Pereyra, Fernando Ariel

Sistema de visión artificial y robótica para la manipulación automática de objetos reciclables

2024

Instituto de Ingeniería y Agronomía

Carrera: Ingeniería en Informática

Universidad Nacional Arturo Jauretche

Instituto de Ingeniería y Agronomía

Carrera de Ingeniería en Informática

PRÁCTICA PROFESIONAL SUPERVISADA
Informe de avance

Sistema de visión artificial y robótica para la manipulación automática de objetos reciclables

Pereyra, Fernando Ariel

Florencio Varela, septiembre 2023

PRÁCTICA PROFESIONAL SUPERVISADA (PPS)

Sistema de visión artificial y robótica para la manipulación automática de objetos reciclables.

Informe Final

DATOS DEL ESTUDIANTE

Apellido y Nombres: Pereyra, Fernando Ariel

DNI: 36.715.434

Nº de Legajo: 34634

Correo electrónico: f91p@outlook.com

Cantidad de materias aprobadas al comienzo de la PPS: 41

PPS enmarcada en el artículo (4 ó 7) de la Resolución (CS) 183/21…

DOCENTE SUPERVISOR

Apellido y Nombres: Prof. Ing. Eduardo Kunysz, Prof. Mg. Jorge Osio

Correo electrónico: ekunysz@unaj.edu.ar, josio@unaj.edu.ar

DOCENTE TUTOR DEL TALLER DE APOYO A LA PRODUCCIÓN DE TEXTOS ACADÉMICOS DE LA UNAJ

Apellido y Nombres: Esp. Prof. Lía Lavigna

Correo electrónico: llavigna@unaj.edu.ar

DATOS DE LA ORGANIZACIÓN DONDE SE REALIZA LA PPS

Nombre o Razón Social: Universidad Nacional Arturo Jauretche

Dirección: Av. Calchaquí 6200, Florencio Varela, (1888) Buenos Aires, Argentina

Teléfono: +54 11 4275-6100

Sector: Programa Tecnologías de la Información y la Comunicación (TIC) en aplicaciones de interés social, Instituto de Ingeniería y Agronomía

TUTOR DE LA ORGANIZACIONAL

Apellido y Nombres: Prof. Mg. OSIO, Jorge

Correo electrónico: josio@unaj.edu.ar

FIRMA DEL COORDINADOR DE LA CARRERA

Apellido y Nombres: Dr. Ing. Martín Morales

Correo electrónico: martin.morales@unaj.edu.ar

RESUMEN

La presente Práctica Profesional Supervisada tiene como objetivo el desarrollo e implementación de un sistema automatizado para el control de un brazo robótico, guiado por visión artificial, destinado al reconocimiento y traslado de objetos reciclables. Dada la creciente importancia del reciclaje en la conservación del medio ambiente, es fundamental optimizar los procesos de clasificación, separación y tratamiento de residuos. Este proyecto busca mejorar la eficiencia en estas tareas mediante el uso de tecnologías avanzadas como la visión artificial y la robótica. El sistema desarrollado consta de dos componentes principales: un módulo de visión, basado en una cámara Raspberry Pi y la librería OpenCV para el procesamiento de imágenes y un módulo de control, que gestiona las acciones del brazo robótico mediante el protocolo UART. El enfoque metodológico incluyó la integración de ambos módulos para permitir la identificación, localización y manipulación de objetos reciclables, garantizando una ejecución precisa y rápida de las tareas de reubicación. Los resultados obtenidos demuestran que el sistema es capaz de identificar y manipular objetos con éxito, mejorando la eficiencia en el proceso de reciclaje. Se concluye que la automatización, a través de la robótica y visión artificial, puede transformar los procesos tradicionales de reciclaje, incrementando la precisión y reduciendo los recursos humanos necesarios.

ABSTRACT

This Supervised Professional Practice aims to develop and implement an automated system for controlling a robotic arm, guided by computer vision, to recognize and relocate recyclable objects. Given the growing importance of recycling for environmental conservation, optimizing the processes of classification, separation, and treatment of waste is essential. This project seeks to improve the efficiency of these tasks by using advanced technologies such as computer vision and robotics. The developed system consists of two main components: a vision module, based on a Raspberry Pi camera and the OpenCV library for image processing and a control module that manages the robotic arm's actions through the UART protocol. The methodology included integrating both modules to allow the identification, localization, and manipulation of recyclable objects, ensuring precise and fast execution of relocation tasks. The results show that the system successfully identifies and manipulates objects, enhancing efficiency in recycling processes. The conclusions highlight

that automation through robotics and computer vision can transform traditional recycling processes, increasing precision and reducing the need for human resources.

Tabla de contenido

TABLA DE FIGURAS

1 Introducción

La propuesta de la presente Práctica Profesional Supervisada (PPS) consistirá en el desarrollo e implementación de un sistema de control de un brazo robótico guiado mediante visión artificial para el reconocimiento y traslado de objetos reciclables.

El reciclaje es una actividad esencial para la conservación del medio ambiente y la reducción de la generación de residuos. Sin embargo, el proceso de reciclaje implica una serie de tareas que requieren una gran cantidad de recursos humanos y materiales, como: la clasificación, la separación, el transporte y el tratamiento de los diferentes tipos de residuos.

Una forma de mejorar la eficiencia y la rentabilidad del reciclaje es mediante el uso de sistemas automatizados que puedan realizar estas tareas de forma rápida y precisa. Para ello, se necesita combinar dos tecnologías clave: la visión artificial y la robótica.

Por una parte, la visión artificial, entendida como la disciplina que se ocupa de dotar a las máquinas de la capacidad de percibir e interpretar el entorno visual mediante el procesamiento de imágenes y videos. Por otra parte, la robótica, que es la ciencia que se encarga de diseñar, construir y programar robots que puedan realizar acciones físicas sobre el mundo real.

El sistema consta de dos componentes principales: un módulo de visión y un módulo de control. El módulo de visión se encarga de capturar las imágenes del entorno donde se encuentran los objetos reciclables, analizarlas mediante algoritmos de procesamiento y extraer la información relevante para su clasificación y localización. Mientras que el módulo de control se encarga de enviar las órdenes adecuadas al brazo robótico, para que pueda agarrar el objeto deseado y depositarlo en el contenedor correspondiente.

El sistema se ha implementado utilizando una cámara Raspberry Pi para la obtención de las imágenes en directo y posterior procesamiento, y un brazo robótico con pinza que realizará el efector final del movimiento de los objetos de una posición a otra. Para el procesamiento de las imágenes se utilizó la librería de Python llamada OpenCV, la cual se comunicará con el sistema de control del brazo mediante el protocolo UART (Universal Asynchronous Receiver-Transmitter, por sus siglas en inglés).

1.1 Objetivos Generales

- Investigar el uso del Procesamiento de Imágenes y Visión Artificial en Ciencias de la Computación.
- Desarrollar un sistema que emplee técnicas de procesamiento de imágenes con el fin de proveer cierta autonomía a un brazo robótico, en un entorno determinado.

1.2 Objetivos Específicos

- Evaluar las distintas alternativas de procesamiento de imágenes para visión artificial.
- Analizar las diferentes funciones de OpenCV, que se utilizarán para el procesamiento de imágenes y probar funcionalidades similares con Deep Learning.
- Analizar las diferentes funciones y librerías disponibles para el control de un brazo robótico.
- Sistematizar la información referente a brazos robóticos y aplicaciones tecnológicas.
- Diseñar un sistema de control, que interactúe con los sensores y manipule los mecanismos motores del robot.
- Desarrollar el software en lenguaje Python, que sea capaz de procesar las imágenes y obtener los correspondientes datos a través de los cuales recibirá instrucciones de dirección del brazo.
- Realizar distintas pruebas de funcionamiento al brazo y a la aplicación de monitoreo para validar la propuesta realizada.

