INSTITUTO TECNOLÓGICO DE LA PAZ
DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN
MAESTRÍA EN SISTEMAS COMPUTACIONALES

DESARROLLO DE UN PROTOTIPO DE VEHÍCULO AUTÓNOMO CON RECONOCIMIENTO Y APROXIMACIÓN HACIA BOTELLAS DE PLÁSTICO EN AMBIENTES CONTROLADOS

QUE PARA OBTENER EL GRADO DE
MAESTRO EN SISTEMAS COMPUTACIONALES

PRESENTA:
DIEGO VAZQUEZ LUCERO

DIRECTORES DE TESIS:
M.C. JORGE ENRIQUE LUNA TAYLOR

LA PAZ, BAJA CALIFORNIA SUR, MÉXICO, AGOSTO 2023.

La Paz, B.C.S., 11/ AGOSTO /2023

DEPI_MSC/055/2023

ASUNTO: Autorización de impresión

C. DIEGO VÁZQUEZ LUCERO,
ESTUDIANTE DE LA MAESTRÍA EN SISTEMAS COMPUTACIONALES,
P R E S E N T E .

Con base en el dictamen de aprobación emitido por el Comité Tutorial de la Tesis denominada: **"DESARROLLO DE UN PROTOTIPO DE VEHÍCULO AUTÓNOMO CON RECONOCIMIENTO Y APROXIMACIÓN HACIA BOTELLAS DE PLÁSTICO EN AMBIENTES CONTROLADOS"**, mediante la opción de tesis (Proyectos de Investigación), entregado por usted para su análisis, le informamos que se **AUTORIZA** la impresión.

A T E N T A M E N T E
Excelencia en Educación Tecnológica®

JUDITH GUADALUPE MARTÍNEZ TIRADO,
JEFA DE LA DIV. DE ESTUDIOS DE POSGRADO E INV.

EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA
TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE LA PAZ
DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

c.c.p. Depto. de Servicios Escolares
c.c.p. Archivo.

*JGMT/icl**

DICTAMEN DEL COMITÉ TUTORIAL

La Paz, B.C.S., **10/AGOSTO/ 2023**

JUDITH GUADALUPE MARTÍNEZ TIRADO,
JEFA DE LA DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN,
P R E S E N T E.

Por medio del presente, enviamos a usted dictamen del Comité Tutorial de tesis para la obtención del grado de Maestro, con los siguientes datos generales:

No. de Control	Nombre
M21310005	DIEGO VÁZQUEZ LUCERO
Maestría en:	SISTEMAS COMPUTACIONALES
Título de la tesis: DESARROLLO DE UN PROTOTIPO DE VEHÍCULO AUTÓNOMO CON RECONOCIMIENTO Y APROXIMACIÓN HACIA BOTELLAS DE PLÁSTICO EN AMBIENTES CONTROLADOS	
DICTAMEN: Se autoriza el trabajo de investigación, en virtud de que realizó las correcciones correspondientes conforme a las observaciones planteadas por este Comité Tutorial.	

A t e n t a m e n t e.
El Comité Tutorial

DR. ISRAEL MARCOS SANTILLÁN MÉNDEZ

MSC. CÉSAR HIGUERA VERDUGO

MC. JORGE ENRIQUE LUNA TAYLOR

c.c.p. Coordinador de la Maestría.
c.c.p. Departamento de Servicios Escolares.
c.c.p. Estudiante.

Boulevard Forjadores de B.C.S. #4720, Col. 8 de Octubre 1ra Sección, C.P. 23080, La Paz, B.C.S. Tel. (612) 12 10424 e-mail: depi_paz@tecnm.mx | lapaz.tecnm.mx

Dedicatoria

"A todas las personas que me han impulsado a conseguir este objetivo".

"A los precursores de las ciencias computacionales".

"A las generaciones futuras".

"A mi hijo".

Agradecimientos

“A mis padres por darme la vida. A mi esposa por su amor incondicional e infinito. A toda mi familia. Todos ellos me han motivado a seguir adelante”.

“Valoro profundamente el esfuerzo, dedicación y paciencia de los miembros de mi comité de tesis y tutorial por sus sabios consejos que sin duda serán pilares en mi futuro desempeño profesional”.

“A todos mis maestros por formarme en esta nueva etapa de mi camino, compartiendo sus conocimientos y despertando en mi de nuevo el interés por la ciencia”.

“A mis amigos sinceros René y Daniel Ruiz Isais, apasionados de la ciencia, que se interesaron en el proyecto, y proporcionaron algunos componentes del dispositivo robotico”.

“A mi segunda casa, mi querido Tec”.

“A mi querida maestra Iliana”.

Resumen

La contaminación por plástico en el entorno marino genera una preocupación cada vez mayor debido a su persistencia y las consecuencias que tiene en el medio ambiente, la vida silvestre y la salud humana. Con el objetivo de aportar soluciones a esta problemática, en este trabajo de tesis se propone el diseño y construcción de un prototipo de robot móvil autónomo para el reconocimiento de botellas de plástico, aplicando una estrategia de segmentación semántica, a través de redes neuronales convolucionales. Así mismo, se propone el diseño de un sistema de navegación basado en lógica difusa, para lograr que el robot se aproxime a las botellas identificadas. Para evaluar estas estrategias, se trabajó en un ambiente controlado, dentro de un edificio académico, con diferentes tipos de botellas incluyendo la presencia de otros objetos. Los resultados de los experimentos muestran un 96.7 % de precisión en las pruebas de reconocimiento, mientras que en las pruebas de aproximación, el robot alcanzó en todos los casos la posición de las botellas identificadas.

Abstract

Plastic pollution in the marine environment is of increasing concern due to its persistence and the consequences it has on the environment, wildlife and human health. With the aim of providing solutions to this problem, this thesis work proposes the design and construction of an autonomous mobile robot prototype for the recognition of plastic bottles, applying a semantic segmentation strategy, through convolutional neural networks. Likewise, the design of a navigation system based on fuzzy logic is proposed, to make the robot approach the identified bottles. To evaluate these strategies, we worked in a controlled environment, inside an academic building, with different types of bottles including the presence of other objects. The results of the experiments show a 96.7 % accuracy in the recognition tests, while in the approximation tests, the robot reached in all cases the position of the identified bottles.

Índice general

Índice de figuras

Índice de tablas

1. INTRODUCCIÓN

1.1. Problemática

El plástico comenzó a comercializarse a gran escala en la década de los 50's del siglo pasado. Fue ganando mercado por su uso práctico, durabilidad y conveniencia; reemplazando al acero en la construcción de autos, al papel y al vidrio en embalajes, a la lana y al algodón en el diseño de ropa, y a la madera en la construcción de muebles (Buteler, 2019). A pesar de las ventajas del uso del plástico, por ser barato, útil y versátil, es desechado sin control, afectando tanto a humanos como a otras especies que habitan principalmente en ambientes marinos.

Estudios realizados en Carolina del Norte y Pacifico Norte de Estados Unidos, encontraron que más de la mitad de las aves marinas tenian partículas de plástico en sus estomagos (Rodolfo, 2015). La Figura 1 muestra una imagen de un ave marina que confunde una botella de plástico con alimento.

Figura 1: Las aves marinas confunden botellas de plástico con alimentos. (Buteler, 2019).

La contaminación degrada y destruye hábitats de playa únicos que son utilizados por animales y plantas. Así mismo, limita la posibilidad de utilizar las playas con fines económicos y recreativos. El plástico en el entorno acuático genera una preocupación cada vez mayor debido

a su persistencia y a las consecuencias que tiene en el medio ambiente, la vida silvestre y la salud humana (Agencia de Protección Ambiental de Estados Unidos, 2020).

Se deben tomar medidas urgentes para disminuir la cantidad de desechos plásticos que llegan al mar y que aunados al calentamiento global, la contaminación por metales y el derrame de sustancias químicas ponen en serio peligro de los ecosistemas marinos de flora y fauna, se espera que al paso que vamos, el problema del plástico se triplicará en el periodo de 2015 y 2025 (Ibérico, 2018). La Figura 2 muestra un ambiente de playa contaminado con botellas de plástico.

Figura 2: Contraste entre la belleza del mar y su ambiente contaminado. (Ibérico, 2018).

México es un país muy diverso y el sexto con mayor turismo a nivel mundial. Entre sus principales destinos cuentan con playas y actividades ecoturisticas que se verían afectadas por la contaminación de plásticos y pérdida de biodiversidad. En México no se cuenta con una legislación a nivel federal respecto al uso del plástico y la legislación para el manejo de residuos actual es insuficiente (NOM-083-SEMARNAT-2003). Sin embargo, distintas entidades federativas han

aprobado leyes locales o han creado iniciativas de ley enfocadas en la regulación de plásticos de un solo uso (Ocampo y Santa Catarina, 2019).

Con el objetivo de aportar soluciones a esta problemática, en este trabajo se propone el diseño y construcción de un robot móvil autónomo, para identificar botellas de plástico y desplazarse hacia estas, con el objetivo futuro de recolectarlas y depositarlas en contenedores para su reciclamiento. Para realizar los experimentos y evaluar la estrategia propuesta, se eligió trabajar en un ambiente controlado, dentro de un edificio académico, con cuatro tipos diferentes de botellas, incluyendo la presencia de otros objetos.

Diferente de otras propuestas recientes con objetivos similares (Kulshreshtha et al., 2021), (De La Torre Muña y Yufra Torres, 2020), (Ramírez Zavalza et al., 2020), (Bai et al., 2018), donde el reconocimiento se realiza al nivel de áreas rectangulares que encierran a los objetos, en este proyecto se aplica segmentación semántica a través de redes neuronales convolucionales (CNNs) (Ronneberger et al., 2015), para reconocer los objetos al nivel de pixeles. Esto permite delimitar el contorno de las botellas, y estimar su área y centroide de forma más precisa. Con estos datos, se aplica posteriormente un algoritmo basado en lógica difusa para aproximar el robot a las botellas identificadas.

1.2. Antecedentes

Los robots móviles se utilizan en una gran variedad de aplicaciones, por ejemplo, en vigilancia, exploración, operaciones de búsqueda y rescate, limpieza, automatización industrial, entre otras. Para esto, deben contar con algoritmos especializados que les permitan navegar a través de su entorno. La navegación de un robot incluye todas las acciones que llevan a que este se desplace desde su posición actual hasta una posición de destino (Lozada et al., 2020), (Agarwal y Bharti, 2020), (Gul et al., 2019).

Los constantes avances en la tecnología de los microprocesadores, sensores, cámaras, entre otros componentes, y el desarrollo de nuevas técnicas de IA, como las redes neuronales artificiales y la lógica difusa, hacen de la robótica móvil uno de los campos con mayor crecimiento (Abeliuk y Gutiérrez, 2021). En (Marin et al., 2021) se presenta un perfil histórico de algoritmos aplicados

a problemas de control, así como de las principales aplicaciones de la robótica móvil.

Actualmente, las redes neuronales artificiales son utilizadas en el control de sistemas con dinámicas complejas, especialmente en sistemas no lineales que varían en el tiempo y que resultan complejos de regular con métodos convencionales (Gachet Páez y Valverde Gil, 2007). En el campo de la robótica, una aplicación de las redes neuronales es en tareas de percepción a través de visión artificial. En este campo, se utilizan para la detección y clasificación de los objetos presentes en el entorno del robot (Perez et al., 2018).

Por otro lado, la lógica difusa se utiliza en sistemas donde los datos y sus relaciones no pueden describirse en términos matemáticos precisos. En este tipo de sistemas, se aplica la lógica difusa pretendiendo emular la estrategia de control que seguiría un experto humano en el control manual de un proceso (Santos, 2011). Dentro de la robótica móvil, una aplicación de la lógica difusa es en tareas de navegación. Por ejemplo, en (Cristino et al., 2019a) proponen la implementación de un sistema de control difuso tipo Mamdani, para la navegación y evasión de obstáculos, sobre una plataforma con recursos computacionales limitados. En (Lozada et al., 2020) presentan una arquitectura basada en control difuso para la selección de acciones de navegación, con el objetivo de que un robot móvil tome decisiones, con base en el nivel de carga de la batería.

La idea de diseñar y construir robot móviles para la recolección de basura no es nueva. Por ejemplo, en (De La Torre Muña y Yufra Torres, 2020) diseñaron y construyeron un robot móvil recolector y compactador de botellas de plástico en playas con arena fina, aplicando una red neuronal tipo MobileNet. En (Ramírez Zavalza et al., 2020) diseñaron y desarrollaron un robot recolector de basura utilizando CNNs con arquitectura SDD y Faster R-CNN.

En (Bai et al., 2018) presentan el diseño de un robot recolector de basura que opera sobre césped. El robot utiliza un sensor inercial, un sistema odométrico y un receptor GPS para producir información de ubicación. Para la identificación y evasión de obstáculos utiliza un sensor ultrasónico, así como una cámara web para detectar y reconocer la basura, a través de una Red Neuronal Convolucional.

En (Kulshreshtha et al., 2021) presentan el diseño de un robot recolector de basura. Los sensores que incluyen son una cámara, un sensor ultrasónico y un módulo GPS. Probaron tres modelos de red neuronal para la detección de objetos (Mask-RCNN, YOLOv4 y YOLOv4-tiny).

En la presente tesis, se propone una estrategia que combina segmentación semántica con lógica difusa, para el reconocimiento de botellas de plástico y aproximación de un robot hacia estas, utilizando únicamente información de una cámara de video. Esta estrategia no se encuentra reportada en algún otro trabajo de la literatura revisada.

1.3. Objetivo general

Desarrollar un prototipo de vehículo autónomo para reconocer y aproximarse a botellas de plástico en un ambiente de controlado.

1.4. Objetivos específicos

- Construir un vehículo con los componentes electromecánicos necesarios para su desplazamiento autónomo en un ambiente controlado.
- Crear un conjunto de imágenes con la segmentación a nivel de pixel de botellas de plástico.
- Entrenar una Red Neuronal Convolucional para el reconocimiento de botellas de plástico.
- Diseñar e implementar un algoritmo basado en lógica difusa para aproximar el vehículo a las botellas identificadas.

1.5. Alcances y limitaciones

El prototipo de robot móvil autónomo desarrollado es capaz de reconocer cuatro tipos de botellas de plástico con una precisión mayor al 95 %, y de desplazarse y aproximarse a estas a una distancia no mayor a quince centímetros, respecto al centroide de las mismas.

Los componentes electromecánicos del vehículo construido son de bajo presupuesto, con capacidades limitadas de carga y fuerza de tracción, por lo que los experimentos se realizaron

en un ambiente controlado, considerando únicamente el desplazamiento sobre piso firme y sin pendientes.

1.6. Hipótesis

Es factible construir y programar un robot autónomo capaz de reconocer botellas de plástico y aproximarse a estas, aplicando segmentación semántica y lógica difusa.

1.7. Justificación

Diferente de otras propuestas recientes con objetivos similares (Kulshreshtha et al., 2021, De La Torre Muña y Yufra Torres, 2020, Ramírez Zavalza et al., 2020, Bai et al., 2018), donde el reconocimiento se realiza al nivel de áreas rectangulares que encierran a los objetos de interés, en este proyecto se aplica segmentación semántica (Ronneberger et al., 2015), para reconocer los objetos al nivel de píxeles. Esto permite delimitar el contorno de las botellas, y estimar su área y centroide de forma más precisa, permitiendo ampliar las estrategias para aproximarse a estas y establecer métodos para su eventual recolección.

Así mismo, se propone un sistema de control basado en lógica difusa, que suministra la potencia necesaria a los motores, para aproximar el vehículo de forma autónoma hacia las botellas reconocidas. Esta combinación, de aplicar segmentación semántica para el reconocimiento de objetos y lógica difusa para aproximarse hacia estos, no se encuentran reportada en algún otro trabajo de la literatura revisada. Una característica relevante de la estrategia propuesta, es su sencillez de implementación y el uso de una cámara de video, como única fuente de datos para localizar y aproximarse a los objetos.

1.8. Organización de la tesis

En el Capítulo uno se describe la problemática que se desea solucionar, objetivos trazados, alcances y limitaciones, planteamiento de la hipótesis y justificación de este proyecto.

En el Capítulo dos se analizan tres de los trabajos relacionados encontrados en el estado del arte, que se enfocan en el diseño y programación de robots recolectores en diferentes ambientes.

En el Capítulo tres se presentan aspectos del marco teórico que involucra el desarrollo del presente trabajo, como son la robótica móvil, programación, diseño mecatrónico, IA, redes neuronales, aprendizaje de máquina, aprendizaje profundo, visión computacional, lógica difusa y segmentación semántica.

En el Capítulo cuatro se explican las fases del proceso de desarrollo del prototipo robótico, como la obtención de las imágenes de entrenamiento, su procesamiento, entrenamiento de la de red neuronal, y las pruebas preliminares de reconocimiento y aproximación.

En el Capítulo cinco se muestran los experimentos realizados para determinar la eficiencia del reconocimiento y aproximación del robot a las botellas detectadas, así como el cálculo del error entre la posición final del robot y el centroide de las botellas.

Finalmente, en el Capítulo seis se presentan las conclusiones del proyecto y recomendaciones de trabajo futuro.

2. TRABAJOS RELACIONADOS

Durante el desarrollo de este proyecto se revisaron diversos trabajos relacionados con los objetivos de esta investigación, principalmente, aquellos que utilizan robots móviles basados en técnicas de la IA, para el reconocimiento y recolección de basura en distintos ambientes. A continuación se presentan los trabajos que consideramos más relevantes.

2.1. Diseño de robot autónomo de recolección de basura al aire libre Usando YOLOv4-Tiny

El robot presentado en (Kulshreshtha et al., 2021) fué diseñado para recolectar basura en ambientes como playas, parques, bosques, y en general, lugares donde se reune la gente con fines de convivencia. Este robot fue desarrollado para trabajar de forma autónoma, logrando detectar la basura, moverse hacia ella y recogerla evitando cualquier obstáculo en el camino. Para ello incluye sensores como cámara tipo webcam, sensores ultrasonicos y módulo GPS (ver Figura 3).

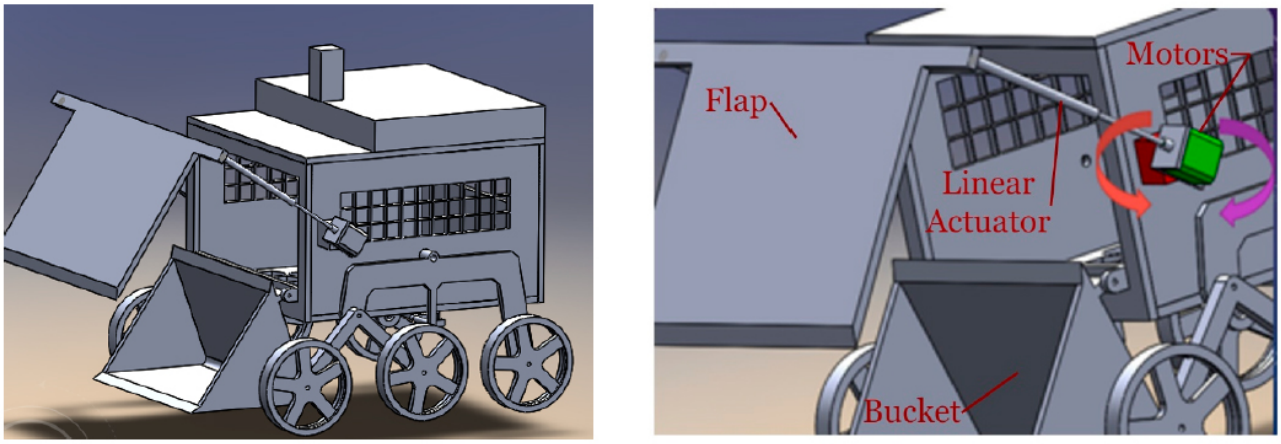


Figura 3: A la izquierda el cuerpo del robot y a la derecha los motor de aleta. (Kulshreshtha et al., 2021)

Se utilizaron las placas Raspberry PI 4 Model B© y Arduino UNO© para la programación de las operaciones de toma de decisiones que se requiere para que el robot realice las tareas cognitivas, como son el reconocimiento y la navegación. Raspberry PI© se encarga del procesamiento de imágenes y las tareas de seguimiento, mientras que Arduino UNO© realiza la

evasión de obstáculos y planificación de trayectorias. En la Figura 4 se muestra el diagrama de la configuración mecatrónica.

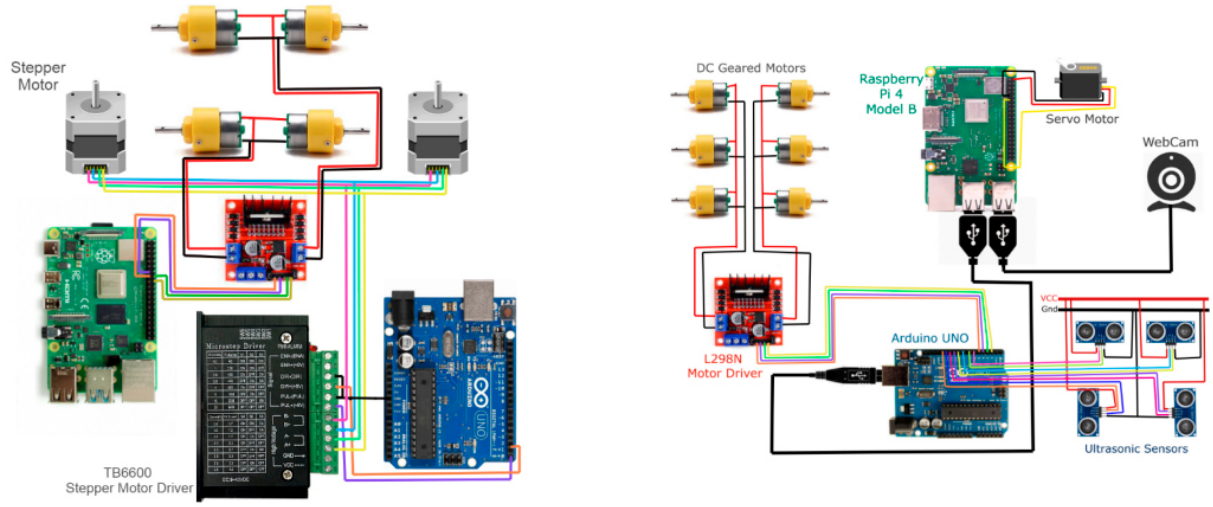


Figura 4: Diagrama de la configuración mecatrónica. (Kulshreshtha et al., 2021).

Para la detección de los objetos de interés probaron tres modelos de red: Mask-RCNN, YOLOv4 y YOLOv4-tiny. Mask-RCNN logró una precisión promedio (mAP) superior al 83 % y un tiempo de detección (DT) de 3973,29 ms, YOLOv4 logró un 97,1 % (mAP) y 32,76 DT, y YOLOv4-tiny logró un 95,2 % y 5,21 ms DT. Finalmente, seleccionaron el modelo YOLOv4-tiny, dado que logra un mAP muy similar al del YOLOv4, pero con un DT mucho menor.

2.2. Diseño de un robot móvil recolector y compactador de botellas de plástico utilizando redes neuronales en playas con arena fina

En el trabajo presentado en (De La Torre Muña y Yufra Torres, 2020), proponen el diseño de un robot móvil recolector y compactador de botellas de plástico en un ambiente de playas con arena fina, utilizando redes neuronales. El sistema de locomoción del robot es tipo oruga y cuenta con dos brazos encargados de recolectar las botellas por medio de una pala. Las botellas recolectadas se llevan a un contenedor para posteriormente ser compactadas. La Figura 5 muestra el diagrama de los componentes electrónicos del robot.

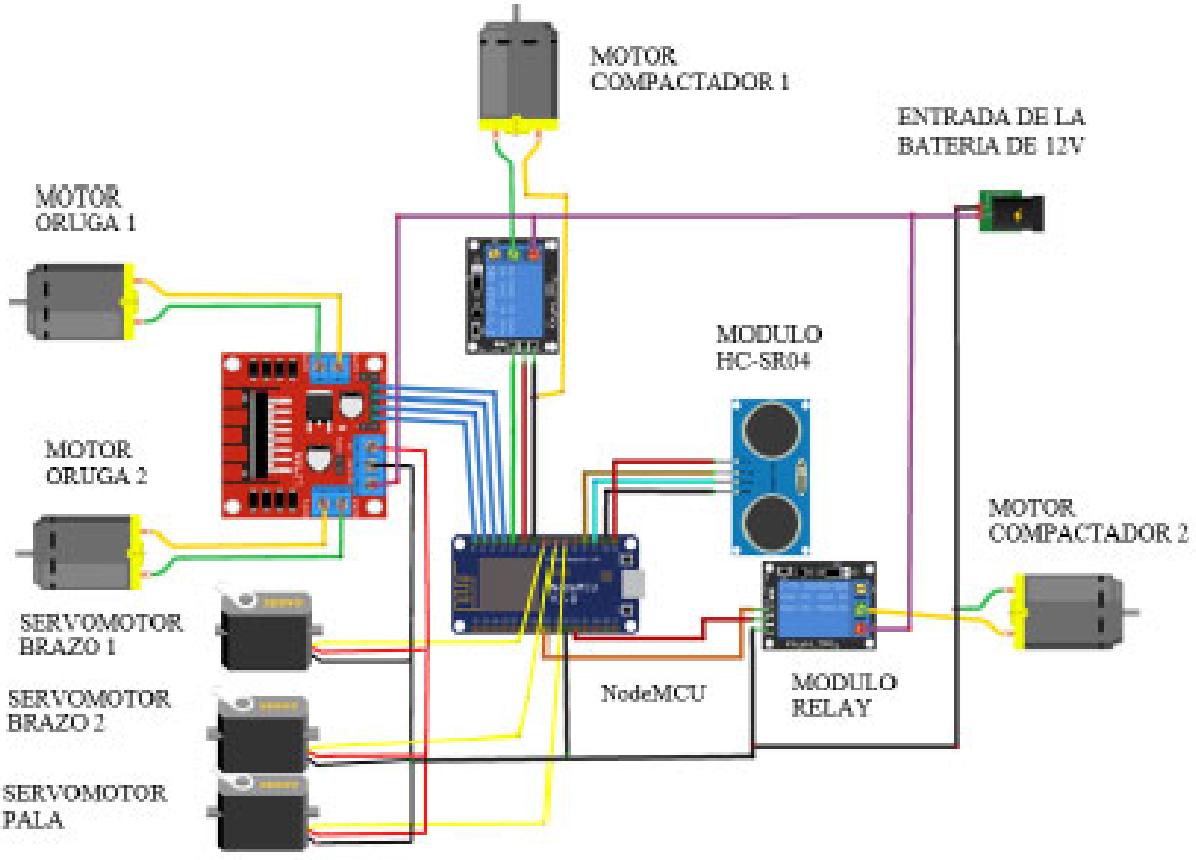


Figura 5: Diseño electrónico de robot recolector y compactador de botellas de plástico. (De La Torre Muña y Yufra Torres, 2020).

