, **Alan O.**, **Alexis** y **Emmanuel C.** por su gran apoyo, su tiempo y por los buenos momentos compartidos.

Resumen

La levitación acústica es una técnica que nos permite suspender objetos en el aire u otro fluido sin la necesidad de un soporte mecánico. La levitación acústica día a día ha ido evolucionando, por ejemplo, en la década de los 30 era utilizada por los físicos y les permitía observar el comportamiento de las ondas sonoras. Actualmente la levitación permite suspender, rotar y trasladar objetos en tres dimensiones. La levitación acústica se puede combinar con diferentes sistemas de medición sin contacto, permitiendo observar las características de las muestras.

Al momento de trabajar con sustancias líquidas en un levitador acústico, la muestra oscila, rota y cambia de tamaño de acuerdo a la presión que está siendo sometida, a simple vista no se puede observar estos sucesos, es la importancia de implementar sistemas de visión por computadora a un levitador acústico, los sistemas de visión por computadora es un sistema que realiza tareas similares al sistema de visión humano, para lograr el funcionamiento del sistema de rastreo de posición y tamaño es necesario la creación de un repositorio de imágenes, en las cuales se seleccionó y clasificó el objeto que se desea ser rastreado por la cámara, posteriormente se realiza el reentrenamiento de la arquitectura You Only Look Once versión 3 (YOLOv3) para obtener los datos requeridos en el proyecto.

Este documento tiene como objetivo principal la implementación de un sistema de rastreo de posición y tamaño de objetos acústicamente levitados. Para la realización del sistema de rastreo se utilizó un levitador acústico uniaxial, una cámara web y una computadora con un sistema operativo Windows. La cámara se posicionó a 1 cm de distancia del levitador acústico y se conectó a la computadora, la computadora tendrá control de la cámara cada vez que el código se ejecute una vez y se podrá detectar las dos clases (Gota de agua y Objeto esférico) y así obtener la posición en coordenadas y tamaño en pixeles del objeto levitado, así mismo se investigaron los conceptos físicos para mejor comprensión sobre la levitación acústica, visualización por computadoras y la manera en que trabaja el algoritmo de la arquitectura YOLOv3. También se investigaron los conceptos básicos de electrónica y programación para el desarrollo e implementación del sistema de rastreo.

Abreviaturas

YOLO You Only Look Once (Sólo se mira una vez)

YOLOV3 You Only Look Only Versión 3

TL Transductores Langevin

IA Inteligencia Artificial

3D Tres dimensiones

2D Dos dimensiones

CV/CV Computer Vision/ Visión por computadora

CNN/RNC Convolutional Neural Networks/Redes Neuronales Convolucionales

R-CNN Regions with Convolutional Neural Networks

FASTER R-CNN Faster Regions with Convolutional Neural Networks

OpenCV Open Computer Vision (Visión artificial abierta)

IBM International Business Machines

BSD Berkeley Software Distribution

Índice de figuras

Índice de tablas

Capítulo I: Introducción

En este capítulo se plasman los antecedentes los cuales son un panorama general del proyecto, se define la problemática a resolver, se propone una solución. También se describe el objetivo general y los específicos, se mencionan los alcances, las limitaciones, y la metodología a seguir para el desarrollo del proyecto. Al final del capítulo se describe la organización de la tesis.

1.1 Introducción

La levitación acústica es un método que nos permite suspender objetos milimétricos y micrométricos en el aire u otro fluido con sonido sin la necesidad de un soporte mecánico. La levitación acústica puede utilizarse para retener partículas de diferentes materiales y tamaños es la razón principal por la cual tiene una amplia gama de aplicaciones en biología, química, ciencia de los materiales e ingeniería.

Los levitadores acústicos se han desarrollado desde los años 70's para la producción de ambientes de microgravedad. Un levitador acústico de un solo eje o levitadores acústicos uniaxiales son los dispositivos principales para producir trampas acústicas, se componen de un transductor emisor y un reflector u otro transductor que opera a la misma frecuencia. Entre el emisor y el reflector existe una distancia múltiple de media longitud de onda donde se genera una onda estacionaria. Las ondas estacionarias ejercen fuerza de radiación acústica que forman trampas acústicas lo que permite que las partículas se leviten.
Al trabajar con diferentes partículas u objetos, al momento que están siendo levitados es útil saber en qué posición se encuentra y el tamaño del objeto, especialmente cuando se trabaja con sustancias líquidas, en este caso gotas de agua dichas muestras oscilan, rotan y cambian de forma debido al cambio de presión que son sometidas. La visión por computadora puede emplearse para ver y analizar el comportamiento de los objetos levitados mediante imágenes o conjuntos de imágenes captados por una cámara. La visión por computadora funciona de manera similar a la visión humana solo que la visión humana tiene la

ventaja de estar entrenada para distinguir objetos, es capaz de visualizar la distancia en que se encuentra un objeto, si está en movimiento o no en el mismo instante, mientras que la visión por computadora entrena a las máquinas para realizar estas funciones a través de cámaras, datos y algoritmos en lugar de retinas, nervios ópticos y una corteza visual.

1.2 Antecedentes

La levitación acústica se ha utilizado desde hace muchos años para la manipulación de materiales gaseosos y líquidos sin necesidad de contacto físico con la superficie del recipiente (Zhao & Wallaschek, 2014). Para la generación de ambientes de microgravedad principalmente se utilizaban levitadores acústicos basados en un transductor piezoeléctrico tipo Langevin y un reflector. Sin embargo, en las últimas décadas, se han desarrollado levitadores basados en arreglos de múltiples transductores compactos y económicos (Morris et al, 2019). En el Laboratorio de Óptica Aplicada (LOA) del Instituto de Ciencias Físicas (ICF), UNAM Morelos, se está trabajando con levitadores acústicos para la levitación de gotas de agua para detectar contaminantes con técnicas ópticas como plomo, mercurio, cadmio y bario. En el 2020 se desarrolló un levitador acústico uniaxial de fases variables para la generación de vórtices. A finales del 2021 se inició la construcción y caracterización de un levitador que desplaza objetos. También se desarrolló un sistema de rastreo que permite rastrear una sola gota de agua para poder analizarla y extraer su información de posición y tamaño en píxeles al momento de ser levitada, estos sistemas comúnmente se conocen como sistemas de visión por computadora.

Visión por computadora: Uno de los momentos más importantes de la visión por computadora fue en la década de los 80 con el desarrollo de la ingeniería informática y la creación de procesadores más rápidos y sofisticados, que dio lugar a microprocesadores capaces de captar, procesar y reproducir imágenes tomadas por una cámara que podía estar conectada de forma remota (Infaimon, 2020). Actualmente se han desarrollado algoritmos con la capacidad de proporcionar información de un objeto a través de una imagen o en un conjunto de imágenes.

1.3 Planteamiento del problema

En el Laboratorio de Óptica Aplicada (LOA) del ICF, cuando se diseñan y caracterizan levitadores acústicos es útil conocer la posición y estabilidad del objeto que está levitando. También cuando se realizan análisis espectroscópicos de sustancias líquidas levitadas, en este caso gotas de agua, no se puede determinar su tamaño estas muestras oscilan, rotan y cambian de tamaño debido a la presión que son sometidas a su evaporación. Estos sucesos no se pueden caracterizar de manera precisa a simple vista; sin embargo, al implementar técnicas de visión por computadora la caracterización de los levitadores sería una tarea más simple.

1.4 Objetivo general

Implementar un sistema de rastreo utilizando transfer learning mediante la arquitectura You Only Look Once (YOLO) que permita determinar en tiempo real la posición y tamaño de objetos acústicamente levitados.

1.5 Objetivos específicos

1. Estudiar y comprender los temas de levitación acústica, visión por computadora, transfer learning, redes neuronales convolucionales y You Only Look Once (YOLO).
2. Reentrenar el sistema de rastreo de objetos con dos clases en tiempo real, a partir del modelo ya entrenado mencionado en la metodología.
3. Evaluar el sistema de rastreo con las dos clases proporcionadas en el entrenamiento.

1.6 Justificación

Al disponer de un sistema de rastreo de posición y tamaño utilizando transfer learning con YOLOv3 en tiempo real a un levitador acústico sería de gran ayuda para los trabajos realizados en el Laboratorio de Óptica Aplicada del Instituto de Ciencias Físicas de la UNAM debido a que el sistema ayudará a obtener el tamaño y la posición en píxeles de los objetos esféricos y las gotas de agua levitadas en tiempo real.

1.7 Hipótesis

La implementación de un sistema de rastreo utilizando YOLOv3 aplicado a objetos acústicamente levitados permitirá obtener el tamaño y la posición en tiempo real de las partículas levitadas, necesario para análisis posteriores de posición y estabilidad.

1.8 Alcances y limitaciones

A continuación, se abordan los alcances y limitaciones que acotan el desarrollo del prototipo.

1.8.1 Alcances

Los alcances que acotan el desarrollo del proyecto son los siguientes:

I. Recopilación de 150 imágenes por clase (objeto) a detectar.
II. Creación del archivo .txt en formato YOLOv3 por imagen en el software LabelImg.
III. Reentrenamiento del sistema de rastreo de objetos con dos clases (Objeto esférico y gota de agua) utilizando transfer learning con YOLOv3.
IV. Visualizar la posición y tamaño del objeto levitado en la computadora.
V. Guardar los datos del objeto levitado de acuerdo al tiempo que el usuario tarda en presionar la tecla correspondiente para pausar la ejecución.

1.8.2 Limitaciones

Las limitaciones que acotan el desarrollo del proyecto son los siguientes:

I. El sistema no podrá identificar al objeto cuando se desplace rápidamente en el levitador acústico.
II. El sistema no permite la conexión con una cámara de alta definición al no contar con los drivers necesarios
III. El sistema no cuenta con una interfaz gráfica para su uso.

1.9 Metodología

La metodología seleccionada para el desarrollo del sistema de rastreo de posición y tamaño de objetos acústicamente levitados es un modelo en cascada que consta de seis etapas, mostradas en la Figura 1.1, y que se describen a continuación.

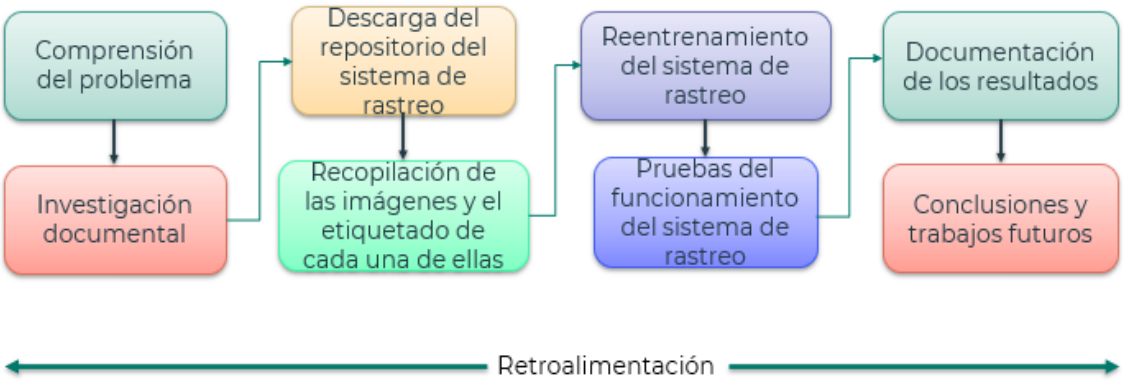

Figura 1.1.- Metodología de solución

Compresión del problema: en esta parte es necesario comprender el ¿por qué implementar el sistema de rastreo de objetos?, ¿en que apoyará la implementación del sistema de rastreo?

Investigación documental: es donde se realiza una investigación de los temas involucrados al problema, en este caso los temas a investigar con mayor profundidad son: visión por computadora, redes neuronales convolucionales, You Only Look Once, transfer learning y levitación acústica. Así mismo se investiga los componentes necesarios para el funcionamiento del sistema, lenguaje de programación y software a utilizar.

Descarga del repositorio del sistema de rastreo: para la implementación del sistema de rastreo de objetos es necesario la descarga del repositorio (un repositorio contiene diferentes archivos de un proyecto (Barrionuevo, 2009)) con nombre “detección de objetos en video” de Alejandro Puig quien es un especialista en Inteligencia Artificial. Se eligió este repositorio porqué el autor al explicar los pasos a seguir y el funcionamiento del código que compone al sistema de rastreo es fácil de comprender, en cambio con otros autores fue un poco complicado

seguir algunos pasos o entender el funcionamiento del código del sistema de rastreo.

Recopilación de las imágenes y el etiquetado de cada una de ellas: Para que el sistema de rastreo funcione con las clases de nuestro interés se tiene que recopilar imágenes que contengan los objetos. Una vez recopiladas las imágenes con la ayuda del software LabelImg se lleva a cabo el etiquetado de cada objeto de acuerdo a su clase creando el archivo .txt con formato YOLO.

Reentrenamiento del sistema de rastreo: ya teniendo las imágenes etiquetadas y descargado el repositorio se realiza el reentrenamiento del sistema de rastreo de objetos.

Pruebas del funcionamiento del sistema de rastreo: una vez finalizado el reentrenamiento se realizan las pruebas para comprobar que el sistema si rastrea las dos clases de nuestro interés.

Documentación de los resultados: al verificar que el sistema rastrea los objetos de nuestro interés, se procede a documentar los resultados obtenidos y así determinar si el prototipo está cumpliendo con cada una de las características propuestas.

Conclusiones y trabajos futuros: por último, se describen los puntos de mayor interés y las conclusiones del desarrollo del proyecto, determinando si cumple con los objetivos propuestos. También se mencionan las potenciales mejoras del sistema de rastreo de objetos que pueden realizarse como trabajo a futuro.

1.10 Organización de la tesis

El documento de tesis está distribuido en cinco capítulos:

Capítulo I.- Introducción

En este capítulo se abordan los antecedentes del proyecto a desarrollar, las características, los objetivos a lograr, justificación, planteamiento del problema, hipótesis, sus alcances y limitaciones, además contiene una introducción para mayor comprensión del proyecto.

Capítulo II.- Marco teórico

En este capítulo se describen los conceptos teóricos que sustentan el desarrollo del proyecto, se detalla el funcionamiento de cada uno de los componentes y del software a utilizar. Permitiendo conocer el panorama general del proyecto y así dar inicio a la implementación de cada etapa del proyecto.

Capítulo III.- Diseño e implementación

En este capítulo se presenta la propuesta de diseño del proyecto, se detalla paso a paso el desarrollo del proyecto, se muestra el algoritmo desarrollado para obtener la posición, el tamaño de la partícula levitada, así como las conexiones entre la cámara, el módulo el perfil y la computadora. También se muestra el levitador acústico utilizado.

Capítulo IV.- Pruebas y resultados

En este capítulo se presentan y describen las pruebas realizadas al sistema de monitoreo de posición y tamaño de un objeto levitado acústicamente. Así mismo se presentan los resultados logrados en cada una de las etapas del diseño del proyecto.

Capítulo V.- Conclusiones y trabajos futuros

En este capítulo se presentan las conclusiones del proyecto de acuerdo con los resultados alcanzados durante su desarrollo. También se da una breve introducción de los trabajos futuros para el mejoramiento del proyecto presente.

Capítulo II.- Marco teórico

En este capítulo se desglosa los elementos teóricos necesarios para el desarrollo del proyecto presente.

2.1. Levitación acústica

La levitación acústica se define como una técnica que nos permite atrapar y suspender objetos micrométricos y milimétricos en el aire u otro fluido a través de ondas ultrasónicas (Contreras, 2021). Las ondas ultrasónicas operan a frecuencias mayores a los 20 kHz, rango no audible para el ser humano (Smoot, 2021). Existen diferentes técnicas para levitar objetos, como la levitación magnética, la eléctrica, la óptica, la aerodinámica y la acústica, sin embargo, al comparar estas técnicas, la levitación acústica presenta una gran ventaja al no requerir de ninguna propiedad magnética y/o eléctrica del material a levitar (Valencia, 2017). Por lo anterior, la levitación acústica se puede emplear en diversas áreas, por ejemplo, en la química analítica, ciencia de los materiales o biología, ya que al realizar estudios de muestras el resultado puede ser alterado por el contacto que se tiene con el contenedor.

