Optimización de la herramienta de procesamiento de imágenes para el sistema Brainlab de HUMANA - Fase IV

Sergio Alejandro Boch Ixén

UVG UNIVERSIDAD DEL VALLE DE GUATEMALA

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Optimización de la herramienta de procesamiento de imágenes para el sistema Brainlab de HUMANA - Fase IV

Trabajo de graduación presentado por Sergio Alejandro Boch Ixén para optar al grado académico de Licenciado en Ingeniería Mecatrónica

Guatemala,

2024

Vo.Bo.:

(f) _____

M. Sc. Carlos Esquit

Tribunal Examinador:

(f) _____

M.Sc. Carlos Esquit

(f) _____

M. Sc. Miguel Enrique Zea Arenales

(f) _____

Ing. Kurt Emmanuel Kellner

Fecha de aprobación: Guatemala, 13 de febrero de 2025.

Prefacio

El camino hacia la culminación de este trabajo ha sido profundamente enriquecedor y desafiante, marcado por aprendizajes, sacrificios, alegrías y el apoyo invaluable de quienes me acompañaron. Este recorrido ha dejado una huella indeleble en mi formación profesional y personal, y por ello, quiero iniciar agradeciendo a Dios, por ser mi guía, brindándome salud, sabiduría y fortaleza para completar este proyecto.

Mi más profundo agradecimiento es para mi familia, especialmente, para mis padres, Gustavo Boch y Teresa Ixén, quienes han sido mi mayor fuente de inspiración y apoyo. Gracias por sus invaluables consejos, su amor incondicional y su aliento constante, incluso en los momentos más difíciles. Cada palabra de ánimo, cada gesto de confianza y cada sacrificio suyo ha sido clave para alcanzar este objetivo.

Quiero agradecer a mi asesor, el PhD. Luis Rivera, por su paciencia, dedicación y guía durante la elaboración de mi trabajo de graduación. Sus aportes han sido fundamentales para la calidad y el enfoque de este proyecto. Además, extiendo un agradecimiento especial al MSc. Miguel Zea, quien con su entusiasmo, creatividad y pasión por la enseñanza hizo que estos últimos años fueran no solo productivos, sino también memorables y entretenidos.

Además, no puedo dejar de mencionar a mis amigos y compañeros, quienes hicieron que este viaje universitario fuera más llevadero y lleno de momentos inolvidables. Gracias por el apoyo mutuo, las risas compartidas y por estar ahí en los momentos más significativos. De manera especial, quiero agradecer a mi gran amigo José Pablo Petion, quien me acompañó desde antes, durante el inicio y durante todo este viaje universitario. Su amistad sincera, su lealtad y su compañía constante fueron un pilar fundamental en esta etapa de mi vida. Gracias por estar presente tanto en los momentos de alegría como en aquellos en los que parecía que nadie más estaba.

Finalmente, quiero agradecer a todas aquellas personas, mencionadas o no, que de alguna manera contribuyeron a este viaje universitario, ya sea con un consejo, una palabra de aliento o simplemente con su presencia. Cada uno de ustedes ha sido parte de este logro y de la persona que soy hoy.

Índice

Lista de figuras

Lista de cuadros

Resumen

Este proyecto se enfoca en la optimización de una herramienta de procesamiento de imágenes y reconocimiento de caracteres, utilizada en el sistema Brainlab de HUMANA, con el objetivo de mejorar su precisión y adaptarla a un entorno práctico mediante su integración con un sistema robótico físico. La necesidad de este trabajo surge de las limitaciones detectadas en proyectos previos, tales como errores en el reconocimiento de caracteres y dificultades para validar el sistema con robots físicos.

Se desarrollaron mejoras en los algoritmos de procesamiento de imágenes para aumentar la exactitud y reducir los tiempos de ejecución. Además, se diseñó una interfaz gráfica simplificada que facilita la configuración y operación del sistema, junto con un protocolo de comunicación confiable para conectar la herramienta con un robot físico, el myCobot 280 M5. Este robot fue elegido por su precisión, flexibilidad y compatibilidad con sistemas modernos de visión por computadora, lo cual permitió realizar pruebas exhaustivas en un entorno controlado.

La metodología incluyó el análisis de algoritmos previos, pruebas con capturas de pantalla reales obtenidas del sistema Brainlab, y simulaciones mediante herramientas especializadas como la librería Robotics Toolbox. Finalmente, se validó la efectividad del sistema en términos de precisión, confiabilidad y facilidad de uso, marcando un avance significativo en la automatización de procesos médicos guiados por imágenes en HUMANA. El proyecto está estructurado en etapas clave: introducción y justificación, análisis del marco teórico, desarrollo de algoritmos y la interfaz gráfica, implementación del protocolo de comunicación, y validación experimental con el robot myCobot 280 M5. Este trabajo contribuye al desarrollo de tecnologías innovadoras para la asistencia quirúrgica, estableciendo bases sólidas para futuras iteraciones y aplicaciones prácticas.

Abstract

This project focuses on optimizing an image processing and character recognition tool used in the Brainlab system at HUMANA, aiming to improve its accuracy and adapt it to a practical environment by integrating it with a physical robotic system. The need for this work arises from limitations identified in previous projects, such as errors in character recognition and challenges in validating the system with physical robots.

Improvements were made to the image processing algorithms to increase accuracy and reduce execution times. Additionally, a simplified graphical interface was designed to facilitate system configuration and operation, along with a reliable communication protocol to connect the tool to a physical robot, the myCobot 280 M5. This robot was chosen for its precision, flexibility, and compatibility with modern computer vision systems, enabling comprehensive testing in a controlled environment.

The methodology included the analysis of previous algorithms, tests with real screenshots obtained from the Brainlab system, and simulations using specialized tools such as the Robotics Toolbox library. Finally, the system's effectiveness was validated in terms of accuracy, reliability, and ease of use, marking a significant advancement in the automation of image-guided medical processes at HUMANA. The project is structured into key stages: introduction and justification, analysis of the theoretical framework, development of algorithms and the graphical interface, implementation of the communication protocol, and experimental validation with the myCobot 280 M5 robot. This work contributes to the development of innovative technologies for surgical assistance, establishing a solid foundation for future iterations and practical applications.

CAPÍTULO 1

Introducción

Este proyecto se centra en la optimización de una herramienta de procesamiento de imágenes y reconocimiento de caracteres utilizada en el sistema Brainlab de HUMANA, así como en su integración con un sistema robótico físico. La necesidad de esta optimización surge de las limitaciones identificadas en trabajos previos, que incluyen problemas de precisión en el reconocimiento de caracteres y dificultades en la validación con entornos físicos, como la integración con sistemas robóticos.

Actualmente, el proceso del sistema Brainlab, utilizado en HUMANA para intervenciones médicas se realiza manualmente, lo que es lento y susceptible a errores. En este proyecto, se busca automatizar y mejorar este procedimiento mediante el desarrollo de algoritmos de procesamiento de imágenes optimizados, una interfaz gráfica de usuario simplificada y un protocolo de comunicación efectivo con el robot.

La metodología empleada incluye la revisión y mejora de los algoritmos existentes, pruebas con un mayor número de imágenes obtenidas directamente del sistema Brainlab, y el desarrollo de un sistema de comunicación entre la herramienta de procesamiento de imágenes y el robot físico. El proyecto se culminó con una validación final que garantiza la precisión, fiabilidad y eficiencia del sistema integrado.

CAPÍTULO 2

Antecedentes

El procesamiento digital de imágenes es un campo de continua evolución. El objetivo del procesamiento de imágenes puede tener a su vez distintos motivos. Algunos de estos podrían ser: mejorar la calidad de imagen, conseguir una representación más eficiente que permita almacenarla en menor espacio sin una pérdida apreciable de calidad, extraer información relevante de cara a interpretar su contenido y tomar decisiones al respecto, etc.