En el trabajo que reportan, realizan el diseño de los componentes del sistema a través de un programa CAD. Incluyen el diseño de cada uno de los sistemas del robot. Describen la parte mecánica, desde los materiales seleccionados considerando factores ambientales, dimensiones

del robot, capacidad de compactación y capacidad de almacenamiento. Para la parte eléctrica realizan el cálculo de la energía requerida para todo el sistema y los componentes eléctricos que se necesita para su alimentación. En la parte electrónica describen los componentes que se requieren para el desplazamiento, levantamiento y compactación de las botellas de plástico. La visualización de las conexiones del circuito la realizan a través del software Fritzing.

El desarrollo del programa lo realizaron con ayuda del software ATOM, con el objetivo de que el robot seleccione solo las botellas de material plástico. Este software utiliza redes neuronales y procesamiento de imágenes a través del microcontrolador Arduino UNO©. En la Figura 6 se muestra el diagrama general de los diferentes componentes de sistema del robot.

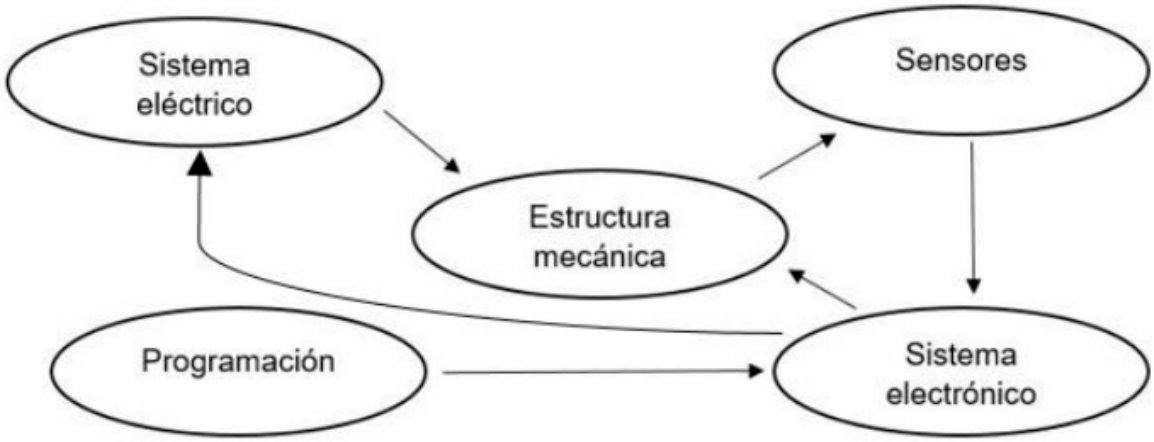


Figura 6: Diagrama general de los componentes del sistemas del robot recolector y compactador de botellas de plástico. (De La Torre Muña y Yufra Torres, 2020).

Para el desarrollo de sus experimentos, estudiaron tres modelos de microcontroladores con ciertas características predefinidas que son: 1) que sea programable en el lenguaje C++, 2) fácil de adquirir y 3) capacidad para comunicación a través de WiFi. Los modelos evaluados fueron Sparkfun Esp8266 thing, EspDuino-32 y Nodemcu Esp8266. Eligieron el modelo Nodemcu Esp8266 que, a pesar de que posee un número limitado de entradas y salidas digitales, estas fueron suficientes para el control del robot, y contiene una cantidad adecuada de memoria para procesamiento. La Figura 7 muestra una imagen de este microcontrolador, con la configuración de los pines utilizados.

Pines disponibles	
en la placa	en el programa
D0	16
D1	5
D2	4
D3	0
D4	2
D5	14
D6	12
D7	13
D8	15
Rx	3
Tx	1
SD3	10

Figura 7: Configuración de pines del NodeMCU. (De La Torre Muña y Yufra Torres, 2020)

Para la programación de las redes neuronales y su entrenamiento utilizaron el software Atom, el cual es de código abierto disponible para macOS, Linux y Windows. Este software permite crear una interfaz amigable y fácil de usar con ayuda de las librerías de P5, ml5 y MQTT.

Para el diseño de la estructura mecánica utilizaron material de aluminio, con un sistema de locomoción tipo oruga. Para la parte eléctrica se dimensionó la capacidad de almacenamiento de la batería basándose en el consumo total del sistema, optando por una batería de 12V y 40Ah de ion-litio ya que ofrece un bajo peso. Para la parte electrónica se diseñó un circuito utilizando el módulo L293N que controla el sistema de locomoción, y dos módulos relevadores para el control de cada motor del sistema de compactación.

Para el sistema de recolección utilizaron tres servomotores de 5 [kg] para levantar y depositar las botellas. El entrenamiento de la red neuronal se realizó mediante la creación de una página Web con el objetivo de ofrecer una interfaz fácil y amigable para el usuario. Además se utilizó el microcontrolador NodeMCU para recibir los datos de la red neuronal mediante WiFi.

2.3. Plataforma de Robot Móvil con navegación autónoma con Arduino UNO© y Raspberry PI©

En el trabajo presentado en (Oltean, 2019) proponen el diseño de un vehículo controlado a través de las tarjetas Raspberry PI© y Arduino UNO©. El vehículo es capaz de moverse en un ambiente 2D, con habilidad de mapeo, navegación, seguimiento y evasión de obstáculos. El diseño de este vehículo tiene fines educativos y de investigación, se pretende que se utilice como base para trabajos futuros.

La plataforma mecánica del robot se basa en un chasis de configuración fija de cuatro ruedas accionado de forma independiente por cuatro motores de corriente continua. Para el movimiento de avance o retroceso del chasis, los cuatro motores se accionan en la misma dirección con la misma velocidad deseada. Para realizar una rotación del chasis, los motores en lados diferentes se accionan en sentido contrario. En el chasis están montadas dos placas acrílico de 15x20 cm que contienen la electrónica e interfaces conectadas a los sensores y motores del robot, distribuidas como se muestra en la Figura 8.

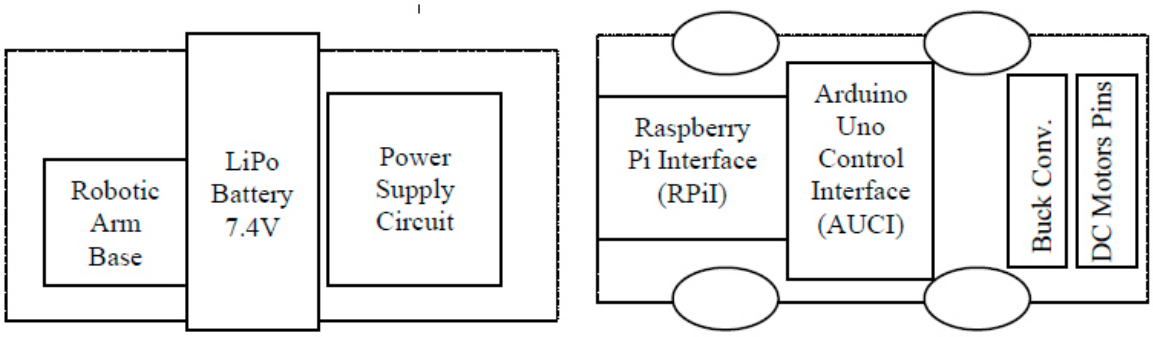


Figura 8: Diagrama de distribución de la electrónica e interfases. (Oltean, 2019).

La placa superior contiene el circuito de alimentación, la batería LiPo (con 4000 [mAh] de capacidad, 7,4 [V]) conectada a él y la base de un brazo robótico. La placa inferior alberga los principales componentes de control, como Raspberry PI© y Arduino UNO© , así como el convertidor Buck. En la parte frontal del robot se colocaron tres sensores infrarrojos, un sensor ultrasónico central para detectar los obstáculos y una cámara Raspberry PI© para identificar

el color y forma de los obstáculos. A cada lado del robot se colocó un sensor ultrasónico para detectar obstáculos laterales.

El robot incluye un brazo robótico de un grado de libertad, con una pinza que tiene una longitud máxima de apertura de 13 cm, que permite levantar y eventualmente transportar los objetos. El brazo y la pinza son accionados por dos servomotores. La Figura 9 muestra el diagrama de bloques de los componentes electrónicos de esta plataforma de robot móvil.

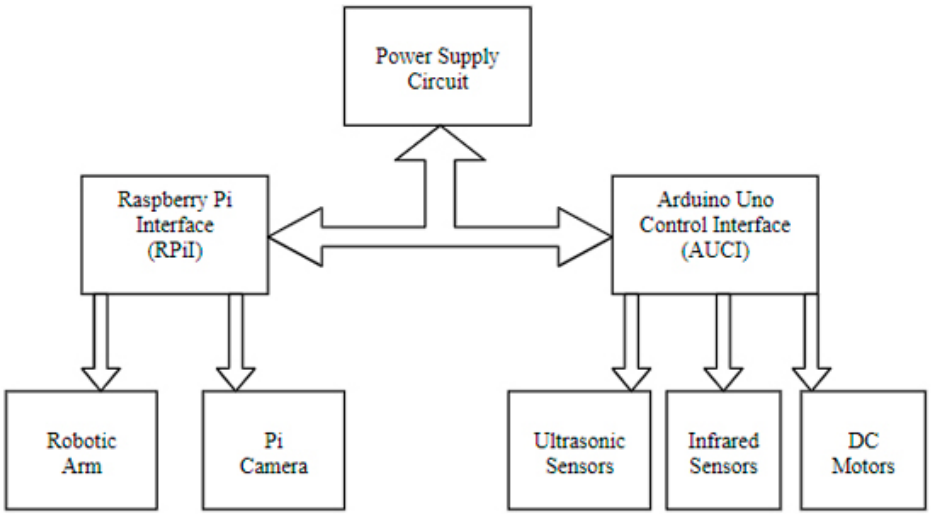


Figura 9: Diagrama de bloques de la parte electrónica del robot. (Oltean, 2019)

La tarjeta Raspberry PI© se programó bajo el sistema operativo Raspbian, para realizar las acciones de control: mapeo, navegación, detección de obstáculos, evasión y transporte. Así mismo, se programó la tarjeta Arduino UNO© para la lectura y control de los sensores ultrasónicos e infrarrojos, servomotores y motores de corriente continua.

El brazo robótico es accionado por dos servomotores. El primer servomotor está ubicado en la base del brazo robótico y asegura un movimiento de 15 cm del primer eslabón del brazo, desarrollando un par máximo de 9,4 [kg/cm] con tensión de alimentación de 4,8 [V]. El segundo servomotor está ubicado en la parte superior del brazo robótico, que tiene la función de atrapar y levantar los objetos. Es capaz de desarrollar un par máximo de 3,17 kg/cm con una tensión de alimentación de 4,8V.

Los autores reportan que el robot móvil diseñado cumple con los requisitos de diseño para esta etapa básica de desarrollo, siendo una solución de bajo costo, confiable y extensible, con la posibilidad de utilizarse en la enseñanza, como soporte didáctico para el estudio de microcontroladores, electrónica, automatización y robótica. Así mismo, consideran que se puede utilizar en actividades de investigación, para estudiar diferentes algoritmos de mapeo, localización, navegación, detección de obstáculos y transporte.

Finalmente, reportan que la plataforma desarrollada puede utilizarse en aplicaciones como: robot guía autónomo en ambientes interiores, uso médico como apoyo a pacientes, aplicaciones militares, embalaje y organización de tarimas en almacenes, transporte de material de desecho, lavandería, alimentos, productos farmacéuticos o correos.

2.4. Robot basado en aprendizaje profundo para recolección automática de basura sobre pasto

En el trabajo presentado en (Bai et al., 2018), diseñaron un robot recolector de basura que opera sobre césped. El robot puede detectar la basura de manera precisa y autónoma mediante el uso de una red neuronal profunda. Además, a través de información de la segmentación del suelo, proponen una estrategia de navegación para guiar al robot durante su desplazamiento. Con las funciones de reconocimiento de basura y navegación automática, se tiene el objetivo de que el robot pueda recolectar basura del suelo en parques o escuelas de manera eficiente y autónoma.

Para localizar la basura, utilizan una red neuronal convolucional que procesa las imágenes capturadas. Utilizan también una red convolucional para detectar y segmentar el suelo, lo cual es útil para la evasión de obstáculos. En la Figura 10 se muestra una imagen del robot recolector.

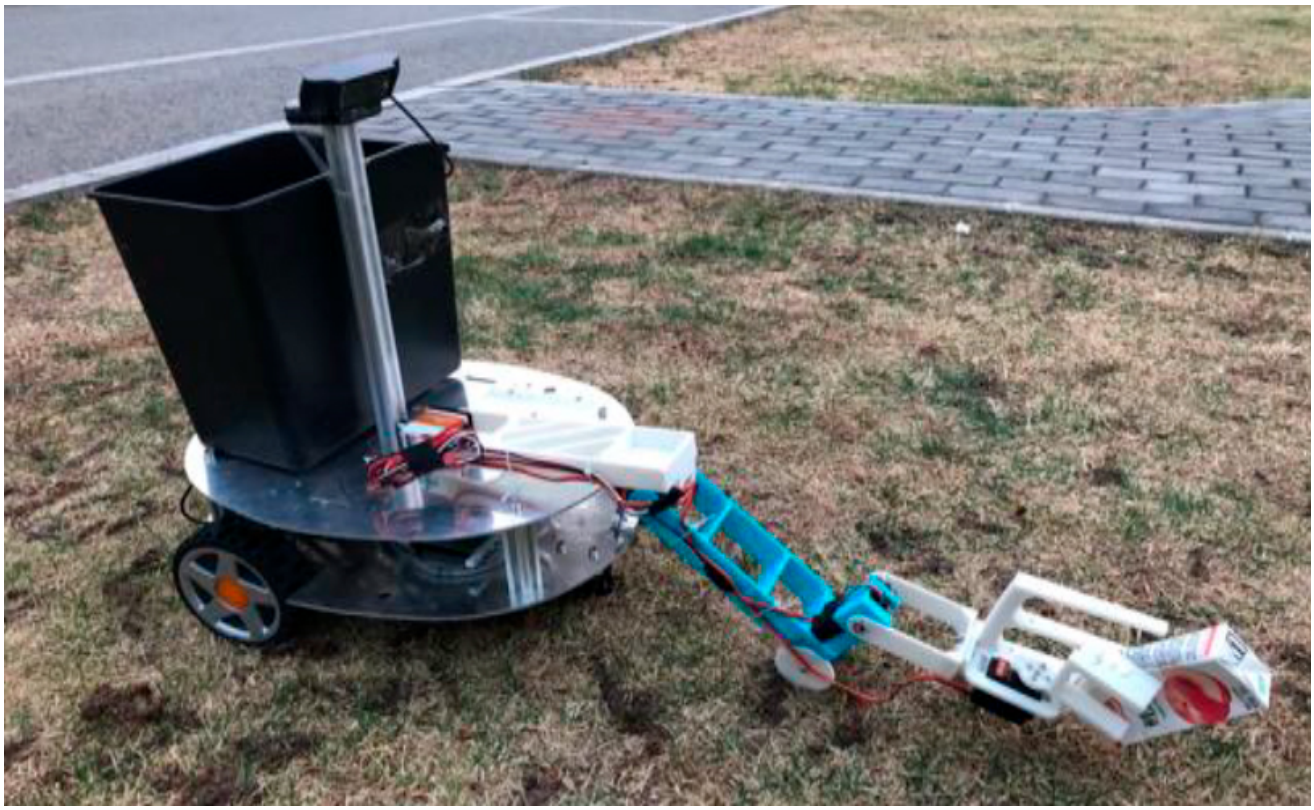


Figura 10: Robot recolectando basura sobre pasto. (Bai et al., 2018).

En la Figura 11 se muestran los principales componentes de hardware del robot. Así mismo,

a continuación de describe brevemente la función de cada uno:

- Sensor inercial, odómetro de vehículo y un receptor GPS proporcionan la ubicación precisa del robot.
- Cámara web para detectar y reconocer la basura, así como para detectar obstáculos.
- Telémetro ultrasónico ayuda para detectar obstáculos de manera más sólida.
- Base del robot es el cuerpo principal del robot, equipado con todos los sensores, controlador, manipulador y contenedor de basura.
- Dos ruedas motrices y una rueda pasiva. Se acciona de manera diferencial para girar el robot.
- El controlador del robot incluye la CPU, la GPU y el controlador del motor.
- La rotación, la traslación hacia adelante o hacia atrás del robot es controlada por el controlador del motor.
- Brazo manipulador para recoger la basura y depositarla en el contenedor.
- Contenedor de basura con ranura de sujeción para montar y desmontar.

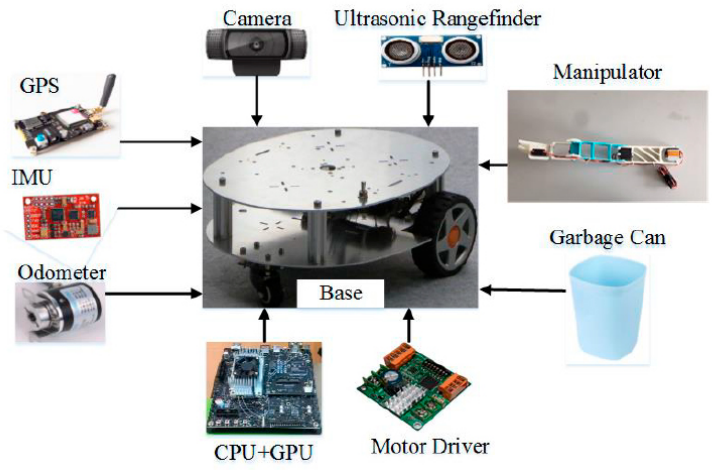


Figura 11: Componentes del robot recolector de basura en pasto. (Bai et al., 2018).

Para la localización de los objetos utilizan la red SegNet, que genera recuadros alrededor de los objetos detectados. Una vez localizados los objetos, aplican la red ResNet para la clasificación y etiquetado semántico de estos. La Figura 12 muestra un ejemplo de este proceso de localización y etiquetado de objetos.

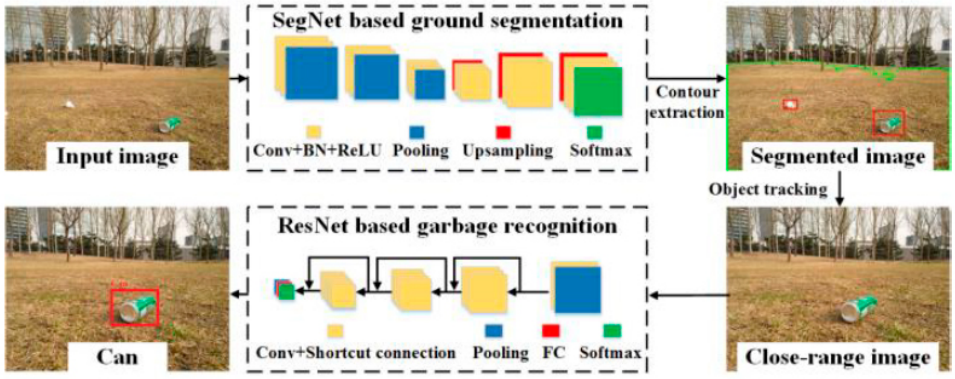


Figura 12: Flujo del procesamiento de la percepción. (Bai et al., 2018).

La imagen segmentada proporciona un área aproximada del suelo y los objetos sobre este. Cuando se detecta un objeto sobre el suelo, el sistema rastreador controla el robot para que se acerque al objeto. En caso de existir múltiples objetos se selecciona el más cercano para rastrear. Una vez que el robot está próximo a un objeto, se utiliza una imagen de corto alcance para su reconocimiento. Si el objeto se reconoce como basura, el manipulador lo recoge, de lo contrario, el robot lo considera como un obstáculo y planeará evitarlo.

Los resultados experimentales muestran que el robot propuesto puede reconocer la basura con precisión, moverse de manera eficiente y recoger la basura de forma autónoma.

3. MARCO TEÓRICO

Para lograr el desarrollo de un vehículo autónomo con capacidad de reconocimiento y aproximación a objetos, se deben de combinar conocimientos de diversos campos de la ciencia y tecnología, como son: la robótica, aprendizaje máquina, aprendizaje profundo, visión por computadora y el control difuso.

3.1. Robótica Móvil

Hoy en día, la robótica móvil es uno de los campos de investigación científica de mayor expansión. Los robots móviles se desarrollaron gracias a la necesidad de tener vehículos capaces de operar en diferentes ambientes. Por lo cual, estos dispositivos deben tener la capacidad de realizar movimientos precisos sobre entornos no estructurados; la información que va recogiendo en su recorrido puede ser incierta y es procedente de sensores (Marin et al., 2021).

Los robots se pueden clasificar según el entorno en el que operan (Figura 13), y por el tipo de actividad que realiza (Figura 14). Comúnmente también se clasifican como fijos y móviles, y ambos trabajan en ambientes diferentes y tienen capacidades muy diferentes. Los robots fijos son elementos manipuladores que realizan tareas repetitivas, tales como soldadura o pintura. Los robots móviles se muevan y realizan tareas en ambientes totalmente diferentes y que pueden cambiar con el tiempo, un ejemplo de estos son los automóviles autónomos (Ben-Ari y Mondada, 2017).

Con los avances en mecatrónica y computación, la Robótica adquiere sofisticadas capacidades sensoriales y motoras, que dan a los nuevos robots la capacidad de adaptarse a ambientes cambiantes, explorarlos y realizar tareas. Al crear una relación entre la IA y la robótica móvil, el dispositivo robótico puede elevar su nivel de autonomía a través del aprendizaje (Andreu-Perez et al., 2018).

Esto refuerza la idea de que se pueden construir robots totalmente autónomos, capaces de realizar con máxima eficiencia tareas como conducir un vehículo, volar en ambientes naturales y artificiales, nadar, caminar, conversar, transportar y recoger objetos, mejorando los que

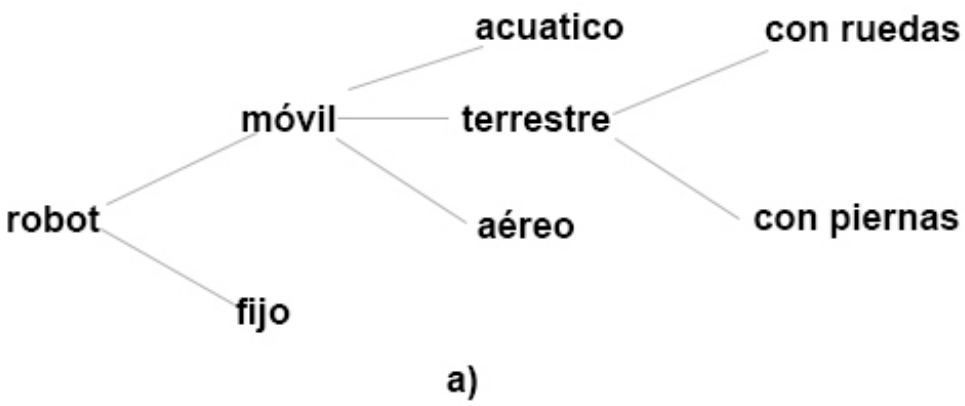


Figura 13: Clasificación respecto al ambiente en el que trabajan. (Ben-Ari y Mondada, 2017).

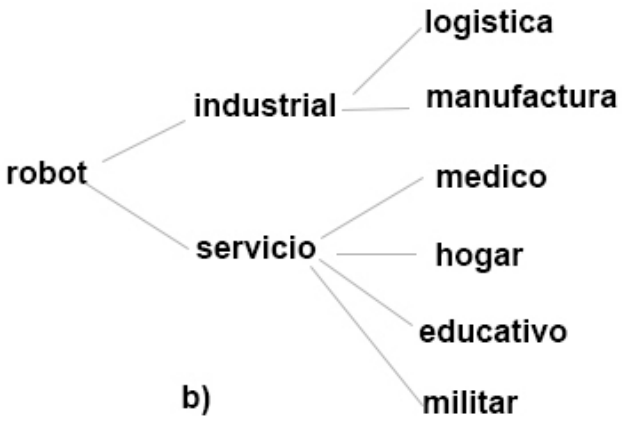


Figura 14: Clasificación respecto a la función que realizan. (Ben-Ari y Mondada, 2017).

existen hoy en día. El desarrollo de cámaras y las técnicas informáticas para el reconocimiento y la navegación y junto al desarrollo de otros componentes como microcontroladores, sistemas geolocalización, motores y bacterias, nos muestra un panorama alentador para estos campos.

Un robot es autónomo cuando tiene la capacidad de determinar por sí solo las acciones a tomar al realizar una tarea, apoyado con un sistema de percepción. También requiere de un sistema de control para coordinar todos los subsistemas que lo componen (Rubio et al., 2019). Las tareas básicas de la robótica móvil consisten en locomoción, percepción y navegación (Alatise y Hancke, 2020).

Figura 15: Robot móvil autónomo trabajando en el campo. (Ben-Ari y Mondada, 2017).

La mecatrónica reúne áreas de la tecnología que involucran sensores, sistemas de medición, sistemas de manejo y actuadores, así como sistemas de microprocesadores (Bolton, 2017), ver Figura 16.

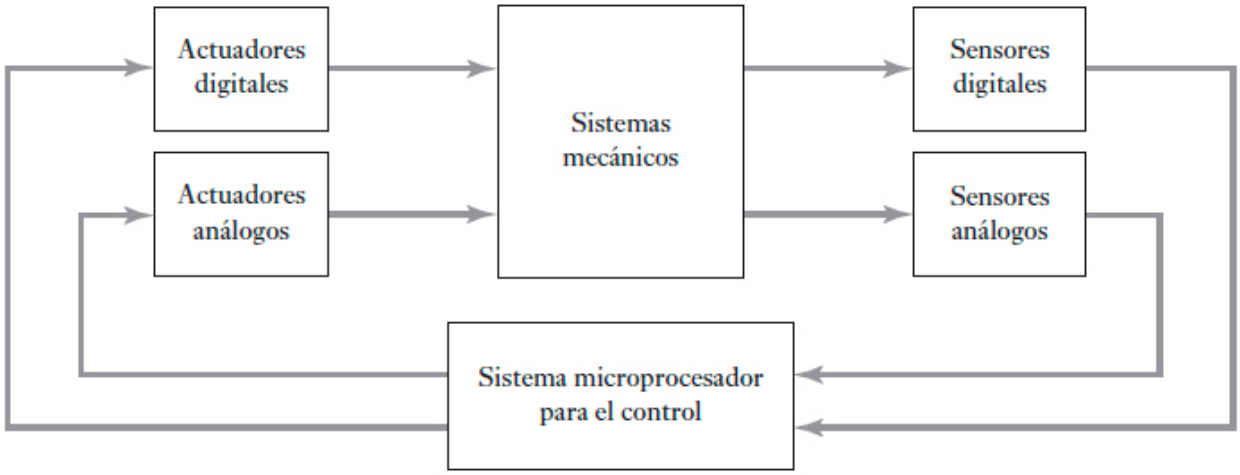


Figura 16: Elementos básicos de un sistema mecatrónico. (Bolton, 2017).