1.3 Tareas para ejecutar

1. **Investigar, analizar y estudiar sobre conceptos de Procesamiento de Imágenes, sus diferentes técnicas y algoritmos.** Para ello, se estudiarán los diferentes conceptos y técnicas, que se aplican en el procesamiento de imágenes para luego analizar las diferentes funciones que nos provee OpenCV. Se pondrá especial énfasis en aquellas que se utilizan para filtrar el ruido y/o colores, detectar bordes (cambios bruscos de intensidad lumínica), binarizar imágenes, etc.
2. **Desarrollar un software que sea capaz de procesar las distintas imágenes obtenidas por una cámara de vídeo y actuar en consecuencia.** Se busca desarrollar un sistema, que sea capaz de obtener imágenes en tiempo real a través

de una cámara propietaria de raspberry pi, procesarlas con OpenCV y obtener datos, que lo ayudarán a decidir qué movimiento realizar. Dichas instrucciones serán entregadas al módulo de control para que accione y/o apague los correspondientes motores del brazo robótico.

3. **Probar los sistemas desarrollados.** Se configurará un brazo robótico para probar el sistema de visión artificial desarrollado para el mismo y comprobar que el sistema puede moverse de forma autónoma para la captura de un objeto.
4. **Obtención de resultados y redacción del informe final.** Se espera listar y catalogar todos los resultados obtenidos a lo largo del presente proyecto para su posterior exposición en un informe final, que sintetice todo el conocimiento adquirido a lo largo de cada etapa del proyecto con su respectivo progreso y datos correspondientes.

2 Módulo de visión

Para realizar el módulo de visión se procede a investigar los conceptos de procesamiento de imágenes y sus diferentes técnicas y algoritmos. La librería utilizada es OpenCV, que provee más de 500 funciones de utilidad para esta tarea. Más adelante se describirán algunas de estas funciones, especialmente las que permiten filtrar el ruido y/o colores, detectar bordes, binarizar imágenes, etc.

2.1 Preprocesamiento de imágenes

El preprocesamiento de imágenes se refiere a un conjunto de técnicas utilizadas para preparar las imágenes antes de su análisis o procesamiento más avanzado. Aquí se realizan operaciones como la eliminación de ruido, la mejora de la calidad, el ajuste de contraste y brillo, la normalización y la transformación de las imágenes para optimizar los resultados de etapas posteriores, como la segmentación y el reconocimiento de patrones. Para llevar a cabo el preprocesamiento de imágenes y video en tiempo real, se desarrolló una aplicación escrita en lenguaje Python utilizando la herramienta OpenCV. Se realiza una conversión a escala de grises de las imágenes para optimizar el proceso de análisis reduciendo la complejidad de los datos y que el enfoque esté solo en las intensidades de los píxeles. También, se ajustó el cuadro de visión de la cámara, acotándolo a la zona de interés donde se localizarían los objetos a detectar. Además, se redujo la resolución de la imagen para mejorar el rendimiento del sistema.

2.2 Segmentación de objetos

La segmentación de objetos es un proceso muy importante en el análisis de imágenes, ya que consiste en la división de una imagen en distintas regiones o segmentos que corresponden a objetos o áreas de interés. Este proceso es vital para la detección, reconocimiento y clasificación de los elementos presentes en una escena.

Para realizar la segmentación de objetos en tiempo real, se emplean diferentes técnicas basadas en la manipulación de los valores de los píxeles que componen la imagen. Algunas de las técnicas más comunes incluyen:

- **Segmentación basada en umbralización**: esta técnica segmenta la imagen en base a un valor de umbral determinado, clasificando los píxeles según su intensidad. Es una de las técnicas más sencillas y utilizadas en aplicaciones de segmentación, especialmente en entornos controlados con contrastes altos entre los objetos y el fondo.
- **Segmentación por contornos**: en OpenCV, la segmentación mediante contornos es una técnica popular para detectar bordes y separar objetos basándose en la geometría de las áreas segmentadas. Los contornos ayudan a identificar las formas y los límites de los objetos, lo que facilita su posterior análisis.
- **Segmentación por regiones**: esta técnica agrupa píxeles con características similares (como color, textura o intensidad) en una misma región. Este enfoque es útil en escenarios complejos donde los objetos tienen colores o texturas similares al fondo.

La implementación estas técnicas de segmentación se logra utilizando funciones de OpenCV, como *cv2.threshold()*, *cv2.findContours()*, y *cv2.bitwise_and()*, para procesar y segmentar de manera eficiente las imágenes en tiempo real.

2.3 Extracción de características

La extracción de características se refiere al proceso de analizar las imágenes capturadas por la cámara Raspberry Pi para identificar patrones, formas, colores y otras propiedades que permitirán clasificar los objetos reciclables. OpenCV ofrece una amplia gama de herramientas para realizar esta tarea. Algunas de las técnicas de extracción de características más comunes que se pueden usar para la aplicación de visión artificial incluyen:

a) Características de forma (Shape)

- **Contornos**: usando la función *cv2.findContours()*, se puede identificar las formas de los objetos en la imagen. Los contornos representan los límites de los objetos y pueden ayudar a determinar su forma (rectangular, circular, etc.).
- **Huella de momentos (Moments)**: los momentos de una imagen o un contorno proporcionan información sobre su forma, como el centro de masa, el área y la orientación. En OpenCV se calculan los momentos utilizando *cv2.moments()*.

b) Características de textura

- **Histogramas de gradientes orientados (HOG)**: el descriptor HOG es una técnica que captura la estructura y la textura de los objetos, utilizando la distribución de los gradientes de la imagen. Se puede usar para identificar patrones y características más complejas.
- **Características de la textura (GLCM)**: las Matrices de Co-ocurrencia de Niveles de Gris (GLCM) son una herramienta que describe la textura en imágenes y puede ser útil para diferenciar materiales reciclables como plástico, vidrio o cartón, que pueden tener diferentes texturas.

c) Características de color

- **Histogramas de color**: los histogramas de color son una de las formas más comunes de representar las características de color de una imagen. Esto puede ser útil si los objetos reciclables tienen colores distintivos. En OpenCV *cv2.calcHist()* calcula el histograma de una imagen en diferentes espacios de color (por ejemplo, HSV o RGB).
- **Espacios de color HSV o LAB**: cambiar el espacio de color de RGB a HSV o LAB puede mejorar la segmentación de objetos con iluminación variable o condiciones de luz complejas.

d) Características de bordes

- **Detección de bordes (Canny, Sobel)**: la detección de bordes es crucial para segmentar y detectar objetos. El algoritmo de Canny (*cv2.Canny()*) o el filtro de Sobel *(cv2.Sobel()*) pueden ayudar a identificar los bordes de los objetos dentro de la imagen.

2.4 Clasificación y Localización de objetos

Una vez extraídas las características de la imagen, estas pueden ser utilizadas para clasificar los objetos reciclables. La clasificación generalmente se realiza utilizando técnicas de aprendizaje automático o redes neuronales. Para esto, las características extraídas de las imágenes pueden ser transformadas en vectores de características que luego se alimentan a un modelo de clasificación como un **SVM**, **k-NN**, o incluso una red neuronal convolucional (CNN) si el sistema está orientado hacia un enfoque más avanzado.