2.1. Investigaciones externas a la Universidad del Valle de Guatemala

En el trabajo de graduación de Aranguren y Vela [1], se presenta el diseño y desarrollo de un sistema de seguimiento de objetos mediante el procesamiento digital de imágenes aplicado al control de robots autónomos. El software permite al usuario seleccionar los clores de la marca que desea monitorear, además le permite cambiar los niveles de luminosidad con los que se presenta la imagen en pantalla. El sistema está conformado por un robot autónomo capaz de realizar movimientos rotacionales y traslacionales de manera de evitar que la marca se pierda en algún momento. El computador posee un software desarrollado en el entorno MATLAB potenciado con tecnología de procesamiento paralelo a través de una tarjeta de video con tecnología CUDA para optimizar el procesamiento de imágenes que permite realizar la identificación del objeto de interés. Se logró controlar el movimiento del robot de tal manera que se tenga un movimiento suave y agradable a la vista. Es importante destacar que la utilización de una tarjeta de video GPU acelera el procesamiento de imágenes, pero siempre y cuando este sea utilizado de la manera correcta, de otro modo el procesamiento puede resultar incluso más lento que antes. El diseño de un robot sencillo de dos grados de libertad y de poco peso simplifica el control del sistema y permite mayor precisión.

2.2. HUMANA

El centro de Epilepsia y Neurocirugía Funcional HUMANA [2] es una organización formada por profesionales en neurociencias que trabajan en beneficio de los pacientes que padecen problemas neurológicos de difícil control: epilepsia, parkinson, tumores cerebrales, columna vertebral, movimientos anormales, entre otros.

HUMANA está basada en un modelo de atención de categoría internacional con experiencia médica y uso de tecnología de vanguardia. En HUMANA se cuenta con unidades equipadas con cámara robótica y equipo de Neuroestimulación visual. Cuentan con un Quirófano diseñado y equipado para la cirugía cerebral y de columna, donde se cuenta con un neuronavegador para realizar procedimientos quirúrgicos con imágenes guiadas por computadora, por lo que la visión por computadora, procesamiento de imágenes y *machine learning* son conceptos que a HUMANA les genera interés y les son de utilidad.

2.3. Brainlab

Brainlab [3] es una empresa pionera de la tecnología médica digital, explora e identifica nuevas posibilidades para desarrollar los mejores tratamientos posibles para los pacientes. En general, su tecnología puede posibilitar que las intervenciones quirúrgicas sean más eficientes, personalizadas y seguras. La tecnología utilizada los impulsa a combinar imágenes médicas y tecnologías innovadoras para garantizar que se aprovechan al máximo en beneficio de los pacientes.

Brainlab desarrolla un software donde correlaciona imágenes conformando un modelo digital del paciente que se optimiza y depura constantemente con ayuda de inteligencia artificial y sofisticados algoritmos. Mediante la combinación del mundo físico y digital, es posible procesar y proporcionar información crítica de individualizada del paciente cuando se necesite.

Para lograr que el máximo número de pacientes se beneficien, Brainlab pone a disposición sus datos y tecnología básica a otras empresas. De este modo, se logran conocimientos de distintas áreas clínicas para propiciar sinergias. En HUMANA utilizan el software de Brainlab para mostrar a los doctores encargados de operaciones del cerebro la posición y el ángulo en que tienen que insertar ciertas agujas. Dicho posicionamiento se realiza de forma manual, con un brazo mecánico. El proceso es lento y es lo que se busca automatizar con el software.

2.4. Investigaciones en Universidad del Valle de Guatemala

En la Universidad del Valle de Guatemala se han realizado diversos estudios y proyectos donde involucran visión por computadora, procesamiento de imágenes y *machine learning*. Con el pasar de los años, se ha buscado automatizar los procesos para hacerlos más eficientes y sencillos de utilizar para usuarios potenciales. Un ejemplo de este estudio es el

trabajo realizado por Portales [4], donde se enfoca en la optimización de la herramienta de procesamiento de imágenes para el sistema Brainlab [3] en HUMANA [2].

La Universidad del Valle de Guatemala ha mantenido una colaboración de investigación con el Centro de Epilepsia y Neurocirugía Funcional HUMANA [2]. Durante este tiempo, se han llevado a cabo varios proyectos que buscan mejorar los procesos dentro del centro, abarcando desde la mejora en el control de los instrumentos hasta el procesamiento de los resultados de los diversos estudios realizados en ese lugar.

La tesis de Cifuentes [5] se enfoca en el diseño e implementación de un mando de control para una versión robotizada del sistema Varioguide de Brainlab. El proyecto se realizó en colaboración con el centro Neurológico HUMANA y la Universidad del Valle de Guatemala para mejorar la eficiencia en cirugías. Se dividió en tres partes. Primero, el análisis de servomotores Dynamixel AX-12A y MX106T, segundo, en el diseño de la estructura mecánica del primer prototipo y tercero, en el diseño del mando de control. El diseño estructural, donde se creó el primer prototipo de la estructura mecánica y se descubrieron limitaciones computacionales y físicas. En el diseño de mando de control se aplicó todo para controlar el brazo mecánico robotizado. El sistema Varioguide se ajusta por etapas, incluyendo ajustes tridimensionales, angulares y escalares. Los comandos de movimiento fueron satisfactorios, pero el problema que se presentó fue la estabilidad del sistema.

En el trabajo de graduación de Juárez [6] se optimizó y fabricó un sistema mecánico de cuatro grados de libertad para sustituir el sistema estereotáxico VarioGuide utilizado en HUMANA. Este proyecto rediseñó los grados de libertad superiores del brazo robótico, utilizando Autodesk Inventor para calcular masas, torques y transmisiones de potencia necesarias para cada articulación. Además, se implementaron sistemas mecánicos como tornillos sinfín-corona y husillos de bolas para mejorar la precisión y fluidez del movimiento. El prototipo funcional, fabricado con técnicas de impresión 3D utilizando ácido poliláctico (PLA), presentó características como auto bloqueo y mayor seguridad en la manipulación. Estos avances sentaron las bases para iteraciones posteriores, buscando optimizar el sistema robótico de asistencia quirúrgica. En la Figura 1 se puede ver una de las piezas fabricadas del sistema de desplazamiento transversal.

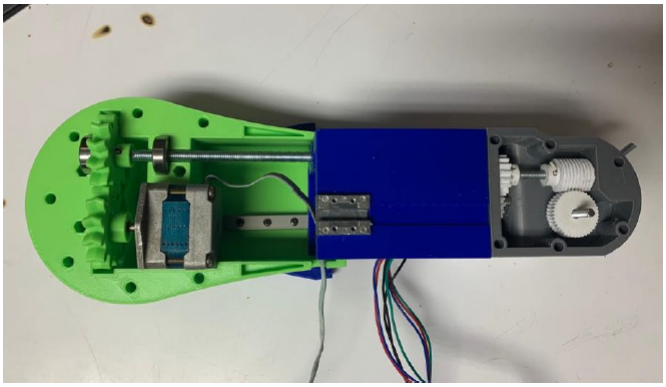


Figura 1: Sistema de desplazamiento transversal [6]

De manera complementaria, Vanegas [7] desarrolló un trabajo de graduación enfocado en el rediseño y validación de los últimos cuatro grados de libertad del brazo robótico

HUMANA, optimizando su integración con el sistema Varioguide-Brainlab. Como ejemplo, el diseño del cuarto grado de libertad se puede ver en la Figura 2. Este proyecto implementó mejoras en la transmisión de potencia y la estructura mecánica, abordando específicamente la precisión de los movimientos finales del brazo robótico. Las pruebas de precisión realizadas confirmaron la efectividad del diseño para aplicaciones quirúrgicas, fortaleciendo los esfuerzos previos en el desarrollo de soluciones robóticas para HUMANA.