Por ejemplo, un robot móvil se considera un sistema mecatrónico ya que para su desarrollo se requiere de la integración de varias disciplinas como son: ingeniería mecánica, ingeniería electrónica e ingeniería en sistemas computacionales. Para desarrollar un robot móvil es necesario considerar el modelado del comportamiento cinemático, modelado dinámico y desarrollo de

algoritmos de control del movimiento y otras acciones que el dispositivo deba realizar (Daniel et al., 2016).

3.2. Elementos del diseño.

Para el correcto funcionamiento de un robot móvil, se deben considerar los elementos físicos y sensoriales (Hardware) además del desarrollo del software, que deben obedecer eficiente y rápidamente, a los grandes niveles de procesamiento de datos que se que se requieren en las tareas de reconocimiento.

NVIDIA Jetson Nano©.

La NVIDIA Jetson Nano© (ver Figura 17) es una plataforma para desarrollo y programación de proyectos que involucran elementos de robótica e inteligencia artificial. Sus principales características son: bajo costo, alto rendimiento en consumo de energía, gran velocidad de procesamiento y su tamaño reducido.

La NVIDIA Jetson Nano© cuenta con un procesador ARM de 64 bits de cuatro núcleos y una GPU con arquitectura 'Maxwell' con 128 núcleos de procesamiento gráfico, además de 4 Gbytes de memoria que ofrece una potencia de hasta 427 Gflops. El diseño de la NVIDIA Jetson Nano© se basa en hardware libre y regularmente se utiliza también en sistemas operativos libres basados en GNU/Linux (Vazquez, 2023).

Al contar con una GPU, esta pequeña computadora permite entrenar y ejecutar en paralelo sistemas de redes neuronales, ofreciendo una rápida ejecución de sistemas de la IA, tales como sistemas de visión computacional, clasificación y reconocimiento de objetos, entre otros.

Arduino UNO©.

La placa de desarrollo Arduino UNO© (ver Figura 18), está basada en el microcontrolador ATmega328. Cuenta con 14 pines numerados del 0 al 13, y cada uno de estos se puede configurar como pin de entrada o como pin de salida digital. El Arduino UNO© es un sistema microcon-

Figura 17: NVIDIA Jetson Nano© . (Vazquez, 2023).

trolador monoplaca, de hardware libre, muy fácil de utilizar y de bajo costo (Herranz, 2015). Los pines que se pueden configurar como salida de esta placa, pueden recibir o proporcionar una corriente de hasta 40 [mAh].

Con la placa Arduino UNO© se puede controlar por medio de programación, sensores, los cuales se utilizan para medir datos del entorno, como son la temperatura, luminosidad, distancia, humedad, entre otras. Estas lecturas se reciben a través de señales eléctricas que Arduino UNO© interpreta y las puede utilizar para realizar alguna acción, por ejemplo, para el encendido o apagado de un motor, o cualquier otro tipo de dispositivo que se desea controlar (Viveros Villada, 2022).

Todos los programas (llamados sketches) escritos para la placa de desarrollo Arduino UNO© contienen las funciones setup() y loop(). En la función setup() suelen declararse las variables y las configuraciones del hardware que se quiere controlar, mientras que la función loop() contiene el bloque principal del código necesario para el desarrollo de una aplicación.

Para facilitar el trabajo de un programador, el equipo de Arduino UNO© ha proporcionado un lenguaje de programación de alto nivel, un compilador y una herramienta de comunicación

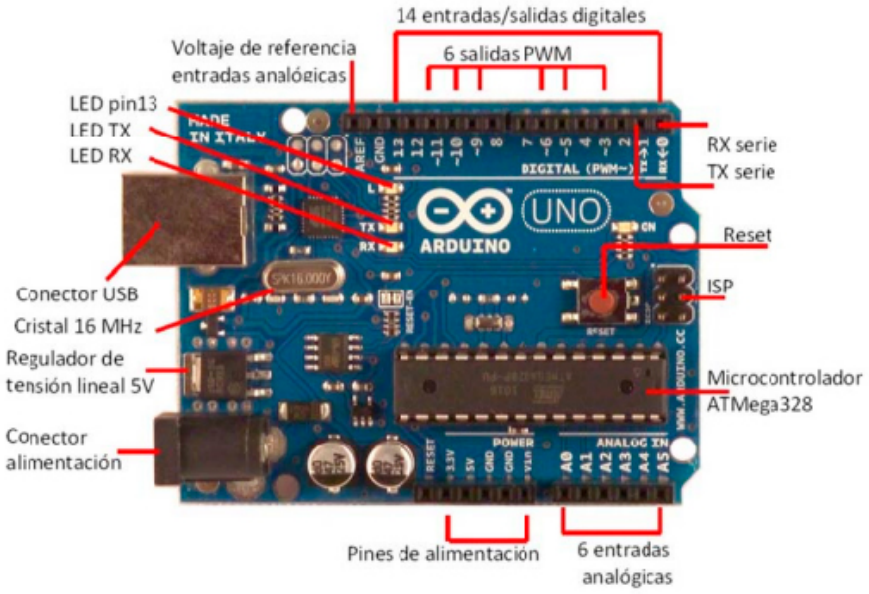


Figura 18: Placa de desarrollo Arduino UNO©. (Herranz, 2015).

para implementar el código de máquina en la memoria de Arduino UNO©. Todas esas herramientas están incluidas en un IDE (Entorno de desarrollo integrado) disponible gratuitamente en el sitio web de Arduino UNO© para su descarga, ver Figura 19.

Lenguaje de programación Python.

Python es un lenguaje de programación de propósito general que actualmente se considera uno de los lenguajes de programación más fáciles para iniciar el aprendizaje de la programación, debido a su estilo sencillo y flexible de codificación. El lenguaje incluye una gran cantidad de métodos integrados como parte de su biblioteca estándar y la facilidad de agregar múltiples librerías para diferentes propósitos.

Sus características principales son: Es fácil y sencillo de aprender, su uso esgratuito y de código abierto, es un lenguaje de programación de alto nivel, tiene una plataforma independiente, es portable, de tipado dinámico, admite paradigmas orientado a procedimientos y objetos, es un lenguaje interpretado, pero puede compilarse, es extensible, incrustado y tiene una amplia biblioteca de funciones (Cutting y Stephen, 2021).

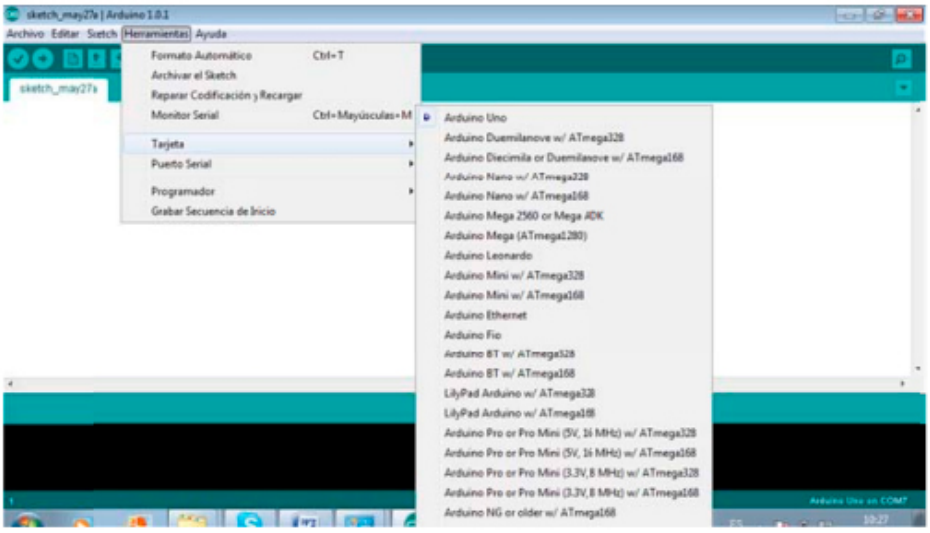


Figura 19: IDE de Arduino. (Herranz, 2015).

De acuerdo a PYPI, Phyton se ubica en primer lugar de popularidad hasta marzo de 2022, y en los últimos cinco años ha crecido 12.1 %, mientras que Java ha perdido 4.5 %, ver Figura 20. Una encuesta reciente de Stack Overflow mostró que Python ha superado lenguajes muy populares como Java, C y C++, llegado a la cima en popularidad y uso (Raschka, 2015, Pulungan et al., 2021).

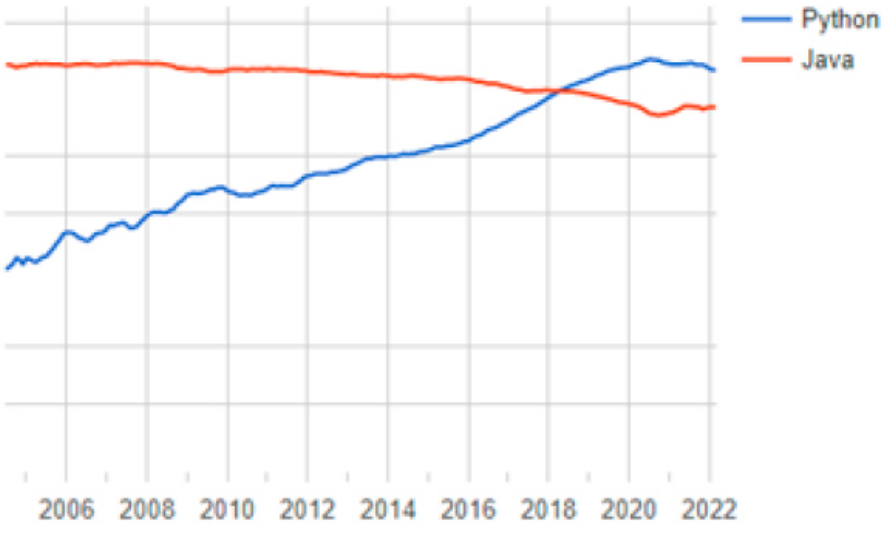


Figura 20: Gráfica de popularidad Python vs Java. (Vazquez, 2023).

Open CV.

OpenCV (Open Source Computer Vision) es una biblioteca multiplataforma gratuita para el procesamiento de imágenes en tiempo real, que se ha convertido en una herramienta estándar para todo lo relacionado con visión por computadora (García et al., 2015). Entre sus aplicaciones incluye: corrección y mejora de imágenes, procesamiento del color, procesamiento de imágenes en video y fotografía computacional, ver Figura 21.

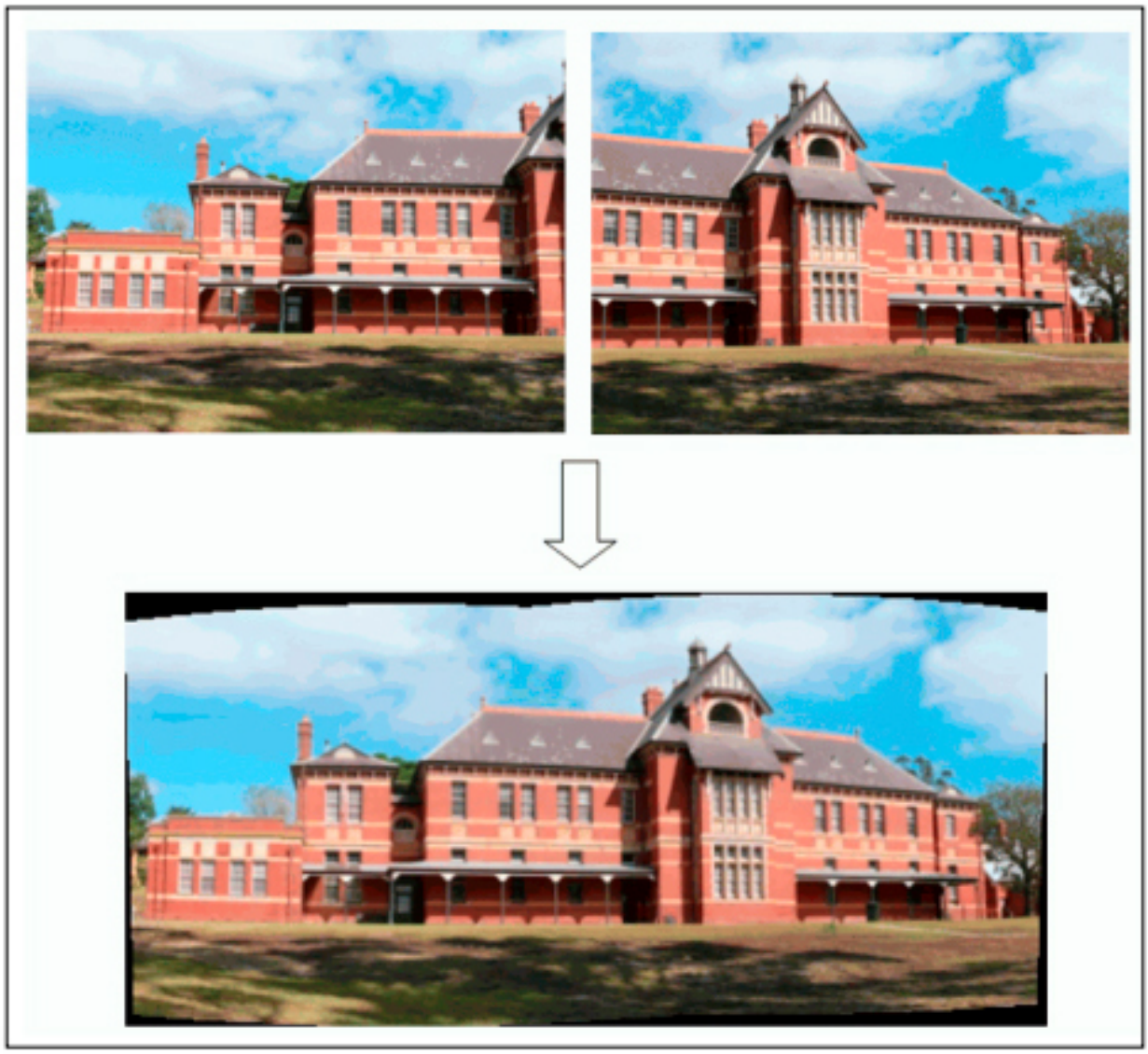


Figura 21: Ejemplo de procesamiento de imágenes en video. (García et al., 2015).

El color de una imagen proporciona información que puede ser utilizada en tareas de visión

computacional, como son el reconocimiento de objetos, seguimiento de trayectoria, detección de característica y la separación por colores de contornos, analizando los valores de los pixeles y comparandolo con otros cercanos.

Un contorno se puede representar de diferentes maneras, dos de las formas más comunes son el uso de polígonos y códigos de cadena de Freeman (Bradski y Kaehler, 2008). El valor de retorno del método cvFindContours es el número total de contornos encontrados. Un paso importante antes de llamar a este método es la binarización de la imagen de entrada. Este proceso requiere un poco de filtrado posterior para limpiar la imagen, de lo contrario, se pueden encontrar demasiados contornos. La Figura 22 muestra la representación de contornos.

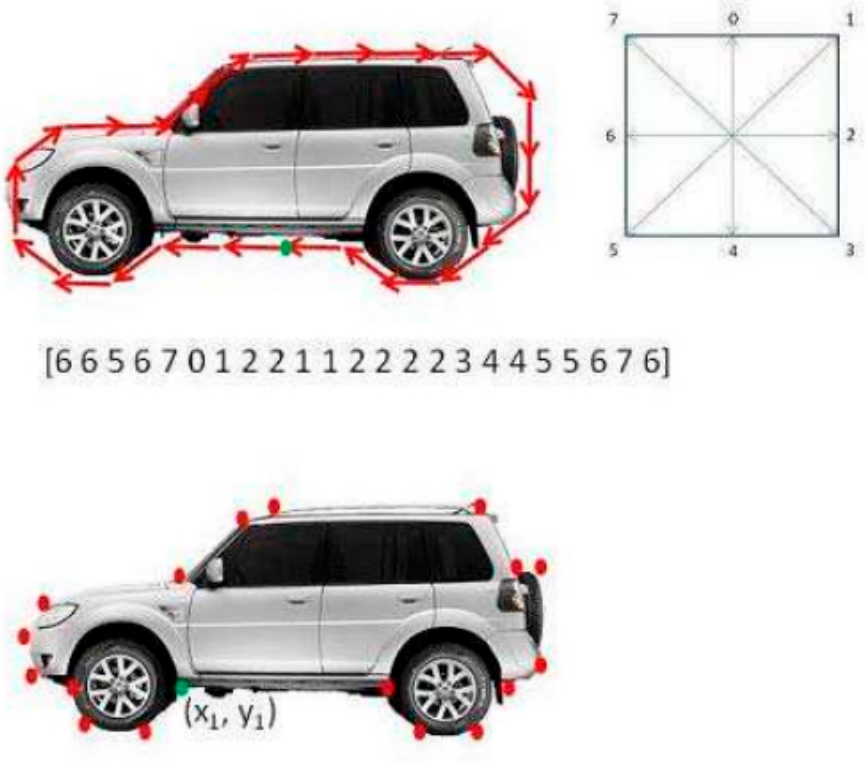


Figura 22: Representación del contorno: arriba el código de cadena de Freeman y abajo un conjunto de puntos para un polígono. (Bradski y Kaehler, 2008)

Para identificar contornos dentro de una imagen se utiliza la función cv2.findContours(), que tiene tres argumentos: el primero es la imagen de origen, el segundo es el modo de recuperación de contorno, el tercero es el método de aproximación de contorno. Como salida devuelve la variable "contornos" con la lista de todos los contornos de la imagen. Cada contorno indivi-

dual es una matriz Numpy de coordenadas (x, y) de puntos (Mordvintsev y Abid, 2014). La Figura 23 muestra un ejemplo de código de como identificar contornos utilizando la función cv2.findContours().

```
import numpy as np
import cv2

im = cv2.imread('test.jpg')
imgray = cv2.cvtColor(im,cv2.COLOR_BGR2GRAY)
ret,thresh = cv2.threshold(imgray,127,255,0)
image, contours, hierarchy = cv2.findContours(thresh,cv2.RETR_TREE,cv2.CHAIN_APPROX_
↪SIMPLE)
```

Figura 23: Ejemplo de uso de la función cv2.findcontours(). (Mordvintsev y Abid, 2014)

.

Como un segundo ejemplo de la librería Open CV, la función cv2.moments() devuelve una variable tipo diccionario de todos los valores de momento calculados a partir del contorno de un objeto. El cálculo de los momentos ayuda a calcular algunas características del contorno detectado en la imagen, como es el centro (coordenadas del centroide) de su masa o el área de esta. La Figura 24 muestra un ejemplo de código del uso de esta función para calcular el contorno, momento y centroide de un objeto reconocido dentro de una imagen.

```
import cv2
import numpy as np

img = cv2.imread('star.jpg',0)
ret,thresh = cv2.threshold(img,127,255,0)
contours,hierarchy = cv2.findContours(thresh, 1, 2)

cnt = contours[0]
M = cv2.moments(cnt)
print M
cx = int(M['m10']/M['m00'])
cy = int(M['m01']/M['m00'])
```

Figura 24: Ejemplo de uso de la función cv2.moments(). (Mordvintsev y Abid, 2014)

.

Pytorch.

Pytorch es una biblioteca de clases y métodos para proyectos de aprendizaje profundo. Por su flexibilidad, se ha convertido en una de las herramientas de ML esenciales para la comunidad científica en la obtención de modelos de aprendizaje (Stevens et al., 2020).

Pytorch es una biblioteca desarrollada por Facebook para la implementación de Redes Neuronales. Una de las principales ventajas de Pytorch es que calcula los gradientes de los pesos de cada capa de la red, a la vez que aplica el modelo sobre los datos de entrenamiento. Otras librerías necesitan calcular de manera estática los gradientes y aplicar posteriormente el algoritmo. Otra de las ventajas de esta librería es que permite crear módulos personalizados, lo que facilita el diseño de la red de acuerdo a la problemática presentada (Cerezo Sánchez et al., 2020).

En (Novac et al., 2022) se presenta una evaluación de la librería PyTorch respecto a TensorFlow. Estas son las dos bibliotecas más utilizadas para diseñar redes neuronales, las cuales son a su vez ampliamente utilizadas en la inteligencia artificial. Lo anterior, es debido a la facilidad de uso, documentación disponible, facilidad de integración, tiempos de entrenamiento, precisión y tempo de ejecución. En las Figuras 25, 26 y 27, se presentan los resultados de las pruebas de tiempo de ejecución, evolución de la eficiencia y evolución del error de estas dos librerías.

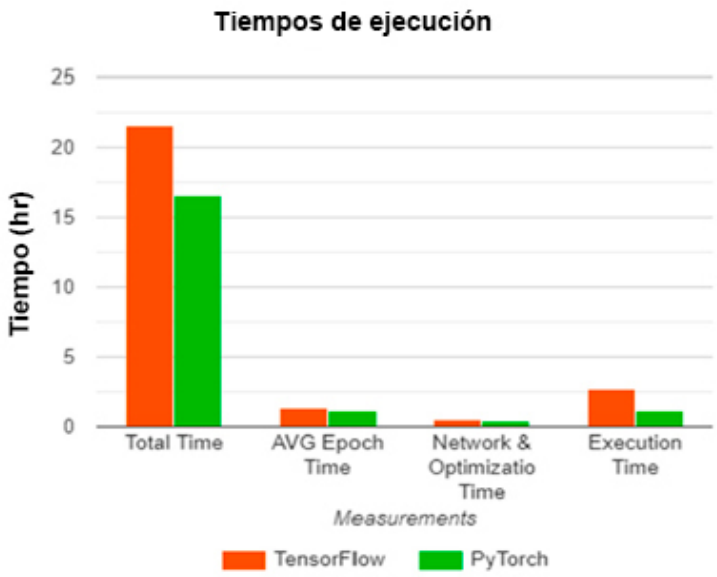


Figura 25: Tiempos de ejecución. (Novac et al., 2022).

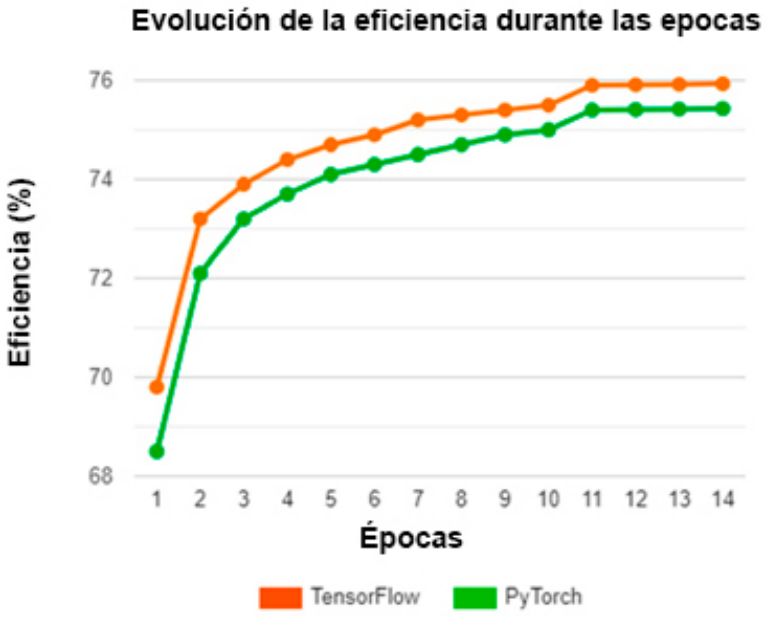


Figura 26: Eficiencia durante épocas. (Novac et al., 2022).

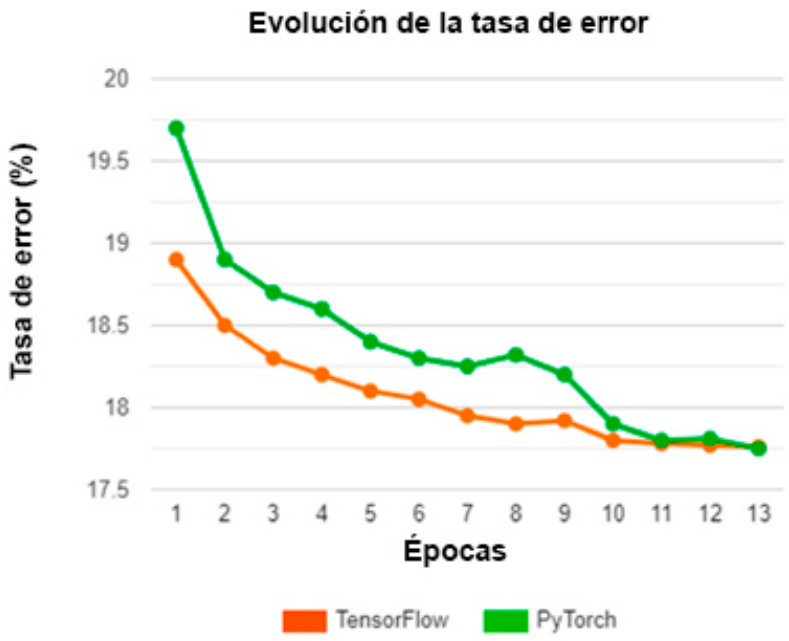


Figura 27: Error durante épocas. (Novac et al., 2022).

3.3. Inteligencia artificial (IA)

La IA es una de las ramas de las ciencias de la computación que más interés ha despertado en la actualidad, debido a su enorme campo de aplicación. La búsqueda de mecanismos que nos ayuden a comprender la inteligencia y realizar modelos y simulaciones de estos, es algo que ha motivado a muchos científicos a elegir esta área de investigación.

En la revisión de la bibliografía se encuentran diferentes propuestas para definir la IA. Sin embargo, no existe una definición que no haya sido debatida y sea aceptada totalmente. Por ejemplo, en (Rouhiainen, 2018) la definen como la capacidad de las computadoras para usar algoritmos, aprender de los datos y utilizar lo aprendido en la toma de decisiones tal y como lo haría un ser humano.

La IA es un campo diverso de investigación, que involucra diferentes subcampos esenciales para su desarrollo. Estos incluyen, entre otros, redes neuronales, lógica difusa, computación evolutiva y métodos probabilísticos. La Figura 28 muestra algunas áreas de aplicación de la IA.