Métodos de levitación acústica

La levitación acústica se clasifica en cinco métodos: los cuales se pueden observar en la Figura 2.1 (Peralta, 2018): a) y b) levitación acústica de onda estacionaria, c) levitación acústica de campo cercano, d) levitación invertida de campo cercano, d) levitación de haz simple.

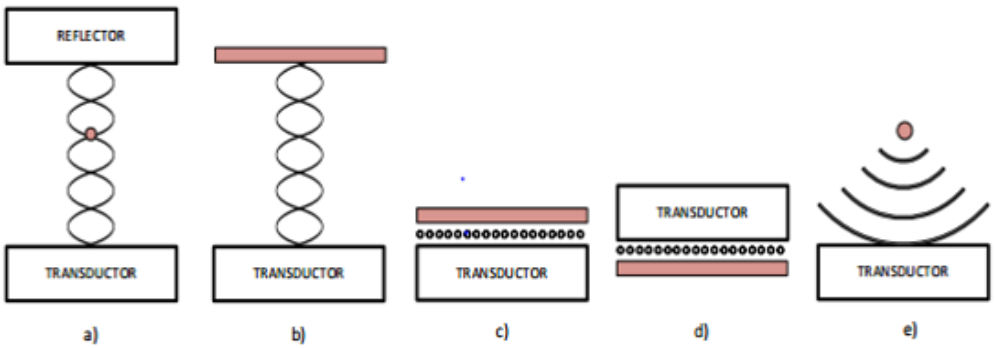

Figura 2.1.- Métodos de levitación acústica (Vara, 2020).

Uno de los métodos más comunes de producir la levitación es a través de ondas estacionarias producidas dentro de levitadores uniaxiales. La onda estacionaria es el resultado de la interferencia generada por las ondas acústicas contrapropagantes y sus múltiples reflexiones dentro de la cavidad.

Los levitadores uniaxiales de cavidad están formados por un transductor que es una superficie vibrante que emite sonido (emisor) y un reflector (o dos transductores opuestos operando a la misma frecuencia), donde las superficies que emiten y reflejan las ondas acústicas están separadas por una distancia múltiplo entero de media longitud de onda (Contreras, 2021). Esto produce reflexiones e interacciones entre todas las ondas generadas, así creando puntos máximos, mínimos y nulos de presión (Folkmann, 2019/2020). La partícula u objeto estaría levitando y alcanzando su punto de equilibrio en las posiciones del eje z donde hay puntos nulos de presión como se puede observar en la Figura 2.2 (Peralta, 2018).

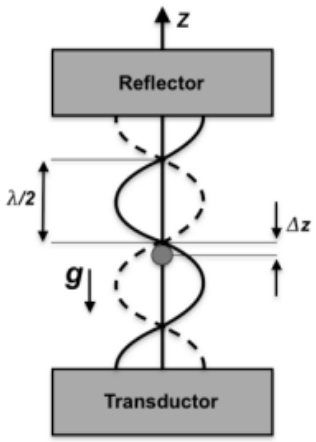

Figura 2.2.- Esquema de un levitador acústico de onda estacionaria entre un reflector y un transductor (Peralta, 2018).

Donde g representa la gravedad que actúa sobre la masa de la partícula, λ es la longitud de onda, $\lambda/2$ es media longitud de onda, tamaño máximo de la partícula a levitar y Δz es el eje en el cual debería de levitar la partícula. La distancia Δz puede ser calculada en función de las propiedades de la partícula y la onda estacionaria.

2.2. Presión acústica

La presión acústica es la diferencia entre la presión instantánea debida al sonido y la presión atmosférica estática. Sus unidades en el sistema internacional son en Pascales (Pa) (Gordillo & Guaraca, 2015).

El oído humano tolera diferentes presiones acústicas, la presión acústica mínima para oído humano es aproximadamente 20 μPa (umbral auditivo) y la máxima es aproximadamente a 60 Pa (umbral de dolor). La presión acústica también utiliza una escala logarítmica comprimida expresada en decibelios, por la razón que la diferencia entre los sonidos intensos y débiles es muy grande (Williams, 2021).

El nivel de presión acústica se calcula con la ecuación 2.1:

$$L_p = 10\log_{10}\left(\frac{p}{p_o}\right)^2 = 20\log_{10}\left(\frac{p}{p_o}\right) dB \qquad ec.2.1$$

donde p_0 es la presión sonora de referencia, p = presión sonora instantánea. La presión acústica de los sonidos audibles en decibelios varía entre 0 dB (umbral de audición) y 120 dB (umbral de dolor).

2.3. Transductores ultrasónicos

Un transductor es un dispositivo capaz de transformar una señal física, mecánica, óptica, etc., en señal eléctrica o viceversa. Los transductores son utilizados en diferentes áreas de trabajo, por ejemplo, en la industria, en la medicina, en la robótica, entre otras, donde son utilizados para obtener información de entornos físicos y químicos para conseguir señales o impulsos eléctricos o viceversa (Altúzar, 2017).

Los principales elementos de un levitador son los transductores ultrasónicos de la familia piezoeléctricos (Marzo et al, 2017), los transductores piezoeléctricos son aquellos que consisten en materiales cristalinos piezoeléctricos, los cuales se contraen ante impulsos eléctricos aplicados en la superficie. Estos transductores no contienen imanes y no utilizan ningún tipo de magnetismo (Altúzar, 2017). En la Figura 2.3 se observa la estructura general de un transductor piezoeléctrico ultrasónico (Herberg & Ghazisaeidi, 2006).

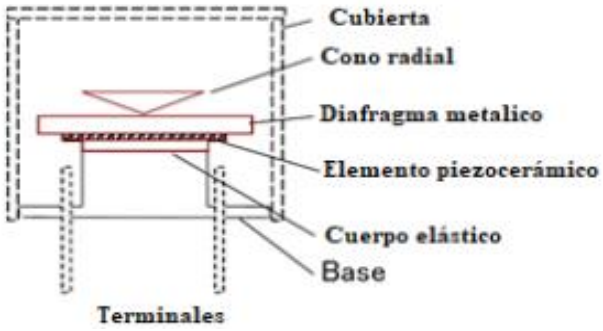

Figura 2.3.- Estructura general de un transductor piezoeléctrico ultrasónico (Herberg & Ghazisaeidi, 2006)

Los transductores ultrasónicos transforman la señal eléctrica de entrada en ondas acústicas. Estos transductores emiten ondas de frecuencias mayores a 20 kHz, rangos audibles más altos que la audición humana y son utilizados ampliamente en ultrasonidos de alta potencia y sonido subacuático (Smoot, 2021). Para operar en el aire, se encontró que los transductores para la medición de distancias brindan una buena potencia acústica, una frecuencia de resonancia constante y están disponibles a un precio bajo (Marzo et al, 2017). Su principal función es detectar objetos a través de las ondas sonoras, su funcionamiento básico se puede observar en la Figura 2.4, donde se tiene un transmisor que envía una señal o pulso ultrasónico, que se refleja sobre un determinado objeto y la flexión es detectada por un receptor de ultrasonidos (Cuamatzi et al, 2010) (Reinaldo, 2008).

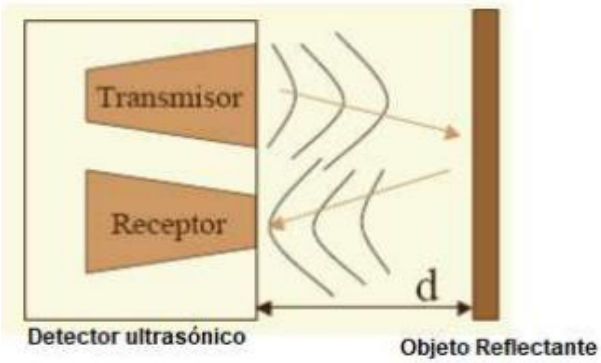

Figura 2.4.- Funcionamiento de un transductor ultrasónico (Reinaldo., 2008).

La mayoría de los sensores ultrasónicos de bajo costo se basan en la emisión de un pulso ultrasónico cuyo lóbulo o campo de acción es de forma cónica, como se puede observar en la Figura 2.5 (Reinaldo, 2008).

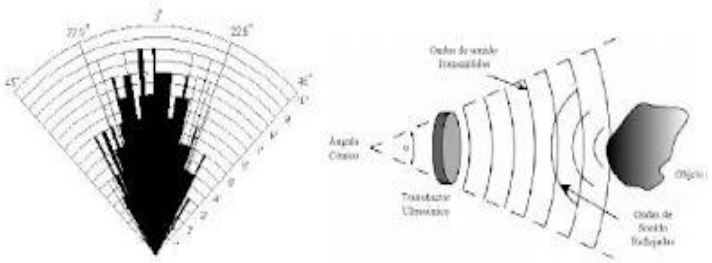

Figura 2.5.- Perspectiva de ángulo de emisión de un pulso ultrasónico (R., 2008)

2.4 Levitadores de arreglos de transductores

Desde el inicio de la historia de la levitación acústica, se han utilizado principalmente transductores Langevin (TL) como fuentes de emisión ultrasónica para la construcción de levitadores acústicos (Contreras, 2021), estos levitadores se tenían que fabricar con una alta tolerancia con frecuencias de resonancia cuidadosamente combinadas. Sin embargo, los transductores Langevin tienen desventajas limitantes para el uso generalizado en la levitación acústica, las cuales son (Marzo et al, 2017) (Morris et al, 2019):

1. Son difíciles de sintonizar con una frecuencia de resonancia específica.
2. El alto voltaje utilizado para impulsarlos es potencialmente peligroso.
3. Normalmente los transductores se calientan debido a la ineficiencia de la transducción y su comportamiento de resonancia es sensible a la temperatura, antes de operar se deben dejar calentar y así perder potencia después de un funcionamiento intenso.

En la última década se han desarrollado distintos tipos de levitadores acústicos con características atractivas. En estos levitadores se han sustituido los TL por arreglos de múltiples transductores más compactos, eficientes y económicos (Marzo et al, 2017) (Contreras, 2021). Al utilizar transductores más compactos permite ordenarlos en arreglos o matrices que operan en fase permitiendo que la superficie emisora del levitador esté discretizada y vibre de manera similar a la superficie continua del TL. Dicha discretización posee dos principales ventajas con respecto a los levitadores basados en un solo transductor (Contreras, 2021):

1. Se pueden diseñar superficies emisoras de mayor área y de geometrías más variadas de manera simple.
2. Se logra la manipulación de los transductores independientemente y se controla su fase electrónicamente.

En los estudios recientes de levitación acústica se ha demostrado que los levitadores de arreglos en fases tienen la capacidad de levitar objetos de alta densidad. Marzo et al en 2017, reportaron un levitador acústico multiemisor de un solo eje llamándolo TinyLev, este levitador produce un atrapamiento estable, es resistente a los cambios de temperatura y humedad, funciona con bajo voltaje, es fácil de operar, puede operar durante períodos de tiempo prolongados y está basado en un diseño de cavidad concéntrica uniaxial, como se puede ver en la Figura 2.6. El levitador TinyLev es capaz de levitar objetos con densidades de hasta 6.5 g/m^3 (Contreras, 2021) (Marzo et al, 2017). En 2021, Contreras y Marzo reportaron un levitador similar al TinyLev, ellos mostraron que, al ajustar finamente la longitud de cavidad y operar en modo resonante, es posible levitar esferas de acero y mercurio, así mostrando una mejora en el desempeño de levitación de las cavidades acústicas basadas en arreglos de fase (Contreras, 2021).

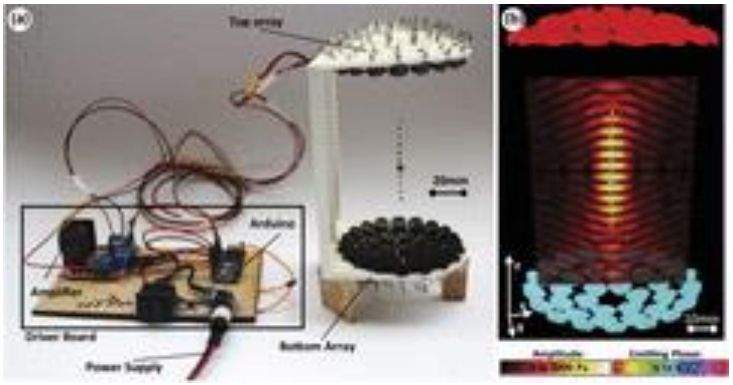

Figura 2.6.- a) Levitador TinyLev, b) Campo acústico simulado; cada círculo representa un transductor de 10 mm de diámetro y el color representa la fase de emisión de los transductores (Marzo, et al, 2017).

2.5 Visión por computadora

La visión por computadora también es conocida como visión artificial, visión de máquina, etc., es una rama de la Inteligencia Artificial (IA) que permite ver, analizar y comprender imágenes o videos a través de computadoras para obtener información de los objetos físicos captados por la cámara (Molleda, 2008). La visión por computadora funciona de manera similar a la visión humana solo que la visión humana tiene la ventaja de estar entrenada para distinguir objetos, saber la distancia que se encuentra, si está en movimiento o no en el mismo instante,

mientras que la visión por computadora entrena a las máquinas para realizar estas funciones a través de cámaras, datos y algoritmos en lugar de retinas, nervios ópticos y una corteza visual (IBM, 2020). En la actualidad la visión por computadora es una tecnología nueva que día a día se va incorporando cada vez más en la vida cotidiana, se aplica en distintos procesos científicos y militares (Montoya & Cruz, 2013). La visión por computador es utilizada por muchas tecnologías, algunas de ellas son el reconocimiento de objetos, detección de eventos, la reconstrucción de una escena (mapping) y de imágenes (Juárez, 2018). Un sistema de visión por computadora es un sistema autónomo que puede realizar las mismas tareas que el sistema de visión humana realiza. Se tiene que tener claro que la entrada del sistema es una imagen o una secuencia de imágenes y la salida es información. Los sistemas tienen la capacidad de extraer o deducir las propiedades y estructura del mundo tridimensional a partir de una o un conjunto de imágenes. Una imagen es la representación bidimensional de una escena del mundo tridimensional (Molleda, 2008). Los principales componentes de un sistema de visión por computadora son los sensores de imagen y un digitalizador. Un sensor de imagen es un dispositivo físico, como puede ser un cámara, la cual captura imágenes para que sean procesadas. Un digitalizador, es conocido como tarjeta digitalizadora para una computadora, este dispositivo es capaz de convertir la señal analógica de salida del sensor de imagen en una señal digital, que puede ser procesada por una computadora (Molleda, 2008).

2.5.1 Aplicaciones de la visión por computadora

La visión por computadora cubre un amplio espectro de aplicaciones ya que permite extraer y analizar información espectral, espacial y temporal de diversos objetos. La información espectral incluye intensidad (escala de grises) y frecuencia (color), la información espacial se trata de aspectos como la forma y posición, ya sea en una, dos o tres dimensiones. La información temporal comprende aspectos estacionarios y dependientes del tiempo.

La visión por computadora es un área con más de 40 años de investigación y cuenta con diversas aplicaciones, por ejemplo, en vehículos autónomos desarrollados por Tesla, para el desarrollo de Robots, en tiendas en línea, como, de Amazon Go al momento de detectar cuando los compradores recogen o

devuelven artículos en existencia, IBM utilizó visión por computadora para crear My Moments para el torneo de golf Masters 2018, también se emplea en sistemas de seguridad entre otras (IBM, 2020).