A continuación, se muestra una lista de métodos y tecnologías para la clasificación de objetos:

- **Clasificación basada en Umbrales de Color:** Utiliza valores de color (RGB, HSV, etc.) para identificar y clasificar objetos según su color predominante.
- **Segmentación de Imagen:** Separa los objetos del fondo utilizando técnicas como umbrales, contornos, o segmentación semántica para aislar las regiones de interés.
- **Histogramas de Gradientes Orientados (HOG):** Extrae características de la forma y la textura de los objetos a través de los gradientes en la imagen, útil para clasificar objetos basados en sus contornos.
- **Redes Neuronales Convolucionales (CNN):** Modelos de aprendizaje profundo que aprenden automáticamente a clasificar objetos a partir de grandes volúmenes de datos de imágenes etiquetadas.
- **Máquinas de Vectores de Soporte (SVM):** Utiliza un algoritmo de clasificación supervisada basado en la creación de un hiperplano óptimo que separa las clases de objetos según las características extraídas.
- **Clasificación por K-Vecinos más Cercanos (K-NN):** Clasifica objetos basándose en la similitud entre el objeto de prueba y los objetos ya etiquetados en el conjunto de datos.
- **Redes Neuronales de Convolución Profunda (Deep Learning):** Extensión avanzada de las CNN, capaz de aprender representaciones jerárquicas más complejas para clasificar objetos con alta precisión en tareas de visión artificial.
- **Análisis de Componentes Principales (PCA):** Técnica de reducción de dimensionalidad, que ayuda a reducir la complejidad de los datos al conservar las características más relevantes para la clasificación.
- **Máquinas de Soporte de Vectores (SVM) con Características de Textura:** Extiende SVM para clasificar objetos a partir de características que describen la textura superficial del objeto, como la energía o entropía.
- **Redes Neuronales Recurrentes (RNN):** Se utilizan cuando se trata de secuencias de imágenes o información temporal (como análisis de movimiento), mejorando la clasificación en sistemas dinámicos.
- **Clasificación por Algoritmos de Clustering (K-Means, DBSCAN):** En lugar de asignar etiquetas predefinidas, estos métodos agrupan objetos similares basándose en características comunes extraídas de las imágenes.

- **Características Locales (SIFT, SURF, ORB):** Detectan puntos clave en una imagen (por ejemplo, bordes o esquinas) que se utilizan para clasificar objetos mediante su correspondencia en otras imágenes.
- **Redes Generativas Adversariales (GANs):** Generan nuevas imágenes de objetos a partir de datos existentes, lo que puede ayudar a mejorar la clasificación mediante la creación de datos de entrenamiento adicionales.
- **Algoritmos de Redes de Convolución Transferida (Transfer Learning):** Utiliza modelos previamente entrenados en grandes conjuntos de datos para aplicar conocimientos adquiridos a un conjunto de datos más pequeño y específico, como el reconocimiento de objetos reciclables.
- **Análisis de Características Texturales (GLCM):** Utiliza la matriz de co-ocurrencia de nivel de gris para analizar la textura de los objetos y ayudar en su clasificación, especialmente útil para materiales con texturas particulares.
- **Detección de Contornos (Canny, Sobel):** Identifica los bordes de los objetos en una imagen, lo que puede ser útil para clasificar formas o estructuras básicas en los objetos.
- **Redes Neuronales de Convolución en Tiempo Real (YOLO, SSD, Faster R-CNN):** Algoritmos de detección de objetos en tiempo real que no solo localizan objetos en una imagen, sino que también los clasifican, permitiendo una identificación y acción rápida.
- **Clasificación Basada en Deep Reinforcement Learning (DRL):** Utiliza aprendizaje por refuerzo profundo para aprender la clasificación de objetos de manera iterativa, donde el modelo recibe retroalimentación por su desempeño.
- **Clasificación mediante Modelos Basados en Árboles (Random Forest, Decision Trees):** Construcción de un conjunto de árboles de decisiones que clasifican objetos basándose en las características extraídas de la imagen.
- **Clasificación Basada en Redes Neuronales Artificiales (ANN):** Emplea una red neuronal tradicional para clasificar objetos de acuerdo con las características extraídas, aunque menos eficiente que las CNN para tareas de visión.

Debido al enfoque automatizado del sistema para el control del brazo robótico, se decidió elegir la técnica de clasificación de objetos **YOLO (You Only Look Once)** por su capacidad para realizar detección y clasificación de objetos en tiempo real con alta precisión.

Una vez que YOLO ha identificado y clasificado los objetos en la imagen, el siguiente paso es determinar su ubicación dentro del marco de visión establecido. La **localización** de objetos es crucial para que el brazo robótico pueda interactuar correctamente con ellos. YOLO, además de identificar objetos, proporciona las **coordenadas de los límites (bounding boxes)** de cada objeto detectado, que son las posiciones de la esquina superior izquierda y la esquina inferior derecha del cuadro delimitador. Estas coordenadas permiten al sistema conocer la posición de los objetos en el espacio de la imagen.

2.4.1 YOLO (You Only Look Once)

YOLO es un algoritmo de aprendizaje automático creado por Joseph Redmon y Ali Farhadi, que utiliza características aprendidas por una red neuronal convolucional (CNN) profunda para la detección de objetos en imágenes. YOLO divide la imagen en una cuadrícula y predice las cajas delimitadoras y las probabilidades de clase para cada celda. A diferencia de los métodos tradicionales que utilizan ventanas deslizantes o propuestas de región, YOLO puede detectar y clasificar objetos en una imagen con una sola pasada. Este algoritmo basado en redes neuronales convolucionales permite no sólo identificar, sino también, localizar los objetos reciclables en el área de visión establecida, lo cual es fundamental para la posterior manipulación por parte del brazo robótico. Su velocidad y efectividad en entornos dinámicos, además de su habilidad para reconocer múltiples objetos simultáneamente, hacen que YOLO sea la solución ideal para clasificar rápidamente los objetos durante el proceso de reciclaje automatizado.

No todo es positivo en el uso de la técnica YOLO, a pesar de las ventajas mencionadas, puede presentar problemas con objetos pequeños o superpuestos, ya que, solo puede predecir un número fijo de cajas por celda. Puede pasar por alto algunos objetos que no encajan bien en la cuadrícula, o dar falsos positivos para las regiones de fondo. Además, requiere muchos datos de entrenamiento con los suficientes recursos computacionales, no siendo eficaz su uso en determinados dominios o dispositivos.

2.5 OpenCV

Figura 1 - Logo de OpenCV

OpenCV es una biblioteca multiplataforma de visión artificial escrita en C/C++ de código abierto, desarrollada por la división rusa de Intel en el centro de Nizhni Nóvgorod, bajo la licencia open source BSD. Cuenta con más de 500 funciones enfocadas al procesamiento de imágenes, como reconocimiento de objetos, detección de movimiento, análisis de imagen/video, visión estérea, calibración de cámaras y visión robótica.

Las operaciones de procesamiento de imágenes en OpenCV se llevan a cabo mediante la manipulación de matrices de píxeles, que representan la imagen digital y tiene una gran cantidad de funciones y herramientas para esta tarea, es por esto que fue elegido para el desarrollo del sistema de visión artificial.

Su instalación requirió de la instalación previa de algunas librerías para que su utilización para el procesamiento de imágenes sea correcta.

2.5.1 Prerrequisitos e instalación

Para que OpenCV funcione correctamente, especialmente en plataformas como Raspberry Pi con el sistema operativo Raspbian, es necesario instalar ciertos paquetes y dependencias previas que aseguren el funcionamiento adecuado de la biblioteca.

Los principales prerrequisitos para la instalación de OpenCV en Raspbian son:

- **libjpeg-dev**: biblioteca necesaria para manejar imágenes JPEG.
- **libtiff5-dev**: proporciona soporte para el manejo de imágenes en formato TIFF.
- **libjasper-dev**: paquete para trabajar con imágenes en formato JPEG2000.
- **libpng-dev**: paquete que permite trabajar con imágenes PNG.
- **libavcodec-dev, libavformat-dev**: bibliotecas que permiten el manejo de flujos de video y códecs de vídeo (necesarias para la manipulación de video en tiempo real).
- **libeigen3-dev**: es una librería que optimiza operaciones matemáticas y es útil para mejorar el rendimiento de los cálculos computacionales de OpenCV.
- **python3-dev**: es necesario para la integración de OpenCV con Python, ya que proporciona los encabezados de desarrollo de Python.