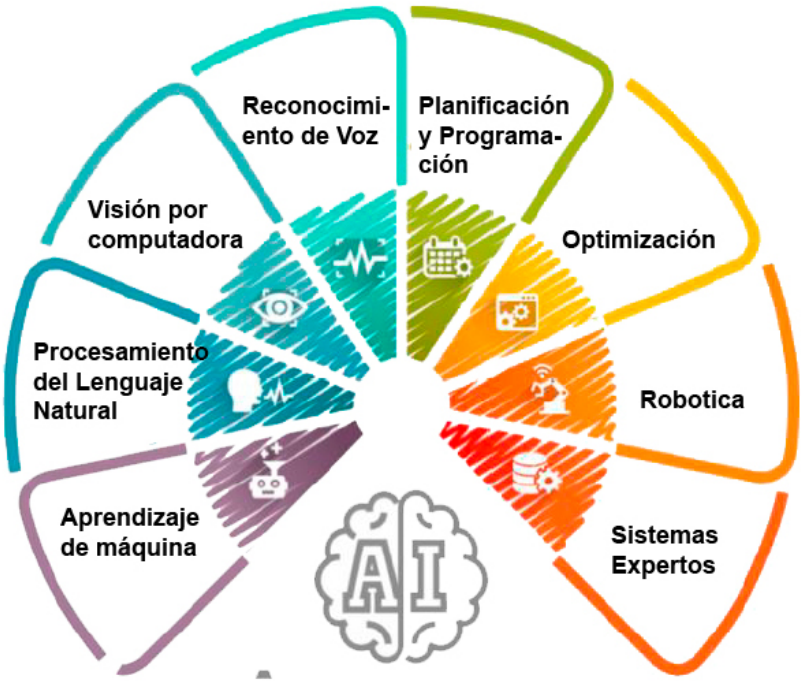


Figura 28: Ejemplos de aplicaciones de la IA. (Rouhiainen, 2018).

Redes Neuronales Artificiales (RNA).

Las RNA tratan de emular el comportamiento del cerebro humano, que aprende a través de la experiencia y la extracción de conocimiento genérico a partir de un conjunto de datos (Beltrán Barba, 2022). Existen diferencias importantes entre el cerebro biológico y el creado en computadores convencionales. Por ejemplo, las computadoras trabajan con microprocesadores que son muy rápidos y son capaces de ejecutar instrucciones complejas de forma fiable, y por otro lado, el cerebro está formado por un número mucho mayor de neuronas, con interacciones mucho más complejas que las modeladas en una computadora. La Figura 29 muestra las partes de una neurona biológica.

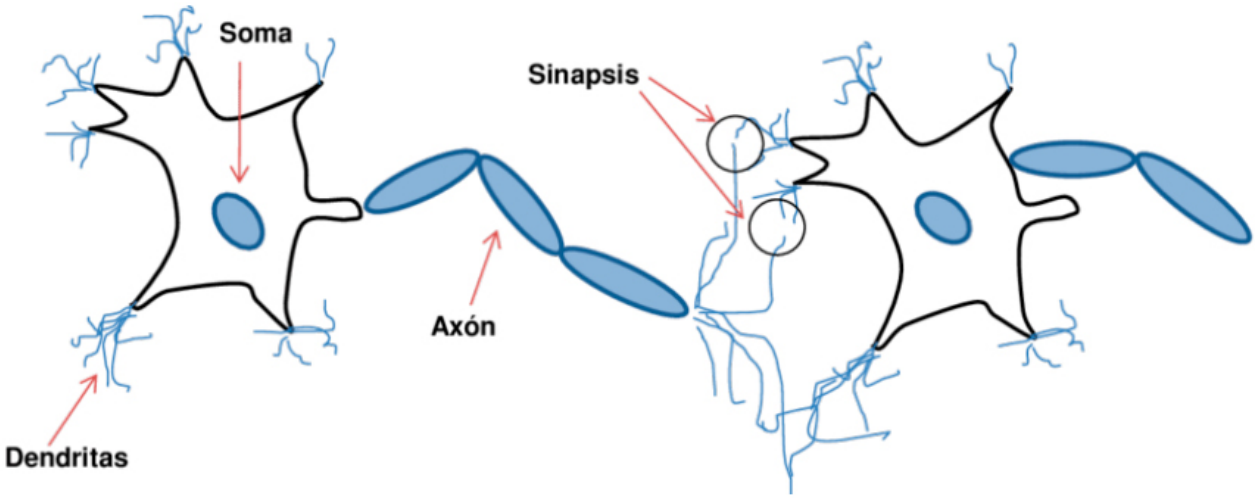


Figura 29: Partes de una neurona biológica. (Beltrán Barba, 2022)

Las RNA se basan en cálculos conexionistas basados en datos. Se componen de elementos informáticos muy simples que están interconectados de una manera que se parece más o menos al tejido nervioso humano. Las redes neuronales se pueden entrenar y se adaptan particularmente bien a los problemas de reconocimiento de patrones (ver Figura 30) y predicción con la ayuda de numerosas muestras de entrenamiento (Pietikäinen y Silven, 2022).

Actualmente, las RNA son utilizadas en el control de sistemas con dinámicas complejas, especialmente en sistemas no lineales que varían en el tiempo y que resultan complejos de regular

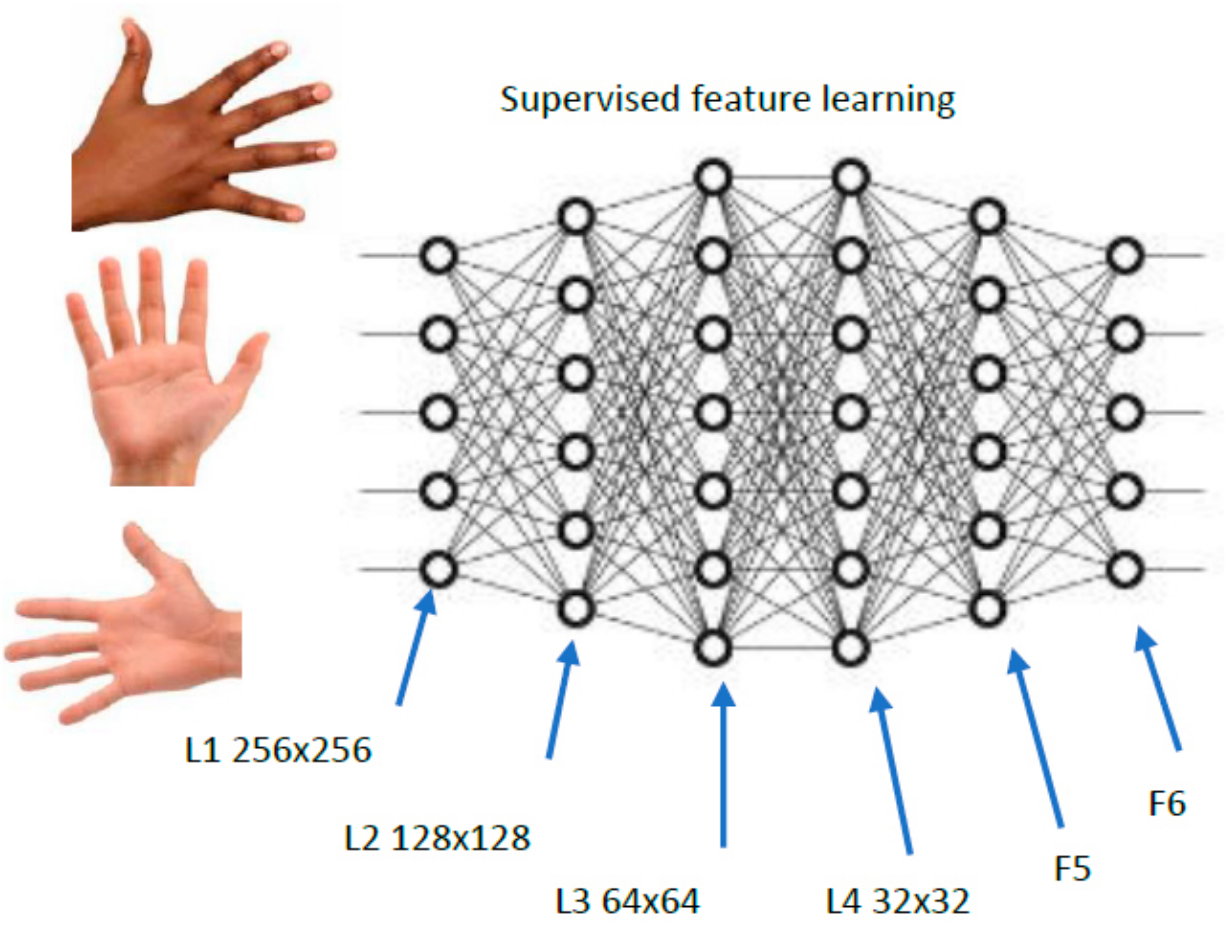


Figura 30: Reconocimiento de patrones con red neuronal artificial. (Pietikäinen y Silven, 2022).

con métodos convencionales (Gachet Páez y Valverde Gil, 2007). En el campo de la robótica, una aplicación de las redes neuronales es en tareas de percepción a través de visión artificial. En este campo, se utilizan para la detección y clasificación de los objetos presentes en el entorno del robot (Perez et al., 2018).

Aprendizaje Automático (ML).

El ML es la rama de la IA enfocada en la programación de sistemas que tienen que ser capaces de aprender de los datos. El término ML fue acuñado por A. Samuel en 1959. Lo definió como el campo de estudio que otorga a las computadoras la capacidad de aprender sin ser programadas explícitamente (Janiesch et al., 2021).

En (Singh et al., 2021) se presenta un amplio panorama de la división y aplicaciones del ML y se resume en la Figura 31.

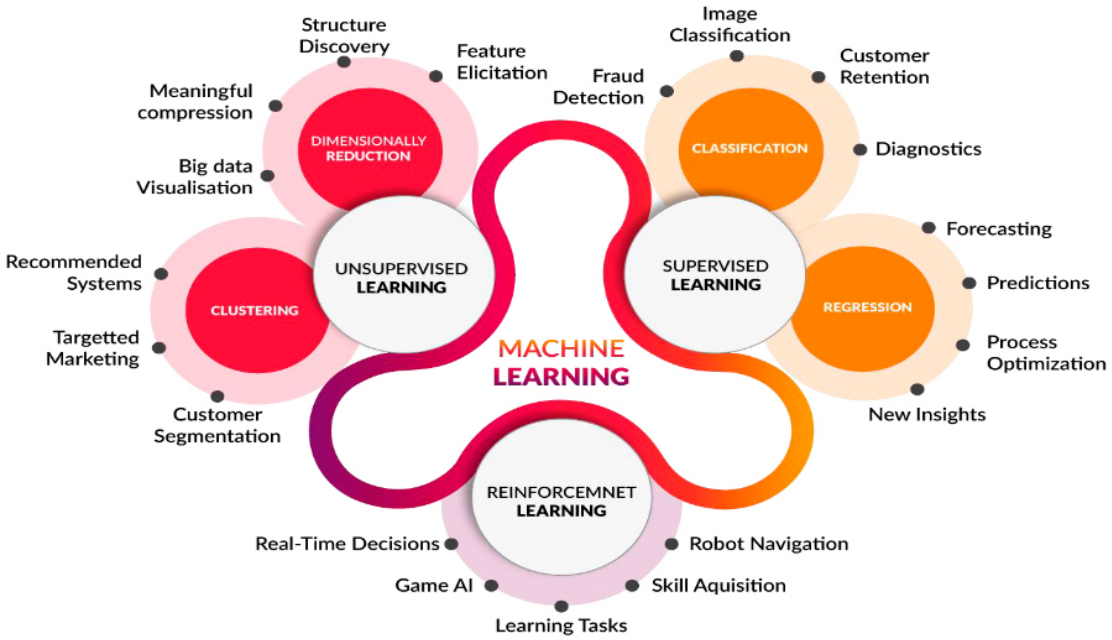


Figura 31: Aplicaciones del ML. (Singh et al., 2021).

Se aplica en campos como robótica, asistentes personales virtuales, juegos por computadora, reconocimiento de patrones, procesamiento de lenguaje natural, minería de datos, predicción de tráfico, redes de transporte en línea, recomendación de productos, predicciónes del mercado de acciones, diagnóstico médico, predicción de fraude en línea, asesoramiento agrícola, refinamiento de resultados de motores de búsqueda, BoT Chatbots para atención al cliente en línea, filtrado de Spam de correo electrónico, sistemas de vigilancia por video, reconocimiento facial entre muchas otras aplicaciones (Ray, 2019).

Generalmente se ocupa de tres tipos de problemas, a saber: clasificación, regresión y agrupación. Dependiendo de la disponibilidad de tipos y categorías de datos de entrenamiento, es posible que se deba seleccionar entre las técnicas disponibles de "aprendizaje supervisado", "aprendizaje no supervisado", "aprendizaje semisupervisado" y "aprendizaje por refuerzo" para aplicar el algoritmo de aprendizaje automático apropiado (Ray, 2019).

Aprendizaje profundo (DL).

El DL es un subcampo específico del aprendizaje automático (ver Figura 32). El DL moderno implica decenas o incluso cientos de capas sucesivas de representaciones y todas aprenden automáticamente a partir de la exposición a los datos de entrenamiento. Mientras tanto, otros enfoques del ML tienden a centrarse en aprender solo una o dos capas de representaciones de los datos; por lo tanto, a veces se les llama aprendizaje superficial (Chollet, 2021).

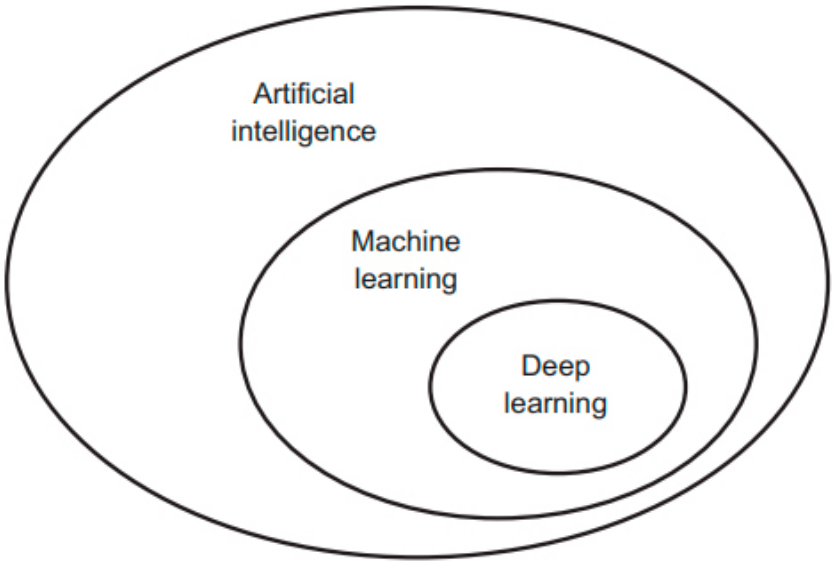


Figura 32: IA, aprendizaje automático y aprendizaje profundo. (Chollet, 2021).

El DL permite que los modelos computacionales de múltiples capas de procesamiento aprendan y representen datos con múltiples niveles de abstracción, imitando cómo el cerebro percibe y comprende la información multimodal, capturando así implícitamente estructuras complejas

de datos a gran escala. El DL es una rica familia de métodos que abarca redes neuronales, modelos probabilísticos jerárquicos y una variedad de algoritmos de aprendizaje de funciones supervisadas y no supervisadas (Voulodimos et al., 2018).

Las redes neuronales convolucionales, son un modelo computacional para la implementación del DL. Están compuestas por capas convolucionales, capas de agrupación y capas totalmente conectadas (Mu y Zeng, 2019). La Figura 33 muestra un diagrama de la arquitectura de una CNN.

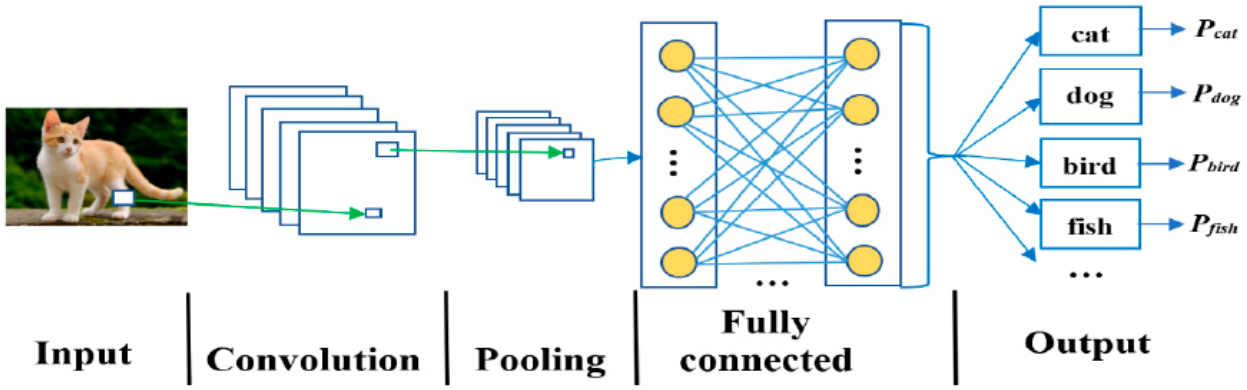


Figura 33: Ejemplo de la estructura de una Red Neuronal Convolucional. (Mu y Zeng, 2019).

La convolución es una operación matemática comúnmente utilizada en el área de procesamiento de imágenes y se realiza entre dos matrices (la entrada y el filtro). El filtro se desliza a lo largo y ancho de la imagen de entrada, aplicando una multiplicación elemento por elemento de ambas matrices, para su posterior suma. El resultado de este proceso es una nueva matriz conocida como mapa de características (Ramírez Zavalza et al., 2020). En la Figura 34 se muestra un ejemplo del proceso de convolución.

El aprendizaje profundo se utiliza en el procesamiento de imágenes digitales para resolver problemas difíciles (por ejemplo, colorización, clasificación, segmentación y detección de imágenes). Los métodos del aprendizaje profundo, como las CNNs (CNNs), mejoran principalmente el rendimiento de la predicción. La clasificación de imágenes es un excelente ejemplo de esto (Krizhevsky et al., 2017, O'Mahony et al., 2020).

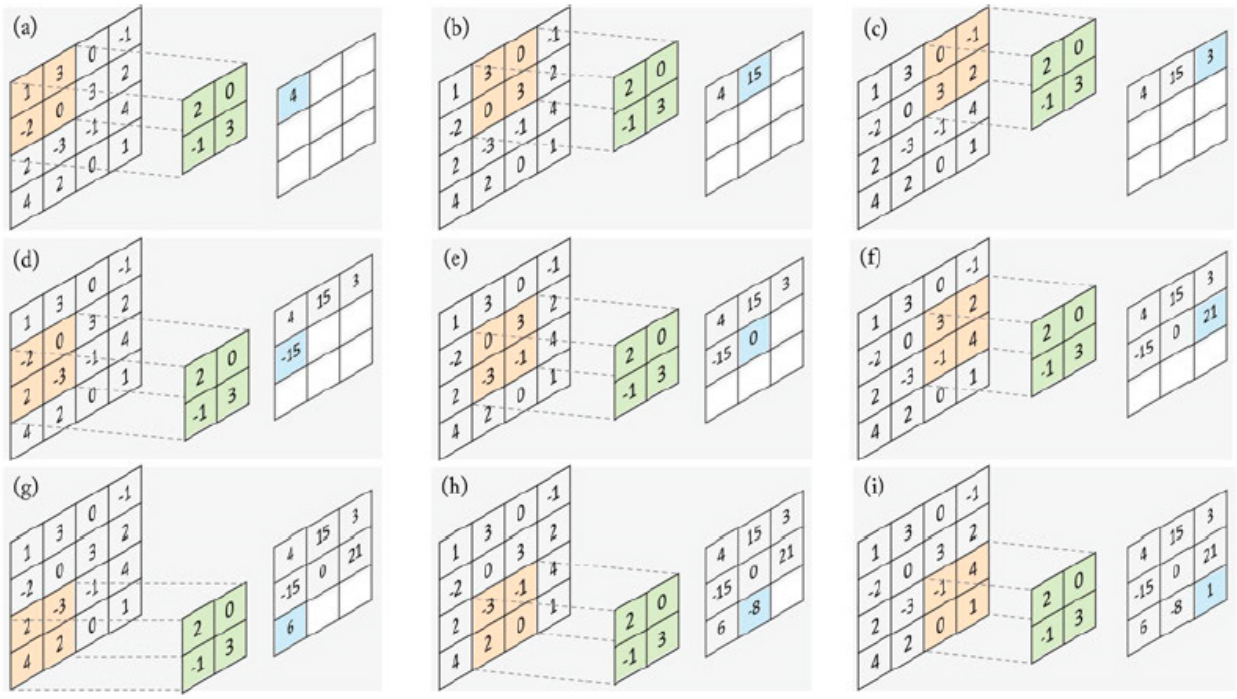


Figura 34: Ejemplo del proceso de convolución. (Ramírez Zavalza et al., 2020)

Otras aplicaciones actuales del aprendizaje profundo incluye: PNL (clasificación de oraciones, traducción, etc.), procesamiento de datos visuales (visión por computadora, análisis de datos multimedia, etc.), procesamiento de voz y audio (mejora, reconocimiento, etc.), análisis de redes sociales y salud (Pouyanfar et al., 2018). La Figura 35 muestra algunos ejemplos de las principales aplicaciones del aprendizaje profundo.

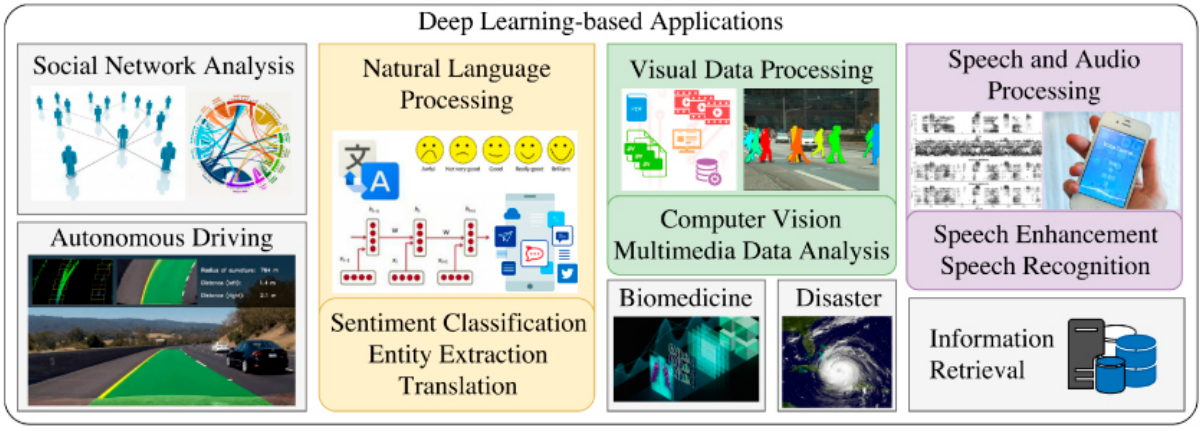


Figura 35: Principales aplicaciones del aprendizaje profundo. (Pouyanfar et al., 2018).

3.4. Visión computacional

La visión por computadora, es el campo de la informática que se ocupa de permitir que las computadoras, dispositivos o máquinas en general, puedan ver, interpretar, manipular y comprender el ambiente que las rodea y pueda tomar acciones ya sea en forma guiada o autónoma, por medio del desarrollo de algoritmos y técnicas para la obtención, procesamiento y análisis de imágenes y videos que pueden provenir de dispositivos como cámaras de video, impagenes fijas, obtenidas por scanner entre otros (Gollapudi, 2019).

La precisión de las aplicaciones de Visión Computacional está determinada por qué tan bien se interpretan las imágenes o los videos. Para esto, se deben cuidar algunos aspectos importantes de las imágenes que pueden hacer que todo el proceso de interpretación de imágenes sea complejo. En la Figura 36 se muestran algunas imágenes difíciles de procesar: a) Imágenes con ilusiones ópticas, b) Imágenes en diferentes ángulos y c) Imágenes en movimiento.

Figura 36: Ejemplos de imágenes difíciles de procesar. (Gollapudi, 2019)

La visión por computadora se ha vuelto común en áreas como la industria automotriz, especialmente para mejorar la seguridad y la funcionalidad de los vehículos nuevos. También se usa ampliamente en el sector biomédico y de la salud, haciendo posible la atención preventiva para el cáncer y otras enfermedades genéticas. Las tiendas en línea y físicas están utilizando la visión por computadora para mejorar la experiencia del cliente, brindar alternativas competitivas y optimizar los procesos.

3.5. Segmentación semántica

La segmentación semántica trata el problema de clasificar uno o todos los objetos presentes en una imagen a nivel de píxeles, asignando un color diferente a cada clase de objeto identificado. Por ejemplo: animales, automóviles, personas, arboles, etc. (Minaee et al., 2021). La Figura 37 muestra la segmentación de una imagen usando el modelo de aprendizaje profundo DeepLabv3.

Después de la segmentación, una imagen está compuesta por un conjunto de grupos de píxeles donde cada grupo representa una región. Estos datos segmentados se pueden transformar en una forma compacta que facilita la descripción de la región y ayuda a comparar y combinar con un patrón dado. Al pensar en un contorno, uno considera un objeto de interés y una línea que lo rodea como un contorno ideal.