2.5.2 Etapas en un proceso de visión por computadora

Como se mencionó anteriormente la visión extrae e interpreta información obtenida de un escenario, para lograr este proceso se requiere de distintas etapas, las cuales son (Montoya & Cruz, 2013) (Molleda,2008):

Adquisición de la imagen: Es el proceso de captura de una imagen digital a través de un dispositivo como una cámara digital, video-cámara, escáner, etc.

Pre-procesamiento: En esta etapa se busca el mejoramiento de la imagen, se corrige la falta de iluminación uniforme, la eliminación del ruido, nitidez de la imagen, realce de ciertos detalles o características de la imagen, regularizar sus colores, sus texturas, etc.

Segmentación: Es el proceso que divide a una imagen en objetos o regiones que sean de nuestro interés de estudio. La segmentación se basa en 3 propiedades:

- Similitud: Los píxeles de un objeto deben de tener valores parecidos.
- Discontinuidad: Los bordes deben de estar bien definidos.
- Conectividad: Los píxeles pertenecientes al mismo objetivo o región deben de estar agrupados.

Extracción de características: Es la obtención de características relevantes convenientes para diferenciar un objeto de otro de una imagen. Estas características pueden ser externas como la forma, perímetro, eje mayor, eje menor, rectángulo mínimo que contiene la región, excentricidad; o internas como el área, centro de gravedad, patrones de texturas (liso, áspero, regular, entre otros) y color (promedio y mediana de niveles de intensidad, máximo y mínimo de valores de intensidad).

Reconocimiento: Es el proceso que clasifica en categorías los objetos presentes en la imagen de acuerdo a sus descriptores de la etapa anterior. Los objetos que presentan descriptores semejantes se agrupan automáticamente en una misma

categoría, clase o con una mínima intervención humana. Para esto se utilizan técnicas, como puede ser la triangulación, se localiza al objetivo en el espacio 3D.

Interpretación: Es el proceso que da sentido o significado a cada categoría de los objetos reconocidos para la comprensión de la escena.

La Figura 2.7 muestra un diagrama de forma general de las etapas consideradas en un proceso de visión por computadora.

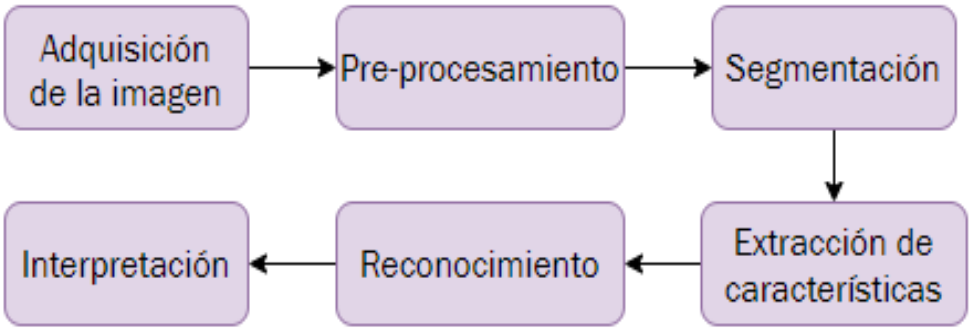

Figura 2.7.- Etapas de un proceso de visión por computadora.

2.5.3 Técnicas de visión por computadora

Las técnicas como la clasificación de imágenes, la detección de objetos, el seguimiento de objetos y la segmentación de imágenes son las cuatro principales técnicas que ayudan a crear visión por computadora ya sea cambiando las técnicas o por separado (Brita, 2021).

Clasificación de imágenes: La clasificación de imágenes tiene como objetivo clasificar el contenido de la imagen de acuerdo a su tipo. Las redes neuronales convolucionales es la arquitectura más utilizada (Brita, 2021). La clasificación de imágenes se puede clasificar en dos tipos:

- Clasificación binaria: Esta clasificación, puede verificar una sola clase de objeto para la imagen dada y obtener un resultado basado en si tiene ese objeto.
- Clasificación multiclase: Esta clasificación puede verificar más de dos clases de objetos en la imagen dada y obtener el resultado basado en cada uno de los objetos.

Para lograr la clasificación de imágenes se tiene que proporcionar imágenes de cada una de las clases a la computadora y así crear un conjunto de datos de entrenamiento. Cada una de las clases tiene propiedades independientes, estas propiedades están representadas por vectores. Estos vectores se entrenan con las RNC (Brita, 2021) (Das, 2021).

Detección de objetos: El funcionamiento de detección de objetos es identificar y ubicar objetos dentro de una imagen o video, esta técnica dibuja cuadros delimitadores que se posicionan en el objeto o los objetos de nuestro interés y así poderlos clasificar. Aunque los cuadros pueden ser de diferentes tamaños, pueden contener imágenes de la misma clase. Si la detección de imágenes contiene una gran cantidad de objetos también se requiere de una cantidad cada vez mayor de potencia informática. Se han desarrollado algoritmos como R-CNN, Fast R-CNN, YOLO, Single Shot MultiBox Detector (SSD) y Redes totalmente convolucionales basadas en regiones para encontrar rápidamente estas ocurrencias (Brita, 2021) (Das, 2021).

Seguimiento de objetos: El seguimiento de objetos rastrea el movimiento de uno o varios objetos de interés en la escena dada. Tradicionalmente se aplica para el monitoreo de video o interacciones del mundo real una vez detectado el objeto inicial. Las técnicas de seguimiento de objetos se pueden dividir en dos categorías de acuerdo al modelo de observación (Brita, 2021) (Das, 2021):

- Método generativo, describe las características aparentes y reduce errores en la reconstrucción de la búsqueda del objeto. Algunos ejemplos de este modelo que intentan encontrar una representación adecuada de los datos originales es el análisis de componentes principales (PCA), el análisis de componentes independientes (ICA), la factorización matricial no negativa (NMF).
- Método discriminativo, se utiliza para diferenciar entre el objeto y el fondo. Este método es el más exacto y poco a poco se convierte en el principal método de seguimiento. Ejemplos de métodos discriminativos son los codificadores automáticos apilados (SAE), las redes neuronales convolucionales y las máquinas de vectores de soporte (SVM).

Segmentación de imágenes: El seguimiento de imágenes es el proceso de dividir una imagen digital en un conjunto de píxeles. El objetivo principal de la segmentación de imágenes es simplificar la presentación de una imagen y facilitar el análisis. Dado que existen muchos enfoques diferentes para la segmentación de imágenes, Mask R-CNN y Fully Convolutional Networks (FCN) pueden usarse para predicciones densas sin capas completamente conectadas (Brita, 2021).

2.6 Redes neuronales convolucionales (RNC)

Las Redes Neuronales Convolucionales (RNC) es un tipo de red neuronal artificial, su funcionamiento es similar a la corteza visual de un cerebro biológico, se componen de neuronas que tienen pesos (los pesos son los valores que tienen todas las conexiones entre las neuronas de la red neuronal) y sesgos que pueden aprender (Barrios,2019). Las RNC utilizan distintas capas y por sí mismas logran reconocer una gran cantidad. Las RNC consisten en múltiples capas de filtros convolucionales de una o más dimensiones (Barrios, 2019). Las RNC han aportado múltiples soluciones al campo de la visión por computador. Principalmente son utilizadas en sistemas de detección y clasificación de objetos, segmentación de imágenes debido al buen desempeño. Funcionan excepcionalmente bien cuando se procesan datos no estructurados en imágenes o vídeos. En el campo de visión por computadora las RNC son muy buenas para las tareas de clasificación de escenas, para la detección o categorización de objetos y clasificación de imágenes en general. Para el desarrollo del proyecto no es necesario profundizar el tema de las RNC, pero sí es importante saber su funcionamiento para comprender el funcionamiento del algoritmo de la arquitectura YOLOv3.

Estructura de las redes neuronales convolucionales

En general, las RNC son construidas con una estructura que contiene 3 tipos distintos de capas, en la Figura 2.8 se observa la estructura general de las redes neuronales convolucionales y como cada una de sus capas trabaja.

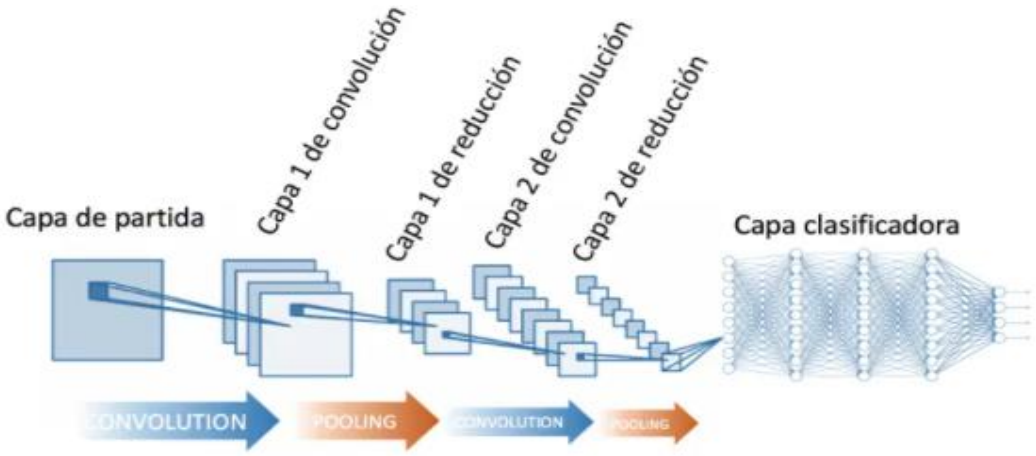

Figura 2.8.- Arquitectura de las redes neuronales convolucionales (Calvo, 2017).

Capa convolucional

La capa convolucional es el componente básico de las RNC y es donde se realizan la mayoría de los cálculos. En esta capa se realiza la operación de convolución, lo que distingue a las redes neuronales de otra red neuronal. Esta operación recibe como entrada la imagen, para luego aplicar sobre ella un filtro o kernel y así devolver un mapa de características de la imagen original, de esta manera se logra reducir el tamaño de los parámetros (Briega, 2019) (Calvo, 2017).

En la Figura 2.9 se observa cómo se lleva a cabo la operación de convolución.

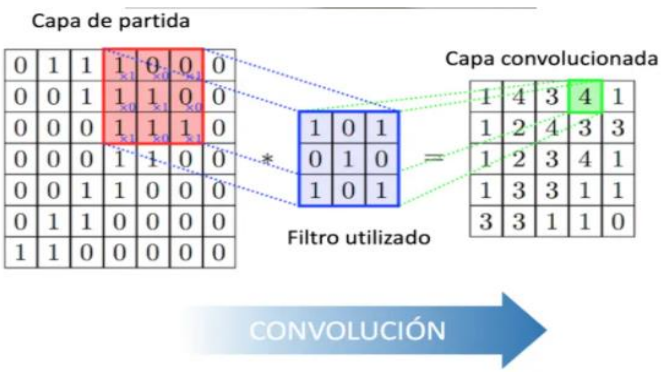

Figura 2.9.- Convolución (Calvo, 2017).

Capa de reducción o pooling

La capa de pooling generalmente se coloca después de la capa convolucional. Su principal utilidad es la reducción de las dimensiones espaciales (ancho por alto) del volumen de entrada para la siguiente capa convolucional. En esta capa la

operación que se lleva a cabo es la reducción de muestreo (pooling), dicha operación barre un filtro a lo largo de toda la entrada, pero este filtro no tiene algún peso (Briega, 2019).

Los dos tipos más comunes de la capa de pooling son:

- **Max-pooling:** A medida que el filtro se mueve a través de la entrada, selecciona el píxel con el valor máximo para enviarlo a la matriz de salida.
- **Average pooling:** A medida que el filtro se mueve a través de la entrada, calcula el valor promedio dentro del campo receptivo para enviarlo a la matriz de salida.

La operación que suele utilizarse en esta capa es máx-pooling (Figura 2.11).

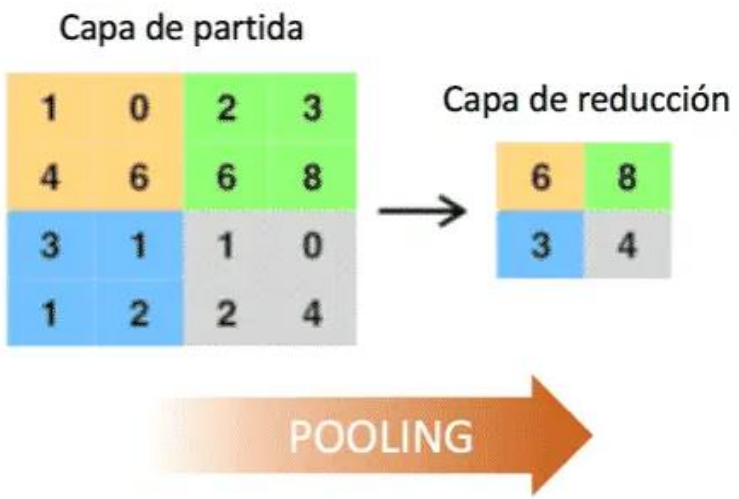

Figura 2.11.- Pooling (Calvo, 2017).

Si en esta capa se pierde mucha información puede que sea de gran beneficio para las RNC. Ayudan a reducir la complejidad, mejorar la eficiencia y a limitar el riesgo de sobreadaptación (Briega, 2019).

Capa clasificadora totalmente conectada

Al finalizar las capas convolucionales y de pooling, las redes utilizan generalmente capas completamente conectadas en la que cada píxel se considera como una neurona separada igual que en una red neuronal regular. En esta capa el número total de neuronas será igual al número de clases a predecir.

2.7 You Only Look Once (YOLO)

YOLO se implementó originalmente utilizando Darknet, que es una red neuronal de código abierto escrito en C y CUDA (Soto, 2017). YOLO es uno de los algoritmos orientado a objetos, este algoritmo es capaz de detectar y reconocer varios objetos en una imagen o en un conjunto de imágenes (video) en tiempo real, para realizar estas acciones YOLO emplea redes neuronales convolucionales. YOLO aplica una única red neuronal convolucional a la imagen completa lo que lo hace diferente de otros algoritmos detectores de objetos como R-CNN o su actualización Faster R-CNN (Bouchard, 2020). Esta red divide la imagen en regiones y predice los cuadros delimitadores y las probabilidades de cada región. Estos cuadros delimitadores se ponderan en función de las probabilidades predichas (Bouchard, 2020) (Soto, 2017). YOLO cuenta con distintas versiones, para este proyecto de tesis, se utilizó la versión 3 de YOLO (YOLOv3).

YOLOv3

YOLOv3 funciona utilizando tres técnicas (Grace, 20121):

Bloques residuales: La Imagen se divide en varias cuadrículas. Cada cuadrícula tiene una dimensión de S x S. En la Figura 2.12 se observa como una imagen de entrada se divide en muchas celdas de cuadrículas de igual dimensión. Cada celda de la cuadrícula detectará y será responsable de los objetos que aparecen dentro de ella.

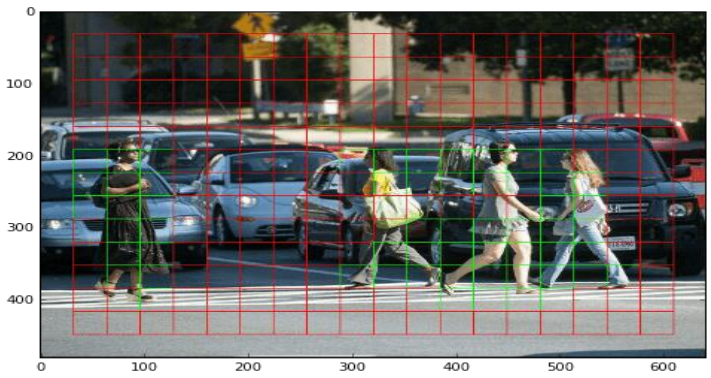

Figura 2.12.- Imagen de entrada divididas en muchas celdas (Grace, 2021).