La instalación de estas librerías y dependencias puede llevarse a cabo con el siguiente comando:

```
C:\Users\f91p>sudo apt-get install build-essential cmake pkg-config libjpeg-dev libtiff5-dev libjasper-de
v libpng-dev libavcodec-dev libavformat-dev libswscale-dev libv4l-dev libxvidcore-dev libx264-dev libfont
config1-dev libcairo2-dev libgdk-pixbuf2.0-dev libpango1.0-dev libgtk2.0-dev libgtk-3-dev libatlas-base-d
ev gfortran libhdf5-dev libhdf5-serial-dev libhdf5-103 python3-pyqt5 python3-dev -y
```

Este comando instalará las librerías necesarias para que OpenCV funcione correctamente, permitiendo el procesamiento de imágenes y video en tiempo real en el sistema Raspbian.

Una vez instaladas las dependencias, se procede a la instalación de OpenCV utilizando pip:

```
C:\Users\f91p>pip install opencv-python
```

2.5.2 Alternativas y/o complementos

OpenCV es una de las bibliotecas de visión artificial más populares y utilizadas en el procesamiento de imágenes y videos en tiempo real, sin embargo, existen otras bibliotecas que también son muy útiles para esta tarea, y algunas de estas son las siguientes:

Figura 2 - Logo de Scikit Image

Scikit Image es una biblioteca o herramienta de procesamiento de imágenes de código abierto, que contiene algoritmos para segmentación, transformaciones geométricas, manipulación del espacio de color, análisis, filtrado y muchos otros.

Figura 3 - Logo de SimpleCV

SimpleCV es una biblioteca de código abierto, que se utiliza para la visión artificial y el aprendizaje automático.

Figura 4 - Logo de NumPy

NumPy es una biblioteca de Python, que proporciona una gran cantidad de funciones matemáticas para el manejo de arreglos y matrices.

Figura 5 - Logo de Imutils

Imutils es una biblioteca, que se utiliza para simplificar la realización de tareas comunes en OpenCV.

2.5.3 Funciones más importantes

Algunas de las funciones más utilizadas de OpenCV son las que se detallan a continuación:

- ***VideoCapture*:** se utiliza para la captura de imágenes y recibe como parámetro el índice de la cámara o la ruta del archivo de video. Si el índice de la cámara que se pasa como parámetro es cero se captura la webcam principal.
- ***read*:** captura una imagen y retorna una tupla con un indicador booleano de éxito y la imagen 'frame' capturada.

- ***rezise*:** cambia el tamaño de la imagen. Recibe como parámetros la imagen original y el tamaño esperado.
- ***Canny*:** con el algoritmo Canny de OpenCV se pueden detectar bordes en la imagen. Recibe como parámetros la imagen original y los valores umbral para la detección de bordes.
- ***Harris*:** Harris es uno de los algoritmos presentes en OpenCV para la detección de características, por ejemplo, esquinas.

2.6 Evaluación del módulo de visión

Se procedió a evaluar el módulo de visión implementado en este sistema, que utiliza un enfoque de visión por computadora basado en un modelo previamente entrenado para la detección y clasificación de objetos reciclables. Este modelo, que consiste en un conjunto de archivos de pesos (.weights), configuración (.cfg) y clases (coco.names), fue entrenado para reconocer una variedad de objetos de interés, tales como plásticos, latas y vidrio, entre otros.

Para la captura de imágenes, se empleó una cámara Raspberry Pi, que permite obtener imágenes en tiempo real con una resolución adecuada para el proceso de detección. Sin embargo, en la fase inicial de pruebas, se utilizó una webcam de PC con conexión USB, para facilitar el desarrollo y optimizar la fase de validación del sistema. Esta cámara proporciona imágenes con suficiente calidad y definición para que el módulo de visión identifique correctamente los objetos en el entorno.

El sistema de visión procesa las imágenes mediante la librería OpenCV, que realiza el preprocesamiento de las imágenes (como la conversión a escala de grises y la reducción de ruido) para la detección de los objetos. Una vez detectados los objetos, se dibujan cajas delimitadoras rectangulares sobre cada uno de estos, lo que permite visualizarlos claramente. Además, el sistema es capaz de proporcionar información adicional sobre cada objeto detectado, como la clase a la que pertenece y su ubicación en la imagen (coordenadas de la caja delimitadora).

Este enfoque permitió una precisa clasificación de los objetos reciclables y una interacción fluida entre el módulo de visión y el módulo de control del brazo robótico.

3 Módulo de control

El módulo de control es el componente esencial encargado de la gestión y ejecución de las acciones del brazo robótico. Este módulo está basado en una Raspberry Pi 4, que actúa como el cerebro del sistema, ejecutando una aplicación desarrollada en Python que contiene las funciones de visión por computadora y el control de movimientos del brazo robótico.

La visión por computadora, implementada mediante la librería OpenCV, procesa las imágenes capturadas por una cámara Raspberry Pi posicionada estratégicamente para monitorizar una cinta transportadora de objetos. Esta cámara proporciona datos en tiempo real que permiten identificar y clasificar los objetos en función de si son o no reciclables, lo cual se utiliza como base para la toma de decisiones del sistema.

Una vez que el módulo de visión identifica un objeto y determina si es reciclable, se envían los comandos correspondientes al servocontrolador del brazo robótico LeArm, quien, siguiendo dichas órdenes, realiza el agarre del objeto en un punto previamente establecido sobre la cinta transportadora. Posteriormente, el brazo robótico traslada el objeto al contenedor correspondiente según su clasificación, asegurando una correcta separación de los materiales reciclables.

3.1 Arquitectura del sistema

3.1.1 Raspberry Pi

Se utiliza el sistema de cómputo ofrecido por la placa Raspberry Pi 4 con el que se realizarán las tareas de procesamiento de la imagen detectada por la cámara de video.

Figura 6 - Módulo de cámara Rasberry Pi.
Fuente: Recuperado de https://ecuarobot.com/wp-content/uploads/2020/02/camera-module.png (2024).

Se hace uso del módulo de cámara de Rasberry Pi para capturar las imágenes que serán analizadas para el reconocimiento y ubicación de los objetos reciclables. Esta estará conectada a la Raspberry Pi mediante un puerto de módulo de cámara.

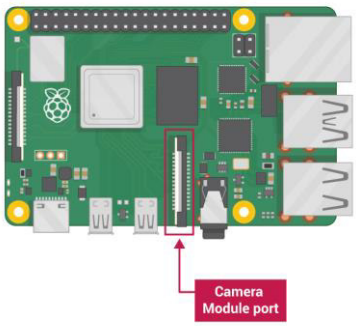


Figura 7 - Placa Rasberry Pi 4.

Fuente: Recuperado de https://ecuarobot.com/wp-content/uploads/2020/02/pi4-camera-port.png (2024).

Las conexiones de la placa son las siguientes:

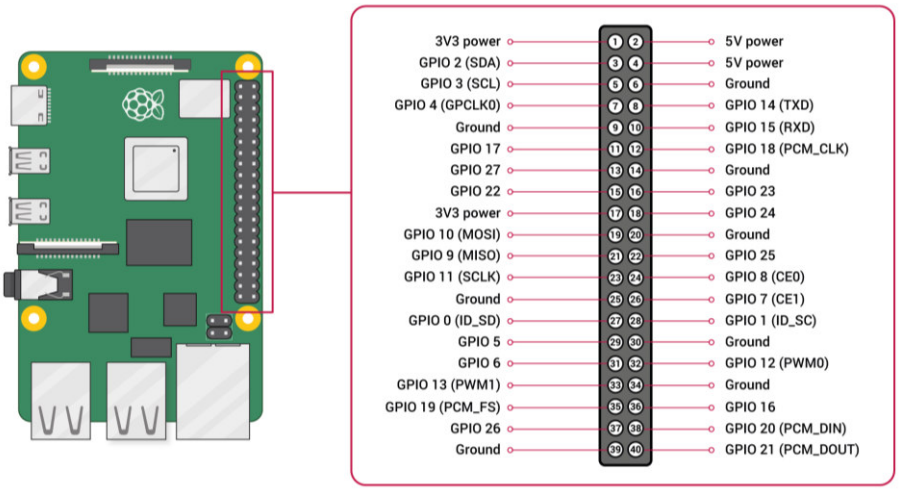


Figura 8 - GPIO y el cabezal de 40 pines.