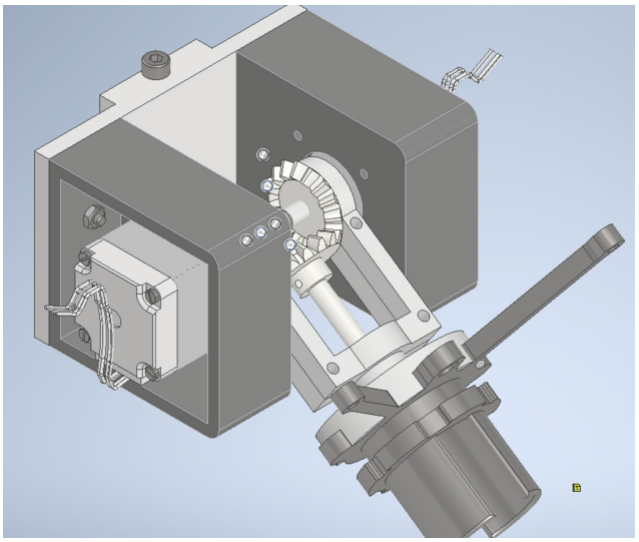


Figura 2: Diseño de uno de los grados de libertad [7]

En la tesis de Galicia [8] se creó un algoritmo de *machine learning* para identificar dígitos de calibración en la pantalla del laboratorio de HUMANA. Se utilizó un modelo de red neuronal convolucional que logró una precisión del 98.6 % en el modelo y del 91.6 % en las pruebas físicas realizadas. Además, se diseñó un mecanismo de anclaje para un sensor óptico en la pantalla del laboratorio, lo que mejoró la efectividad y redujo el tiempo del sistema de calibración VarioGuide. Este mecanismo puede apreciarse en la Figura 3. Se implementó un sistema de procesamiento de imágenes para reconocer variables en la pantalla de la computadora, junto con un algoritmo para calificar cámaras y determinar la más adecuada para el reconocimiento de características necesarias. Además, se diseñó un formato TCP para enviar los datos de calibración leídos del sistema de Varioguide a un microcontrolador ESP32, facilitando la transferencia de información.

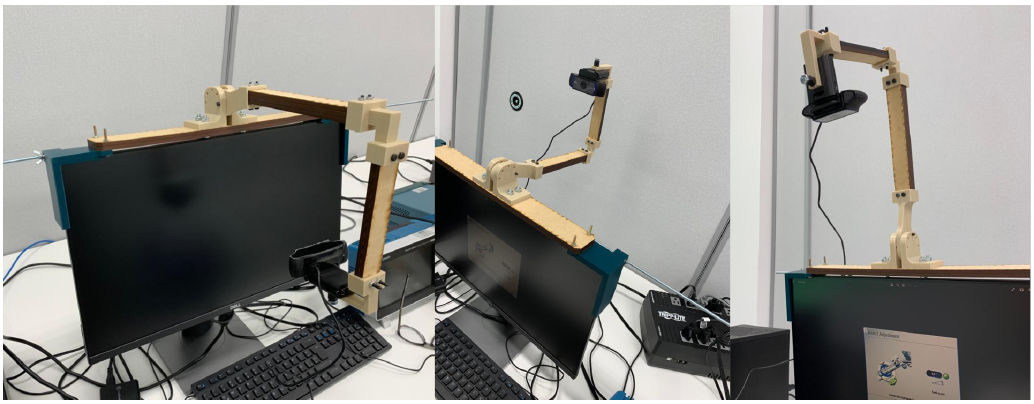


Figura 3: Ensamblaje de prototipo de acople [8]

En la tesis de Portales [4], se trabajó en la optimización de algoritmos para el reconocimiento óptico de caracteres, centrándose en los ángulos proporcionados por el sistema de Brainlab utilizado por HUMANA. Se desarrolló una interfaz para facilitar el procesamiento e identificación de estos ángulos, junto con un sistema de servidor-cliente que conecta la computadora principal con un sistema embebido para recortar y transmitir imágenes.

El proyecto empleó el motor Tesseract como herramienta principal y Asprise OCR como herramienta secundaria para el reconocimiento de caracteres. Se realizaron pruebas preliminares con diferentes tipos de textos para verificar su funcionamiento. Además, se evaluó una tercera herramienta llamada OCR Space, aunque sus resultados no fueron satisfactorios.

Se diseñaron cuatro interfaces: dos para ejecutar todo el procesamiento en la misma interfaz en la computadora principal, utilizando Tesseract y Asprise OCR, respectivamente, y otras dos para trabajar con el servicio de servidor-cliente, implementado en el sistema embebido, cada una utilizando un motor diferente. Los resultados en este caso fueron satisfactorios realizando pruebas con un Arduino que se pueden apreciar en la Figura 4, sin embargo, no se comprobó con un robot real.

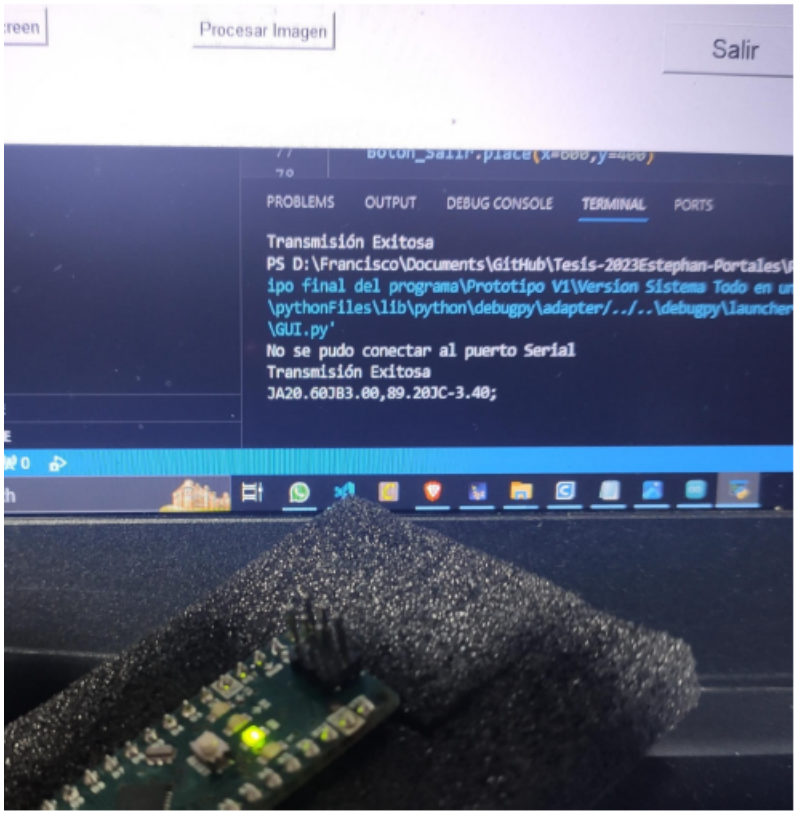


Figura 4: Recepción de datos exitosa con Arduino [4]

CAPÍTULO 3

Justificación

En los trabajos de graduación para el grado de licenciatura de Galicia [8] y Portales [4] se logró un avance significativo en la interfaz para el procesamiento de imágenes y en la herramienta para procesamiento óptico de caracteres con Tesseract. Sin embargo, los resultados obtenidos no fueron totalmente concluyentes. Se tuvieron algunas limitaciones en cuanto a la calidad de imágenes por cámara web y para la primera toma de capturas, en cuanto a la validación con el entorno físico, ya no se realizaron pruebas con un brazo robótico.

Existen métodos de optimización que fueron implementados de manera limitada o simplemente no fueron implementados en el sistema de procesamiento e interfaz de Portales [4], que pudieron haber hecho la interfaz más amigable y eficiente. Por lo anterior, en este trabajo se planteó la necesidad de desarrollar un sistema de captura de pantalla más eficiente que pueda cumplir con los requisitos de exactitud y confiabilidad validado con un brazo róbotico. Así mismo, se buscó la simplificación de la interfaz para que sea aun más amigable y eficiente para el público objetivo que estará utilizándola.

Validar el procesamiento de imágenes para ángulos con un sistema robótico físico fue crucial para asegurar la precisión y exactitud del sistema, así como su integración adecuada. Estas pruebas permitieron verificar la funcionalidad, medir el rendimiento, detectar y corregir errores, y optimizar tanto los algoritmos de procesamiento como los controles del sistema robótico. Se buscó garantizar que el sistema puede operar en condiciones del mundo real de manera confiable, confirmando su preparación para aplicaciones prácticas.

CAPÍTULO 4

Objetivos

4.1. Objetivo general

Optimizar la herramienta de procesamiento de imágenes y reconocimiento de caracteres para el sistema Brainlab de HUMANA, y utilizar los resultados del reconocimiento de caracteres para el posicionamiento de un sistema róbotico físico.