Regresión de cuadro delimitador: Un cuadro delimitador es un contorno que resalta un objeto en una imagen. Cada cuadro delimitador de la imagen consta de los siguientes atributos: Ancho, altura, clase (como persona, automóvil, peatón, etc.) y centro del cuadro delimitador. YOLO utiliza una regresión de cuadro delimitador único para predecir la altura, el ancho, el centro y la clase de los objetos. En Figura 2.13 se muestra un ejemplo de un cuadro delimitador.

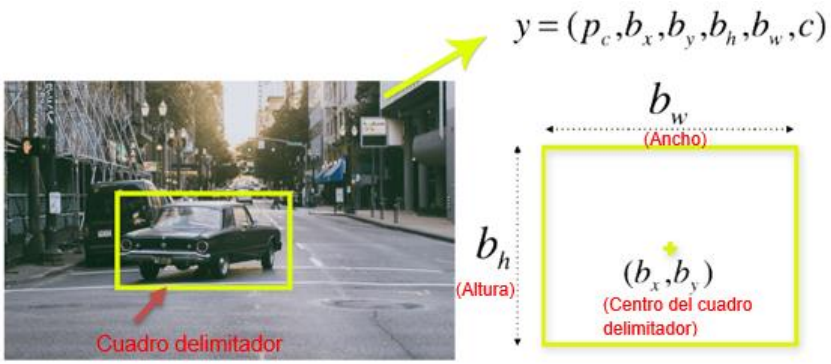

Figura 2.13.- Ejemplo de un cuadro delimitador (Grace, 2021).

Intersección sobre unión (IOU): La IOU en la detección de objetos se encarga de describir cómo se superponen los cuadros delimitadores. En la Figura 2.14 se observa un ejemplo de cómo funciona IOU, el cuadro delimitador azul es el cuadro provisto, el cuadro verde es el cuadro delimitador real. YOLO utiliza la IOU para proporcionar un cuadro delimitador de salida que rodea perfectamente los objetos.

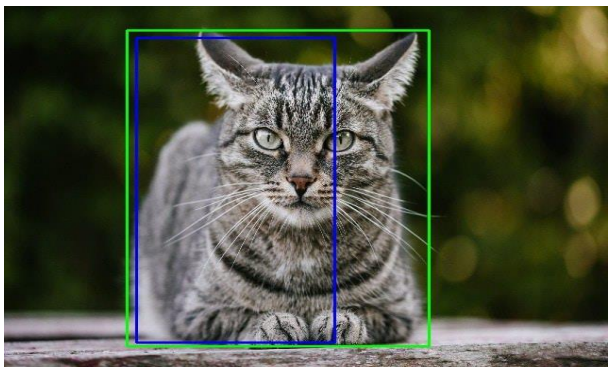

Figura 2.14.- Funcionamiento de la IOU (Grace, 2021).

En la Figura 2.15 se muestra cómo se aplican las tres técnicas para la obtención de los resultados finales de la detección, es decir, se observa el funcionamiento general del procesamiento de imágenes de YOLO.

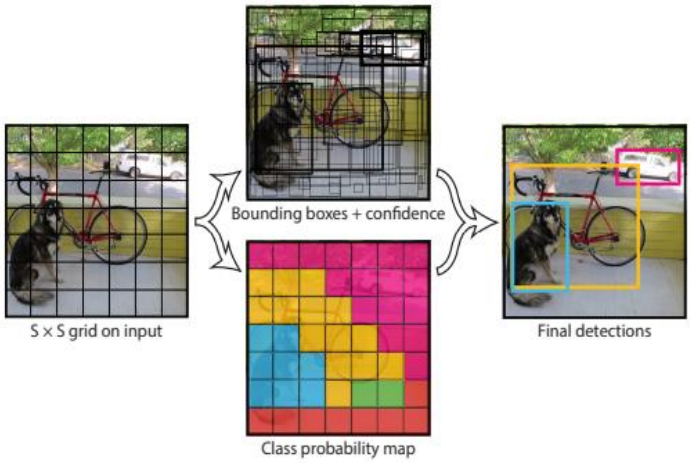

Figura 2.15.- Visión general del procesamiento de imágenes de yolo (Soto, 2017).

YOLOv3 está compuesto por 53 capas neuronales convolucionales, así recibiendo el nombre de Darknet-53, en Figura 2.16 se observa su arquitectura.

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Figura 2.16.- Arquitectura de Darknet-53 (Silva, 2020).

El algoritmo YOLOv3 se puede aplicar en (Grace, 2021):

- **Conducción autónoma:** Para detectar objetos que se encuentren alrededor de los automóviles como vehículos, personas, y señales de estacionamiento. La detección de objetos en los automóviles autónomos evita accidentes, ya que el automóvil no depende de un conductor humano.

- **Fauna:** YOLO se utiliza para poder detectar diferentes tipos de animales que habitan en los bosques o en diversos ecosistemas.
- **Seguridad**: Yolo también puede utilizarse en sistemas de seguridad para hacer cumplir la seguridad en una cierta área.

2.8 Python

Figura 2.17.- Logo de Python. *(Robledano, 2019).*

Python fue creado por el informático Guido Van Rossum a finales de los ochenta y es un lenguaje de programación de alto nivel, interpretado, orientado a objetos y de uso generalizado con una semántica dinámica integrada, para propósitos generales. Cuenta con una licencia de código abierto lo que permite ser utilizado en distintos sistemas operativos (Álvarez, 2003). Python es un lenguaje sencillo, legible y elegante. De esta manera se considera un lenguaje ideal para trabajar con grandes volúmenes de datos ya que, al ser un lenguaje versátil multiplataforma y multiparadigma, favorece su extracción y procesamiento. Se emplea en plataformas de alto tráfico, por ejemplo, Google, YouTube o Facebook. Actualmente Python es uno de los lenguajes más utilizados en inteligencia artificial y Ciencia de datos (Robledano, 2019).

2.9 OpenCV

Figura 2.18.- Logo de OpenCV

OpenCV es una biblioteca de código abierto de visión por computadora desarrollada por Intel, y está disponible para múltiples plataformas, como: Linux,

Windows, Android, Mac. También cuenta con soporte para diferentes lenguajes como, Python, Java, C, C + +, entre otros (de Programación, 2021). OpenCv al ser un producto con licencia Berkeley Software Distribution (BSD) permite ser usada libremente por empresas, grupos de investigación y por los organismos gubernamentales (García, 2015). OpenCv cuenta con más de 2500 algoritmos optimizados, lo que incluye un amplio conjunto de algoritmos por visión por computadora y aprendizaje autónomo con tecnología de última generación. Estos algoritmos se pueden utilizar para detectar, reconocer y clasificar rostros, animales y objetos, para extraer modelos 3D de objetos, encontrar imágenes similares de una base de datos de imágenes, unir imágenes para producir una alta resolución de imagen de una escena completa, etc (García, 2015).

2.11 PyTorch

Figura 2.19.- Logo de PyTorch, Intelligent, 2020):

PyTorch es una biblioteca de aprendizaje profundo de código abierto basado en python, PyTorch fue lanzado oficialmente en 2016 por un equipo del laboratorio de investigación de facebook, siendo una pieza fundamental para el desarrollo de relevantes aplicaciones de inteligencia artificial, como Autopilot de Tesla y el Pyro de Uber. Se basa en Torch, que está escrito en el lenguaje de programación Lua. Pytorch permite utilizar el código de python nativo y otras bibliotecas python como numpy (Chaudhary et al, 2020) (Intelligent, 2020).

PyTorch es uno de los framework de aprendizaje profundo más populares a nivel mundial por las siguientes razones (Intelligent, 2020):

- PyTorch está diseñado para integrarse perfectamente con Python y sus bibliotecas populares como NumPy el cual es fácil de aprender.
- PyTorch es un framework fácil de aprender a diferencia de otros de Deep Learning. Esto se debe a que su sintaxis y aplicación son similares a muchos lenguajes de programación convencionales como Python.

- Aunque PyTorch es un framework relativamente reciente, ha desarrollado muy rápidamente una comunidad dedicada de desarrolladores. Esto se debe a su documentación que está muy bien organizada y es útil para los principiantes.
- PyTorch está profundamente integrado en Python, hasta tal punto que muchas herramientas de depuración de Python se pueden utilizar fácilmente en el código de PyTorch.

2.12 Anaconda

Anaconda es una suite de código abierto que abarca una serie de aplicaciones y librerías y conceptos diseñados para el desarrollo de la ciencia de los datos de Python. Al trabajar con Anaconda permite crear diferentes entornos que permiten codificar en lenguaje Python python o en lenguaje R. estos entornos también son conocidos como Entornos de Desarrollo Integrados (IDE, por sus siglas en inglés). Los IDE contienen muchas funciones útiles para escribir, editar y depurar código, visualizar e inspeccionar datos, almacenar variables, presentar resultados y colaborar en proyectos (Rolon-Mérette, 2020).

2.13 LabelImg

LabelImg es una herramienta de anotación de imágenes gráficas, está escrita en python y Qt para su interfaz gráfica. La información de las anotaciones se guarda como archivos XML en formato PASCAL VOC, formato utilizado por Imagenet. También es compatible con los formatos YOLO y Create ML (tzutalin, 2015).

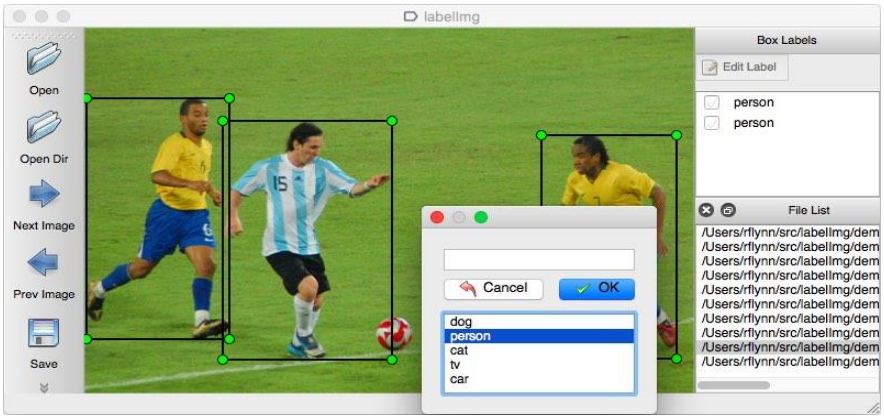

Figura 2.20.-LabelImg (tzutalin, 2022)

Capítulo III.- Diseño e implementación

En este capítulo se describen los requisitos necesarios para el desarrollo del sistema de rastreo de posición y tamaño de objetos acústicamente levitados. Es importante establecer las características que se buscan en cada uno de los requisitos antes de proceder al diseño, con el fin de obtener un producto que cumpla con todo lo mínimo requerido. Así mismo, se presenta la arquitectura física para mayor comprensión del funcionamiento del prototipo y se menciona cada una de las etapas del prototipo abarcando desde la recopilación de imágenes hasta el funcionamiento del sistema de rastreo.

3.1 Requisitos del sistema de rastreo de posición y tamaño de objetos acústicamente levitados.

A continuación, se describe e identifica cada uno de los requisitos del desarrollo del prototipo del sistema de rastreo:

3.1.1 Restricciones de diseño

Las restricciones de diseño son requisitos que limitan el desarrollo al crear el producto. Se etiquetan como RDx, siendo x el número del requisito.

RD1. You Only Look Once (YOLO)

YOLO es uno de los algoritmos de visión artificial orientado a objetos, este algoritmo es capaz de detectar, reconocer y clasificar diferentes objetos en una imagen o en un conjunto de imágenes en tiempo real, para realizar estas acciones YOLO emplea una red neuronal convolucional.

RD2. Python

Python es un lenguaje programación de alto nivel orientado a objetos con el cual se pueden desarrollar aplicaciones de todo tipo, es un lenguaje sencillo de aprender debido a la similitud que tiene con el lenguaje humano.

3.1.2 Requisitos funcionales nominales

Requisito para el funcionamiento del prototipo en situaciones normales. Se etiquetan como FN.x, siendo x el número de requisitos.

FN1. Reentrenamiento de la arquitectura YOLO con una clase (gota de agua y Objeto esférico)

Para el reentrenamiento del algoritmo de la arquitectura YOLOv3, es necesario recopilar 150 imágenes de cada una de las clases y con la ayuda del software Labellmg para obtener el archivo .txt de cada una de las imágenes.

FN2. Detección de gota de agua levitada

Se muestran en pantalla las coordenadas en pixeles de la posición y el tamaño del cuadro delimitador donde se encuentra la gota de agua levitada desde el momento que es detectada hasta el momento que el usuario suspende la ejecución del programa o el objeto se mueva de manera rápida.

FN2. Detección del objeto esférico

Se muestran en pantalla las coordenadas en pixeles de la posición y el tamaño del objeto esférico levitado desde el momento que es detectada hasta el momento que el usuario suspende la ejecución del programa.

3.1.3 Requisitos de calidad

Exigencias en la calidad que se piden explícitamente para el prototipo. En esta categoría se engloban los requisitos de rendimiento, accesibilidad y facilidad de uso. Se etiquetan como CA.x, siendo x el número de requisito.

CA1. Detección del objeto en tiempo real

El sistema de rastreo detectará objetos de las dos clases reentrenadas en tiempo real mediante una cámara web de bajo costo, al detectar el o los objetos que están siendo levitados en la terminal del software utilizado en este caso spider se mostrarán las coordenadas (x1, y1, x2, y2), el ancho y alto (los datos obtenidos son en pixeles) del cuadro delimitador donde se encuentra el objeto, estos datos son guardados en un archivo .txt.

3.1.4 Requisitos de evolución

Requisitos para el diseño del producto con el objetivo de facilitar la adaptación a exigencias o condiciones que puedan surgir en el futuro. Se etiquetan como EV.x, siendo x el número de requisito.

EV1. Ingresar nueva clase para que el sistema lo detecte

EV1. Ingresar nueva clase para que el sistema lo detecte

Se realiza un nuevo repositorio de imágenes, se crean sus archivos .txt en formato YOLO con ayuda del software LabelImg u otro software en el que se puedan trabajar en formato YOLO, y se vuelve a entrenar el sistema con las nuevas clases a detectar.

3.1.5 Requisitos de proyecto

Requisitos que afectan y condicionan el proceso de desarrollo del prototipo. Se etiquetan como PR.x, siendo x el número de requisito.

PR1. Tiempo de desarrollo

El tiempo de desarrollo del prototipo no debe de exceder a los 6 meses ya que es el tiempo establecido por la Universidad Politécnica del Estado de Morelos.

3.1.6 Requisitos de soporte

Requisitos que deben ser cumplidos por el cliente (a diferencia de los anteriores). Se etiquetan como SO.x, siendo x el número de requisito.

SO1. Contar con un levitador acústico uniaxial de arreglos de transductores

Para el desarrollo del prototipo es necesario contar con un levitador acústico por medio del cual se comprobará el funcionamiento y se obtendrán los resultados del sistema de rastreo de posición y tamaños de objetos acústicamente levitados.

SO2. Contar con los objetos de cada una de las clases (esfera y gota de agua)

Para lograr el rastreo de los objetos levitados es necesario contar con objetos esféricos y con una jeringa con agua para poder introducir la gota de agua en el levitador acústico.

SO3. Contar con una PC con un sistema operativo Windows

Para lograr el reentrenamiento de la arquitectura YOLOv3 es necesario contar con un sistema operación

3.2. Arquitectura física

Mediante esta arquitectura física se explica la parte física del prototipo del sistema de rastreo (Figura 3.1). El levitador acústico uniaxial se encuentra aproximadamente a 5 cm de distancia de la cámara web, esta distancia se obtuvo midiendo la distancia de la cámara al levitador acústico. La cámara web se conecta a un puerto USB del CPU de la computadora para ser controlada por el sistema de rastreo y así obtener la información requerida de los objetos levitados en tiempo real.