Fuente: Recuperado de https://www.raspberrypi.com/documentation/computers/raspberry-pi.html (2024).

Se utiliza una carcasa para protección de la placa Raspberry Pi 4 junto con un cooler 30x30 5v para una mejor disipación del calor que esta genera mientras está encendida y en ejecución de la aplicación.

3.1.2 Brazo robótico

El brazo robótico utilizado es el **Robot LeArm** que cuenta con seis servomotores con un alto grado de precisión, permitiendo mucha flexibilidad en su movimiento, pudiendo agarrar objetos en cualquier dirección.

Figura 9 - LeArm Robot.

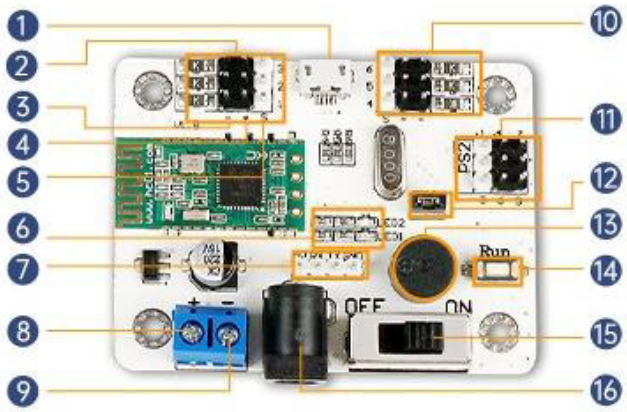
Fuente: Elaboración propia, basada en la guía de observación.

Se lo puede controlar mediante un Joystick inalámbrico, o bien, de manera remota a través de una aplicación en su versión móvil, disponible tanto en Android como en iOS y de escritorio para Windows, las cuales se describen más adelante.

Servomotor: Un servomotor, también conocido como servo, es un actuador rotativo o motor que se puede mover en un ángulo, posición y a una velocidad determinada en cada momento, algo que no puede hacer un motor eléctrico normal. En definitiva, utiliza un motor normal y lo combina con un sensor para la retroalimentación de posición.

Servocontrolador del brazo robótico GTQ278.88: El servocontrolador GTQ278.88 es la placa utilizada para controlar los servomotores del brazo robótico. Algunas de sus características son:

- Voltaje: opera a 5V.
- Canales: tiene 6 canales con los que puede controlar múltiples servos.
- Conexiones de entrada: soporta Bluetooth 4.0, proporcionando una opción de conectividad inalámbrica para control remoto o comunicación con otros dispositivos, soporte para controlador PS2 (un joystick de PlayStation 2) y pines TX, RX y GND que permiten conectar las señales de control provenientes de una fuente como la Raspberry Pi.
- Tipo de señal: usa señales PWM (Pulse Width Modulation) para controlar la posición de los servomotores.
- Interfaz de comunicación: UART mediante los pines físicos TX (Transmisión) y RX (Recepción).
- Protección:
    - Protección contra Sobre Corriente: esta protección ayuda a evitar daños en los servos y en la placa en caso de que se produzcan sobrecorrientes.
    - Alarma de Bajo Voltaje: soporta una alarma de bajo voltaje, que avisa cuando la alimentación está por debajo de un nivel seguro para el funcionamiento del dispositivo, ayudando a prevenir apagones inesperados o fallos de funcionamiento.
    - Protección de Aislamiento de Señal: la protección de aislamiento de señal ayuda a reducir la interferencia y a proteger las señales de control, garantizando una comunicación estable y reduciendo el riesgo de daños por señales eléctricas no deseadas.
- Tamaño y Peso:
    - Tamaño de montaje: 49mm x 34mm
    - Peso: 20g



1. Conector Android
2. Puertos de conexión de servos 1-3 con protección contra sobrecorrientes
3. Servo Negativo
4. Servo Positivo
5. Terminal de señal
6. LED1: Indicador de encendido
   LED2: Indicador de comunicación
7. Puerto de comunicación de desarrollo secundario
8. VCC+
9. VCC-
10. Puertos de conexión de servos 4-6 con protección contra sobrecorrientes
11. Puerto receptor del mando PS2
12. Interruptor de alarma de baja tensión
13. Alarma sonora de baja presión
14. Botón de funcionamiento sin conexión
15. Interruptor de encendido
16. Puerto de alimentación DC

Figura 10 - Servocontrolador GTQ278.88.

Fuente: Elaboración propia, basada en la guía de observación.

3.1.2.1 Conexión de la Raspberry Pi con el brazo robótico

Para realizar la conexión de la placa Raspberry Pi 4 con el brazo robótico “Robot LeArm” se consideran los niveles de voltaje con los que operan ambos dispositivos. Los pines GPIO de la PI operan a 3.3V y el servocontrolador del brazo trabaja con un nivel lógico de 5V.

Para evitar daños o mal funcionamiento se utiliza un adaptador de nivel lógico para garantizar que las señales sean seguras y comprendidas correctamente por ambos dispositivos. Esto es necesario especialmente cuando se quiere trasmitir datos mediante el pin TX del dispositivo de 5V al pin RX de recepción de datos que opera a 3.3V, ya que el primero puede enviar una señal que supera los 3.3V y dañar el receptor. Este adaptador se conecta entre los dos dispositivos y maneja la conversión de voltaje automáticamente. Esto ayuda a protegerlos y garantiza una comunicación fiable entre ellos.

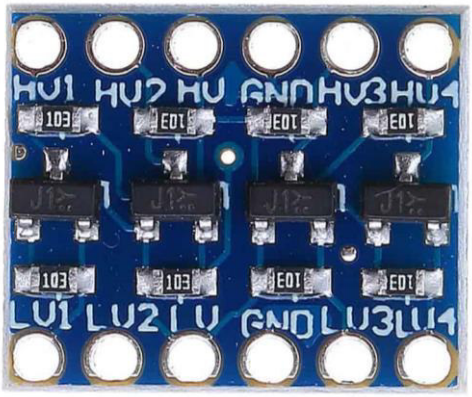


Figura 11 - Adaptador de nivel lógico.
Fuente: Recuperado de https://www.pcboard.ca/image/cache/catalog/products/logic-level-converter/logic-level-converter-05-800x800.jpg (2024).

También se realiza la conexión de los pines GND (Tierra) de ambos dispositivos. Esto es necesario, ya que permite que las señales de voltaje se interpreten correctamente.

3.1.2.2 Aplicación oficial

El software del brazo robótico, tanto en su versión móvil (disponible solo en algunos dispositivos) como en su versión de escritorio, permite una interacción y control de sus movimientos de manera remota.

La versión móvil se conecta con el robot mediante la tecnología Bluetooth 4.0, mientras que la versión de escritorio lo hace mediante un mini cable USB, cuyos drivers se instalan automáticamente en el sistema operativo Windows.

Las versiones del software según plataforma son las siguientes:

- LeArm V2.0 (Windows)
- LeArm V1.2 (Android/iOS)

3.1.2.2.1 Acciones preprogramadas

El software tiene 10 acciones preprogramadas para visualizar el movimiento del brazo. Las acciones que se llevan a cabo son la de rotación de los correspondientes servomotores en un ángulo y tiempo definidos. Según estas acciones sincronizadas se dota al brazo robótico de nuevas posibilidades y características como, por ejemplo, agarrar un objeto, tarea básica para la cual fue concebido.