4.2. Objetivos específicos

- Revisar los algoritmos de procesamiento de imágenes desarrollados en fases anteriores con el fin de optimizar el reconocimiento de caracteres y mejorar su precisión.
- Validar la herramienta de procesamiento de imágenes y reconocimiento de caracteres desarrollada en la fase anterior con más imágenes, obtenidas con capturas de pantalla, del sistema Brainlab de HUMANA.
- Validar el protocolo de comunicación y envío de comandos propuesto en la fase anterior con un sistema robótico físico.
- Crear una interfaz de usuario fácil de usar para el manejo y configuración de la herramienta desarrollada.

CAPÍTULO 5

Alcance

El alcance de este proyecto se limitó a la optimización de la herramienta de procesamiento de imágenes, el reconocimiento de caracteres utilizados por el sistema Brainlab en HUMANA y su validación en el sistema robótico físico. Esto incluye la revisión y mejora de algoritmos de procesamiento de imágenes previamente desarrollados, la validación con un conjunto ampliado de imágenes capturadas del sistema, y la creación de un protocolo de comunicación que permita la interacción efectiva entre la herramienta y un sistema robótico físico.

Para la validación de los procedimientos y algoritmos desarrollados en este proyecto, se utilizó un robot provisional debido a que el diseño final del brazo robótico destinado para su integración completa con el sistema Brainlab de HUMANA aún se encuentra en desarrollo.

Además, el proyecto abarca el desarrollo y la integración de una interfaz gráfica de usuario (GUI) que facilite la configuración y el uso de la herramienta por parte de los usuarios. La validación final se realizó mediante pruebas con el robot myCobot 280 M5. No se abordaron desarrollos fuera del ámbito de la herramienta actual ni se consideraron la implementación con otros tipos de robots o sistemas diferentes al sistema Brainlab.

Cabe mencionar que el alcance de este proyecto se vio limitado por la falta de acceso a un número mayor de imágenes reales proporcionadas por el sistema Brainlab en HUMANA. Estas imágenes eran necesarias para llevar a cabo una validación más exhaustiva del reconocimiento de caracteres y procesamiento de imágenes. La disponibilidad limitada de estos recursos visuales restringió la capacidad para realizar pruebas adicionales y aumentar la precisión del sistema en escenarios reales. Esta limitación se tuvo en cuenta al diseñar las pruebas de validación, ajustando los métodos para trabajar con los recursos disponibles sin comprometer los objetivos principales del proyecto.

CAPÍTULO 6

Marco teórico

6.1. Imagen digital

Una imagen se define como una función de dos dimensiones *f(x,y)* donde x e y son las coordenadas de un plano que contiene todos los puntos de la misma, y *f(x,y)* es la amplitud en el punto *(x,y)* a la cual se le llama intensidad o nivel de gris de la imagen en ese punto. En el caso de que tanto las coordenadas x e y como los valores de intensidad de la función f sean discretos y finitos, se habla de una imagen digital.

Una imagen digital está compuesta de un número finito de elementos y cada uno tiene una localidad y un valor particular. A estos elementos se les llama puntos elementales de la imagen o píxeles, siendo este último el término comúnmente utilizado para denotar la unidad mínima de medida de una imagen digital [9].

6.2. Análisis y procesamiento digital de imágenes

El Análisis digital de imágenes es el área de la ingeniería que se encarga de la extracción de mediciones, datos o información contenida en una imagen. Un sistema de análisis de imágenes se distingue debido a que tiene como parámetro de entrada una imagen, y cuyo resultado es comúnmente una salida numérica, en lugar de otra imagen. Esta salida es la información referente al contenido de la imagen de entrada.

Para llegar desde la imagen original al conjunto de parámetros e información extraída de la misma, es necesario pasar por distintas etapas de procesamiento y filtrado donde se analiza la imagen y se adecúa para cierta aplicación específica.

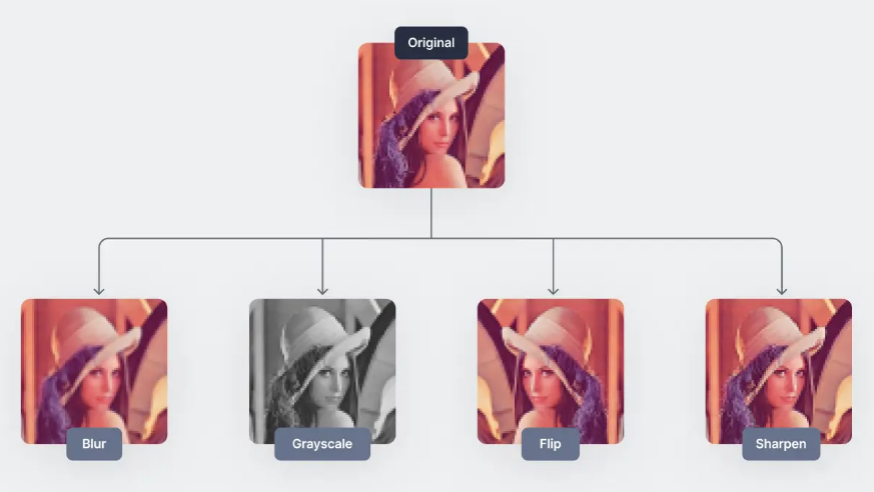


Figura 5: Ejemplos de procesamientos de imágenes típicos [10]

Estas herramientas se organizan según el nivel de procesamiento que se desea realizar para analizar la información contenida en una imagen digital.

- **Preprocesamiento.** Operaciones para adaptar la información de una imagen y tener mejor análisis en pasos posteriores. Ejemplos de procesamiento son las operaciones de brillo y contraste.
- **Segmentación.** Operaciones para hacer una partición de la imagen en varias regiones que representen la información necesaria para el problema a resolver.
- **Detección de objetos y clasificación.** Determinación y clasificación de objetos contenidos en la imagen.
- **Análisis de imagen.** Obtener información de alto nivel acerca de lo que la imagen muestra.

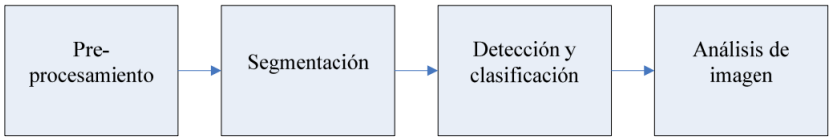


Figura 6: Niveles de procesamiento [11]

El conjunto de métodos de procesamiento de imágenes está dividido en tres grandes grupos:

- **Algoritmos en el dominio espacial:** se refiere a métodos que procesan una imagen píxel por píxel, o también tomando en cuenta un conjunto de píxeles vecinos.
- **Algoritmos en el dominio de la frecuencia:** frecuentemente, estos métodos son aplicados sobre los coeficientes resultantes de la transformada de Fourier de una imagen.

- **Algoritmos de extracción de características:** a diferencia de los dos grupos anteriores, los algoritmos de extracción de características están enfocados al análisis de imágenes para la extracción de atributos y regiones de interés, separación de objetos del fondo, detección de bordes o formas, entre otros.

Para el procesamiento de imágenes, también existen ciertas técnicas que pueden representar un beneficio al aplicarlas, siempre y cuando se deseé cierta información o no. Algunas de ellas son:

- **Escala de grises vs. imágenes a color:** las imágenes en escala de grises son útiles para reducir la carga computacional y enfocarse en estructuras, formas y datos que contenga la imagen. En caso contrario, las imágenes a color son beneficiosas cuando el color nos aporta algún tipo de información o es relevante para el análisis [12].