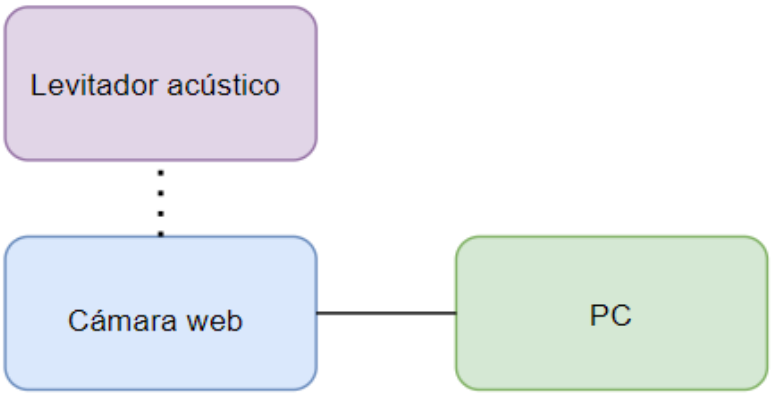

Figura 3.1.- Arquitectura física en bloques.

Levitador acústico, es donde se levitan los objetos que se requieren detectados por el sistema de rastreo.

Cámara web, por medio de la cámara se obtiene la entrada de video en tiempo real al sistema para que el objeto de interés sea detectado y así obtener la información de interés.

PC, es la encargada de tener el software que controlará al sistema de rastreo y en cuya terminal mostrará la información obtenida a través de la detección del objeto.

Para mayor comprensión del sistema de rastreo en la Figura 3.2 se observa detallada la arquitectura.

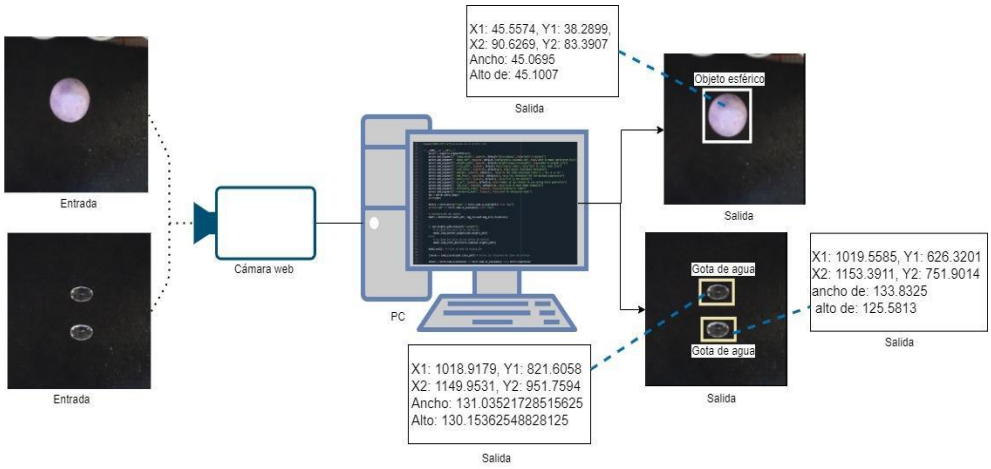

Figura 3.2.- Arquitectura Física más detallada.

Entrada, fotograma con objetos de cada una de las clases.

Salida, es el resultado obtenido a partir de la entrada al sistema de rastreo.

3.3 Etapas del sistema de rastreo

3.3.1. Etapa uno: Repositorio de imágenes a utilizar y creación de los archivos .txt en formato YOLO de cada imagen.

Se crea una carpeta con nombre custom, dentro de esa carpeta creamos dos carpetas más una con el nombre “images” y la otra con el nombre “labels”.

En la carpeta “images” se recopilan 300 imágenes en formato .jpg (150 imágenes por clase) que se utilizaran para el reentrenamiento de las dos clases (Gota de agua y Objeto esférico) a detectar como se observa en la Figura 3.3.

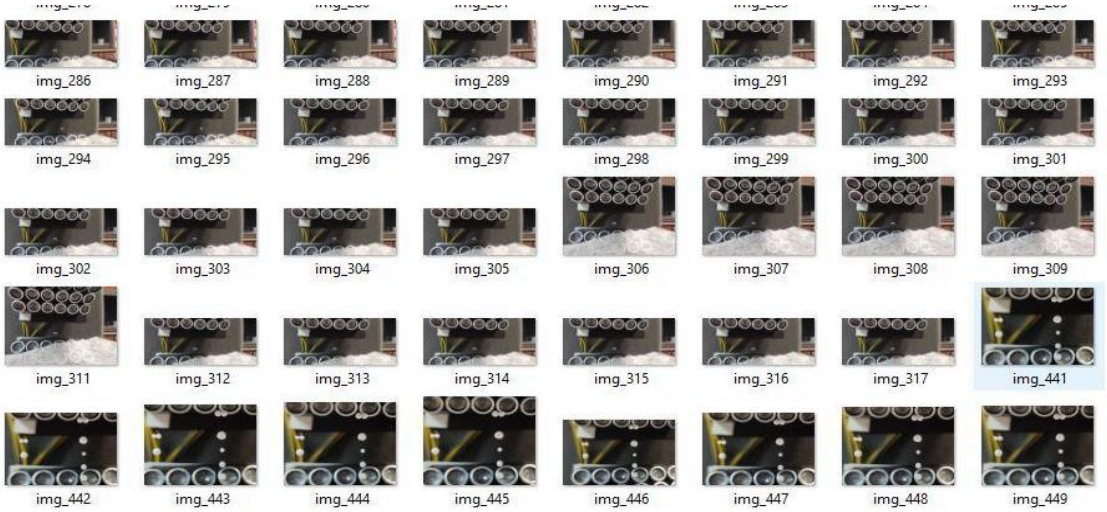

Figura 3.3.- Repositorio de las imágenes para el entrenamiento de la arquitectura YOLO

En el software LabelImg se realiza el etiquetado de las imágenes, es decir, se selecciona cada objeto de nuestro interés como se observa en las Figuras 3.4 y 3.5.

Ya dentro del editor visual:

- Se selecciona el directorio donde se encuentran las imágenes (cusmtom/images).
- Se selecciona el directorio donde se guardarán los archivos .txt (custom/labels).
- Se cambia el tipo de formato con el que se estará trabajando, en este caso se seleccionó el formato YOLO como se puede observar en recuadro verde de la Figura 3.4, porque se está trabajando con el algoritmo de la arquitectura YOLOv3.
- Se dibuja la caja (bounding-box) sobre la gota de agua o sobre el objeto esférico, se escribe el nombre de la clase a la que corresponde y se guarda como se puede observar en las Figuras 3.4 y 3.5. Este procedimiento se realiza en todas las imágenes.

Figura 3.4.- etiquetado de la clase Gota de agua

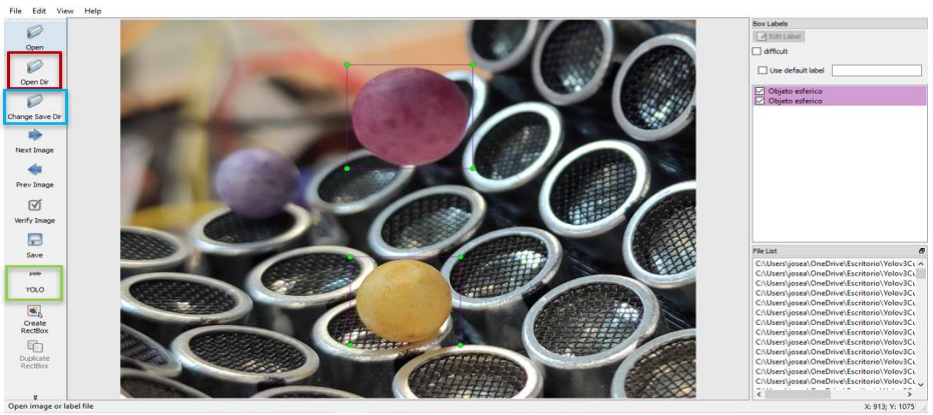

Figura 3.5.- Etiquetado de la clase Objeto esférico

En cada archivo .txt contiene información del número de la clase que corresponde al objeto seleccionado (en este caso el 0 da referencia a la clase gota de agua mientras que el 1 a la clase Objeto esférico) y las coordenadas donde esta etiquetado el objeto de la imagen real. En la Figura 3.6 se puede observar un ejemplo de dicha información, las coordenadas son obtenidas en píxeles.

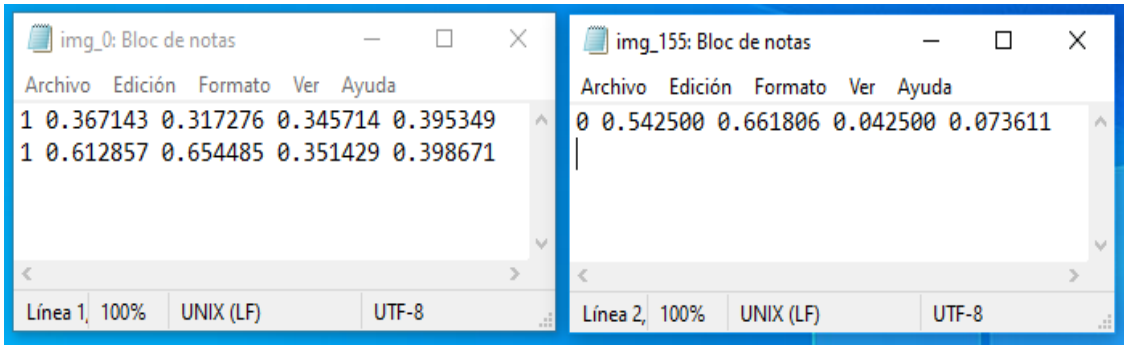

Figura 3.6.- Información obtenida en el etiquetado de los archivos .txt con formato YOLO del número de clase y las coordenadas donde se encuentran el o los objetos en la imagen

3.3.2 Etapa dos: Descarga del repositorio detección de objetos

Para llevar a cabo el reentrenamiento de YOLOv3 se hizo uso del repositorio detección de objetos en video de Alejandro Puig quien es un especialista en IA, el repositorio se encuentra en el siguiente enlace: https://github.com/puigalex/deteccion-objetos-video.

Se descarga el repositorio antes mencionado, el cual tendrá las siguientes carpetas y archivos como se observa en la Figura 3.7:

Nombre	Fecha de modificación	Tipo	Tamaño
config	29/07/2020 06:00 p. m.	Carpeta de archivos	
data	29/07/2020 06:00 p. m.	Carpeta de archivos	
utils	29/07/2020 06:00 p. m.	Carpeta de archivos	
weights	29/07/2020 06:00 p. m.	Carpeta de archivos	
.gitignore	29/07/2020 06:00 p. m.	Archivo de origen ...	2 KB
deteccion_video	29/07/2020 06:00 p. m.	Archivo de origen ...	6 KB
detect	29/07/2020 06:00 p. m.	Archivo de origen ...	6 KB
models	29/07/2020 06:00 p. m.	Archivo de origen ...	15 KB
README	29/07/2020 06:00 p. m.	Archivo de origen ...	5 KB
requirements	29/07/2020 06:00 p. m.	Documento de te...	1 KB
split_train_val	29/07/2020 06:00 p. m.	Archivo de origen ...	1 KB
test	29/07/2020 06:00 p. m.	Archivo de origen ...	4 KB
train	29/07/2020 06:00 p. m.	Archivo de origen ...	7 KB

Figura 3.7.- Carpetas y archivos que contiene la carpeta del repositorio descargado.

A continuación, se describe el funcionamiento de cada uno de los archivos y el contenido de cada una de las carpetas del repositorio que se utiliza para lograr el reentrenamiento de YOLOv3 con las dos clases:

Carpeta **config** contiene 4 archivos, coco.data, create_custom_model.sh, custom.data y yolov3.cfg, de estos archivos los únicos que se utilizaran para el reentrenamiento son create_custom_model.sh y custom.data. El archivo create_custom_model.sh genera un archivo .cfg que contiene la información sobre la red neuronal para correr las detecciones, es decir, nos ayuda a crear el modelo personalizado de acuerdo al número de clases que se desean reentrenar, y el archivo custom.data contiene el número de clases (ese número se modifica de acuerdo al número de clases que se están reentrenando) y el direccionamiento de los archivos train.txt, valid.txt y clases.names los cuales se crearán al momento de hacer el re-entrenamiento, más adelante, se hace mención sobre estos archivos.

En la carpeta **data** se encuentra una carpeta llamada custom que contiene información de las clases que el autor reentreno, es la razón por la cual se elimina y se reemplaza por la carpeta custom que se creó en la sección 3.1, en esta carpeta se crea el archivo classes.names que contiene el nombre de las dos clases tal y como se escribieron en el etiquetado (Gota de agua y Objeto esférico).

La carpeta **utils** contiene diferentes archivos .py, entre ellos se encuentra los archivos: _init_.py que es necesario para que la ejecución del código se realice de manera correcta, dataset.py su funcionalidad es gestionar el conjunto de datos, como la transformación y la carga de los datos para que sean consumidos de manera correcta por el sistema y utils.py este archivo contiene las funciones referentes para la carga de pesos de la red Darnet, también es el encargado de dibujar la salidas en las imágenes detectadas, así como dibujar la intersección de la unión IOU.

En la carpeta **weights** se encuentra dos archivos, un archivo con el nombre download_weights.sh que contiene el enlace de los pesos de la red neuronal del modelo de YOLOv3. En la referencia (Puig, 2021) el autor menciona que los pesos son los valores que tienen todas las conexiones entre las neuronas de la red neuronal YOLOv3. Estos modelos computacionalmente son muy pesados de entrenar desde cero al utilizar el modelo entrenado es una buena opción. También

en esa carpeta se encuentra un archivo con nombre download_darknet.sh donde se encuentra el enlace de la estructura de pesos YOLOv3 y así poder hacer transfer learning sobre los pesos.

El archivo **deteccion_video.py** contiene el algoritmo, con el cual se logra la detección de los objetos en video o en tiempo real.

El archivo **detect.py** realiza la misma función que el archivo anterior solo que con imágenes.

En el archivo **models.py** es donde se define la red Darnet y las funciones YOLO asociadas (cajas de detección y la función de pérdida) y así poder crear la red neuronal la cual es la base del detector de objetos.

El archivo **README.MD** contiene las indicaciones del autor con el fin de realizar el reentrenamiento de YOLOv3 correctamente del sistema de detección de objetos.

En el archivo **requirements.txt** se encuentran todas las librerías necesarias con sus respectivas versiones para el correcto funcionamiento de YOLOv3, solo que en las librerías torch y torchvision sus versiones fueron cambiadas por las más actuales.

El archivo **split_train_val.py** genera los archivos train.txt y val.txt en la carpeta custom, los cuales contienen la dirección de cada una de las imágenes, dividiéndolas entre el 90% (train.txt) y el 10% (val.txt) del total de todas las imágenes.

El archivo **test.py** ayuda a evaluar el rendimiento del sistema y a determinar el desempeño del entrenamiento o del sistema de detección de objetos.

Por otro lado, el archivo **train.py** es el código que se utiliza para el proceso de entrenamiento. En este código, se encuentra definida la arquitectura de la red neurona convolucional, así como los hiperparámetros utilizados.

3.3.3 Etapa tres: Reentrenamiento de la arquitectura YOLOv3

Creación del ambiente en anaconda prompt

1.-Se coloca la dirección donde se encuentra la carpeta del repositorio que anteriormente se descargó como se observa en la Figura 3.8 en el primer rectángulo amarillo.

2.- Se crea el ambiente en anaconda prompt (al trabajar con un ambiente en anaconda ayuda a tener orden con las paqueterías de python), el ambiente llevará por nombre *rentrenamientoyolo* el cual trabajará con la versión 3.6 de python, para lograr la creación del ambiente se ejecuta el siguiente comando:

conda create rentrenamientoyolo python=3.6

En el segundo rectángulo amarillo de la Figura 3.8 se observa que el ambiente en anaconda se creó con éxito.