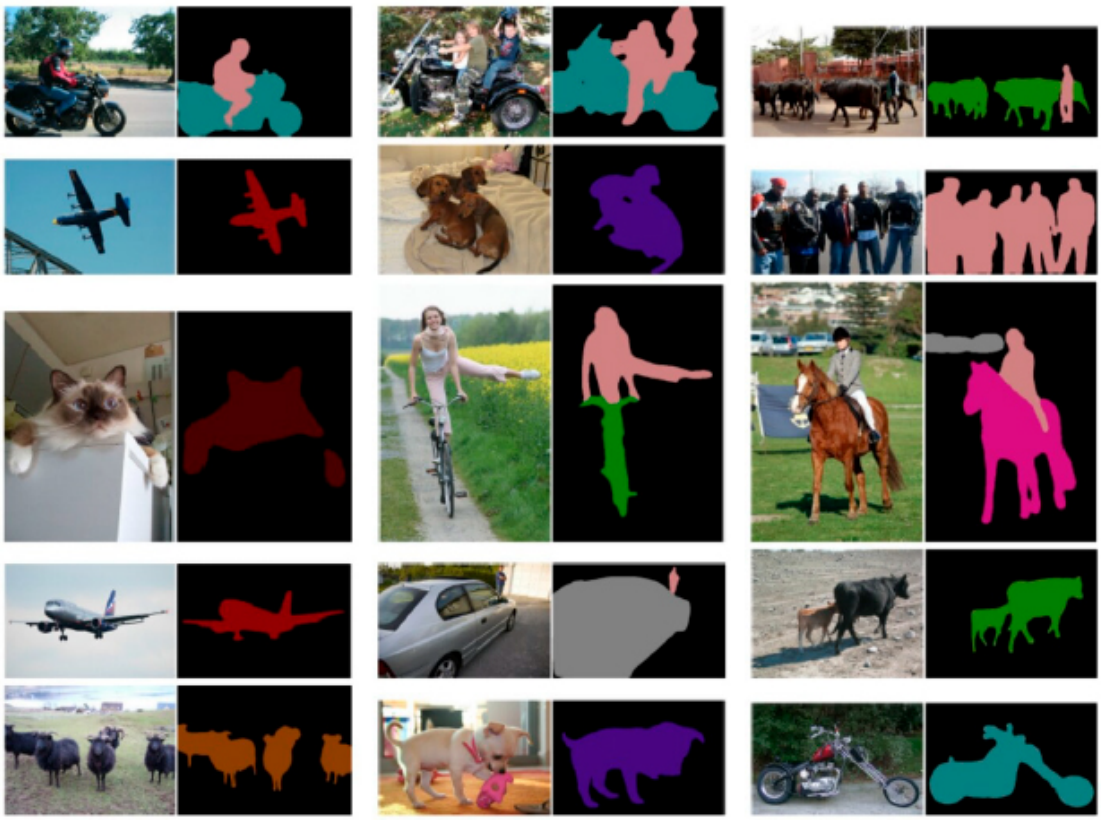


Figura 37: Resultado de segmentación semántica usando el modelo DeepLabv3. (Minaee et al., 2021)

3.6. Sistemas difusos

La teoría difusa se basa en la teoría de los conjuntos difusos introducida por el profesor Lotfi Asker Zadeh en 1965 (Zadeh, 1999). En la teoría de conjuntos difusos, un elemento tiene un grado de pertenencia (Tremante y Brea, 2014). Un miembro puede pertenecer al conjunto en un mayor o menor grado y se define asignando a cada elemento del mismo, un número entre 0 y 1, el cuál indica el grado de membresía o pertenencia al conjunto (Aguilar Jáuregui y Peredo Macías, 1999), ver Figura 38.

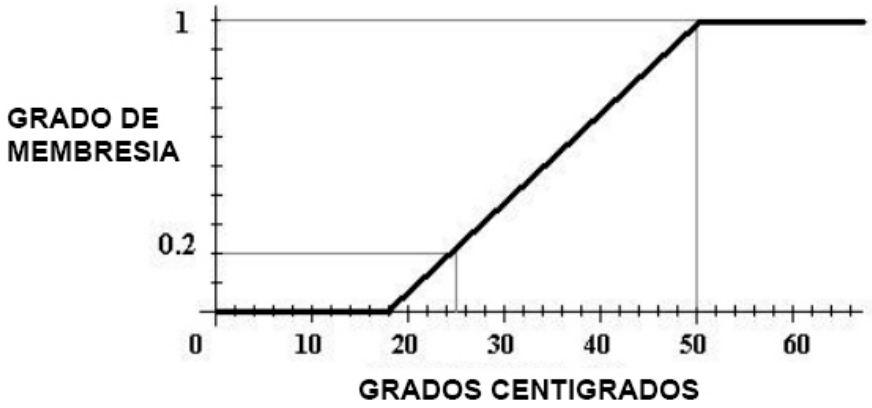
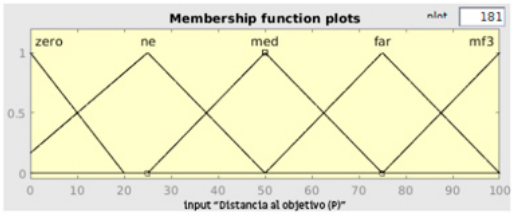


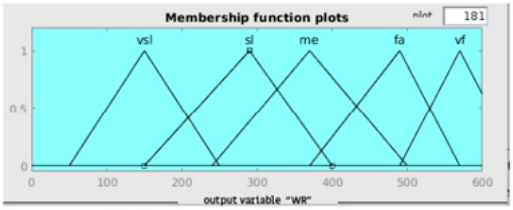
Figura 38: Ejemplo de función de membresia. (Aguilar Jáuregui y Peredo Macías, 1999)

La lógica difusa busca crear aproximaciones matemáticas en la resolución de ciertos tipos de problemas. Pretendiendo producir resultados exactos a partir de datos imprecisos, por lo cuál son particularmente útiles en aplicaciones electrónicas o computacionales (Morales Luna, 2002). La lógica difusa pretende emular la estrategia de control que seguiría un experto humano en el control manual de un proceso (Santos, 2011).

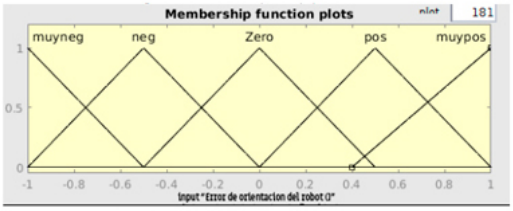
Dentro de la robótica móvil, una aplicación de la lógica difusa es en tareas de navegación. Por ejemplo, en (Cristino et al., 2019b) proponen la implementación de un sistema de control difuso tipo Mamdani, para la navegación y evasión de obstáculos, sobre una placa Arduino UNO©. La Figura 39 muestra los conjuntos y reglas difusas implementados para el sistema de navegación.



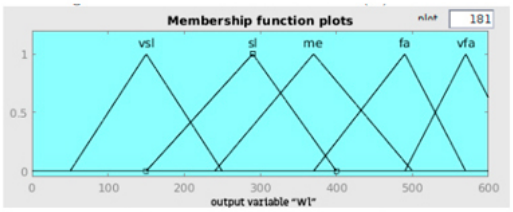
Distancia al objetivo



Potencia de motor derecho



Error de orientación



Potencia de motor izquierdo

alpha\P	zero	ne	med	far	veryfar
muyneg	Rvsl	Rvsl	Rsl	Rm	Rm
	Lm	Lm	Lf	Lf	Lvf
neg	Rvsl	Rvsl	Rvs	Rs	Rsl
	Lsl	Lsl	Lm	Lf	Lf
Zero	Rvsl	Rsl	Rm	Rf	Rvf
	Lvsl	Lsl	Lm	Lf	Lvf
pos	Rsl	Rsl	Rm	Rf	Rf
	Lvsl	Lvsl	Lvsl	Lsl	Lsl
muypos	Rm	Rm	Rf	Rf	Rvf
	Lvsl	Lvsl	Lsl	Lm	Lsl

Reglas difusas de navegación

Figura 39: Conjuntos, funciones de membresia y reglas difusas. (Cristino et al., 2019b)

4. DESARROLLO DEL PROTOTIPO ROBÓTICO

El desarrollo del prototipo se dividió en seis etapas: 1) Diseño mecatrónico, 2) Creación de un conjunto de imágenes de entrenamiento, 3) Diseño y entrenamiento de la red neuronal, 4) Implementación del sistema de reconocimiento de las botellas, 5) Implementación del algoritmo de aproximación y 6) Registro de las operaciones del sistema.

4.1. Diseño Mecatrónico

Como primera fase del proyecto, se diseñó y construyó un prototipo de robot móvil sobre un chasis tipo oruga, dotándolo con los componentes de electrónicos necesarios, para que trabaje de forma autónoma y con alto grado de eficiencia. En la Figura 40 se muestra un diagrama de los componentes de hardware del robot y se detallan a continuación:

- Chasis con llantas sobre oruga. Es una estructura de tipo comercial fabricada en aleación de aluminio que soporta la base para la electrónica integrando un sistema de locomoción que consiste dos motorreductores DC de 9-12 [V] y una banda de desplazamiento tipo oruga.

- Módulo de cámara IMX219. Provee al sistema de los fotogramas necesarios para realizar el procesamiento digital durante el reconocimiento y aproximación.

- NVIDIA Jetson Nano© B01. cuya función es recibir las imágenes de la cámara, sobre las cuales aplica un modelo entrenado de red neuronal convolucional, para reconocer las botellas presentes. Posteriormente, identifica los contornos de las botellas, calcula el área de cada una, y obtiene el centroide de la botella cuyo contorno tiene mayor área. Finalmente, la NVIDIA Jetson Nano© transmite el área y el centroide a una tarjeta Arduino UNO©, vía protocolo Serial.

- Arduino UNO©. Recibe el área y coordenadas del centroide, y aplica un sistema de inferencia difusa, para calcular la potencia que debe aplicar a cada motor dependiendo de su distancia y posición angular, con el objetivo de que el robot se aproxime a la botella reconocida con mayor área.

Figura 40: Componentes electrónicos del prototipo propuesto.

- Puente H L298N. Recibe señales PWM desde la tarjeta Arduino UNO© y administra el voltaje necesario a los motores.
- 2 Power Bank 5 y 2.5 [mAh]. La primera Power Bank (5 [mAh]) suministra energía a la NVIDIA Jetson Nano©, y la segunda (2.5 [mAh]) lo hace con la pantalla de 7 pulgadas.
- 2 pilas de litio 18650 3.7 [V]. Suministran energía al Puente H L298N.
- Pantalla de 7 pulgadas. Se utiliza como interfaz gráfica entre el usuario y el sistema Operativo.
- Base para la electrónica. Se utilizaron dos laminas de acrílico de tipo comercial que se adaptaron para contener los componentes electrónicos.

- Sensor ultrasónico. Determina la aproximación mínima de 15cm a la que el motor ya se encuentra cerca del objetivo y debe detenerse.
- Teclado y mouse inalámbrico. Permiten realizar ordenes al sistema operativo como es la ejecución del programa y la orden para detener el proceso.

La Figura 41 muestra una imagen con todos sus componentes instalados y funcionando en pruebas de reconocimiento y navegación en ambiente controlado.

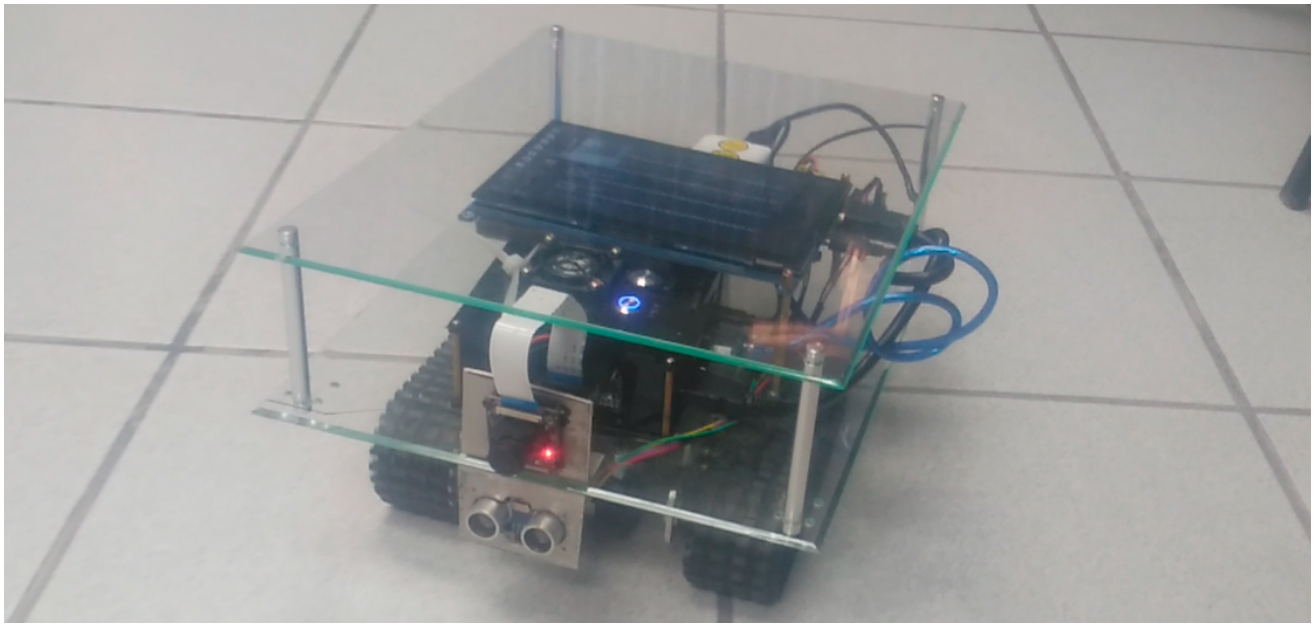


Figura 41: Robot ensamblado con todos sus componentes.

4.2. Generación del conjunto de imágenes de entrenamiento

Para el entrenamiento y pruebas de reconocimiento de la red, se capturaron 532 imágenes, con el robot estático y en movimiento, utilizando cuatro tipos diferentes de botellas de plástico, en un ambiente controlado. Las imágenes se almacenaron con un tamaño de 320x240 píxeles, en formato JPG (ver Algoritmo 4.1).

Algorithm 4.1: Programa para tomar fotos con el robot en movimiento

```
   entrada: Cámara
   salida  : Imagen capturada en movimiento
1  Tecla = 13
2  while Tecla <> ESC do
       Techa = Lee del teclado
       if Tecla = 8 then
           Envía movimiento AL FRENTE
       if Tecla = 6 then
           Envía movimiento DERECHA
       if Tecla = 4 then
           Envía movimiento IZQUIERDA
       if Tecla = 2 then
           Envía movimiento REVERSA
       if Tecla = 5 then
           Envía movimiento CERO
       Envía cadena a la placa Arduino por PuertoSerial
   procedure GiroMotores(potDer, senDer, potIzq, senIzq)
       IN1, IN2, IN3, IN4 = LOW
       ENA = potIzq
       ENB = potDer
       if senIzq = 1 then
           IN1 = HIGH
       if senDer = 1 then
           IN4 = HIGH
   end procedure
```

Para cada imagen, se realizó manualmente la segmentación de las botellas que aparecen en la misma, utilizando el software VGG Image Annotator (VIA) versión 2.0.111, disponible gratuitamente en linea. La Figura 42 muestra la ventana de este programa.

Figura 42: Programa VGG Image Annotator.

Posteriormente, se generó un archivo en formato JSON, con la definición de los polígonos que corresponden a los contornos de las botellas segmentadas y a partir de estos contornos, se aplicó una rutina escrita en Python, para generar archivos en formato PNG, con las imágenes segmentadas nombradas como "máscaras" (ver Algoritmo 4.2).

Algorithm 4.2: Generación de archivos en formato PNG con las imágenes segmentadas

```
entrada: Dataset de entrenamiento
salida  : Imágenes PNG con botellas segmentadas
ruta = '/Dataset/Imagenes/'
archivo_json = Abre archivo ruta + 'poligonos.json'
datos_json = Carga archivo json
for imgId in rango(1, imgs) do
    nombre_img = Lee nombre de la imagen imgId
    regiones = Lee lista de regiones en imgId
    imagen = Lee y abre imagen ruta + nombre_img
alto = Número de filas de imagen
ancho = Número de columnas de imagen
mascara = Crea matriz de ancho X alto X 3
for i in rango(1,numero de regiones-1) do
    region = Lee la región i de regiones
    clase = Lee la clase asociada a region
    if clase = 'botella' then
        R,V,A = 128,0,0
    else
        R,V,A = 0,0,0
    if tipo_region = 'polygon' then
        total_puntos = Lee cantidad de puntos de region
        contornos = Crea matriz de total de puntos elementos
        for i in rango(1,total_puntos-1) do
            x = Lee coordenada x del punto i
            y = Lee coordenada y del punto i
            contorno[i] = crea punto(x,y)
        for i in rango(1,ancho-1) do
            for j in rango(1,alto-1) do
                if puntos esta en area de contorno' then
                    mascara[j,i,0] = A
                    mascara[j,i,1] = V
                    mascara[j,i,2] = R
Guarda mascara como archivo PNG en ruta
```

En estos archivos, el color de cada píxel corresponde a la clase de objeto al que pertenece. Se eligió el color rojo (128,0,0 en formato RGB), para las botellas, y color negro (0,0,0), para el resto de los objetos, considerados como fondo. En la Figura 43 se muestran ejemplos de imágenes y su máscara correspondiente.

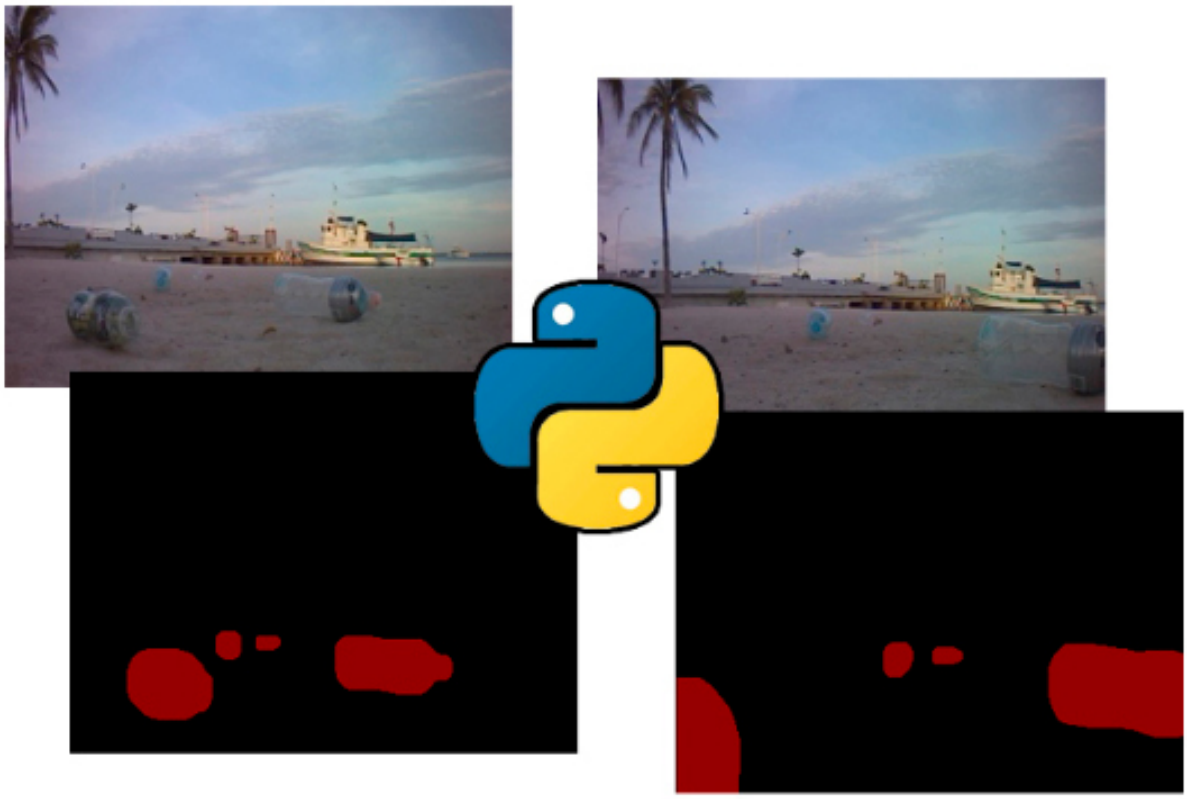


Figura 43: Algoritmo, imagen original e imagen segmentada.

A las máscaras se les sustituyó los tres canales de color de cada pixel, por un valor único que representa la clase de objeto a la que pertenece. Finalmente, las imágenes, tanto las originales como las máscaras, se almacenaron en formato NPY. Del total de imágenes, se utilizó el 75 % para entrenamiento y el 25 % para pruebas de reconocimiento.

Posteriormente, las intensidades de color de las imágenes originales se normalizaron con valores entre 0 y 1. Finalmente, las imágenes normalizadas, tanto las originales como las máscaras, se almacenaron de nuevo en formato NPY (ver Algoritmo 4.3 y 4.4).

Algorithm 4.3: Generación de archivos en formato PNG con las imágenes segmentadas

entrada: Imágenes segmentadas
salida : Imágenes en formato NUMPY
ruta = '/Dataset/Imagenes'

archivos = Lee lista de archivos en ruta
for *archivo,* **in** rango(1,archivos) **do**
    imagen = abre y carga imagen ruta+archivo+'.jpg'
    imagen = convierte formato BGR a RGB de imagen
    imagen = imagen/255.0
    guarda imagen en ruta con formato Numpy
color_etiqueta ← llama a mapa_color_etiqueta() Algoritmo 4.4

for *archivo,* **in** rango(1, archivos) **do**
    máscara = abre y carga imagen ruta + archivo + '.png'
    máscara = convierte formato BGR a RGB de máscara
    idx = ((máscara[:, :, 0] * 256 + máscara [:, :, 1]) * 256 + máscara [:, :, 2])
    máscara = ColorEtiqueta[idx]
    guarda máscara en ruta con formato Numpy

Algorithm 4.4: Función que regresa el color de la etiqueta

entrada: Imágenes segmentadas
salida : Imágenes en formato NUMPY
mapa_color = [[0, 0, 0],[128, 0, 0]] // Colores de las clases de objetos que se desea reconocer

color_etiqueta = crea tensor de 256^3elementos

for *i, mapa* **in** rango(1, mapa_color) **do**
    color_etiqueta[(mapa[0]*256+mapa[1])*256+mapa[2]] = i
Retorna color_etiqueta

4.3. Entrenamiento de la red

El reconocimiento de las botellas se implementó a través de segmentación semántica, aplicando una red neuronal tipo U-Net (Ronneberger et al., 2015). Para esto, se tomó como base el modelo ResNet18 con los pesos preentrenados. A este modelo base, se le eliminaron las dos últimas capas (FC y Softmax), y se le agregaron cuatro capas convolucionales transpuestas, combinadas con cuatro nuevas capas convolucionales.

En la Figura 44 se muestra la arquitectura de la red utilizada. La codificación para definir la arquitectura y el entrenamiento de la red neuronal, se basó en el trabajo presentado en (Sensio, 2020).

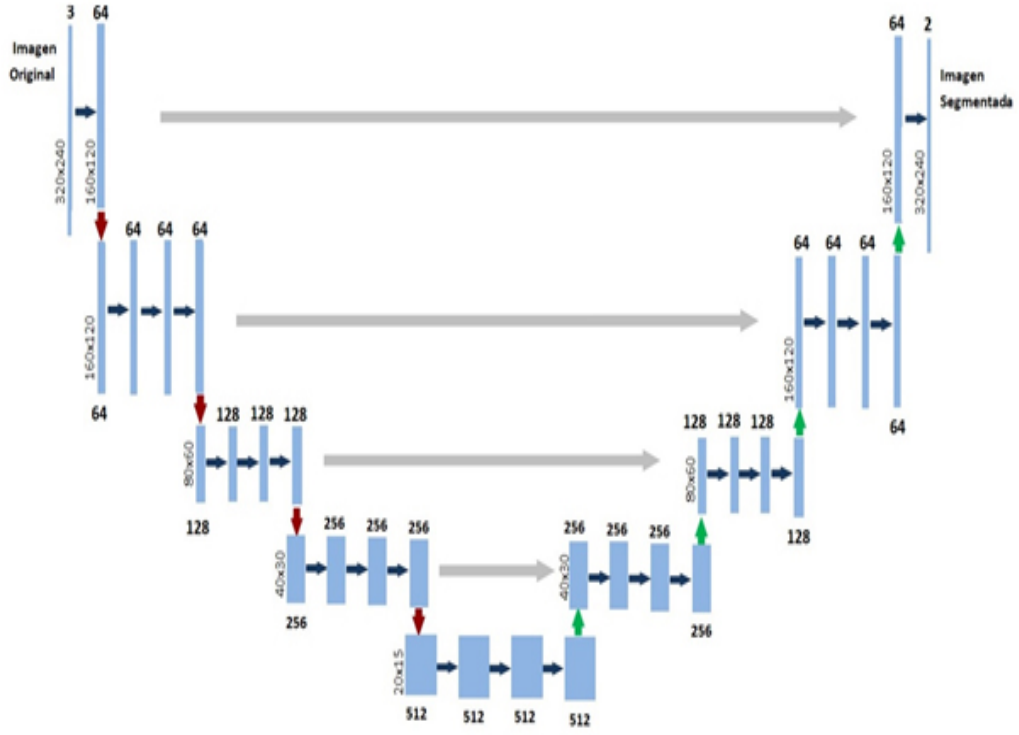


Figura 44: Arquitectura de red U-Net utilizada.

Para la ejecución de los entrenamientos de la red, se configuró un equipo con capacidad de cómputo paralelo, basado en la tarjeta GPU NVIDIA GeForce GTX Titan X, tarjeta madre Gigabyte H81M-H, Procesador Intel Core i7 3.6Ghz x8, Memoria RAM 16GB, Disco Duro SSD 240GB.

La implementación de la red y su entrenamiento, se realizó en lenguaje Python versión 3.9.7, utilizando las librerías Pytorch versión 1.9.0, Torchvisión versión 0.10.0 y OpenCV 4.5.5, sobre el sistema operativo Ubuntu 20.04. Los parámetros usados en el algoritmo de entrenamiento se muestran en la Tabla 1.

Función de optimización	Adam
Función de pérdida	BCEWithLogitsLoss
Factor de aprendizaje (lr)	0.0001
Factor de decaimiento (wd)	0.001
Tamaño de lote	10
Número de épocas	100

Tabla 1: Parámetros de entrenamiento de la red

Para la obtención de los parámetros finales, se realizaron 12 entrenamientos diferentes, evaluando combinaciones de los parámetros, durante 100 épocas cada uno. El tiempo de ejecución de cada entrenamiento fue de alrededor de 6 segundos por época, dando un total aproximado de 10 minutos por las 100 épocas. Se seleccionó el modelo generado que mostró los mejores resultados.

En la Figura 45 se muestra el comportamiento de la función de pérdida durante el entrenamiento, aplicada sobre las muestras de entrenamiento (línea azul), y sobre las muestras de prueba (línea verde). Al final de las 100 épocas, la función de pérdida para las muestras de entrenamiento fue de 0.026 y para las muestras de pruebas de 0.050.