  Algunos beneficios para las imágenes en escala de grises son:

  - **Reducción de complejidad computacional:** las imágenes en escala de grises contienen un solo canal de intensidad, en lugar de los tres canales (RGB) que se encuentran en las imágenes a color. Esto reduce considerablemente la cantidad de datos que necesita ser procesada, lo que disminuye la carga computacional y el tiempo de procesamiento.
  - **Memoria y almacenamiento:** el tamaño de las imágenes en escala de grises es mucho menor en comparación con las imágenes a color. Esto se traduce en un uso más eficiente de la memoria y almacenamiento, lo que es crucial en aplicaciones donde el rendimiento es un factor importante.
  - **Foco en características estructurales:** en muchos casos de visión por computadora, como detección de bordes, segmentación de objetos o reconocimiento de patrones, la información clave proviene de la estructura y las formas en la imagen, que son fácilmente capturadas en escala de grises. El color puede no agregar información relevante para estas tareas.
  - **Reducción de ruido en el modelo:** en algunas aplicaciones, los colores pueden introducir ruido innecesario que distrae de los patrones fundamentales en la imagen. Trabajar con escala de grises puede ayudar a eliminar esa distracción y concentrarse en las características más importantes.

  Algunas aplicaciones comunes para imágenes en escala de grises son:

  - Detección de bordes.
  - Reconocimiento óptico de caracteres (OCR).
  - Reconocimiento de objetos basado en formas.
  - Procesamiento rápido en sistemas de recursos limitados.

  En el caso del procesamiento con imágenes a color, los beneficios que se tienen son los siguientes:

- **Más información:** las imágenes a color contienen más información que puede ser importante para tareas donde el color tenga un papel importante, como la clasificación de objetos que dependen de variaciones en los colores, como lo puede ser un clasificador de frutas.
- **Mejora la diferenciación de objetos:** en escenarios donde objetos diferentes tienen colores similares en escala de grises, las imágenes a color pueden ayudar a diferenciar mejor estos objetos, ya que los colores proporcionan un nivel adicional de separación que no está disponible en escala de grises.
- **Mejores resultados en tareas especificas:** algunas tareas de visión por computadora, como la segmentación semántica o el reconocimiento de escenas naturales, dependen de la riqueza de la información proporcionada por los colores. Los modelos que incluyen información de color a menudo tienen mejor rendimiento en estos casos.
- **Características adicionales:** las imágenes a color permiten aprovechar métricas y características derivadas de los espacios de color, como el espacio HSL o HSV, que pueden ser útiles para tareas de segmentación o detección de objetos basadas en color [12].

Algunas aplicaciones comunes para imágenes a color son:

- Clasificación y reconocimiento de objetos donde el color es importante.
- Segmentación semántica basada en características cromáticas.
- Aplicaciones de monitoreo ambiental o agricultura de precisión.
- Reconocimiento facial o detección de piel, donde el color juega un rol crítico.

- **Umbralización:** la umbralización es una técnica sencilla y efectiva para la segmentación de imágenes, que consiste en convertir una imagen en escala de grises en una imagen binaria. Este proceso involucra definir un valor de umbral para distinguir entre los píxeles que forman los objetos de interés y el fondo.

- **Contraste:** el contraste en una imagen se refiere a la diferencia de brillo entre los elementos de la imagen. En procesamiento de imágenes, mejorar el contraste es fundamental para resaltar características importantes, como bordes y detalles que son vitales para el reconocimiento de caracteres.

  Esto consiste en modificar la gama de valores de intensidad en la imagen para aumentar la visibilidad de los detalles. Esto se puede hacer utilizando diferentes técnicas, como el estiramiento de histograma, que redistribuye los valores de intensidad para abarcar todo el rango posible (de 0 a 255 en imágenes de 8 bits) [13].

- **Reducción de ruido:** las imágenes, especialmente aquellas obtenidas de documentos escaneados o fotografías, a menudo contienen ruido. El ruido puede tomar la forma de puntos, líneas no deseadas, variaciones en la intensidad, etc., que pueden interferir en el reconocimiento de caracteres.

  - **Filtros de suavizado:** los filtros como el filtro de media o mediana son ampliamente utilizados para reducir el ruido en una imagen. El filtro de media reemplaza

el valor de cada píxel con el promedio de los valores circundantes, suavizando variaciones bruscas en la intensidad. El filtro de mediana, en cambio, reemplaza el valor del píxel con la mediana de los valores circundantes, lo que es efectivo para eliminar el ruido impulsivo.

- **Filtros gaussianos:** los filtros gaussianos utilizan una función gaussiana para suavizar la imagen, reduciendo el ruido sin eliminar las características importantes. Este tipo de filtro es especialmente útil cuando se necesita preservar bordes mientras se elimina el ruido [14].

6.3. Visión por computadora

Visión es la ventana al mundo de muchos organismos. Su función principal es reconocer y localizar objetos en el ambiente mediante el procesamiento de imágenes. La visión por computadora es el estudio de estos procesos, para entenderlos y construir máquinas con capacidades similares [15].

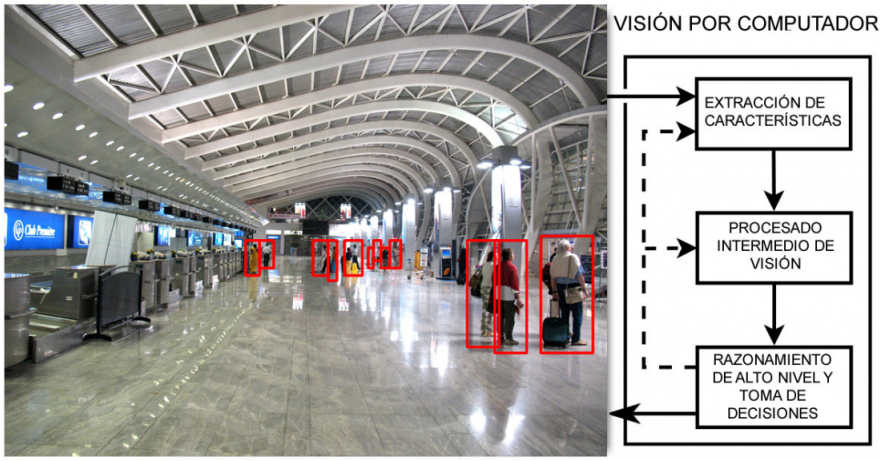


Figura 7: Visión por computadora y cadena de procesamiento típica [9]

El objetivo de la visión por computadora es extraer características de una imagen para su descripción e interpretación por la computadora. Por ejemplo:

- Determinar la localización y tipo de objetos en la imagen.
- Construir una representación tridimensional de un objeto.
- Analizar un objeto para determinar su calidad.
- Descomponer una imagen u objeto en diferentes partes.

En visión se busca obtener descripciones útiles para cada tarea a realizar. La tarea demandará modificar ciertos atributos.

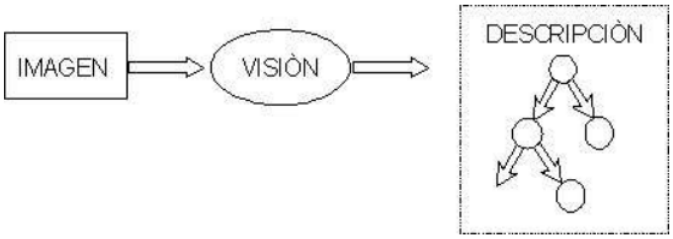


Figura 8: Esquema general de visión por computadora [15]

Nótese que en la Figura 8 la imagen de entrada es procesada para extraer los atributos, obteniendo como salida una descripción de la imagen analizada.

6.4. Python

Python es un lenguaje de programación de alto nivel, interpretado y multipropósito. Es ampliamente utilizado en las aplicaciones web, el desarrollo de software, la ciencia de datos y el *machine learning*.

En cuanto a la realización de tareas y *machine learning* en Python, consiste en extraer conocimientos valiosos a partir de los datos, mientras que el enseña a las computadoras a aprender automáticamente de los datos y a efectuar predicciones precisas [16].

6.5. Open CV

Open CV es una librería de computación visual para el procesamiento de imágenes en Pyhton. Esta biblioteca proporciona herramientas para realizar operaciones de procesamiento de imágenes, como el filtrado, la detección de bordes, el reconocimiento de características, el seguimiento de objetos, etc. Estas herramientas permiten desarrollar aplicaciones de visión artificial, como el reconocimiento facial, el seguimiento de objetos, etc.