3.1.2.2.2 Acciones personalizadas

La aplicación permite programar acciones con el objetivo de realizar movimientos personalizados al brazo robótico.

Para lograr esto de manera óptima se realizó una inspección del funcionamiento del brazo y se observó detenidamente cómo respondía cada uno de sus servomotores a los estímulos provenientes de las acciones personalizadas que se programaban. De esta manera, se pudo constatar cómo influyen los valores establecidos en el software con la respuesta específica del brazo robótico, los ángulos de rotación a los que corresponde cada uno y el tiempo necesario para realizar un movimiento lo más natural posible.

Para este proyecto se propone que los servomotores del brazo tengan una rotación de 180°, 90° hacia la izquierda y -90° hacia la derecha, siendo el ángulo 0 la posición por defecto del robot. Cada ángulo está asociado a un valor dentro del software del brazo robótico, siendo el ángulo 0 equivalente al valor 1500, el ángulo 90° equivalente a 2500 y el -90° a 500. Ese es el rango de valores que maneja el software para los servomotores 6 a 2. Para el servomotor 1, servomotor que maneja la apertura y cierre de la garra del brazo, sus valores de rotación van de 1500 a 2500, siendo 1500 la garra completamente abierta y 2500 la garra completamente cerrada.

Para poner a prueba al brazo robótico se procedió a programar diferentes acciones que en su conjunto realizaban la tarea de recoger un objeto exactamente a 90° a la derecha de este, levantarlo quedando el brazo completamente erguido (posición por defecto) para finalmente volver a dejar el objeto exactamente en donde lo agarró. Después de ir definiendo las acciones y probar las respuestas del brazo se logra cumplir con esta tarea, que es una tarea elemental en este proyecto.

3.1.2.3 Aplicación Python

Se desarrolló una aplicación en el lenguaje de programación Python llamada **EcoArm** cuyo objetivo es procesar imágenes capturadas mediante un módulo de cámara web y determinar los movimientos correspondientes a cada servomotor del brazo robótico. El procesamiento de las imágenes se lleva a cabo utilizando la biblioteca **OpenCV**, herramienta ampliamente utilizada en el ámbito de la visión por computadora.

La técnica empleada para la detección y clasificación de objetos en las imágenes es **YOLO (You Only Look Once)** en su versión 3, un modelo avanzado de redes neuronales

convolucionales centrado en la identificación rápida y precisa de objetos. Para este caso específico, se utiliza un modelo preentrenado, que permite reconocer objetos reciclables, tales como: plásticos, vidrios, papeles y cartón.

La comunicación entre la aplicación y el servocontrolador del brazo se realiza mediante el protocolo de comunicación serial **'LSC Series Servo Controller Communication Protocol V1.2'**, el cual permite una correcta y eficiente interacción entre los componentes del sistema y un control adecuado de los movimientos del brazo robótico.

A continuación, se muestra el diagrama de flujo de la aplicación EcoArm:

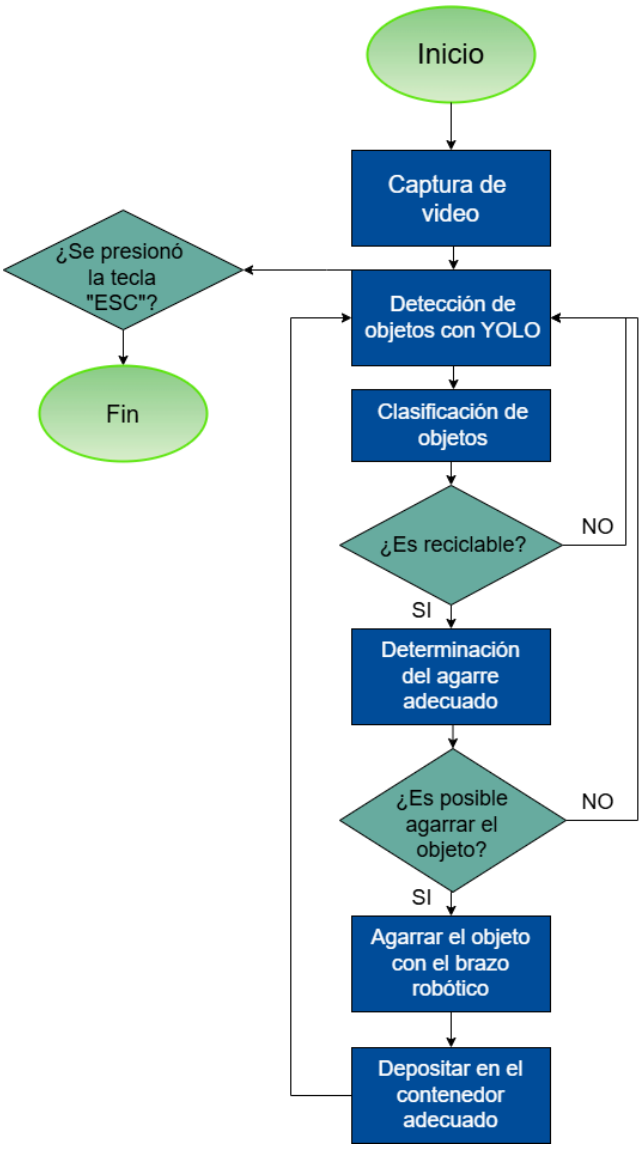


Figura 12 - Diagrama de flujo de la aplicación EcoArm.
Fuente: Elaboración propia, basada en la práctica.

3.2 Planificación de trayectorias

Para realizar las tareas definidas en este proyecto es necesario crear en tiempo real acciones que establezcan el movimiento correcto, que debe hacer el brazo en cada momento. Estas tareas comienzan con la detección y clasificación de objetos visualizados a través de una cámara, posteriormente se determina la distancia de ese objeto para saber si está al alcance del brazo y, por último, se programan las acciones pertinentes para realizar el movimiento de los servomotores en ángulo y tiempo precisos con el fin de que el brazo pueda agarrar el objeto. En este caso lo objetos que identificará y agarrará el brazo robótico serán reciclables, por lo que la misión que este tendrá una vez agarrado el objeto será volcarlo en el contenedor apropiado.

3.3 Control del efector final

Se define al efector final como a la ejecución física de la tarea encomendada al brazo robótico. Este proceso se debe dar de manera controlada teniendo en cuenta las dimensiones del objeto que será agarrado con la garra del brazo y la trayectoria que este debe seguir hasta ser depositado a su ubicación final.

Para controlar el efector final del brazo se realizan los cálculos de la dimensión, posición y distancia del objeto. Se analizan estos datos y se determina, primero, si es reciclable, y después, si es factible agarrar el objeto o no. De ser así, se procede a precisar el movimiento exacto que debe hacer el brazo para realizar su tarea.

3.4 Integración con el módulo de visión

Para llevar a cabo la integración con el módulo de visón es necesario que ambos módulos hablen el mismo idioma para que la comunicación se pueda dar de manera correcta. El idioma que entiende el servocontrolador del brazo robótico está definido por el

protocolo de comunicación serial (LSC Series Servo Controller Communication Protocol V1.2).

3.4.1 Protocolo de comunicación

A continuación, se describe el funcionamiento del protocolo de comunicación serial a partir del cual el brazo puede comprender cómo se espera que se mueva. Los datos son leídos por el servocontrolador byte por byte, y estos son:

- **Header:** dos 0x55 consecutivos indica que llegaron los paquetes de datos.
- **Length:** es igual al número de parámetro N más un comando y más la longitud en bytes que ocupa la propia longitud de datos. Eso significa que la longitud de los datos es igual al parámetro N más 2 (Length = N + command + a byte length = N + 2).
- **Command:** contiene varias instrucciones de control.
- **Parameter:** se utiliza para agregar información de control

Los pines que transmiten los datos del controlador del usuario, que en este caso es la placa Raspberry Pi 4b, están conectados al pin RX del servocontrolador del brazo y ambos conectores a tierra (GND) conectados entre sí.