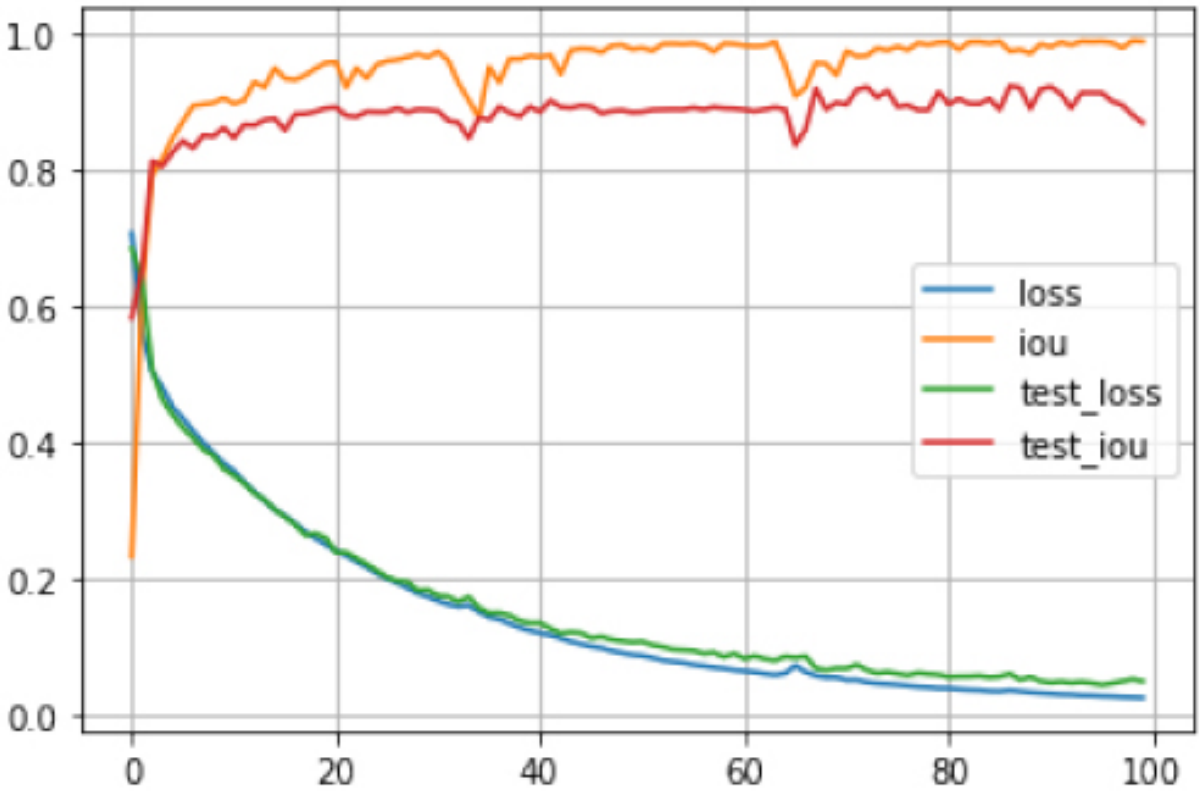


Figura 45: Comportamiento de la función de pérdida durante el entrenamiento.

4.4. Reconocimiento de las botellas

Una vez generado el modelo de la red entrenada, se almacenó en un archivo con formato Pytorch (.pt), y se cargó sobre la NVIDIA Jetson Nano© . El proceso de reconocimiento de las botellas sobre la NVIDIA Jetson Nano©, inicia con la captura de una imagen (frame) a través de la cámara, para pasarla al sistema de inferencia de la red.

A partir de la salida de la red, se obtiene la máscara inferida, correspondiente a la imagen de entrada. Posteriormente, a partir de la máscara inferida, se generan los contornos de las botellas reconocidas, a través de la librería OpenCV. Después se obtiene el área de cada contorno, y se calcula el centroide del contorno con área mayor (ver Algoritmo 4.5 y 4.6).

Algorithm 4.5: Sistema de reconocimiento sobre la Jetson NANO

```
Entrada: Imágenes segmentadas
Salida  : Imágenes en formato NUMPY
Usar Algoritmo 4.6 para Definir Clase HiloCamara
puerto_serial = crea y configura objeto de puerto serial
camara = crea y configura objeto camara
hiloCamara = crea objeto HiloCamara(camara)
inicia ejecución de hiloCamara
modelo = carga el modelo entrenado de la red
pasa el modelo a la GPU
activa el modo de inferencia del modelo
while Tecla ¡¿ ESC do
    if hiloCamara.ultima_imagen ≤ Nulo then
        imagen = hiloCámara.última_imagen
        imagen = convierte formato BGR a RGB de imagen
        imagen_torch = convierte imagen a formato Torch
        imagen_torch = permuta la tercera dimensión de imagen_torch como primera
        dimensión
        imagen_torch = imagen_torch.float()/255
        imagen_torch = agrega una dimensión a imagen_torch
        imagen_cuda = se transfiere imagen_torch a la GPU
        salida = Evalua imagen_cuda con el modelo. Extrae del lote la única imagen
        procesada.
        máscara = genera matriz con la clase inferida por pixel
        máscara = pasa máscara a la CPU en formato Numpy
        máscara = se filtra la clase de interés (botellas)
        contornos = genera contorno de las botellas en máscara con findContours
        áreaMayor = 0
        contornoMayor = Nulo
        for contornoencontornos : do
            área = calcula el área de contorno con contourArea de OpenCV
            if area > areaMayor then
                areaMayor = area
                contornoMayor = contorno
            if areaMayor > 0 then
                momentos = calcula los momentos de contornoMayor
            if moments = True then
                centroideX = calcula centroide de contornoMayor
    cadena = " < area > "+str(areaMayor)+" < cx > "+
    str(centroideX)+" < fin > "
    Envía cadena a la placa Arduino por PuertoSerial
hiloCamara.activo = Falso
sincroniza finalización con hiloCamara
libera recursos de Camara y PuertoSerial
```

Algorithm 4.6: Definir Clase HiloCamara

Entrada: Camara
Salida : Clase HiloCamara
Definir Clase HiloCamara:

function *Constructor*(camara)

camara = parametro camara

ultima_imagen = Nulo

activo = Verdadero

end function

function *Ejecucion*()

while *activo* = *true* **do**

ultima_imagen = Lee frame del buffer de la cámara

Pausa un milisegundo

end function

Lo anterior, considera que la botella reconocida con área mayor, es la que, posiblemente, se encuentra más próxima al robot, y por lo tanto se convierte en el objetivo de alcance del mismo. El tiempo requerido para completar el proceso de reconocimiento por la NVIDIA Jetson Nano©, es de aproximadamente 0.35 segundos, por lo que el robot es capaz de procesar tres imágenes por segundo, lo cuál resultó suficiente para lograr un desplazamiento fluido hacia las botellas.

En la Figura 46 se muestra un ejemplo de imagen de entrada, y la imagen segmentada por la red, con la generación del contorno y coordenadas del centroide de la botella con mayor área.

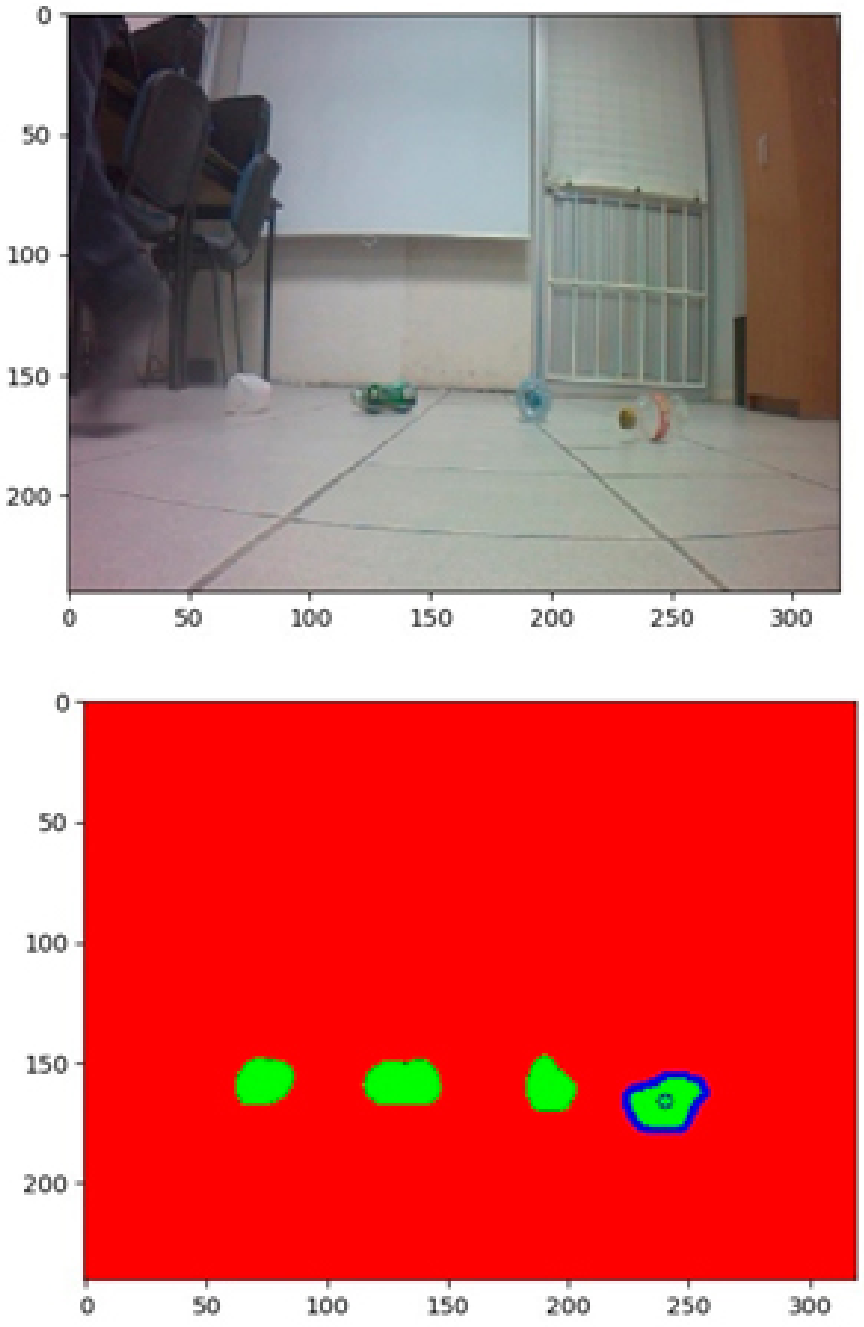


Figura 46: Reconocimiento de botellas y cálculo de centroide.

4.5. Algoritmo de aproximación

Para la aproximación a las botellas se diseño un sistema basado en lógica difusa. El objetivo de este sistema es calcular las potencias de ambos motores del robot, para su desplazamiento por tracción diferencial. La entrada para este proceso son el área de la botella objetivo (*área*)y la coordenada X del centroide de esta en la imagen (*posX*). Además, se dotó al robot de un sensor ultrasónico para determinar la distancia de la botella respecto al robot, para de esta forma determinar cuando se encuentra lo suficientemente cerca y detenerse.

Para esto, el área y el centroide de la botella se transmiten de la NVIDIA Jetson Nano© a la tarjeta Arduino Uno©, vía protocolo serial. Con estos datos, la tarjeta Arduino UNO© ejecuta el algoritmo basado en lógica difusa, implementado en C++. La salida del sistema difuso son las potencias (*potDer* y *potIzq*) de los motores, las cuáles se envían a estos a través de un puente H.

Para implementar este sistema, se definieron tres conjuntos difusos. Un primer conjunto define la función de pertenencia para la variable lingüística de entrada área (ver Figura 47). Un segundo conjunto para la variable de entrada posX (ver Figura 48), y un conjunto para la variable lingüística de salida potencia (ver Figura 49).

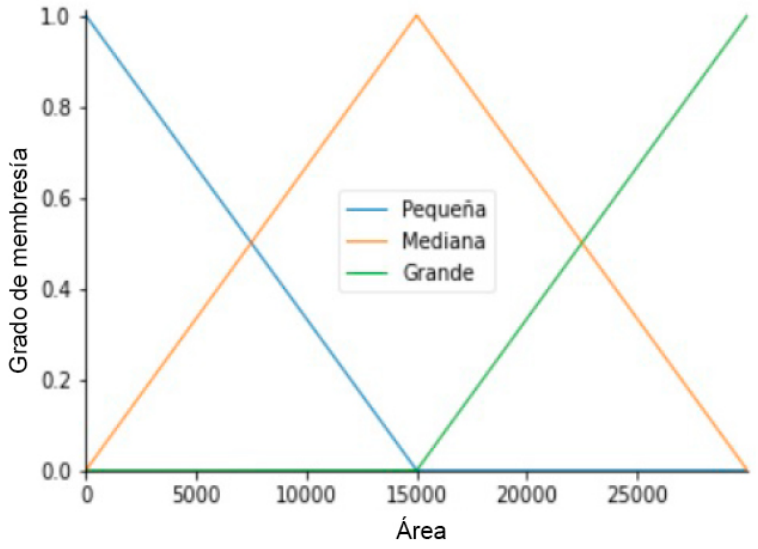


Figura 47: Conjuntos difusos para el área de la botella.

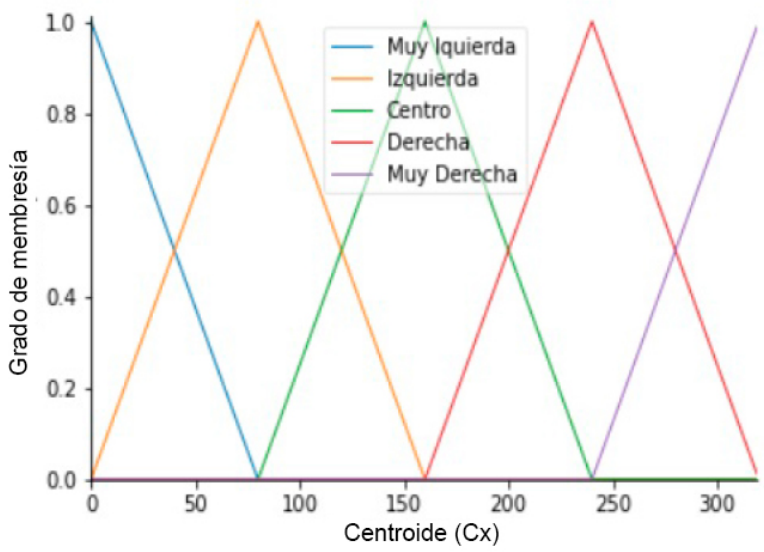


Figura 48: Conjuntos difusos para el centroide de la botella.

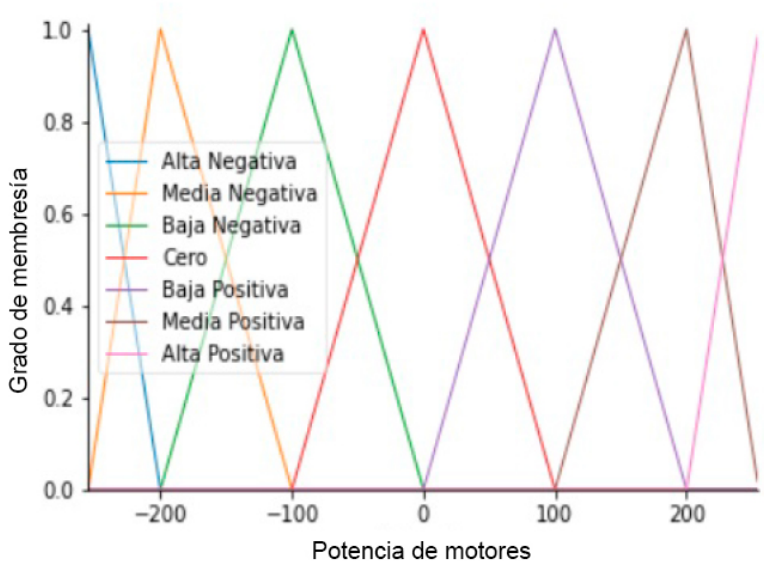


Figura 49: Conjuntos difusos para la Potencia.

En la Tabla 2, se muestra la definición de las reglas para la inferencia difusa de la potencia del motor derecho, y en la Tabla 3 se muestra lo correspondiente para el motor izquierdo.

El sistema de inferencia utilizado es de tipo Mamdani, y para la defusificación se aplica el método Singleton (ver Algoritmos 4.7, 4.8 y 4.10) .

Área \ Pos X	Muy Izq	Izquierda	Centro	Drerecha	Muy Der
Pequeña	Media	Media	Alta	Baja	Baja
Mediana	Baja	Baja	Media	Cero	Cero
Grande	Baja	Baja	Cero	Cero	Baja

Tabla 2: Reglas de inferencia para la potencia del motor derecho

Área \ Pos X	Muy Izq	Izquierda	Centro	Drerecha	Muy Der
Pequeña	Baja	Baja	Alta	Media	Media
Mediana	Cero	Cero	Media	Baja	Baja
Grande	Baja	Cero	Cero	Baja	Baja

Tabla 3: Reglas de inferencia para la potencia del motor izquierdo

Algorithm 4.7: Función de giro de los motores en Arduino

entrada: Potencias de giro y sentido de giro de motores
salida : Giro de motores
IN1 = número de pin asociado a IN1 del puente H ;
IN2 = número de pin asociado a IN2 del puente H ;
ENA = número de pin asociado a INA del puente H ;
IN3 = número de pin asociado a IN3 del puente H ;
IN4 = número de pin asociado a IN4 del puente H ;
ENB = número de pin asociado a INB del puente H ;

```
procedure GiroMotores(potDer, senDer, potIzq, senIzq)
    IN1, IN2, IN3, IN4 = LOW ;
    ENA = potIzq
    ENB = potDer
    if senIzq = 1 then
        IN1 = HIGH
    else
        IN2 = HIGH
    if senDer = 1 then
        IN4 = HIGH
    else
        IN3 = HIGH
end procedure
```

Algorithm 4.8: Función del sistema difuso para cálculo de las potencias de los motores en Arduino UNO©

Entrada: Area mayor y coordenadas del centroide
Salida : Potencias de los motores

```
conDifArea = [[0,0,15000], [0,15000,30000], [15000,30000,30000]]
conDifCx = [[0,0,80], [0,80,160], [80,160,240], [160,240,320], [240,320,320]]
conDifPot = [[-255,-255,-200], [-255,-200,-100], [-200,-100,0], [-100,0,100], [0,100,200],
[100,200,255],[200,255,255]]
reglasPotDer = [[5,5,6,4,4],[4,4,5,3,3], [4,4,3,3,2]]
reglasPotIzq = [[4,4,6,5,5], [3,3,5,4,4], [2,3,3,4,4]]
procedure CalculoPotencias(area, centr_x)
    pertW = crea matriz de cuatro elementos flotantes
    potMedDerW = crea matriz de cuatro elementos flotantes
    potMedIzqW = crea matriz de cuatro elementos flotantes
    contPertW = crea matriz de cuatro elementos enteros
    contPertW = 0
    for i = 0 hasta 2 do
        if area > conDifArea[i][0] y area < conDifArea[i][2] then
            for j = 1 hasta n do
                if centr_x > conDifCx[j][0] y centr_x < conDifCx[j][2] then
                    potMedDerW[contPertW] = conDifPot[reglasPotDer[i][j]][1]
                    potMedIzqW[contPertW] = conDifPot[reglasPotIzq[i][j]][1]
                    if area < conDifArea[i][1] then
                        auxWA = (area-conDifArea[i][0]) /
                    (conDifArea[i][1]-conDifArea[i][0])
                    else
                        auxWA = (conDifArea[i][2]-área) /
                    (conDifArea[i][2]-conDifArea[i][1])
                    if centr_x < conDifCx[j][1] then
                        auxWB = (centr_x-conDifCx[j][0]) / (conDifCx[j][1]-conDifCx[j][0])
                    else
                        auxWB = (conDifCx[j][2]-centr_x) / (conDifCx[j][2]-conDifCx[j][1])
                    pertW[contPertW] = mínimo(auxWA, auxWB)
                    incrementa contPertW
algoritmo5
```

Algorithm 4.9: Función del sistema difuso para cálculo de las potencias de los motores en Arduino. Continuación...

Entrada: Continúa algoritmo 4.8
Salida : Potencias de los motores

```
sumaNumeDer = 0
sumaNumeIzq = 0
sumaDeno = 0
for i = 0 hasta contPertW − 1 do
    sumaNumeDer += potMedDerW[i] × pertW[i]
    sumaNumeIzq += potMedIzqW[i] × pertW[i]
    sumaDeno += pertW[i]
potDer = sumaNumeDer / sumaDeno
potIzq = sumaNumeIzq / sumaDeno
if potDer > 0 then
    senDer = 1
else
    senDer = 0
if potIzq > 0 then
    senIzq = 1
else
    senIzq = 0
potDer = absoluto(potDer)
potIzq = absoluto(potIzq)
Retorna potDer, senDer, potIzq, senIzq
```

Algorithm 4.10: Módulo principal del sistema de navegación sobre el Arduino.

Entrada: Area mayor y coordenadas del Centroide vía Puerto Serial
Salida : Ejecución del sistema difuso
procedure ConfiguraArduino

Velocidad de transferencia del puerto serial = 115200

Tiempo de espera de datos en el puerto serial = 10 ms

Pines de salida IN1,IN2,IN3,IN4,ENA,ENB

end procedure

procedure Ciclo Principal

if *hay Datos en Puerto Serial* **then**

cadena = lee cadena del puerto serial, hasta salto de línea

if *Cadena contiene* " $< area >$ " **then**

posEtiArea = posición de la subcadena " $< area >$ "

posEtiCx = posición de la subcadena " $< cx >$ "

posEtiFin = posición de la subcadena " $< fin >$ "

area = subcadena entre las posiciones posEtiArea+6 y posEtiCx, en formato entero

centr_x = subcadena entre las posiciones posEtiCx+4 y posEtiFin, en formato entero

if $area > 0$ *y* $centr\_x > 0$ **then**

potDer,senDer,potIzq,senIzq = cálculoPotencias(area, centr_x)

giroMotores(potDer,senDer,potIzq,senIzq)

Retardo de un milisegundo

end procedure

4.6. Registro de las operaciones del sistema

Con el objetivo de evaluar la eficiencia de las acciones del sistema desarrollado, se incluyó un registro de las operaciones que realiza el robot durante el tiempo de operación. La Figura 50 muestra un ejemplo de un registro generado durante pruebas reales. Los datos que se almacenan son:

	A	B	C	D	E	F	G	H	I	J	K	L	M
1		nExp	Hora	Area	Cx	Cy	DX	Ag	PotDer	PotIzq	Ds	Ap	tRec
16	14	3	1664908114	2058	160	125	0	-0.22	247	247	90	0	0.3900 Seg
17	15	3	1664908115	2067.5	160	126	0	-0.23	247	247	84	0	0.3960 Seg
18	16	3	1664908115	2354	149	126	11	-2.87	226	204	75	0	0.3969 Seg
19	17	3	1664908116	2657.5	146	127	14	-3.6	221	195	68	0	0.3937 Seg
20	18	3	1664908116	3164.5	148	132	12	-3.17	222	199	60	0	0.4022 Seg
21	19	3	1664908116	3890.5	149	130	11	-2.91	221	200	52	0	0.3955 Seg
22	20	3	1664908117	4700	156	129	4	-1.22	230	221	45	0	0.4019 Seg
23	21	3	1664908117	5836	158	129	2	-0.74	230	225	40	0	0.3994 Seg
24	22	3	1664908118	8704	164	130	4	0.69	207	216	28	0	0.4019 Seg
25	23	3	1664908118	12062	170	134	10	2.09	179	199	21	0	0.3946 Seg
26	24	3	1664908119	10682	177	95	17	4.16	0	0	13	1	0.4066 Seg

Figura 50: Registro de operaciones del sistema.

- Número de experimento (Exp),
- Hora, área mayor (Área).
- Coordenada x del centroide del objeto reconocido (Cx).
- Coordenada y del centroide del objeto reconocido (Cy).
- Desplazamiento del centroide con respecto al centro de la imagen (DX).
- Ángulo en grados entre el centroide del objeto detectado y el centro de la imagen (Ag).
- Potencia asignada al motor derecho (potDer).
- Potencia asignada al motor izquierdo (potIzq).
- Distancia entre el robot y el objeto (Ds).
- Indicador de aproximación realizada (Ap).
- Duración del reconocimiento y aproximación (tRec).

Los datos para este registro son generados en la tarjeta Arduino UNO©, y transmitidos para su almacenamiento en formato JSON a la NVIDIA Jetson Nano©, para posteriormente exportarlo a formato CSV. El análisis de este archivo permite identificar comportamientos extraños del robot durante las pruebas, así como comparar la eficiencia del sistema.

Con esta información se pueden realizar seguimientos gráficos acerca del comportamiento de las potencias suministradas a los motores durante la aproximación, así como el ángulo de aproximación. Las gráficas fueron generadas con lenguaje Python y la librería matplotlib.

La Figura 51 muestra el comportamiento de las potencias asignadas a los motores derecho e izquierdo durante el reconocimiento y aproximación a una botella que inicialmente se encuentra cerca (90cm) con un ángulo de -0.22 grados, según datos registrados en la Figura 50.

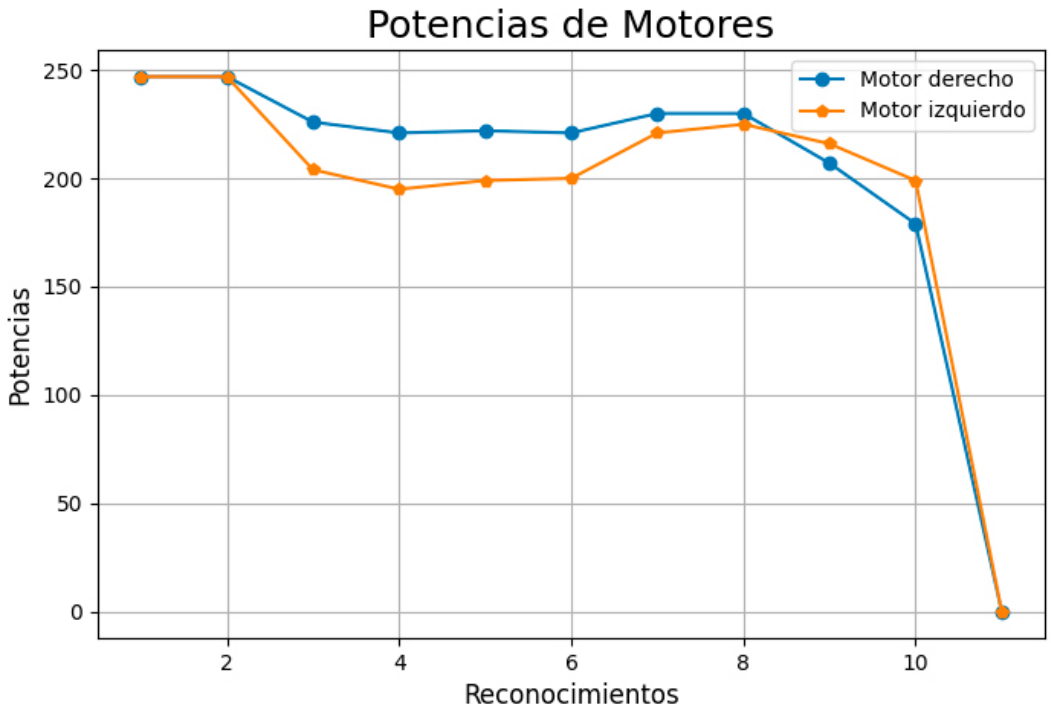


Figura 51: Comportamiento de las potencias suministradas a los motores

Mientras que en la Figura 52 se muestra el comportamiento de el ángulo de aproximación a una botella que inicialmente se encuentra cerca (90cm) con un ángulo de -0.22 grados, según datos registrados en la Figura 50.