6.6. Reconocimiento óptico de caracteres (OCR)

La tecnología de reconocimiento de caracteres, OCR (Optical Character Recognition), engloba a un conjunto de técnicas basadas en estadísticas, en las formas de los caracteres, transformadas y en comparaciones, que complementándose entre sí, se emplean para distinguir de forma automática entre los diferentes caracteres alfanuméricos existentes [17].

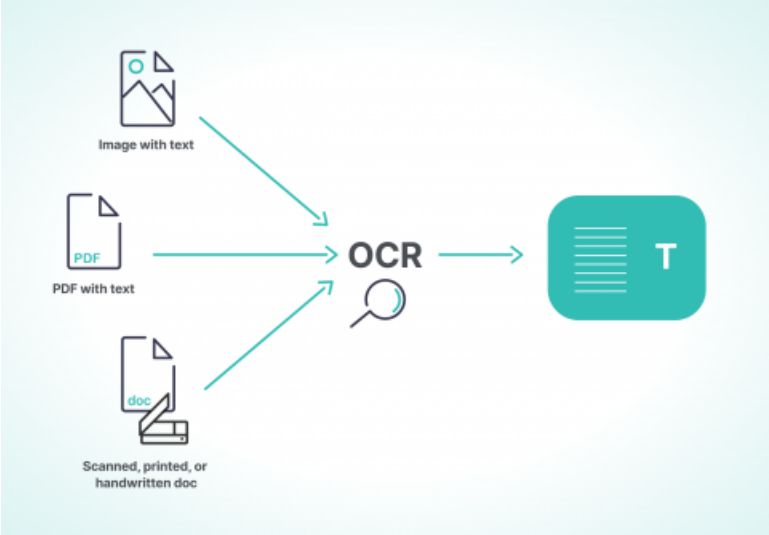


Figura 9: Diagrama de reconocimiento óptico de caracteres [18]

En todo sistema de reconocimiento óptico de caracteres (OCR) se distinguen al menos estas 4 etapas:

- Adecuación de la imagen (preproceso).
- Selección de la zona de interés (segmentación).
- Representación digital de la imagen (extracción de características).
- Distinción del carácter contenido en la imagen (reconocimiento).

Para cada una de las cuatro etapas es posible aplicar multitud de técnicas ya existentes o desarrollar alguna específica en función de las condiciones en las que se representan los datos de entrada, que en el caso de OCR se puede traducir por las imágenes de entrada [19].

6.7. Tesseract

Tesseract es un motor OCR de código abierto que extrae texto impreso o escrito de las imágenes. Fue desarrollado originalmente por Hewlett-Packard, y su desarrollo fue adquirido después por Google. Con Tesseract OCR, es posible extraer texto de las imágenes con un reconocimiento eficaz de patrones y caracteres y en línea del motor de OCR.

Es capaz de reconocer más de 100 idiomas y cuenta con una integración de IA a través de la Red Neuronal LSTM para detectar y reconocer mejor el contenido del texto de diferentes tamaños.

Uno de los aspectos principales de Tesseract es que es compatible con muchos lenguajes de programación y estructuras que utilizan wrappers como Pytesseract, también conocido como Python-Tesseract [20].

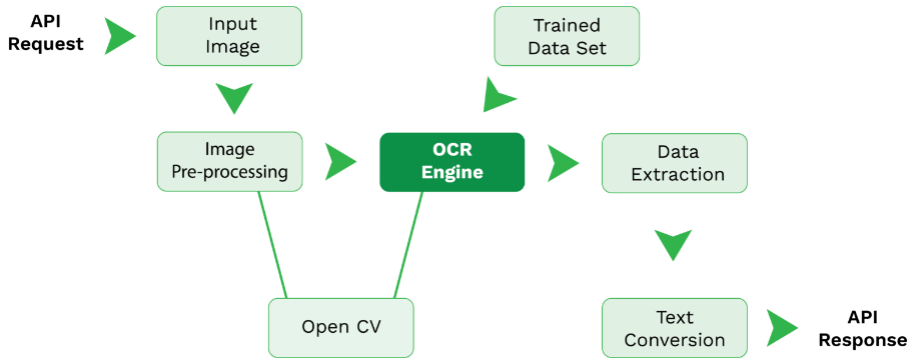


Figura 10: Fases del proceso Tesseract OCR [20]

6.8. Universal Asynchronous Receiver-Trasmitter (UART)

UART significa receptor/transmisor asíncrono universal y definir un protocolo, o conjunto de reglas para intercambiar datos en serie entre dos dispositivos. El UART solo utiliza dos cables entre el transmisor y receptor para transmitir y recibir en ambas direcciones. La comunicación puede ser simplex (los datos se envían en una sola dirección), semidúplex (cada lado transmite, pero solo uno a la vez), o dúplex completo (ambos lados pueden transmitir en simultáneo). Los datos en el UART se transmiten en la forma de tramas [21].

El protocolo UART se utiliza ampliamente en una variedad de aplicaciones, desde la comunicación serie básica entre dispositivos embebidos hasta la interfaz con periféricos en sistemas informáticos y de comunicaciones. Su simplicidad, flexibilidad y fiabilidad lo convierten en una opción popular para le interconexión de dispositivos electrónicos en numerosos contextos industriales y de consumo.

6.9. Sistemas embebidos

Un sistema embebido posee hardware de computador junto con software embebido como uno de sus componentes más importantes. Es un sistema computacional dedicado para aplicaciones o productos. Puede ser un sistema independiente o parte de un sistema mayor, y dado que usualmente su software está embebido en ROM, no necesita memoria secundaria como un computador. Un sistema embebido tiene tres componentes principales:

- Hardware.
- Un software primario o aplicación principal. Este software o aplicación lleva a cabo una tarea en particular, o en algunas ocasiones una serie de tareas.
- Un software operativo que permite supervisar las aplicaciones, además de proveer los mecanismos para la ejecución de procesos. En muchos sistemas embebidos es requerido que el sistema operativo posea características de tiempo real.

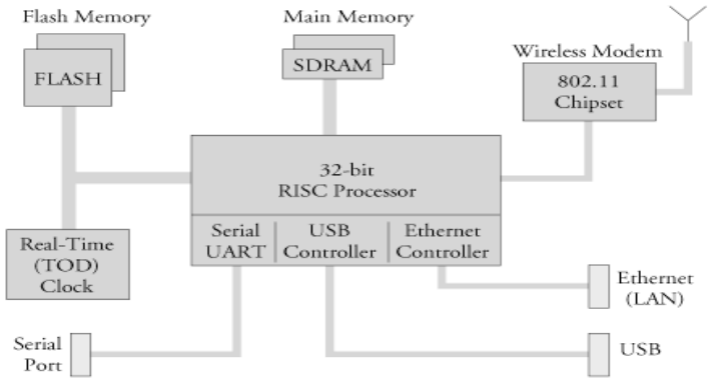


Figura 11: Ejemplo de un sistema embebido [22]

6.10. Sistemas robóticos

La robótica es una de las expresiones de la tecnología cuya aplicación se ha extendido a diversos contextos de la vida del hombre. La robótica, como tecnología que es, constituye el saber y hacer sobre los robots, esto implica el uso del conocimiento de diversas áreas para el diseño, construcción, ensamble y puesta en funcionamiento de un robot con un fin especifico.

6.10.1. Robótica en medicina

En medicina, algunas de las aplicaciones se encuentran en robots que son teleoperados por médicos especializados, ubicados en cualquier parte del mundo y quienes realizan la intervención quirúrgica con precisión microscópica. Entre los beneficios obtenidos cabe mencionar: disminución de costos por desplazamiento médico y la precisión en las acciones al eliminar ruidos como el temblor de las manos; en los desarrollos macrobióticos se resalta la construcción de prótesis que reemplazan eficientemente partes del cuerpo humano [23].

6.10.2. myCobot

myCobot es el robot colaborativo de seis ejes más pequeño y ligero del mundo. Se utiliza para aumentar la productividad y para el desarrollo secundario según las demandas de los usuarios.

myCobot es compacto pero potente; puede combinarse con una variedad de efectores finales para adaptarse a diferentes tipos de escenarios de aplicación. Además, admite el desarrollo secundario de software multiplataforma. Basado en diferentes tipos de aplicaciones, ofrece interfaces de código abierto que permiten el reconocimiento de objetos, reconocimiento facial, reconocimiento de imágenes, entre otros.