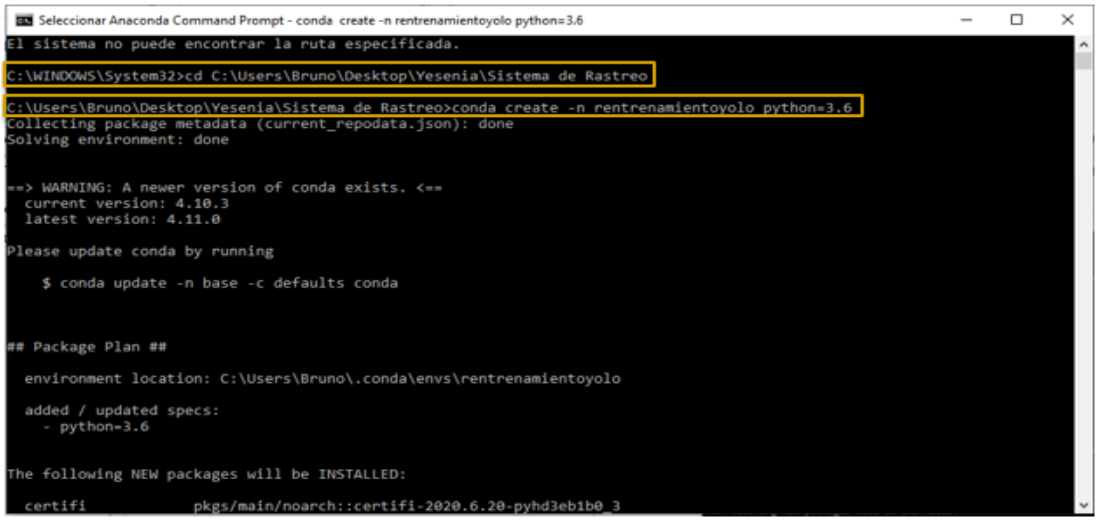

Figura 3.8.- Creación del ambiente en anaconda prompt

3.- Se activa el ambiente *rentrenamientoyolo* ejecutando conda activate rentrenamientoyolo (Figura 3.9), en el rectángulo rojo de la Figura 3.9 se observa que el ambiente se activó correctamente. Al activar el ambiente, se asegura que se está trabajando en el ambiente correcto al momento de realizar la instalación de todas las paqueterías necesarias para el funcionamiento del prototipo.

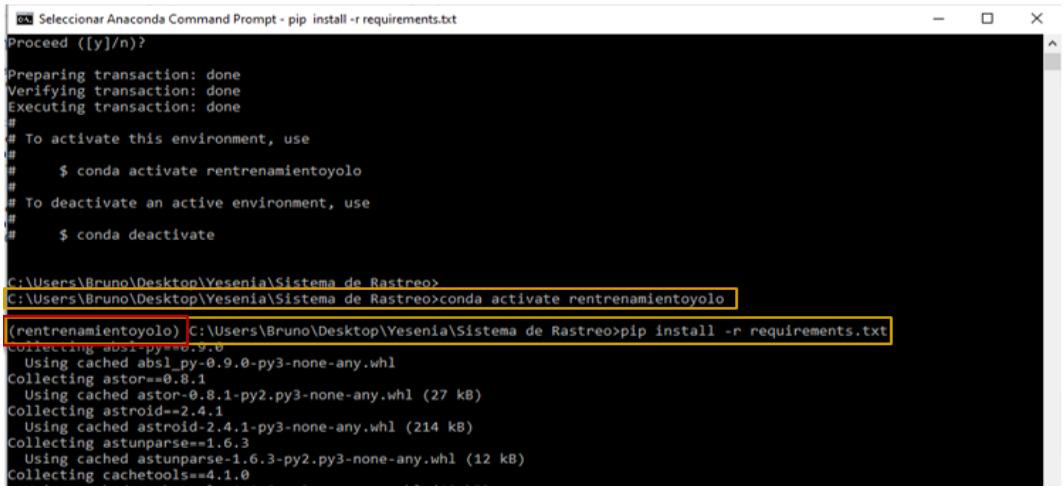

Figura 3. 9.- Activación del ambiente e instalación de todas las paqueterías necesarias

4.- Una vez dentro del ambiente se instalan las paqueterías ejecutando pip install -r requirements.txt (Figura 3.9).

Reentrenamiento de la arquitectura YOLOv3 con las dos clases (Gota de agua y Objeto esférico)

5.- Al ejecutar cd config se direcciona a la carpeta config, se ejecuta bash create_custom_model.sh para generar el archivo yolov3-custom2c.cfg y posteriormente se ejecuta cd.. para salir de la carpeta y regresar a la dirección de la carpeta principal (Figura 3.10).

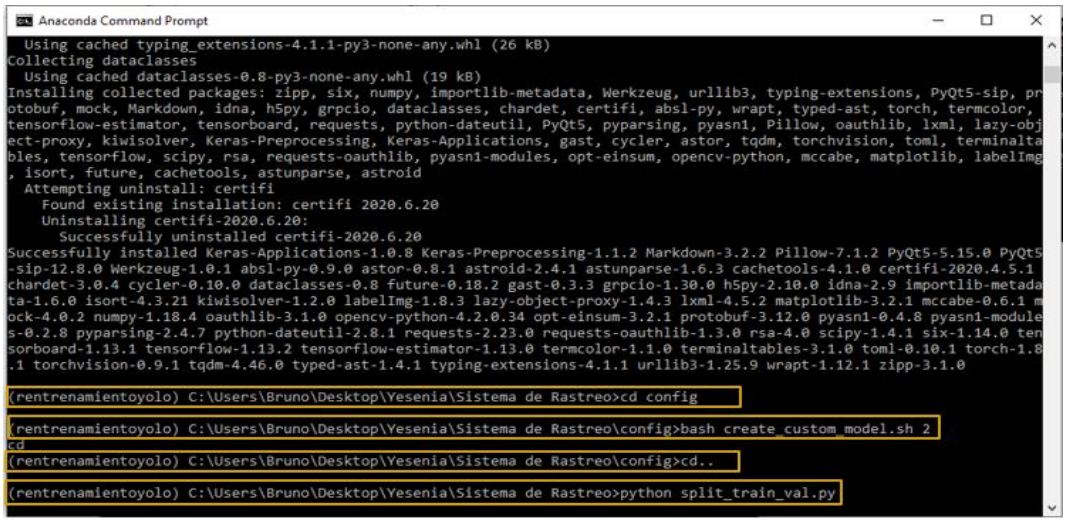

Figura 3.10.- Ejecución de los comandos del punto 5 y 6, para la creación de los archivos yolov3-custom2c.cfg, train.txt y val.txt.

6.- Ejecutamos python split_train_val.py para la generación de los archivos train.txt y val.txt.

7.- Ejecutamos python train.py –model_def config/yolov3-custom2c.cfg –data_config config/custom.data –pretrained_weights weights/darknet53.conv.74 –batch_size 6 para comenzar el reentrenamiento del sistema de detección de las dos clases (Gota de agua y Objeto esférico) (Figura 3.11).

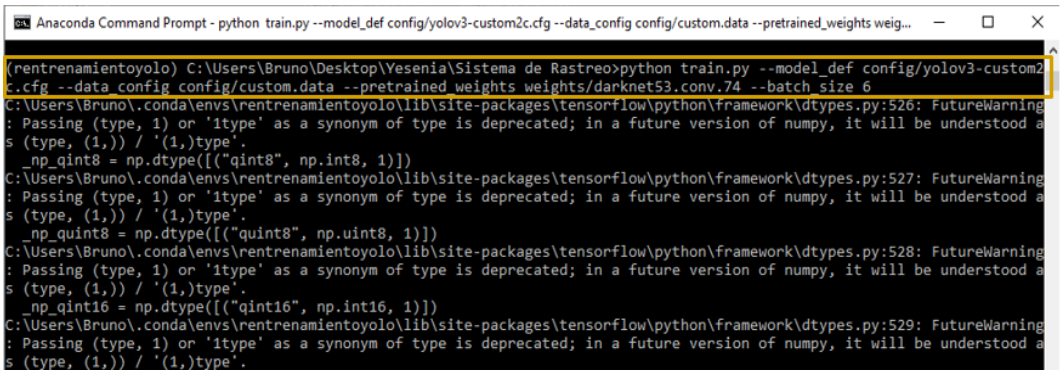

Figura 3.11.- Ejecución del comando del punto 7 para el reentrenamiento del sistema de detección de las dos clases (Gota de agua y Objeto esférico)

En la Figura 3.12 se observa que el sistema de detección de objetos ya inicializó el reentrenamiento y lo está realizando de manera correcta. También se observa en qué número de epoch (época) y batch (lote) se está ejecutando, cada vez que termina de ejecutar una época se genera un archivo .pth que contiene los pesos de la red neuronal de esa época en específico, el procedimiento se realizara 100 veces y todos los archivos generados se guardaran en la carpeta checkpoints que se genera al ejecutar el comando anterior.

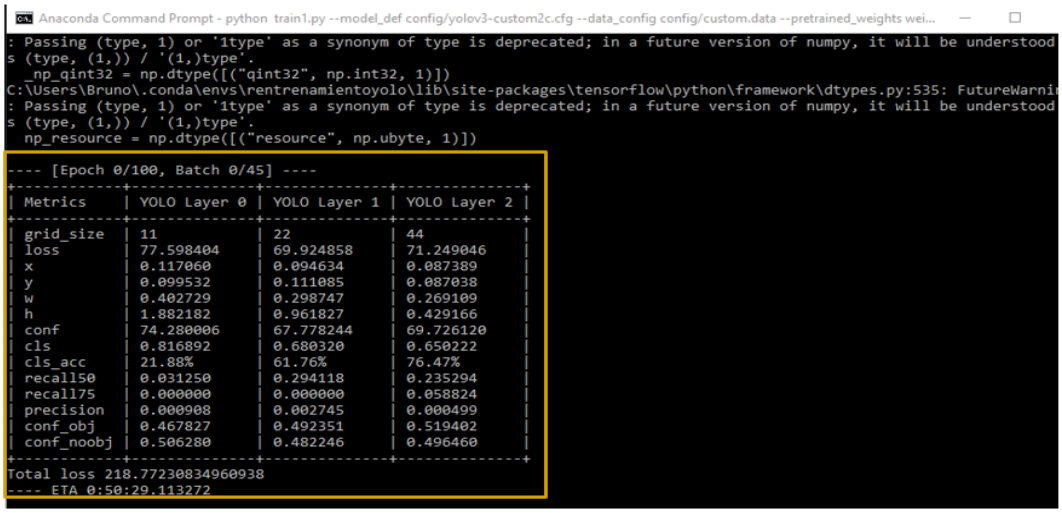

Figura 3.12.- Inicialización del reentrenamiento del sistema de detección de objetos.

8.- Al verificar que las 100 épocas hayan terminado ejecutamos python deteccion_video.py –model_def config/yolov3-custom2c.cfg –checkpoint_model checkpoints/yolov3_ckpt_99.pth –class_path data/custom/classes.names –weights_path checkpoints/yolov3_ckpt_99.pth –conf_thres 0.85. Este comando ejecuta el archivo deteccion_video.py para detectar objetos de las clases reentrenadas.

La instrucción:

- yolov3_ckpt_99.pth indica la época que se utilizara para comprobar el funcionamiento del sistema.
- conf_thres determina el número (del 0 al 1) de confianza del sistema al detectar un objeto de una de las clases entrenadas, entre más alto el número, mayor es la precisión del sistema y eso determina que el sistema está reentrenado correctamente.

9.- Al trabajar con el archivo yolov3_ckpt_99.pth y así lograr el funcionamiento de la manera correcta del sistema con las clases reentrenadas siempre se tendrá que ejecutar el comando del punto anterior desde el prompt de anaconda, para poder ejecutarlo desde cualquier software compatible con el lenguaje python y con anaconda, es necesario que en el archivo train.py cambiamos las líneas de código que se ven en la Figura 3.13 por las líneas de código de la Figura 3.14 para lo la conversión del archivo .pth a .weight.

```
55      # Initiate model
56      model = Darknet(opt.model_def).to(device)
57      model.apply(weights_init_normal)
58
59      # If specified we start from checkpoint
60      if opt.pretrained_weights:
61          if opt.pretrained_weights.endswith(".pth"):
62              model.load_state_dict(torch.load(opt.pretrained_weights))
63          else:
64              model.load_darknet_weights(opt.pretrained_weights)
65
```

Figura 3.13.- Líneas de código del archivo train.py original para generar los archivos .pth

```
55      # Initiate model
56      model = Darknet(opt.model_def).to(device)
57      model.apply(weights_init_normal)
58      model.load_state_dict(torch.load("checkpoints/yolov3_ckpt_44.pth", map_location=torch.device('cpu')))
59      Darknet.save_darknet_weights(model, 'weights/newYolov3.weights', cutoff=-1)
60
61      if opt.pretrained_weights:
62          if opt.pretrained_weights.endswith(".weights"):
63              model.load_darknet_weights(opt.pretrained_weights)
64          else:
65              model.load_state_dict(torch.load(opt.pretrained_weights))
66
```

Figura 3.14.- Líneas de código reemplazadas del archivo train.py original para lograr la conversión del archivo .pth a .weights

10.- Guardamos el documento train.py y se ejecuta el comando python train.py –model_def config/yolov3-custom.cfg –data_config config/custom.data (Figura 3.15), nuevamente se realiza el entrenamiento solo que esta vez con una época (Figura 3.16), al terminar el entrenamiento se genera un archivo con el nombre newyolov3.weights en la carpeta weights.