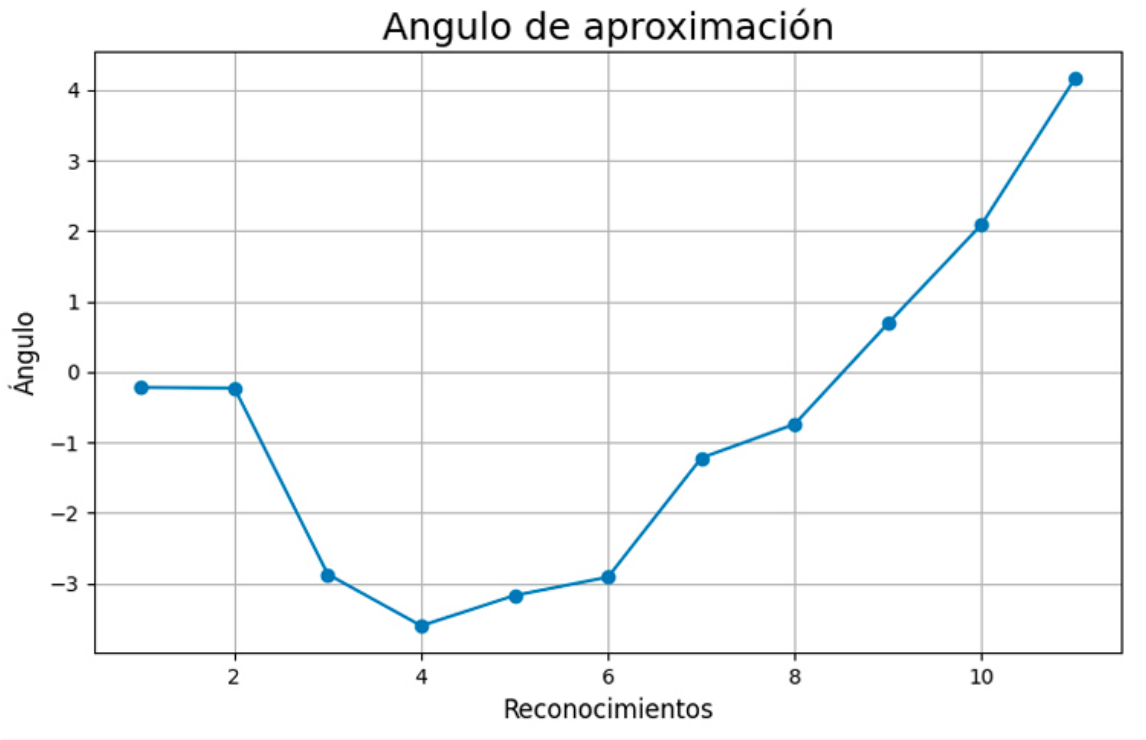


Figura 52: Comportamiento del ángulo de aproximación

Un segundo experimento es registrado en la Figura 53 y muestra el reconocimiento y aproximación a una botella que inicialmente se encuentra alejado 4m aproximadamente de 17.33 grados. Se puede observar que el registro de distancia (Ds) que se calcula mediante el uso de un sensor ultrasónico muestra valores no aceptables (72) lo cuál puede deberse a la calidad de sensor, ya que el error se corrige conforme el robot se aproxima a su objetivo (Figuras 54 y 55).

	A	B	C	D	E	F	G	H	I	J	K	L	M
1		nExp	Hora	Area	Cx	Cy	DX	Ag	PotDer	PotIzq	Ds	Ap	tRec
129	127	10	1664908704	689	233	122	73	17.33	112	200	72	0	0.4378 Se
130	128	10	1664908705	658	233	121	73	17.34	112	200	385	0	0.4123 Se
131	129	10	1664908705	692	210	125	50	11.78	153	215	72	0	0.4606 Se
132	130	10	1664908706	739	186	124	26	6.03	195	229	72	0	0.4705 Se
133	131	10	1664908706	809	174	127	14	3.12	215	236	72	0	0.4713 Se
134	132	10	1664908707	802	170	127	10	2.16	223	239	72	0	0.4708 Se
135	133	10	1664908707	806.5	166	130	6	1.17	229	241	72	0	0.4707 Se
136	134	10	1664908708	1115.5	164	129	4	0.7	233	242	72	0	0.4631 Se
137	135	10	1664908708	1181.5	162	126	2	0.25	241	246	127	0	0.4068 Se
138	136	10	1664908708	1321.5	160	124	0	-0.21	250	250	119	0	0.4063 Se
139	137	10	1664908709	1423	159	122	1	-0.43	247	245	109	0	0.3958 Se
140	138	10	1664908709	1617.5	156	122	4	-1.15	240	231	99	0	0.3979 Se
141	139	10	1664908710	1882	160	122	0	-0.19	248	248	90	0	0.3936 Se
142	140	10	1664908710	2072	160	123	0	-0.2	247	247	81	0	0.3955 Se
143	141	10	1664908710	2353	158	124	2	-0.69	242	237	72	0	0.4008 Se
144	142	10	1664908711	2898.5	160	124	0	-0.21	244	244	64	0	0.4037 Se
145	143	10	1664908711	3592	162	125	2	0.26	233	237	56	0	0.3957 Se
146	144	10	1664908712	4218.5	163	123	3	0.52	226	233	49	0	0.3944 Se
147	145	10	1664908712	5719.5	166	124	6	1.23	210	223	39	0	0.4008 Se
148	146	10	1664908712	7659	168	123	8	1.72	197	214	30	0	0.3959 Se
149	147	10	1664908713	10668.5	174	129	14	3.1	173	199	23	0	0.3968 Se
150	148	10	1664908713	13160.5	178	134	18	4.01	0	0	15	1	0.4008 Se

Figura 53: Registro de operaciones del sistema.

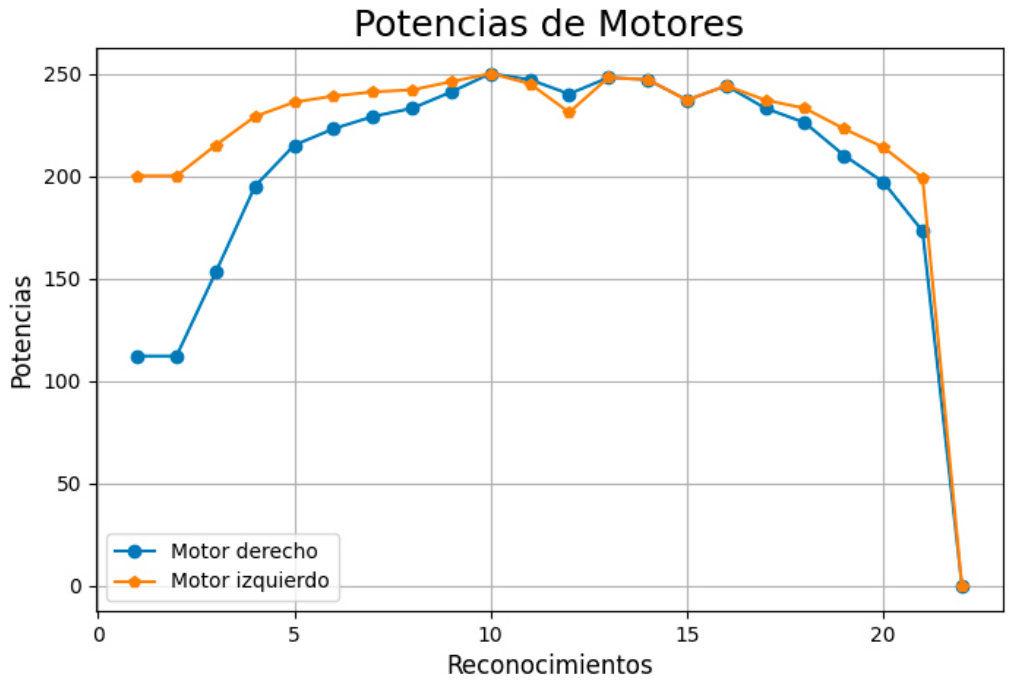


Figura 54: Registro de operaciones del sistema.

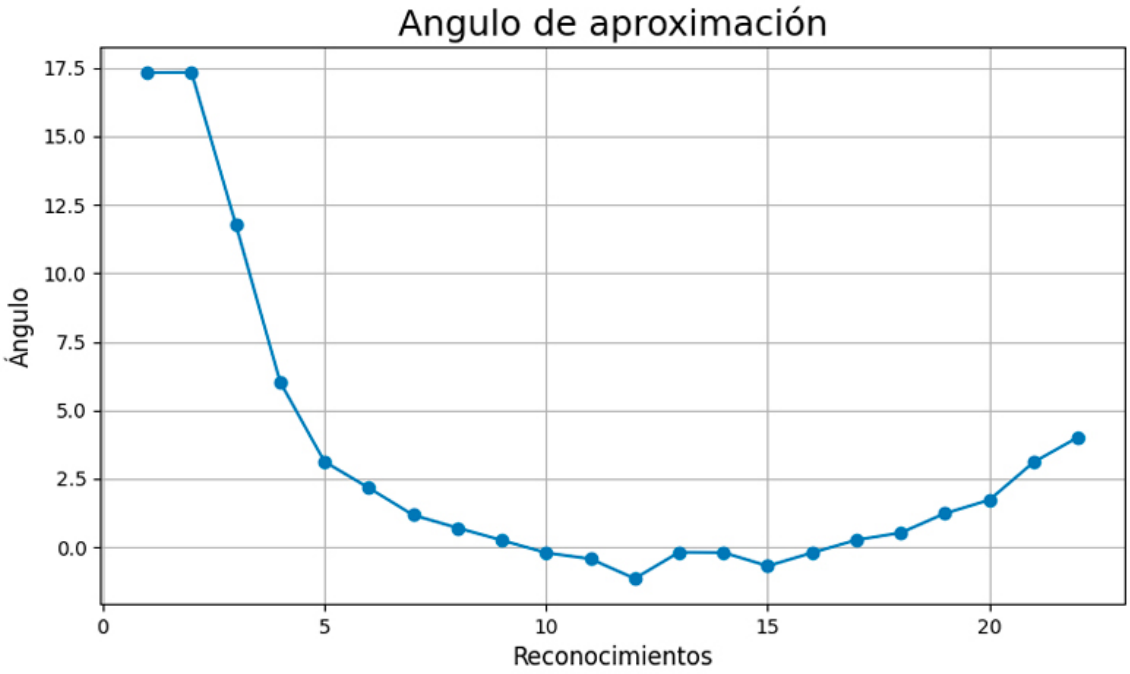


Figura 55: Registro de operaciones del sistema.

5. EXPERIMENTOS Y RESULTADOS

Los experimentos del sistema propuesto se dividieron en tres tipos: 1) pruebas para evaluar la capacidad de reconocimiento, 2) pruebas generales de aproximación a las botellas, y 3) pruebas para determinar la eficiencia del sistema.

5.1. Pruebas de reconocimiento

Para validar el método de reconocimiento se realizaron 50 pruebas, en las cuales se colocó el robot y las botellas en posiciones y orientaciones diferentes. De esta forma, se tomaron 50 imágenes, donde cada una puede contener desde cero hasta las cuatro botellas. Las imágenes pueden contener otros objetos del entorno, como reguladores, botes de basura, cables, mesas, sillas, entre otros. De las 50 imágenes de prueba, 32 contienen botellas y 18 no contienen. En total, se distribuyeron 42 botellas dentro del grupo de 32 imágenes. En las Figuras 56 y 57 se muestran ejemplos de pruebas de reconocimiento. Para esta prueban colocaron otros objetos entre las botellas.

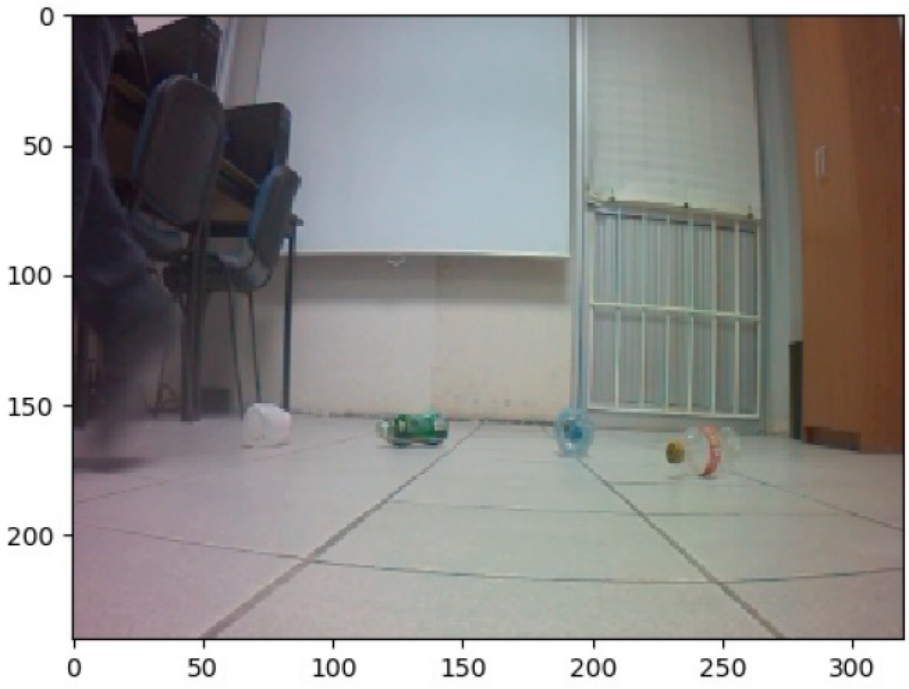


Figura 56: Pruebas de reconocimiento sin otros objetos.

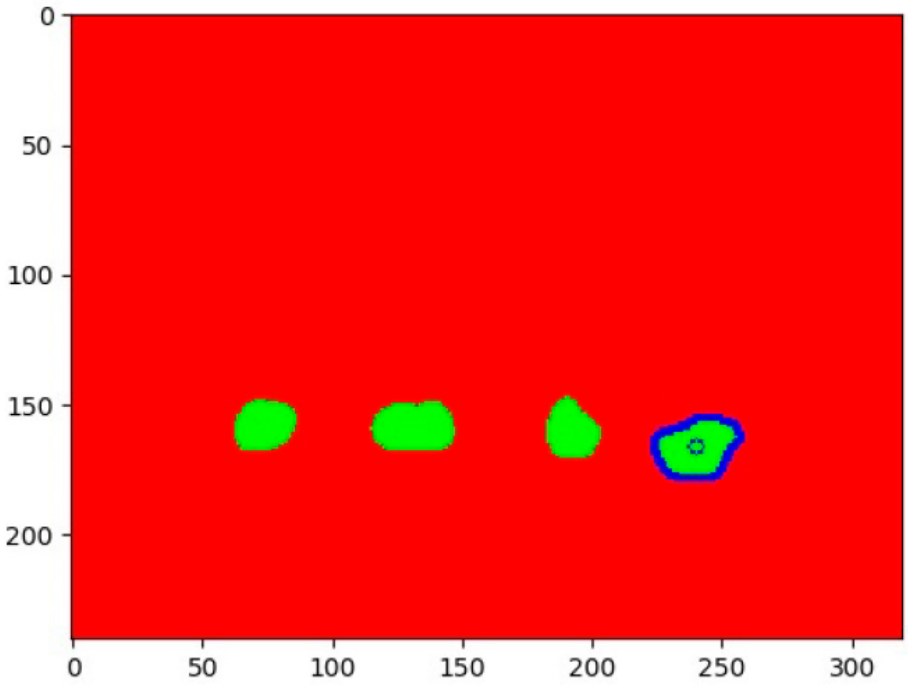


Figura 57: Pruebas de reconocimiento junto a otros objetos.

Para capturar la imagen de la Figura 56, no se colocaron otros objetos entre las botellas (sí en el entorno). Para la imagen de la Figura 58, se colocaron un regulador, un bote de basura y una silla, entre las botellas, ya que son objetos comunes al ambiente controlado en el que se realizaron los experimentos. Como resultado del reconocimiento, el sistema funcionó perfectamente detectando las 4 botellas y descartando los demás objetos.

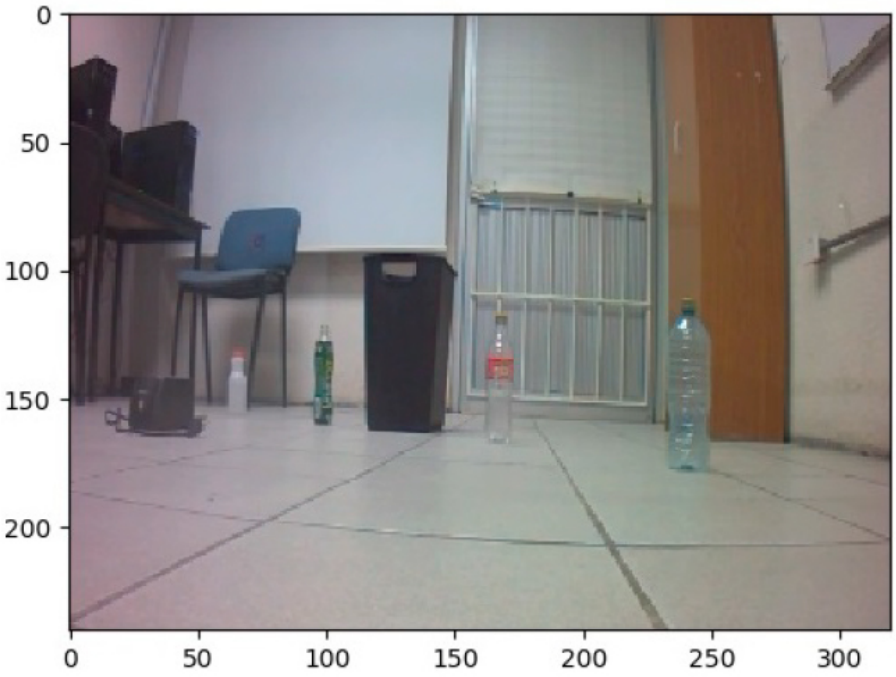


Figura 58: Pruebas de reconocimiento sin otros objetos.

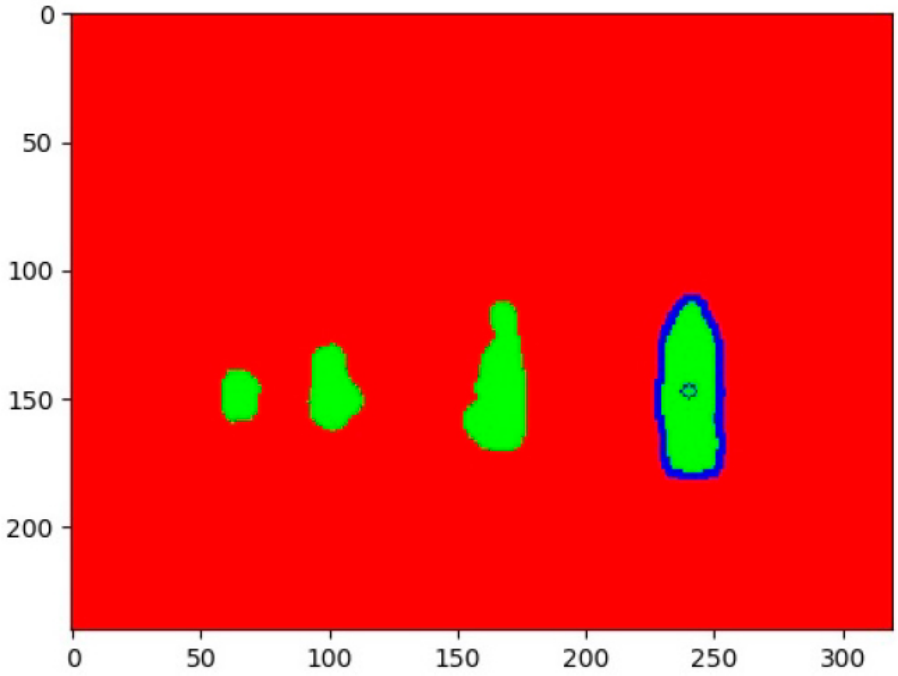


Figura 59: Pruebas de reconocimiento junto a otros objetos.

A continuación se muestran los resultados de las pruebas de reconocimiento realizadas con las 50 imágenes. Para la evaluación se consideraron tres métricas: precision (Ecuación 1), recall (Ecuación 2) y accuracy (Ecuación 3). Estás métricas se basan en los valores: true positive (TP), false positive (FP), true negative (TN) y false negative (FN).

$$PR = \frac{TP}{TP + FP} \tag{1}$$

$$RC = \frac{TP}{TP + FN} \tag{2}$$

$$AC = \frac{TP + TN}{TP + FP + TN + FN} \tag{3}$$

Se consideraron como TP, los casos donde la red reconoce botellas, que realmente sí se encuentran en las imágenes. Como FP, los casos donde la red reconoce botellas, que no se encuentran en las imágenes. Como TN, los casos donde la red no reconoce botellas en imágenes que no contienen alguna botella. Y como FN, los casos donde la red no reconoce botellas, que sí se encuentran en las imágenes.

TP	41
FP	1
TN	18
FN	1
Precision	97.6
Recall	97.6
Accuracy	96.7

Tabla 4: Resultados de las pruebas de reconocimiento

Los resultados de las pruebas de reconocimiento se muestran en la Tabla 4, y exhiben un 96.7 % de precisión en el reconocimiento de las botellas. Este porcentaje de precisión en el reconocimiento, permitió la navegación fluida y directa del robot hacia las botellas. Además, es competitivo con respecto a los resultados reportados en otros trabajos recientes con objetivos similares.

5.2. Pruebas básicas de aproximación

Las pruebas básicas para evaluar el sistema de aproximación, consistieron en observar las trayectorias que siguió el robot para aproximarse a las botellas colocadas en 20 posiciones distintas. Las botellas se colocaron de tal forma que forzaran al robot a seguir distintas trayectorias dentro de un espacio de dos por tres metros. En todos los casos las botellas se colocaron dentro del campo de visión del robot.

Estas pruebas básicas se realizaron antes de colocar el sensor ultrasónico al robot. De tal forma que se utilizó un área de 30,000 píxeles de la botella objetivo, como parámetro para detener el robot. En todas las pruebas realizadas bajo estas condiciones, el robot alcanzó las posiciones de las botellas siguiendo trayectorias casi directas (ver Figura 60). Esto es, una vez que se logró reconocer una botella por la red neuronal, en el 100 % de las pruebas, el robot se desplazó hasta la posición de estas.

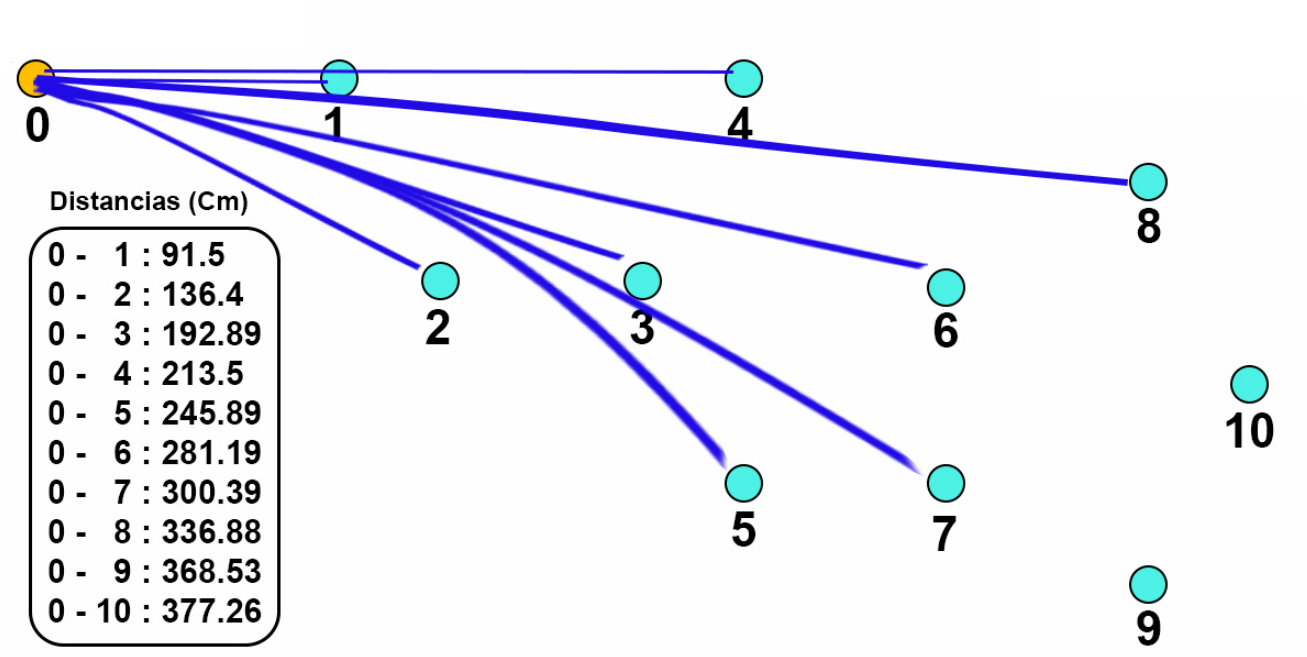


Figura 60: Diseño de experimento de aproximación.

Además de las pruebas con las botellas en posiciones estáticas, se realizaron pruebas con las botellas en movimiento mientras el robot se desplazaba hacia estas. En todos los casos, se realizó el seguimiento y aproximación mientras que las botellas no se salieron del campo de visión,

el robot ajustó eficientemente su trayectoria para dirigirse a las nuevas posiciones dinámicas, generando una impresión de que les da seguimiento. En la Figura 61 se ilustra este experimento.