Puede programarse en varios lenguajes de programación, como Python, C++, Arduino, C##, JS, etc. También ofrece diferentes métodos de conexión, como USB, Wi-Fi y Bluetooth

[24].

Figura 12: Robot myCobot [24].

6.10.3. OWI-007

El OWI-007, también conocido como *Robotic Arm Edge*, es un *kit* de construcción de brazo robótico diseñado para propósitos educativos y de entretenimiento. El brazo robótico puede ser controlado manualmente a través de un panel de control incluido, lo que permite a los usuarios experimentar con sus capacidades y aprender sobre principios de mecánica, electrónica y programación de robots.

El OWI-007 es utilizado en entornos educativos como una herramienta para enseñar conceptos de robótica y tecnología a estudiantes de todas las edades. Además, se emplea en contextos de entretenimiento, donde los aficionados a la robótica pueden disfrutar construyendo y experimentando con este dispositivo [25].

Figura 13: Robot OWI-007 [25]

CAPÍTULO 7

Selección de sistema robótico físico

En este proyecto se buscaba validar la herramienta de procesamiento de imágenes con un sistema robótico físico. El OWI-007 y el myCobot 280 M5 fueron los principales candidatos para esto. Ambos, diseñados con propósitos educativos y de desarrollo, presentan diferencias significativas en términos de capacidades, tecnología y aplicaciones.

Uno de los aspectos clave en la selección del robot fue la posibilidad de emular los movimientos del brazo mecánico de HUMANA. Esto es esencial para validar los algoritmos de procesamiento de imágenes en condiciones similares a las que se encuentran en el entorno real de aplicación. El robot seleccionado debía permitir movimientos precisos y repetibles para replicar las acciones del brazo de HUMANA, además de contar con una compatibilidad adecuada para sensores. Estas capacidades son críticas para asegurar que el sistema pueda adaptarse en el futuro al brazo robótico final desarrollado específicamente para este proyecto.

Para tomar la decisión de qué robot utilizar, se tomaron en cuenta ciertas consideraciones importantes, tales como la disponibilidad de los robots en el departamento, la compatibilidad con las herramientas de procesamiento de imágenes, la facilidad de integración con el sistema Brainlab y las capacidades técnicas de cada modelo. En las siguientes secciones se presentan los robots considerados, detallando sus características y evaluando su idoneidad para los objetivos de este proyecto.

7.1. OWI-007

El OWI-007, aunque es un excelente recurso educativo para introducirse en robótica, resultó no ser adecuado para la validación de la herramienta de procesamiento de imágenes.

- **Falta de sensores integrados:** el OWI-007 no incluye sensores integrados de fábrica.

La validación de herramientas de procesamiento de imágenes generalmente requiere datos de sensores como cámaras, sensores de profundidad o LIDAR, que no se encuentran disponibles en este modelo sin modificaciones significativas.

- **Control básico y manual:** el brazo se controla mediante un control remoto con interruptores, lo que limita la capacidad de automatización y precisión. La herramienta de procesamiento de imágenes realizada requiere de una interfaz sofisticada y precisa para interactuar con el hardware.
- **Tecnología antigua:** el OWI-007 fue diseñado hace muchos años y la tecnología utilizada es bastante básica en comparación con los estándares actuales. Esto incluye motores de corriente continua simples sin retroalimentación, lo que puede limitar la precisión y repetibilidad de las acciones necesarias para pruebas robustas de procesamiento de imágenes.
- **Compatibilidad y soporte:** actualmente, el modelo de OWI-007 con el que se contaba ha sido descontinuado y reemplazado por versiones más nuevas; el soporte y la compatibilidad con software y hardware modernos resultaron ser limitados. Las actualizaciones y el soporte técnico pueden no estar disponibles, lo que complica la integración con tecnologías actuales.

7.2. myCobot 280 M5

El myCobot M5 resultó ser una opción más adecuada para la validación de la herramienta de procesamiento de imágenes.

- **Compatibilidad con sensores y cámaras:** el myCobot M5 está diseñado para ser compatible con diversos sensores y cámaras, lo que es crucial para aplicaciones de procesamiento de imágenes y visión por computadora. La capacidad de integrar cámaras y otros sensores permite que el robot recopile datos visuales en tiempo real, necesarios para validar y probar algoritmos de procesamiento de imágenes.
- **Precisión y repetibilidad:** el myCobot M5 ofrece alta precisión y repetibilidad en sus movimientos, lo cual es esencial para validar herramientas de procesamiento de imágenes. Las pruebas de procesamiento de imágenes a menudo requieren movimientos exactos y repetidos para asegurar que los algoritmos funcionen correctamente en diferentes condiciones. De hecho, uno de los puntos principales del proyecto es buscar que el robot sea lo más preciso posible para ubicarse en la posición exacta solicitada.
- **Programabilidad: y flexibilidad:** el myCobot M5 es altamente programable y se puede controlar mediante diversas plataformas de software como ROS (Robot Operating System), Python, Arduino y otros lenguajes de programación. Esta flexibilidad permite que los desarrolladores puedan implementar y probar algoritmos de procesamiento de imágenes de manera eficiente y adaptada a las necesidades específicas.
- **Múltiples grados de libertad:** el myCobot cuenta con seis grados de libertad y es capaz de realizar movimientos complejos y variados, simulando escenarios del mundo

real donde se implementan herramientas de procesamiento de imágenes. Esta capacidad es fundamental para probar algoritmos en condiciones que se asemejen a las aplicaciones prácticas que buscamos.

- **Integración con inteligencia artificial:** actualmente, la inteligencia artificial se está dando más a conocer y está siendo mucho más implementada en muchas aplicaciones del mundo cotidiano. El myCobot M5 es compatible con entornos de trabajo de inteligencia artificial que se pueden utilizar junto con herramientas de procesamiento de imágenes. Esto permite la implementación de algoritmos avanzados de aprendizaje automático y visión por computadora, facilitando pruebas más complejas y detalladas.
- **Facilidad de uso y documentación:** el myCobot M5 viene con una amplia documentación y soporte comunitario, lo que facilita la configuración e integración de sistemas de procesamiento de imágenes. La disponibilidad de tutoriales y ejemplos prácticos ayuda a los desarrolladores a iniciar rápidamente los proyectos.
- **Tamaño compacto y portabilidad:** el diseño compacto y ligero del myCobot M5 permite su uso en entornos de laboratorio y en campo, proporcionando flexibilidad en el lugar de prueba y validación de herramientas de procesamiento de imágenes. Esta portabilidad es útil para aplicaciones móviles y para probar sistemas en diferentes ubicaciones.
- **Costo-efectividad:** en comparación con otros robots industriales, el myCobot M5 ofrece una solución costo-efectiva para la validación de herramientas de procesamiento de imágenes. Su precio accesible permite que más laboratorios e instituciones puedan utilizarlo sin comprometer el presupuesto.

7.3. Comparación clave

Comparar el OWI-007 con el myCobot 280 M5 revela diferencias significativas en tecnología, funcionalidad y capacidades que hacen al myCobot 280 M5 una opción mucho más adecuada para aplicaciones modernas, incluyendo la validación de herramientas de procesamiento de imágenes.

- **Precisión y control:** el myCobot 280 M5 supera ampliamente al OWI-007 en términos de precisión y control, gracias a sus motores paso a paso y sensores avanzados. Durante el proceso de este proyecto y de pruebas de envío de datos se observó como el movimiento de las juntas en el myCobot 280 M5 es muy limpio, aunque dentro del mundo de la robótica se puede considerar un robot sencillo, sus movimientos son muy precisos.
- **Capacidad de carga y movilidad:** el myCobot 280 M5 tiene una mayor capacidad de carga y más grados de libertad, lo que le permite realizar tareas más complejas y diversas.
- **Compatibilidad con tecnologías modernas:** el myCobot 280 M5 está diseñado para ser compatible con entornos de programación modernos y puede integrarse con

tecnologías de visión por computadora, lo cual es crucial para validar herramientas de procesamiento de imágenes.