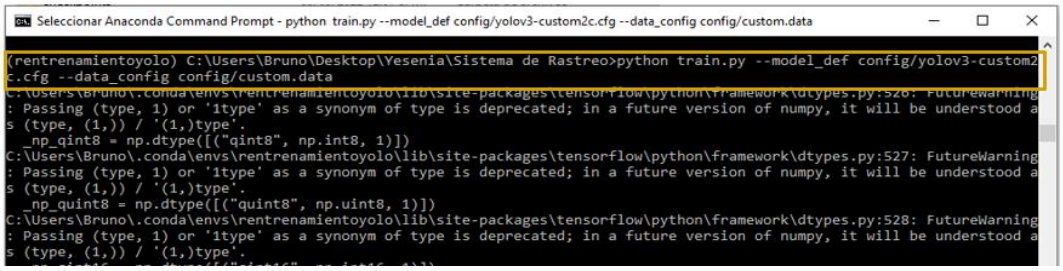

Figura 3.15.- Ejecución del comando python train.py –model_def config/yolov3-custom.cfg –data_config config/custom.data para la generación del archivo newyolov3.weights

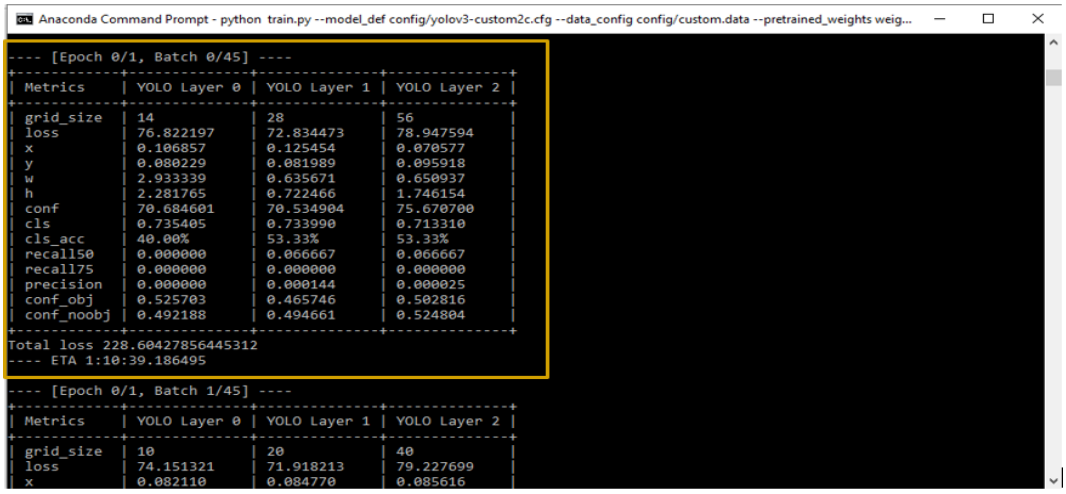

Figura 3.16.- Inicialización del reentrenamiento del sistema y así generar el archivo newyolov3.weights

11.- Por último, algunos de los parámetros del bloque if __name__ == "__main__" del archivo deteccion_video.py como se observan en la Figura 3.17 son

cambiados por los de la Figura 3.18, y así el sistema de rastreo se podrá ejecutar en el software de preferencia sin la necesidad del prompt de anaconda y poder realizar la detección de los objetos de las dos clases reentrenadas.

```
if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument("--image_folder", type=str, default="data/samples", help="path to dataset")
    parser.add_argument("--model_def", type=str, default="config/yolov3.cfg", help="path to model definition file")
    parser.add_argument("--weights_path", type=str, default="weights/yolov3.weights", help="path to weights file")
    parser.add_argument("--class_path", type=str, default="data/coco.names", help="path to class label file")
    parser.add_argument("--conf_thres", type=float, default=0.8, help="object confidence threshold")
    parser.add_argument("--webcam", type=int, default=1,  help="Is the video processed video? 1 = Yes, 0 == no" )
    parser.add_argument("--nms_thres", type=float, default=0.4, help="iou thresshold for non-maximum suppression")
    parser.add_argument("--batch_size", type=int, default=1, help="size of the batches")
    parser.add_argument("--n_cpu", type=int, default=0, help="number of cpu threads to use during batch generation")
    parser.add_argument("--img_size", type=int, default=416, help="size of each image dimension")
    parser.add_argument("--directorio_video", type=str, help="Directorio al video")
    parser.add_argument("--checkpoint_model", type=str, help="path to checkpoint model")
```

Figura 3.17.- Parámetros originales del bloque if del archivo deteccion_video.py

```
if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument("--image_folder", type=str, default="data/samples", help="path to dataset")
    parser.add_argument("--model_def", type=str, default="config/yolov3_custom2c.cfg", help="path to model definition file")
    parser.add_argument("--weights_path", type=str, default="weights/newyolov3.weights", help="path to weights file")
    parser.add_argument("--class_path", type=str, default="data/custom/classes.names", help="path to class label file")
    parser.add_argument("--conf_thres", type=float, default=0.8, help="object confidence threshold")
    parser.add_argument("--webcam", type=int, default=1,  help="Is the video processed video? 1 = Yes, 0 == no" )
    parser.add_argument("--nms_thres", type=float, default=0.4, help="iou thresshold for non-maximum suppression")
    parser.add_argument("--batch_size", type=int, default=1, help="size of the batches")
    parser.add_argument("--n_cpu", type=int, default=0, help="number of cpu threads to use during batch generation")
    parser.add_argument("--img_size", type=int, default=416, help="size of each image dimension")
    parser.add_argument("--directorio_video", type=str, help="Directorio al video")
    parser.add_argument("--checkpoint_model", type=str, help="path to checkpoint model")
```

Figura 3.18.- Parámetros nuevos del bloque if del archivo deteccion_video.py

Una vez realizados todos estos pasos, el sistema de rastreo estará listo para la realización de las pruebas necesarias para determinar si el funcionamiento está siendo correcto.

Capítulo IV.- Pruebas y resultados

En este capítulo se muestran las pruebas que se llevaron a cabo para la verificación del correcto funcionamiento del sistema de rastreo de posición y tamaño de objetos acústicamente levitados, y se presentan los resultados obtenidos a través de las pruebas aplicadas.

4.1 Pruebas del sistema de rastreo de posición y tamaño de un objeto levitado acústicamente

A fin de comprobar el desempeño del sistema de rastreo de objetos mediante el algoritmo de la arquitectura YOLOv3 se utilizaron fotografías, vídeos y video en tiempo real. Para cada una de las pruebas, se tomó en cuenta el parámetro conf_thres (umbral de confianza del objeto) que es uno de los parámetros más importantes para el desempeño del sistema, el cual toma valores de 0 a 1, donde 1 indica que el sistema tiene un funcionamiento ideal al detectar los objetos de interés.

Para lograr la ejecución de las pruebas, se utilizó una cámara web de bajo costo, un levitador acústico uniaxial con el que se levitan y trasladan los objetos y una computadora con sistema operativo Windows 10 con las siguientes características; procesador Intel® core™ i5-3550 CPU @ 3.30 GHZ de 64 bits, 8 GB de Memoria RAM.

En la Figura 4.1, se observa el espacio de trabajo y el levitador acústico que se utilizó para la realización de las pruebas, y así obtener los resultados del desempeño del sistema de rastreo de objetos.

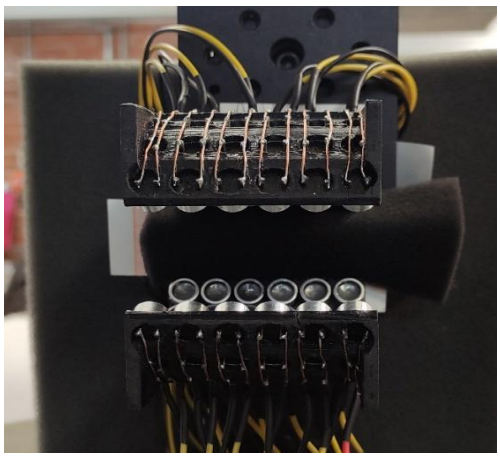

Figura 4.1.- Levitador acústico uniaxial

Las pruebas realizadas se dividieron en 2 secciones; 1) rastreo de las dos clases reentrenadas que corresponden a cuando los objetos están siendo levitados y 2) cuando están siendo trasladados, en ambos casos con diferente umbral de confianza.

4.1.1 Rastreo de las dos clases (Gota de agua y Objeto esférico) cuando los objetos están siendo levitados con diferente umbral de confianza.

En la Figura 4.2, se observan un fotograma de cada una de las clases. Estos fotogramas fueron extraídos del video en tiempo real tomado mediante la cámara web para representar la entrada al sistema de rastreo. Las pruebas se realizaron con diferente umbral de confianza (0.6, 0.85 y 0.95), entre más alto sea el umbral de confianza el sistema tiene un mejor desempeño.

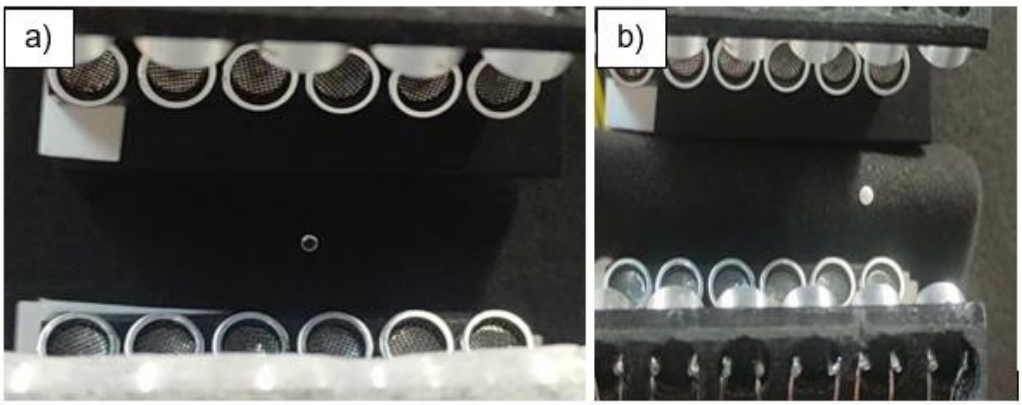

Figura 4.2 a) fotograma de la clase Gota de agua extraído del video en tiempo real, b) fotograma clase Objeto esférico extraído del video en tiempo real

Prueba 1. Umbral de confianza a 0.6

En la Figura 4.3 y Figura 4.4 se observa que los objetos son detectados, al trabajar con un umbral de confianza a 0.6 el sistema tiende a ser poco preciso, como se puede ver en la Figura 4.3 el sistema se confunde y crea doble cuadro delimitador en el objeto, como consecuencia la información obtenida es errónea. En la Figura 4.4 el sistema confunde los transductores por objetos esféricos, al realizar esa acción el sistema dará información de los objetos que no son solicitados por el usuario.

Figura 4.3.- Fotograma resultante de la clase Gota de agua con umbral de confianza de 0.6

En la Tabla 1, se visualiza la información obtenida de los cuadros delimitadores en píxeles de la Figura 4.3 al ser levitado.

Nota: solo se tomó en cuenta los tres primeros decimales de la información obtenida de los cuadros delimitadores para todas las tablas.

Tabla 1.- Información de los cuadros delimitadores de la Figura 4.3

N. de la clase	Gota de agua	Gota de agua
X1	545.992	528.842
Y1	553.808	563.460
X2	581.680	587.460
Y2	587.563	613.143
Ancho	35.687	58.618
Alto	33.755	49.682

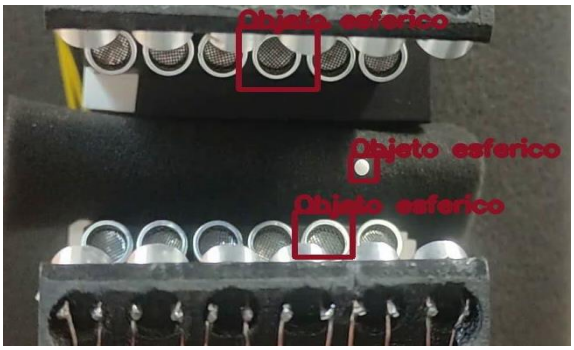

Figura 4.4.- Fotograma resultante de la clase Objeto esférico con umbral de confianza de 0.6

En la Tabla 2, se visualiza la información obtenida de los cuadros delimitadores en píxeles de la Figura 4.4 al ser levitado.

Tabla 2.- Información de los cuadros delimitadores de la Figura 4.4

N. de la clase	Objeto esférico	Objeto esférico	Objeto esférico
X1	594.214	530.050	658.549
Y1	533.761	276.568	455.679
X2	660.781	620.867	688.880
Y2	596.848	363.211	490.986
Ancho	66.567	90.816	30.331
Alto	63.087	86.643	35.306

Prueba 2. Umbral de confianza a 0.85

En la Figura 4.5 y 4.6 se observa que los objetos son detectados, a pesar de trabajar con un umbral de confianza a 0.85 el sistema tiende a ser poco preciso al detectar a los objetos de la clase gota de agua con dicho umbral, como se observa en la Figura 4.5 el sistema se confunde fácilmente y crea doble cuadro delimitador en la gota de agua, sin embargo, para la clase objeto esférico el sistema es más preciso (Figura 6).

Figura 4.5.- Fotograma de salida de la clase Gota de agua con umbral de confianza de 0.85

En la Tabla 3 se visualiza la información obtenida de los cuadros delimitadores en píxeles de la Figura 4.5 al ser levitado.

Tabla 3.- Información de los cuadros delimitadores de la Figura 4.5

N. de la clase	Gota de agua	Gota de agua
X1	529.867	551.243
Y1	564.680	551.222
X2	588.556	586.262
Y2	614.100	585.982
Ancho	58.689	35.018
Alto	49.420	34.759

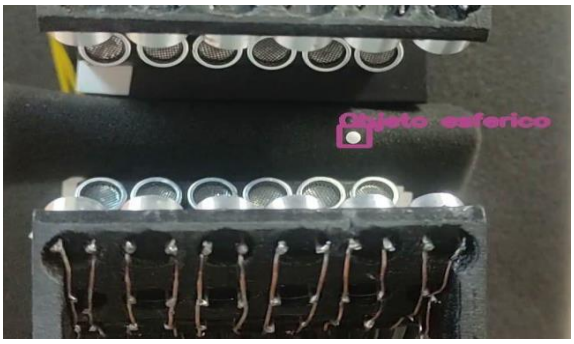

Figura 4.6.- Fotograma de salida de la clase Objeto esférico con umbral de confianza de 0.85

En la Tabla 4 se visualiza la información obtenida del cuadro delimitador en píxeles de la Figura 4.6 al ser levitado.

Tabla 4.- Información del cuadro delimitador de la Figura 4.6

N. de la clase	Objeto esférico
X1	650.859
Y1	451.712
X2	684.387
Y2	488.774
Ancho	33.528
Alto	37.061

Prueba 3. Umbral de confianza a 0.95

En las Figuras 4.7 y 4.8 se observa que los objetos son detectados, al trabajar con un umbral de confianza a 0.95 el sistema tiende a ser más preciso al detectar a los objetos. Al momento de realizar las pruebas con este número de umbral de confianza el sistema no confundió a los traductores por objetos esféricos y no creo cuadros delimitadores falsos como se puede observar en las Figuras 4.4 y 4.5, pero en ocasiones no lograba detectar al objeto.

Figura 4.7.- Fotograma de salida de la clase Gota de agua con umbral de confianza de 0.95

En la Tabla 5 se visualiza la información obtenida del cuadro delimitador en píxeles de la Figura 4.7 al ser levitado.

Tabla 5.- Información del cuadro delimitador de la Figura 4.7

N. de la clase	Gota de agua
X1	545.906
Y1	554.100
X2	581.990
Y2	587.950
Ancho	36.084
Alto	33.850

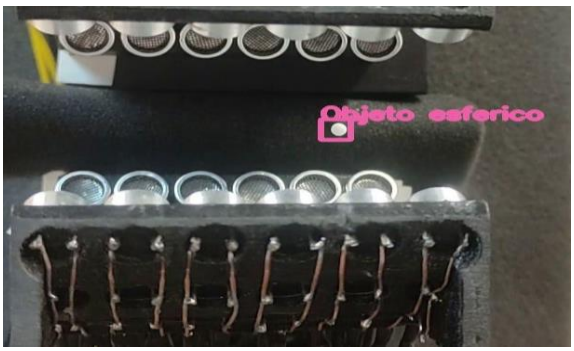

Figura 4.8.- Fotograma de salida de la clase Objeto esférico con umbral de confianza de 0.95

En la Tabla 6 se visualiza la información obtenida del cuadro delimitador en píxeles de la Figura 4.8 al ser levitado.

Tabla 6.- Información del cuadro delimitador de la Figura 4.8

N. de la clase	Objeto esférico
X1	635.180
Y1	454.665
X2	667.375
Y2	490.808
Ancho	32.195
Alto	36.143

4.1.2 Rastreo de datos de las dos clases (Gota de agua y Objeto esférico) con diferente umbral de confianza de objetos acústicamente trasladados.

En la Figura 4.2 se observa el fotograma de cada una de las clases, estos fotogramas fueron extraídos del video en tiempo real tomado mediante la cámara

web para representar la entrada al sistema de rastreo. Las pruebas se realizaron con diferente umbral de confianza (0.6, 0.85 y 0.95), entre más alto sea el umbral de confianza el sistema tiene un mejor desempeño.

Prueba 1. Umbral de confianza a 0.6

En las Figuras 4.9 y 4.10 se observa que los objetos son detectados, al trabajar con un umbral de confianza a 0.6 el sistema tiende a ser poco preciso. En la Figura 4.10 el sistema confunde los transductores por objetos esféricos, también crea cuadros delimitadores demasiados grandes tanto al detectar al objeto como a los transductores, por lo cual se obtendrá información errónea, en cambio para la clase gota de agua si la detecta satisfactoriamente.