Si la transmisión de datos al servocontrolador por parte del usuario es correcta, el LED azul 2 en el controlador parpadeará una vez para indicar que se han recibido los datos correctos. Si el usuario transmite los datos incorrectos, entonces el LED azul 2 no tendrá ninguna reacción y mantendrá el brillo, luego el zumbador emitirá un pitido dos veces para recordarle al usuario que hay un error con los datos.

Ejemplo de la interacción entre la Respberry pi y el sistema de control del brazo en la ejecución del comando, que permite el movimiento de los servomotores:

Nombre: **CMD_SERVO_MOVE**

Valor: 3

Longitud = n_servos_a_controlar * 3 + 5 = n_parámetros + 2

Parámetros

- Parámetro 1: el número de servos a controlar
- Parámetro 2: 8 bits menos significativos del valor de tiempo
- Parámetro 3: 8 bits más significativos del valor de tiempo

- Parámetro 4: número de identificación del servo
- Parámetro 5: 8 menos significativos del valor de posición del ángulo
- Parámetro 6: 8 bits más significativos del valor de posición del ángulo
- Parámetro …: el formato es el mismo que el parámetro 4, 5 y 6, controla las posiciones de ángulo de diferentes ID.

Ejemplo:

1. Controlar el giro del servo n° 2 a la posición 2000 y el giro del servo n° 9 a la posición 2300 en 800 ms.

Header	**Length**	**Command**	**Parameter**
0x55 0x55	0x0B	0x03	0x02 0x20 0x03 0x02 0xD0 0x07 0x09 0xFC 0x08

Sistema / Parám	**Decimal**	**Hexadecimal**		**Binario**	
		MSB	**LSB**	**MSB**	**LSB**
P1	2	0x02		00000010	
P2 y P3	800	0x03	0x20	00000011	00100000
P4	2	0x02		00000010	
P5 y P6	2000	0x07	0xD0	00000100	10110000
P7	9	0x09		00001001	
P8 y P9	2300	0x08	0xFC	00001000	11111100

3.5 Evaluación del módulo de control

Para evaluar el módulo de control se procedió a integrar los dispositivos que lo componen mediante la conexión serial (TX y RX) entre la placa Raspberry Pi 4 y el

servocontrolador del brazo robótico. Para ello, se desarrolló una función encargada de estructurar y preparar los datos a ser enviados al servocontrolador, formando paquetes de información en formato de bytes. Esta estructura se definió respetando el protocolo de comunicación serial través de la interfaz UART para asegurar una correcta transmisión de los comandos.

Una vez implementada la función de transmisión, se envió un comando desde la aplicación Python que controla el sistema al servocontrolador del brazo y la respuesta del sistema fue satisfactoria, ya que el brazo ejecutó las acciones indicadas de acuerdo con las especificaciones previamente establecidas.

A continuación, se definió una ubicación dentro del marco de la escena en la cual los objetos, transportados por una cinta, serían detenidos en un punto específico frente al brazo robótico. En este punto, el módulo de visión del sistema realiza un análisis para determinar si el objeto es reciclable o no. Dependiendo de este análisis, el brazo depositaría el objeto en el contenedor correspondiente, ubicado a un lado del sistema.

Con el marco de la escena y la ubicación de los contenedores definidos, se procedió a colocar un objeto en el punto de análisis del brazo robótico. Posteriormente, se desarrolló la función encargada de mover el brazo de manera precisa y fluida, trasladando el objeto hacia el contenedor correspondiente. Este movimiento fue revisado y debidamente ajustado para garantizar la correcta ejecución de las tareas, considerando tanto los tiempos de respuesta como la trayectoria del brazo. Los ajustes necesarios fueron realizados mediante pruebas iterativas de los comandos enviados y observando las respuestas del sistema, lo que permitió optimizar el comportamiento del brazo robótico y garantizar un desplazamiento eficiente.

4 Conclusiones

En primer lugar, con relación a los principales aportes del proyecto, se puede concluir que se investigaron diferentes alternativas para visión por computadora y se eligió la considerada más adecuada según el sistema al que se implementaba (cámara y módulo de control). Para el procesamiento de las imágenes obtenidas por la cámara, se configuró y optimizó una Raspberry Pi 4, que ejecuta una aplicación en Python, utilizando la librería OpenCV. Al utilizar una Raspberry Pi 4 y una cámara de bajo costo, el proyecto demuestra cómo las tecnologías accesibles pueden ser implementadas de manera efectiva en sistemas industriales. Esto promueve la democratización de la tecnología en áreas clave como el reciclaje.

La implementación de un modelo preentrenado basado en la técnica YOLO para la clasificación de objetos permitió identificar si estos son reciclables y determinar el tipo de contenedor adecuado para cada uno. Según esta clasificación el sistema decide y orienta el movimiento del brazo robótico para depositar los objetos en los lugares correctos. Este sistema no solo mejora la eficiencia en la clasificación de materiales reciclables, sino que también reduce el error humano y optimiza el proceso de recolección, lo que podría mejorar significativamente la tasa de reciclaje en industrias o centros de tratamiento de residuos.

En segundo lugar, se incluyen recomendaciones para la mejora del sistema, y esta consistiría en dotar al brazo robótico de la capacidad de determinar con mayor precisión la posición exacta, el tamaño y el peso de los objetos. Esto permitiría evaluar si el objeto puede ser agarrado de manera segura por la pinza, considerando tanto su tamaño como su peso, y si se encuentra dentro del alcance del brazo. Para ello, sería necesario desarrollar una función que calcule el movimiento y la trayectoria del brazo, así como la manipulación del objeto por el efector final, garantizando así una mayor precisión en el proceso de agarre y traslado.

Además, el brazo robótico podría integrar una función que le permita ajustar su operación a la velocidad de la cinta transportadora, de modo que pueda realizar la separación de los objetos reciclables mientras la cinta sigue en movimiento. Esto optimizaría el proceso al permitir que los objetos sean clasificados y agarrados sobre la marcha, lo que incrementaría significativamente la eficiencia y reduciría los tiempos de operación.

Se podría implementar un sistema de retroalimentación en tiempo real incorporando sensores adicionales para monitorear constantemente la fuerza de agarre y la estabilidad del objeto durante la manipulación. De esta forma, si el objeto es demasiado resbaladizo o tiene una forma irregular, el sistema podría ajustar automáticamente el agarre o solicitar un reajuste en la pinza.

La integración de un sistema de visión estéreo o 3D podría permitir al sistema obtener una visión más completa del entorno, mejorando la capacidad de estimar la distancia y el volumen de los objetos, lo que contribuiría a un agarre más preciso y a una mejor toma de decisiones en cuanto a la manipulación.

Tener en cuenta la eficiencia energética para el uso que el sistema representa en sus horas operativas sería de gran utilidad. Se podrían optimizar los tiempos de inactividad del brazo o hacer uso de modos de bajo consumo cuando no se está manipulando activamente un objeto.

5 Reflexión sobre la Práctica Profesional Supervisada

La Práctica Profesional Supervisada (PPS) fue una experiencia que me permitió mostrar los conocimientos aprendidos a lo largo de la carrera de Ingeniería en Informática en la UNAJ.

El conocimiento en programación fue esencial a lo largo de todo el proyecto, ya que las bases adquiridas en Fundamentos de Programación, Algoritmos y Programación y, Complejidad Temporal, Estructura de Datos y Algoritmos me permitieron escribir código eficiente, lo cual fue muy importante para la interacción fluida entre los diferentes componentes del sistema (cámara, algoritmo de visión en Python, y brazo robótico).