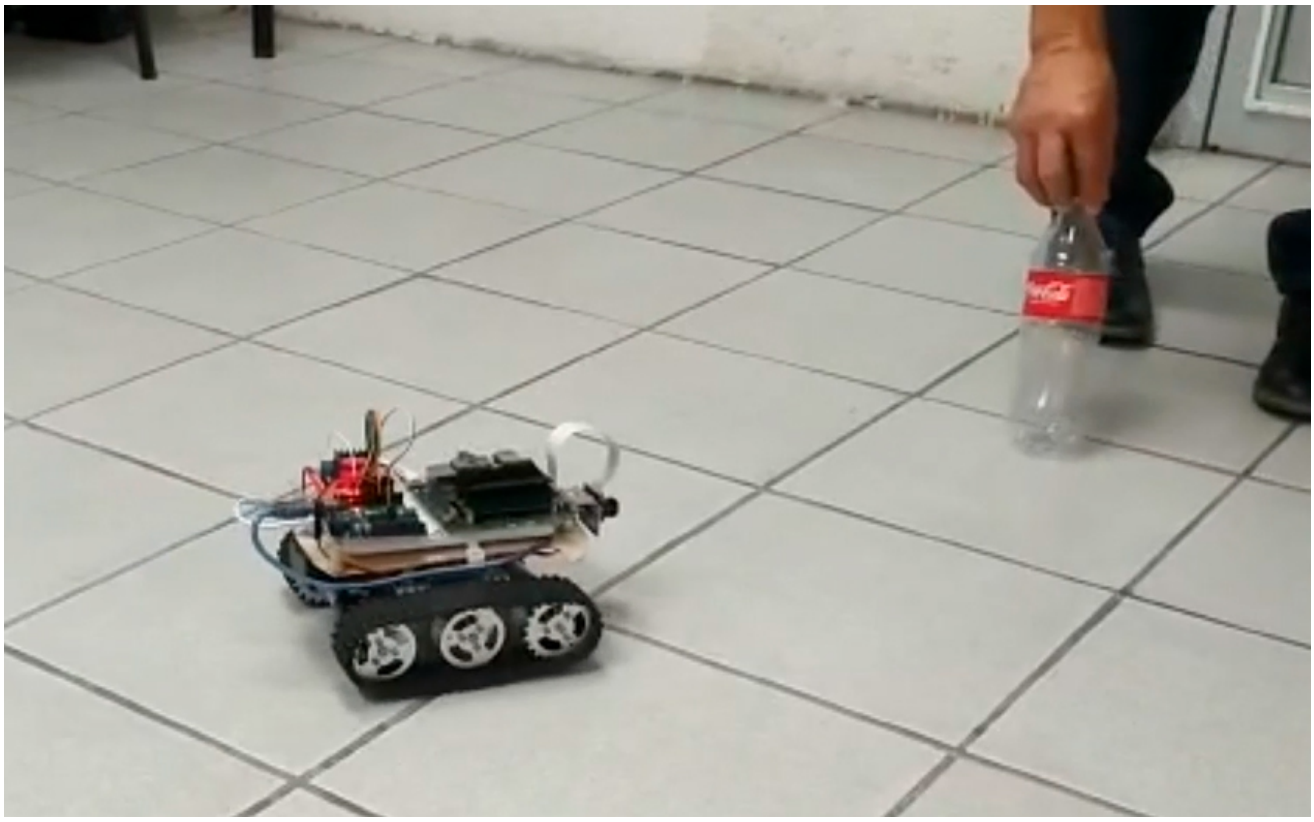


Figura 61: Experimento de reconocimiento y aproximación con botellas en movimiento.

5.3. Pruebas de eficiencia del sistema

Para evaluar la eficiencia del sistema, además de los datos obtenidos previamente, como potencias, área, tiempo de reconocimiento y centroide; se midieron y registraron el ángulo y la distancia de las botellas respecto al robot. Estos valores se calcularon durante el recorrido del robot y en la posición final de este.

En el caso de la distancia se midió en centímetros directamente con el sensor ultrasónico. Para calcular el ángulo se generó una función, a través de regresión lineal. Esta función relaciona la coordenada (X,Y) en píxeles del centroide de la botella en la imagen, con el ángulo en grados en el que se encuentra físicamente la cámara del robot respecto al centroide de la botella.

5.3.1. Regresión lineal para el cálculo del ángulo

Para aplicar la regresión lineal se colocó el robot en una posición fija (punto amarillo de la Figura 62), y se dibujaron 20 puntos de color negro en el piso frente al robot. Para cada uno de los puntos se midió manualmente el ángulo de estos con respecto al robot y se tomó una imagen de estos. En cada una de las imágenes se obtuvo y almacenó el centroide de los puntos en píxeles. Con estos datos se aplicó la regresión lineal.

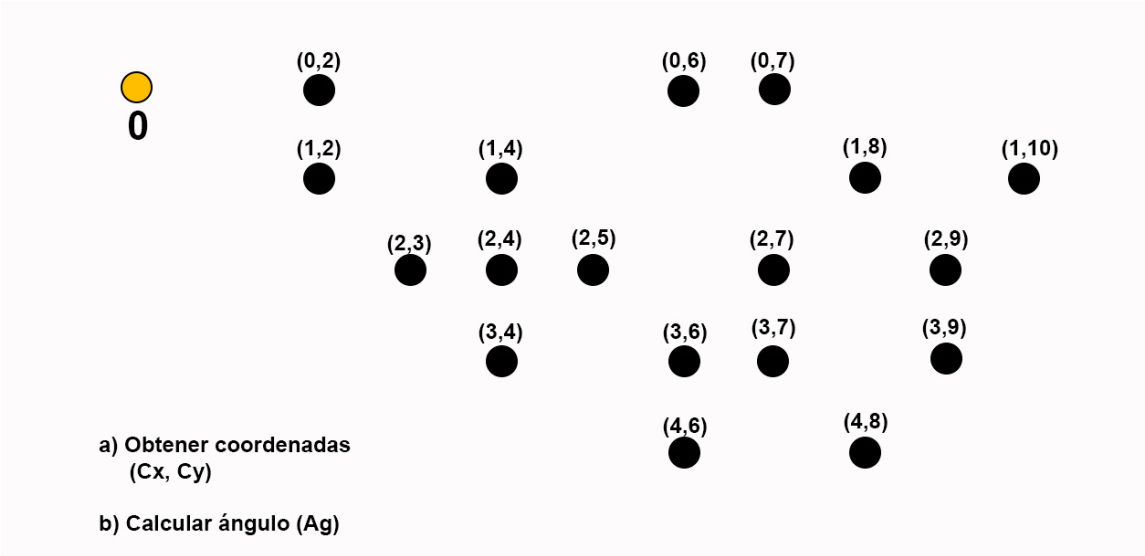


Figura 62: Experimento para obtener la función del ángulo de aproximación.

Con esta función se puede inferir el ángulo de los objetos respecto al frente al robot, tomando como dato de entrada la coordenada del centroide de estos en la imagen. En la Figura 63 se muestran los datos y la función obtenida.

x	y	z
272.19	178.80	26.26
271.88	153.60	26.57
301.25	159.60	33.69
252.81	149.60	21.80
313.44	150.40	36.87
271.88	144.40	26.57
199.38	148.40	9.46
300.63	143.20	33.69
219.69	156.40	14.04
271.88	140.40	26.57
227.19	144.00	15.95
246.88	141.60	20.56
189.38	144.00	7.13
238.13	140.40	18.43
182.81	141.60	5.71
212.50	141.60	12.53
160.94	188.40	0
158.75	147.60	0

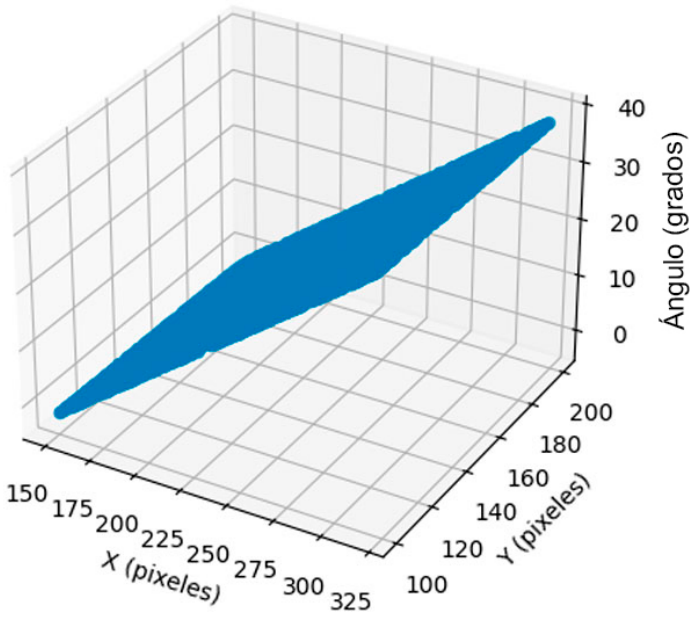


Figura 63: Modelo de regresión obtenido para cálculo del ángulo.

Una vez obtenida la función para el cálculo del ángulo (Ecuación 4), se desarrollaron los experimentos para evaluar la eficiencia de aproximación del sistema.

$$Z = 0.24x - 0.01y - 37.27 \tag{4}$$

Para esto, se colocó el robot en el punto 0 (punto amarillo de la Figura 64), y se realizaron 30 pruebas colocando las botellas en 10 puntos seleccionados al azar y mostrados en color azul en la figura, realizando tres pruebas por cada punto. Para cada una de las 30 pruebas se midió manualmente la distancia entre la posición de la cámara y el centro de la botella y se calculó el ángulo final del recorrido del robot.

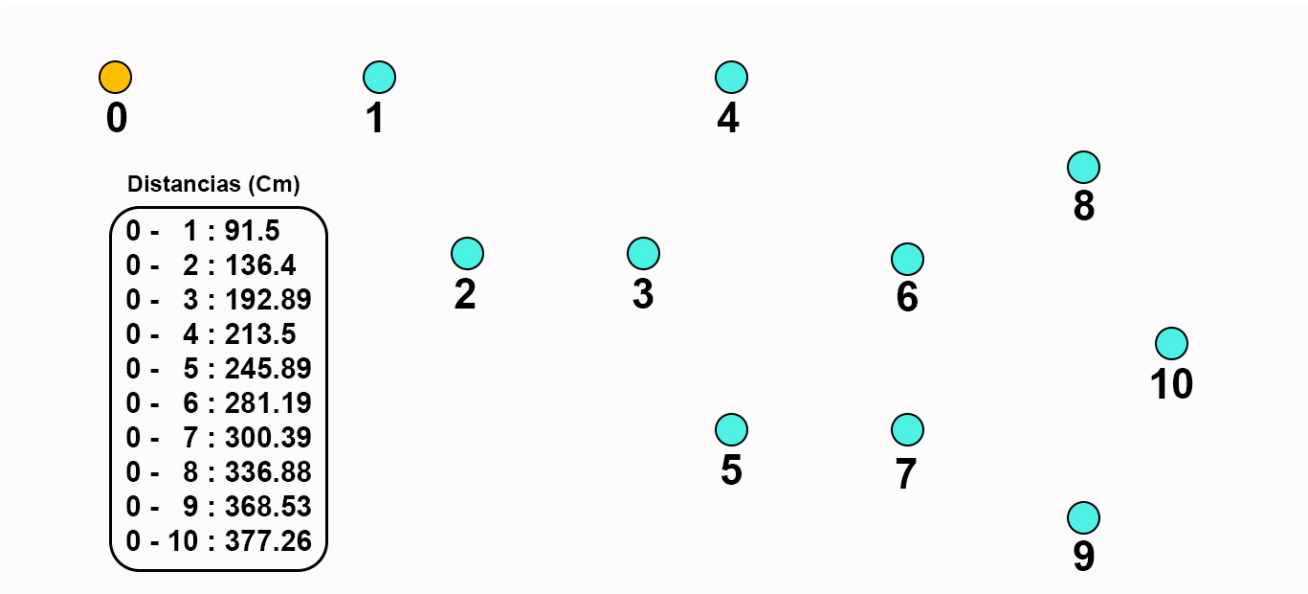


Figura 64: Experimento para pruebas de eficiencias de aproximación.

La Figura 65 muestra los resultados obtenidos de este experimento. En cada fila de los resultados se muestra el promedio de las tres pruebas realizadas por cada posición. Se muestra el número de experimento (Exp), la distancia en centímetros (Ds) y el ángulo en grados (Ag) del frente del robot respecto al centro la botella, así como la distancia horizontal del frente del robot respecto al centro de la botella (Dx). En color rojo se muestra el "error" más grande obtenido de todas las pruebas.

ERROR (Dx) EN Cms

Exp	Ds	Ag	Dx
1	10.83	2.95	0.52
2	12.33	3.10	0.64
3	12.00	4.64	0.78
4	7.30	3.03	0.36
5	9.83	2.09	0.35
6	11.00	4.27	0.82
7	7.00	2.27	0.28
8	11.16	4.46	0.85
9	7.30	6.65	0.76
10	12.50	3.65	0.78

$$Dx = Ds * Tan(Ag)$$

Dx: Cms
Ag: Grados
Ds: Cms

Figura 65: Cálculo del error de aproximación.

6. Conclusiones y trabajo futuro

En este trabajo se propuso el diseño, construcción y programación de un robot autónomo, con reconocimiento y aproximación hacía botellas de plástico, como un primer acercamiento en la búsqueda de soluciones al problema de contaminación de nuestras playas por botellas de plástico aplicando técnicas de IA.

Para el sistema de reconocimiento se entrenó una red neuronal tipo U-Net basada en la ResNet18, implementada con las librerías Pytorch y TorchVision, sobre la tarjeta NVIDIA Jetson Nano© B01. El sistema de aproximación hacía las botellas, se diseño a través de vision computacional y lógica difusa, y se implementó en C++ sobre una tarjeta Arduino UNO©.

Los experimentos se realizaron en un ambiente controlado, utilizando cuatro tipo diferentes de botellas. En las pruebas de reconocimiento se obtuvo un 96.6 % de precisión, mientras que en las pruebas de aproximación, el robot alcanzó en todos los casos la posición de las botellas, a una distancia no mayor de 15 cm y con un ángulo no mayor de 5 grados, lo cual se considera suficientemente cercano para aplicar, en un trabajo futuro, una estrategia para la recolección de las mismas.

Por motivo de los resultados experimentales obtenidos, se concluye que se confirma la hipótesis de la factibilidad construir y programar un robot autónomo capaz de reconocer botellas de plástico y aproximarse a ellas en un ambiente controlado usando técnicas de IA.

Una característica a destacar de esta propuesta, es que el sistema resulta sencillo de implementar, ya que únicamente requiere de las imágenes capturadas por la cámara para guiar el recorrido del robot hacia una aproximación precisa, y de un sensor ultrasónico para determinar el momento de parada del mismo.

Una vez probado el método en un ambiente controlado, como trabajo futuro se propone:

- Reconfigurar el sistema de tracción para que permita la movilidad del dispositivo en ambiente de playa.

- Mejorar el conjunto de datos de entrenamiento con imágenes de mayor resolución y hacer una segmentación mas precisa de cada una de las imágenes y realizar de nuevo el proceso de entrenamiento buscando un mejor modelo.

- Incorporar al robot un sistema de evasión de obstáculos y un sistema de recolección de las botellas.

- Reprogramar el sistema de aproximación en lenguaje Python y ejecutarlo sobre la NVIDIA Jetson Nano© . Esto permitirá eliminar la tarjeta Arduino UNO© y deberá mejorar el tiempo de reconocimiento.

Bibliografía

[Abeliuk y Gutiérrez, 2021] Abeliuk, A. y Gutiérrez, C. (2021). Historia y evolución de la inteligencia artificial. *Revista Bits de Ciencia*, (21):14–21.

[Agarwal y Bharti, 2020] Agarwal, D. y Bharti, P. S. (2020). Nature inspired evolutionary approaches for robot navigation: Survey. *Journal of Information and Optimization Sciences*, 41(2):421–436.

[Agencia de Protección Ambiental de Estados Unidos, 2020] Agencia de Protección Ambiental de Estados Unidos, h.-i.-d.-l.-p.-d.-l.-p. (2020). La importancia de la protección de las playas.

[Aguilar Jáuregui y Peredo Macías, 1999] Aguilar Jáuregui, M. E. y Peredo Macías, C. (1999). Introducción a la teoría de conjuntos difusos "fuzzy set"(compendio). *Polibits*, 22:11–13.

[Alatise y Hancke, 2020] Alatise, M. B. y Hancke, G. P. (2020). A review on challenges of autonomous mobile robot and sensor fusion methods. *IEEE Access*, 8:39830–39846.

[Andreu-Perez et al., 2018] Andreu-Perez, J., Deligianni, F., Ravi, D., y Yang, G.-Z. (2018). Artificial intelligence and robotics. *arXiv preprint arXiv:1803.10813*.

[Bai et al., 2018] Bai, J., Lian, S., Liu, Z., Wang, K., y Liu, D. (2018). Deep learning based robot for automatically picking up garbage on the grass. *IEEE Transactions on Consumer Electronics*, 64(3):382–389.

[Beltrán Barba, 2022] Beltrán Barba, R. (2022). Modelos de aprendizaje profundo. aplicaciones con r y python.

[Ben-Ari y Mondada, 2017] Ben-Ari, M. y Mondada, F. (2017). *Elements of robotics*. Springer Nature.

[Bolton, 2017] Bolton, W. (2017). *Mecatrónica*. Alpha Editorial.

[Bradski y Kaehler, 2008] Bradski, G. y Kaehler, A. (2008). *Learning OpenCV: Computer vision with the OpenCV library*. .ºReilly Media, Inc.".

[Buteler, 2019] Buteler, M. (2019). Qué es la contaminación por plástico y por qué nos afecta a todos? *Desde la Patagonia difundiendo saberes, 16(28), 56-60*.

[Cerezo Sánchez et al., 2020] Cerezo Sánchez, S. et al. (2020). Herramientas modernas en redes neuronales: la librería pytorch. B.S. thesis.

[Chollet, 2021] Chollet, F. (2021). *Deep learning with Python*. Simon and Schuster.

[Cristino et al., 2019a] Cristino, D. J. S., Lozano, H. M., y Espino, E. R. (2019a). Sistema de navegación y evasión de obstáculos aplicando un sistema de control difuso en una placa arduino uno. *Research in Computing Science*, 148.

[Cristino et al., 2019b] Cristino, D. J. S., Lozano, H. M., y Rubio-Espino, E. (2019b). Sistema de navegación y evasión de obstáculos aplicando un sistema de control difuso en una placa arduino uno. *Res. Comput. Sci.*, 148(10):291–303.

[Cutting y Stephen, 2021] Cutting, V. y Stephen, N. (2021). A review on using python as a preferred programming language for beginners.

[Daniel et al., 2016] Daniel, G. S., Emilio, V. S. J., Guillermo, D. D., y Arturo, G. G. (2016). Diseño mecatrónico de un robot móvil.

[De La Torre Muña y Yufra Torres, 2020] De La Torre Muña, K. y Yufra Torres, J. G. C. (2020). Diseño de un robot móvil recolector y compactador de botellas de plástico utilizando redes neuronales en playas con arena fina.

[Gachet Páez y Valverde Gil, 2007] Gachet Páez, D. y Valverde Gil, R. (2007). Identificación de sistemas dinámicos utilizando redes neuronales rbf.

[García et al., 2015] García, G. B., Suarez, O. D., Aranda, J. L. E., Tercero, J. S., Gracia, I. S., y Enano, N. V. (2015). *Learning image processing with OpenCV*. Packt Publishing Birmingham.

[Gollapudi, 2019] Gollapudi, S. (2019). *Learn computer vision using OpenCV*. Springer.

[Gul et al., 2019] Gul, F., Rahiman, W., y Nazli Alhady, S. S. (2019). A comprehensive study for robot navigation techniques. *Cogent Engineering*, 6(1):1632046.

[Herranz, 2015] Herranz, J. C. H. (2015). Una mirada al mundo arduino. *Tecnología y desarrollo*, 13:21.

[Ibérico, 2018] Ibérico, E. (2018). [https://www.eliberico.com/los-desechos-plasticos-mar-se-duplicaran-10-anos/].

[Janiesch et al., 2021] Janiesch, C., Zschech, P., y Heinrich, K. (2021). Machine learning and deep learning. *Electronic Markets*, 31(3):685–695.

[Krizhevsky et al., 2017] Krizhevsky, A., Sutskever, I., y Hinton, G. E. (2017). Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90.

[Kulshreshtha et al., 2021] Kulshreshtha, M., Chandra, S. S., Randhawa, P., Tsaramirsis, G., Khadidos, A., y Khadidos, A. O. (2021). Oatcr: Outdoor autonomous trash-collecting robot design using yolov4-tiny. *Electronics*, 10(18):2292.

[Lozada et al., 2020] Lozada, E. L., Espino, E. R., Azuela, J. H. S., y Ponce, V. H. P. (2020). Selección de acciones para la navegación de un robot móvil basada en fuzzy q-learning. *Research in Computing Science*, 149:79–89.

[Marin et al., 2021] Marin, J. G. G., Díaz, D. B., y Torres, J. S. S. (2021). *Una revisión sobre la evolución de la robótica móvil*.

[Minaee et al., 2021] Minaee, S., Boykov, Y. Y., Porikli, F., Plaza, A. J., Kehtarnavaz, N., y Terzopoulos, D. (2021). Image segmentation using deep learning: A survey. *IEEE transactions on pattern analysis and machine intelligence*.

[Morales Luna, 2002] Morales Luna, G. (2002). Introducción a la lógica difusa. *Centro de Investigación y Estudios Avanzados. México*.

[Mordvintsev y Abid, 2014] Mordvintsev, A. y Abid, K. (2014). Opencv-python tutorials documentation. *Obtenido de https://media. readthedocs. org/pdf/opencv-python-tutroals/latest/opencv-python-tutroals. pdf*.

[Mu y Zeng, 2019] Mu, R. y Zeng, X. (2019). A review of deep learning research. *KSII Transactions on Internet and Information Systems (TIIS)*, 13(4):1738–1764.

[Novac et al., 2022] Novac, O.-C., Chirodea, M. C., Novac, C. M., Bizon, N., Oproescu, M., Stan, O. P., y Gordan, C. E. (2022). Analysis of the application efficiency of tensorflow and pytorch in convolutional neural network. *Sensors*, 22(22):8872.

[Ocampo y Santa Catarina, 2019] Ocampo, M. y Santa Catarina, C. (2019). Plásticos en los océanos. *INCyTU Of. Inf. Científica y Tecnológica para el Congreso de la Unión*, (34):6.

[Oltean, 2019] Oltean, S.-E. (2019). Mobile robot platform with arduino uno and raspberry pi for autonomous navigation. *Procedia Manufacturing*, 32:572–577.

[O'Mahony et al., 2020] O'Mahony, N., Campbell, S., Carvalho, A., Harapanahalli, S., Hernandez, G. V., Krpalkova, L., Riordan, D., y Walsh, J. (2020). Deep learning vs. traditional computer vision. In *Advances in Computer Vision: Proceedings of the 2019 Computer Vision Conference (CVC), Volume 1 1*, pages 128–144. Springer.

[Perez et al., 2018] Perez, J. A., Deligianni, F., Ravi, D., y Yang, G.-Z. (2018). Artificial intelligence and robotics. *arXiv preprint arXiv:1803.10813*, 147.

[Pietikäinen y Silven, 2022] Pietikäinen, M. y Silven, O. (2022). Challenges of artificial intelligence–from machine learning and computer vision to emotional intelligence. *arXiv preprint arXiv:2201.01466*.

[Pouyanfar et al., 2018] Pouyanfar, S., Sadiq, S., Yan, Y., Tian, H., Tao, Y., Reyes, M. P., Shyu, M.-L., Chen, S.-C., y Iyengar, S. S. (2018). A survey on deep learning: Algorithms, techniques, and applications. *ACM Computing Surveys (CSUR)*, 51(5):1–36.

[Pulungan et al., 2021] Pulungan, A. B., Nafis, Z., Anwar, M., et al. (2021). Object detection with a webcam using the python programming language. *Journal of Applied Engineering and Technological Science (JAETS)*, 2(2):103–111.

[Ramírez Zavalza et al., 2020] Ramírez Zavalza, K. A., RAMIREZ ZAVALZA, K. A., et al. (2020). Robot inteligente recolector de basura asistido por redes neuronales artificiales. Master's thesis.

[Raschka, 2015] Raschka, S. (2015). *Python machine learning*. Packt publishing ltd.

[Ray, 2019] Ray, S. (2019). A quick review of machine learning algorithms. In *2019 International conference on machine learning, big data, cloud and parallel computing (COMITCon)*, pages 35–39. IEEE.

[Rodolfo, 2015] Rodolfo, E. (2015). Mar de plástico: Una revisión de los problemas del plástico en el mar. *Marine and Fishery Sciences (MAFIS), 27, 83-105*.

[Ronneberger et al., 2015] Ronneberger, O., Fischer, P., y Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer.

[Rouhiainen, 2018] Rouhiainen, L. (2018). Inteligencia artificial. *Madrid: Alienta Editorial*.

[Rubio et al., 2019] Rubio, F., Valero, F., y Llopis-Albert, C. (2019). A review of mobile robots: Concepts, methods, theoretical framework, and applications. *International Journal of Advanced Robotic Systems*, 16(2):1729881419839596.

[Santos, 2011] Santos, M. (2011). Un enfoque aplicado del control inteligente. *Revista Iberoamericana de Automática e Informática Industrial RIAI*, 8(4):283–296.

[Sensio, 2020] Sensio, J. (2020). Visión artificial-segmentación semántica. sensio inteligencia artificial.

[Singh et al., 2021] Singh, A., Singh, P., y Tiwari, A. (2021). A comprehensive survey on machine learning. *Journal of Management and Service Science (JMSS)*, 1(1):1–17.

[Stevens et al., 2020] Stevens, E., Antiga, L., y Viehmann, T. (2020). *Deep learning with PyTorch*. Manning Publications.

[Tremante y Brea, 2014] Tremante, P. y Brea, E. (2014). Una visión de la teoría difusa y los sistemas difusos enfocados al control difuso. *Ingeniería Industrial. Actualidad y Nuevas Tendencias*, 4(12):121–136.

[Vazquez, 2023] Vazquez, J. A. E. (2023). Nvidia jetson nano, un mini pc para desarrollo de robótica e inteligencia artificial. *Inicio*, 1(1):1–4.

[Viveros Villada, 2022] Viveros Villada, E. (2022). Arduino desde cero.

[Voulodimos et al., 2018] Voulodimos, A., Doulamis, N., Doulamis, A., Protopapadakis, E., et al. (2018). Deep learning for computer vision: A brief review. *Computational intelligence and neuroscience*, 2018.

[Zadeh, 1999] Zadeh, L. A. (1999). Fuzzy logic = computing with words. *Computing with Words in Information/intelligent Systems 1: Foundations*, pages 3–23.