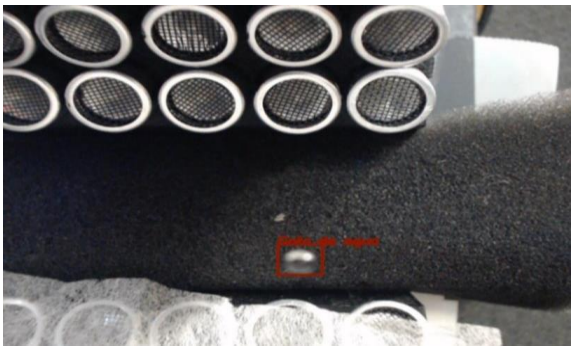

Figura 4.9.- Fotograma de salida de la clase Gota de agua con umbral de confianza de 0.6

En la Tabla 7 se visualiza la información obtenida del cuadro delimitador en píxeles de la Figura 4.9 al ser trasladado.

Tabla 7.- Información del cuadro delimitador de la Figura 4.9

N. de la clase	Gota de agua
X1	586.193
Y1	673.969
X2	681.373
Y2	759.343
Ancho	95.179
Alto	85.3739

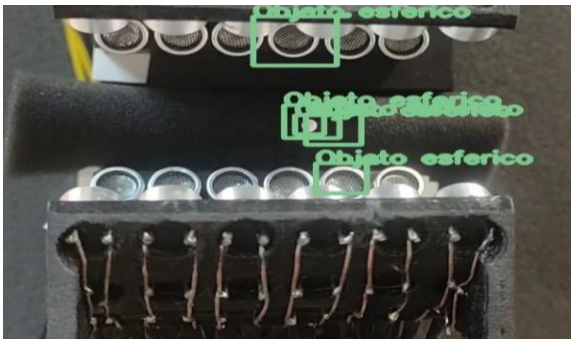

Figura 4.10.- Fotograma de salida de la clase Objeto esférico con umbral de confianza de 0.6

En la Tabla 8 se visualiza la información obtenida de los cuadros delimitadores en píxeles de la Figura 4.10 al ser trasladada.

Tabla 8.- Información de los cuadros delimitadores de la Figura 4.10

N. de la clase	Objeto esférico	Objeto esférico	Objeto esférico
X1	529.269	587.179	598.314
Y1	275.446	450.125	538.458
X2	621.544	648.130	654.610
Y2	363.802	505.453	592.897
Ancho	92.275	60.950	56.295
Alto	88.355	55.328	54.439

Prueba 2. Umbral de confianza a 0.85

En las Figuras 4.11, 4.12 y 4.13 se observa que los objetos son detectados, al trabajar con un umbral de confianza a 0.85 el sistema no se confunde fácilmente. Sin embargo, al detectar la gota de agua el cuadro delimitador se ajusta a su tamaño, pero cuando se trata del objeto esférico el cuadro delimitador es más grande por lo cual se obtendrá información errónea. En la Figura 4.13 se observa que al ser trasladado el objeto el sistema crea doble cuadro delimitador, así mismo también se obtiene información errónea.

Figura 4.11.- Fotograma de salida de la clase Gota de agua con umbral de confianza de 0.85

En la Tabla 9 se visualiza la información obtenida del cuadro delimitador en píxeles de la Figura 4.11 al ser trasladado.

Tabla 9.- Información del cuadro delimitador de la Figura 4.11

N. de la clase	Gota de agua
X1	540.097
Y1	558.311
X2	609.587
Y2	619.709
Ancho	69.489
Alto	61.397

Figura 4.12.- Fotograma de salida de la clase Objeto esférico con umbral de confianza de 0.85

En la Tabla 10 se visualiza la información obtenida del cuadro delimitador en píxeles de la Figura 4.12 al ser trasladado.

Tabla 10.- Información del cuadro delimitador de la Figura 4.12

N. de la clase	Objeto esférico
X1	570.220
Y1	442.023
X2	618.210
Y2	489.649
Ancho	47.990
Alto	47.626

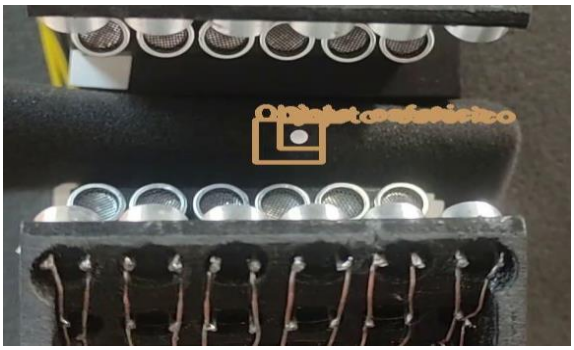

Figura 4.13.- Fotograma de salida de la clase Objeto esférico con umbral de confianza de 0.85

En la Tabla 11, se visualiza la información obtenida de los cuadros delimitadores en píxeles de la Figura 4.13 al ser trasladada.

Tabla 11.- Información de los cuadros delimitadores de la Figura 4.13

N. de la clase	Objeto esférico	Objeto esférico
X1	566.547	541.103
Y1	446.816	443.759
X2	609.300	609.052
Y2	489.417	511.208
Ancho	42.753	67.948
Alto	42.600	67.448

Prueba 3. Umbral de confianza a 0.95

En la Figura 4.14 y 4.15 se observan objetos detectados con un umbral de confianza a 0.95. Con estas condiciones el sistema de rastreo tiende a ser más preciso que en los casos previos. En las pruebas realizadas con este umbral de confianza el sistema no confundió a los traductores por objetos levitados como se puede observar en la Figuras 410, pero en ocasiones no lograba detectar al objeto en cada fotograma, y en otras ocasiones tardaba entre 1 s y 2 s para detectar al objeto durante su traslación.

Figura 4.14.- Fotograma de salida de la clase Gota de agua con umbral de confianza de 0.95

En la Tabla 12 se visualiza la información obtenida del cuadro delimitador en píxeles de la Figura 4.14 al ser trasladado.

Tabla 12.- Información del cuadro delimitador de la Figura 4.14

N. de la clase	Gota de agua
X1	444.500
Y1	623.485
X2	504.456
Y2	674.359
Ancho	59.956
Alto	50.874

Figura 4.15.- Fotograma de salida de la clase Objeto esférico con umbral de confianza de 0.95

En la Tabla 13 se visualiza la información obtenida del cuadro delimitador en píxeles de la Figura 4.15 al ser trasladado.

Tabla 13.- Información del cuadro delimitador de la Figura 4.15

N. de la clase	Objeto esférico
X1	576.478
Y1	450.432
X2	613.936
Y2	488.784
Ancho	37.458
Alto	38.351

Capítulo V.- Conclusiones y trabajos futuros

5.1 Conclusiones

YOLOv3 es un algoritmo de detección de objetos más rápido y preciso a comparación de otros algoritmos detectores, puede correr hasta 45 cuadros (fotogramas) por segundo (FPS). Para lograr el reentrenamiento del algoritmo de la arquitectura YOLOv3 se utilizó un equipo de cómputo con un sistema operativo Windows 10, una memoria RAM de 8GB y procesador intel® core ™ i5, se intentó hacer el reentrenamiento en otro equipo de cómputo, pero fue demasiado lento el reentrenamiento. Posterior al reentrenamiento de las dos clases de interés (Gota de agua y Objeto esférico) se realizaron pruebas para verificar si el sistema de rastreo de posición y tamaño de un objeto acústicamente levitado implementado era capaz de detectar los objetos con su respectiva clase. Las pruebas se realizaron con umbrales de confianza distintos, se observó que si el umbral de confianza es más cercano a 1 el rastreo es más preciso, pero si es cercano al 0 el rastreo tiende a no ser preciso, lo que provoca que el sistema confunda a los transductores por objetos esféricos o crea más de un cuadro delimitador en el mismo objeto, como se puede observar en la sección IV.- Pruebas y resultados cuando el umbral es de 0.65 en ambos casos el sistema crea doble cuadros delimitadores, cuadro delimitadores falsos o confunde a los transductores por objetos esféricos, sin embargo al tener un umbral de 0.95 tiende a ser más preciso.

En conclusión, la hipótesis no se cumple a su totalidad por lo antes mencionado, pero no se descarta la posibilidad que al reentrenar el sistema con más imágenes y épocas se logre su obtener un mejor desempeño. Desafortunadamente, por falta de tiempo, no se pudo volver a hacer el reentrenamiento con más de 150 imágenes de cada clase y comprobar el modelo tendría un mejor desempeño.

5.2Trabajos a futuro

A continuación, se expresan algunos de los trabajos futuros del prototipo sistema de rastreo de posición y tamaño de objetos acústicamente levitados.

1. Reentrenamiento de nuevas clases.

   El prototipo cuenta con dos clases reentrenadas (Gota de agua y Objeto esférico) pero en el Laboratorio de Óptica Aplicada (LOA) del Instituto de Ciencias Físicas (ICF) se trabaja con distintos objetos, para que el sistema pueda detectarlos a todos los objetos se tiene que hacer nuevamente el reentrenamiento del sistema de rastreo de objetos, clasificándolos con el nombre de cada objeto de interés o de acuerdo a la forma del objeto, por ejemplo, redondo, cuadrado o triangular.

2. Interfaz gráfica

   Al desarrollar una interfaz al sistema de rastreo hará que el sistema tenga control al momento de ser ejecutado, también permitirá comunicarse con una cámara de buena calidad y así obtener mejores resultados. Debido al tiempo de desarrollo del prototipo y al no contar con el equipo necesario de cómputo no se logró la implementación de una interfaz gráfica.

Bibliografía

Altúzar Meza, R. (2017). *Transductores*.

Alvaréz, M. (2003). Qué es Python. Desarrolloweb. Retrieved from https://desarrolloweb.com/articulos/1325.php

Andrade, M. A. B., Marzo, A., & Adamowski, J. C. (2020). Acoustic levitation in mid-air: Recent advances, challenges, and future perspectives. *Applied Physics Letters*, *116*(25). https://doi.org/10.1063/5.0012660

Andrade, M. A. B., Pérez, N., Adamowski, J. C. (2018). Review of Progress in Acoustic Levitation. In Brazilian Journal of Physics (Vol. 48, Issue 2, pp. 190–213). Springer New York LLC. https://doi.org/10.1007/s13538-017-0552-6

Andrade, M. A. B., Okina, F. T. A., Bernassau, A. L., & Adamowski, J. C. (2017). Acoustic levitation of an object larger than the acoustic wavelength. *The Journal of the Acoustical Society of America*, *141*(6), 4148–4154. https://doi.org/10.1121/1.4984286

Barrios, J. (2019, 18 junio). Redes neuronales convolucionales son un tipo de redes neuronales. https://www.juanbarrios.com/redes-neurales-convolucionales/

Barrionuevo, L. (2009, March). *Repositorio*. http://glossarium.bitrum.unileon.es/Home/repositorio/repository

Briega, L. (2019). Visión por computadora. https://iaarbook.github.io/vision-por-computadora/#redes-neuronales-convolucionales

Brita. (2021). Visión por Computadora o Visión Computación: Guía 2021. https://brita.mx/vision-por-computadora-o-vision-computacion-guia-2021

Chaudhary, A., Chouhan, K. S., Gajrani, J., & Sharma, B. (2020). Deep Learning With PyTorch (pp. 61–95). https://doi.org/10.4018/978-1-7998-3095-5.ch003

Contreras, V. (2021). *Desarrollo de levitadores acústicos uniaxiales*.

Cuamatzi, X., Jimenéz, M. A., Navarrete, F. J. (2010). SISTEMA DE PROXIMIDAD DE PROXIMIDAD ULTRASÓNICO.

Das, P. (2020). The 5 Most Amazing Computer Vision Techniques to Learn. https://www.analyticsinsight.net/the-5-most-amazing-computer-vision-techniques-to-learn/

Folkmann, C. Á. (2019/2020). *LEVITADOR ACÚSTICO MULTIEMISOR NO RESONANTE DE EJE ÚNICO Mantenimiento electrónico S21ME*.

García Barajas, Alberto (2015). *Instituto Politécnico Nacional unidad profesional interdisciplinaria de ingeniería "Prototipo de sistema para seguimiento de objetos con visión artificial.*

Gordillo, J. S., Guaraca, L. E. (2015, enero). "Determinación de niveles de presión sonora (nps) generados por las aeronaves, en el sector sur del aeropuerto mariscal lamar de la ciudad de cuenca".

Grace K. (2021). Introduction to YOLO Algorithm for Object Detection. https://www.section.io/engineering-education/introduction-to-yolo-algorithm-for-object-detection/

Hedberg, C., & Ghazisaeidi, H. (2006). Variation of High Power Air Transducer.

IBM. (2020). ¿Qué es la visión artificial?. https://www.ibm.com/topics/computer-vision

Intelligent. (2020). ¿Conoces PyTorch? La herramienta de open source con la que puedes crear redes neuronales | ITELLIGENT INFORMATION TECHNOLOGIES. https://itelligent.es/es/conoces-pytorch-herramienta-open-source-la-puedes-crear-redes-neuronales/

Juárez Álvarez, Leonardo. (2018). "Reconocimiento de objetos y rostros con técnicas de visión por computadora e inteligencia artificial" Informe de la Práctica de Entrenamiento Industrial Nombre de la Empresa o Institución: CIDESI Presenta

Marzo, A., Barnes, A., & Drinkwater, B. W. (2017). TinyLev: A multi-emitter single-axis acoustic levitator. *Review of Scientific Instruments*, *88*(8). https://doi.org/10.1063/1.4989995

Molleda Meré, Julio. (2009). Técnicas de visión por computador para la reconstrucción en tiempo real de la forma 3D de productos laminados. Biblioteca de la Universidad de Oviedo.

Morris, R. H., Dye, E. R., Docker, P., & Newton, M. I. (2019). Beyond the Langevin horn: Transducer arrays for the acoustic levitation of liquid drops. In Physics of Fluids (Vol. 31, Issue 10). American Institute of Physics Inc. https://doi.org/10.1063/1.5117335

Peralta, J. C. (2018, junio). *Espectroscopia de rompimiento inducido por láser en gotas acústicamente levitadas*.

Puig, A. (2021). *puigalex/deteccion-objetos-video: Detección de objetos sobre video usando PyTorch*. https://github.com/puigalex/deteccion-objetos-video

Reinaldo. (2008, 16 mayo). Sensor Ultrasónico: Diseño de un sistema de medición basado en un Sensor Ultrasónico. http://sensorultrasonico.blogspot.com/2008/05/diseo-de-un-sistema-de-medicion-basado.html

Robledano, Á. (2019). Qué es Python: Características, evolución y futuro | OpenWebinars. https://openwebinars.net/blog/que-es-python/

Rolon-Mérette, D., Ross, M., Rolon-Mérette, T., & Church, K. (2020). Introduction to Anaconda and Python: Installation and setup. The Quantitative Methods for Psychology, 16(5), S3–S11. https://doi.org/10.20982/tqmp.16.5.s003

Smoot, J. (2021). *The Basics of Ultrasonic Sensors | CUI Devices*. https://www.cuidevices.com/blog/the-basics-of-ultrasonic-sensors

tzutalin. (2015). *tzutalin/labelImg: LabelImg is a graphical image annotation tool and label object bounding boxes in images*. https://github.com/tzutalin/labelImg

Valencia, R. A. (2017). *Sistema de levitación ultrasónica para análisis espectroscópico de muestras líquidas Como Requisito para obtener el grado de: Maestro en Optomecatrónica*.

Montoya, J.E., Cruz Henao, C.C. (2013). Visión por computador aplicado al módulo de inspección de la célula minitek de la facultad de ingeniería industrial de la universidad tecnológica de pereira universidad tecnológica de pereira facultad de ingenierias electrica, electronica, fisica y ciencias de la computación ingeniería de sistemas y computación.

Williams, M. (2021, 26 enero). Fundamentos de Acústica: Presión Sonora, Potencia Sonora e Intensidad Sonora | A escala. https://onscale.com/blog/fundamentals-of-acoustics-sound-pressure-sound-power-and-sound-intensity