También lo aprendido en Organización y Arquitectura de Computadoras y en Lenguajes Formales y Autómatas para la comprensión general de las interconexiones entre dispositivos buscando un trabajo conjunto eficiente entre estos y el software correspondiente. Las materias de Redes para la comprensión de protocolos de comunicación fueron útiles para garantizar una correcta transmisión de datos entre componentes. Las metodologías de trabajo en materias como Ingeniería de Software o Proyecto de Software también fueron fundamentales para la realización de esta PPS.

La UNAJ ha permitido que tenga los espacios necesarios para el aprendizaje y desarrollo de los proyectos durante mi formación académica, adquiriendo de esta manera las herramientas necesarias para llevar adelante esta PPS. Fue muy importante el acompañamiento del tutor Jorge Osio, para avanzar en la realización de la PPS y contar con los recursos necesarios para hacer funcionar el sistema a desarrollar.

6 Bibliografía

Brita IA. Cómo implementar inteligencia artificial para resolver tareas de procesamiento de imágenes. Recuperado de https://brita.mx/como-implementar-ia-para-resolver-tareas-de-procesamiento-de-imagenes/

Datapeaker. Procesamiento de imágenes con OpenCV. Recuperado de https://datapeaker.com/big-data/procesamiento-de-imagenes-opencv-procesamiento-de-imagenes-con-opencv/

Diwan, T; Anirudh, G; Tembhurne, J. (2022, 8 de agosto). Object detection using YOLO: challenges, architectural successors, datasets and applications. Recuperado de https://link.springer.com/article/10.1007/s11042-022-13644-y

IBM. ¿Qué es la visión artificial?. Recuperado de https://www.ibm.com/es-es/topics/computer-vision

Imutils. (2023). Logo. Recuperado de https://pypi.org/static/images/logo-small.2a411bc6.svg

ICHI.PRO. Detección de objetos: comparación de velocidad y precisión (Faster R-CNN, R-FCN, SSD, FPN, RetinaNet y YOLOv3). Recuperado de https://ichi.pro/es/deteccion-de-objetos-comparacion-de-velocidad-y-precision-faster-r-cnn-r-fcn-ssd-fpn-retinanet-y-yolov3-92519592026969

ICHI.PRO. Red neuronal convolucional (CNN) y su aplicación: todo lo que necesita saber. Recuperado de https://ichi.pro/es/red-neuronal-convolucional-cnn-y-su-aplicacion-todo-lo-que-necesita-saber-266752320713701

Kandu, R. (2023, 17 de enero). YOLO: Algorithm for Object Detection Explained [+Examples]. Recuperado de https://www.v7labs.com/blog/yolo-object-detection

Kempf, R (2021, 28 de mayo). Los conceptos básicos del procesamiento de imágenes. Recuperado de https://www.azion.com/es/blog/conceptos-basicos-del-procesamiento-de-imagenes/

LinkedIn. (2023, 27 de julio). ¿Cuáles son algunos de los últimos desarrollos y desafíos en la investigación y aplicaciones de YOLO?

MathWorks. Object Detection Using SSD Deep Learning. Recuperado de https://www.mathworks.com/help/deeplearning/ug/object-detection-using-ssd-deep-learning.html

MathWorks. ¿Qué es el reconocimiento de imágenes?. Recuperado de https://es.mathworks.com/discovery/image-recognition-matlab.html

MathWorks. ¿Qué es el reconocimiento de objetos?. Recuperado de https://la.mathworks.com/solutions/image-video-processing/object-recognition.html

Microsoft Learn. (2022, 13 de octubre). Detección de objetos mediante R-CNN más rápida. Recuperado de https://learn.microsoft.com/es-es/cognitive-toolkit/object-detection-using-faster-r-cnn

Morgunov, A. (2023, 2 de agosto). Object Detection with YOLO: Hands-on Tutorial. Recuperado de https://neptune.ai/blog/object-detection-with-yolo-hands-on-tutorial

NumPy. (2023). Logo. Recuperado de https://numpy.org/images/logo.svg

OpenCV. (2023). Logo. Recuperado de https://opencv.org/wp-content/uploads/2022/05/logo.png.

Openhacks. LeArm Robot. Recuperado de https://www.openhacks.com/page/productos/id/3343/title/LeArm-Robot#lightbox[videos]/0/

Programmer Click. Explicación detallada de la función detectMultiScale de opencv. Recuperado de https://programmerclick.com/article/59831383897/

Programmer Click. Resumen de aprendizaje de R-CNN, SPP-NET, Fast-R-CNN, Faster-R-CNN, YOLO, SSD, R-FCN (actualización continua). Recuperado de https://programmerclick.com/article/10031709234/

PyPro | Ciencia de Datos. Visión Artificial con OpenCv. Recuperado de https://www.pypro.mx/app/curso/vision-artificial-con-opencv/procesamiento-de-imagenes-con-opencv

Raspberry Pi. (2020). [foro]. Interfacing a Raspberry Pi board with LewanSoul LeArm. Recuperado de https://forums.raspberrypi.com/viewtopic.php?t=264521

Raspberry Pi. (2024). [web page]. Raspberry Pi Documentation. Recuperado de https://www.raspberrypi.com/documentation/computers/getting-started.html#setting-up-your-raspberry-pi

Reason.rown. (2022, 15 de agosto). SSD Model for Deep Learning. Recuperado de https://reason.town/ssd-model-deep-learning/

Rosebrock, A. (2018, 26 de febrero). Face detection with OpenCV and deep learning. Última actualización el 4 de julio de 2021. Recuperado de https://pyimagesearch.com/2018/02/26/face-detection-with-opencv-and-deep-learning/

ROS Tutorial. Tutorial visión artificial con OpenCV y Python. Recuperado de http://rostutorial.com/vision-artificial-intro/

SAS. Reconocimiento de imágenes: ¿Qué es y por qué es importante?. Recuperado de https://www.sas.com/es_es/insights/analytics/computer-vision.html

Scikit Image. (2023). Logo. Recuperado de https://scikit-image.org/_static/img/logo.png

SimpleCV. (2023). Logo. Recuperado de http://simplecv.org/assets/SM_logo_color.png

Sojasingarayar, A. (2022, 29 de agosto). Faster R-CNN vs YOLO vs SSD — Object Detection Algorithms. IBM Data Science in Practice. Recuperado de https://medium.com/ibm-data-ai/faster-r-cnn-vs-yolo-vs-ssd-object-detection-algorithms-18badb0e02dc

Tan, L; Huangfu, T; Wu, L; Chen. (2021, 22 de noviembre). Comparison of RetinaNet, SSD, and YOLO v3 for real-time pill identification. BMC Medical Informatics and Decision Making. Recuperado de https://bmcmedinformdecismak.biomedcentral.com/articles/10.1186/s12911-021-01691-8

Tech Lib. Procesamiento de imágenes - Definición y explicación. Recuperado de https://techlib.net/techedu/procesamiento-de-imagenes/

Unipython. Aprende Visión Artificial con OpenCV y YOLO. Recuperado de https://unipython.com/cursos/aprende-vision-artificial-con-opencv/

Viola, P; Jones, M. (2001). Detección rápida de objetos utilizando una cascada potenciada de características simples.

Viso.ai. YOLOv3: Real-Time Object Detection Algorithm (Guide). Recuperado de https://viso.ai/deep-learning/yolov3-overview/

Wikipedia. Procesamiento digital de imágenes. Recuperado de https://es.wikipedia.org/wiki/Procesamiento_digital_de_im%C3%A1genes

Wikipedia. Visión artificial. Recuperado de https://es.wikipedia.org/wiki/Visi%C3%B3n_artificial