- **Aplicaciones avanzadas:** mientras que el OWI-007 es adecuado para educación básica, el myCobot 280 M5 es apto para investigación avanzada, desarrollo de prototipos y aplicaciones industriales ligeras.

Después de evaluar distintas opciones, se concluyó que el myCobot 280 M5 es una herramienta más poderosa y versátil para aplicaciones modernas, especialmente aquellas que requieren la validación de herramientas de procesamiento de imágenes y otras tecnologías avanzadas. Para asegurar la funcionalidad de los controles y algoritmos desarrollados en este proyecto, se llevaron a cabo pruebas de validación utilizando este robot provisional. Dado que el brazo robótico final aún no estaba disponible, el myCobot 280 M5 fue seleccionado para realizar pruebas preliminares en un entorno controlado, lo cual permitió verificar la funcionalidad de los algoritmos de control y procesamiento de imágenes, proporcionando así una validación efectiva de las herramientas implementadas.

Es importante señalar que el myCobot 280 M5 se está utilizando como una solución provisional mientras se completa el desarrollo del brazo robótico final. Este robot fue elegido debido a su capacidad de emular, en cierta medida, los movimientos necesarios para la validación de los algoritmos y herramientas de procesamiento de imágenes desarrollados en este proyecto. Aunque no es un reemplazo exacto del brazo de HUMANA, el myCobot 280 M5 permite realizar pruebas preliminares y evaluar la funcionalidad en un entorno controlado. Una vez que el brazo robótico final esté disponible, se prevé que los algoritmos y configuraciones validados con el myCobot 280 M5 puedan adaptarse con mínimas modificaciones al nuevo sistema, facilitando así una transición fluida.

Es importante mencionar que una de las áreas donde se invirtió gran parte de los esfuerzos fue con la integración del sistema con dicho sistema robótico. Se utilizó la librería pymycobot para conectar el sistema al myCobot 280 M5, lo que permite que las decisiones basadas en el reconocimiento de caracteres se traduzcan en acciones físicas. Adicionalmente, se implementaron herramientas avanzadas como roboticstoolbox y spatialmath para realizar simulaciones y cálculos especiales. Esto permitió validar el comportamiento del sistema en entornos simulados antes de realizar pruebas físicas, asegurando un diseño más robusto.

7.4. Interpretación del myCobot 280 M5

En el contexto del robot myCobot 280 M5, las juntas representan los puntos de conexión entre los diferentes segmentos del brazo robótico. Estas juntas permiten movimientos relativos entre las partes, controlados por motores, y son esenciales para realizar movimientos precisos y repetibles. En este proyecto, se analizaron tres juntas principales: junta 1, junta 2 y junta 3, cada una con una función específica dentro del sistema.

- **Junta 1:** es la base del brazo robótico y permite el movimiento de rotación. Este movimiento es crucial para posicionar el brazo en una dirección general, sirviendo como el primer grado de libertad para alcanzar el objetivo deseado.

- **Junta 2:** conecta la base del brazo con el siguiente segmento y proporciona movimiento en el eje vertical. Este movimiento permite al brazo alcanzar diferentes alturas, ajustando su posición según la tarea a realizar.
- **Junta 3:** ubicada en la parte superior del brazo robótico myCobot 280 M5, es responsable de realizar movimientos precisos de rotación y ajuste angular. Esta articulación juega un papel clave en la orientación final del extremo del brazo, especialmente para posicionar herramientas o sensores en la dirección correcta.

En la Figura 14 se presenta un esquema del robot myCobot 280 M5, donde se señalan las juntas 1, 2 y 3. Este esquema sirve como referencia visual para entender cómo cada articulación contribuye a los movimientos del robot y cómo los datos de estas juntas se utilizan en el contexto de este proyecto.

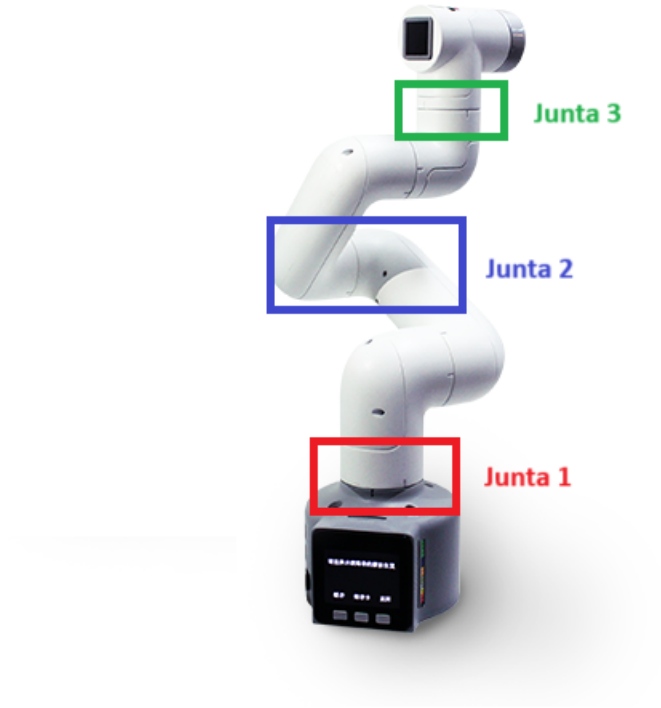


Figura 14: Indicación de interpretación de juntas en el myCobot 280 M5

CAPÍTULO 8

Visión por computadora

La visión por computadora fue la herramienta principal para este proyecto, utilizando distintas funcionalidades de la librería OpenCV. Algunas herramientas que nos ofrece esta librería es cargar, leer, aplicar filtros y almacenar imágenes, tener estas opciones hizo que el procesamiento de las mismas fuera más sencillo en términos de trabajo computacional. Este proyecto fue trabajado en una *laptop* ASUS ROG Strix G GL531GT-UB74. Para ver la lógica implementada en código de esta y otras partes, visitar en el apartado de Anexos 15, el repositorio en GitHub.

8.1. Validación de captura de pantalla

La validación de capturas de pantalla es un paso crucial en el flujo de trabajo del proyecto, especialmente porque se trata de extraer datos precisos desde imágenes para su posterior procesamiento o control de sistemas robóticos. En estas secciones se describen los métodos y criterios utilizados para asegurar que las capturas de pantalla procesadas contengan la información correcta y que dicha información sea interpretada con precisión por el sistema.

Durante de las primeras fases de este proyecto, se trabajó con imágenes que eran capturadas por una cámara web ubicada frente a la pantalla donde se proyectaba la información del posicionamiento de las juntas. El problema de realizar esto era que el posicionamiento de la cámara, luz en el ambiente, reflejo, etc., complicaban la correcta obtención de la imagen y por ende complicaban el reconocimiento de caracteres. Para solucionar esto, desde la fase III se propuso experimentar y validar obtener imágenes mediante capturas del pantalla. En la Figura 15 se puede ver la forma de las capturas obtenidas.

Figura 15: Muestra de captura de pantalla del sistema Brainlab

Uno de los avances más significativos que se implementó en el código nuevo fue la incorporación de nuevas librerías. Mientras que el código anterior utilizaba únicamente herramientas básicas como cv2 y numpy para el procesamiento de imágenes, además de pytesseract que se redacta en el capítulo 8.3, para el reconocimiento óptico de caracteres, se decidió ampliar este conjunto para incluir ImageOps de la librería PIL. Esto permitió realizar preprocesamiento avanzado de imágenes, como la inversión de colores y ajustes de contraste, con el fin de optimizar la calidad de las imágenes antes de su análisis. Estos ajustes resultaron fundamentales para mejorar la precisión del motor OCR, especialmente en casos donde las imágenes tenían bajo contraste.

Teniendo en cuenta esto, era necesario contar con una interfaz para realizar todo el procedimiento de análisis. Se tomó como base la interfaz desarrollada en la fase anterior, esta permite al usuario obtener las imágenes que contienen los datos de configuración de manera rápida y eficaz.

El sistema Brainlab consta de tres juntas con distintos ángulos de configuración, por ende existen tres tipos de capturas de pantalla. La junta 1 consta solamente de un ángulo de configuración, la junta 2 consta de un ángulo de configuración y uno de desplazamiento lineal y la junta 3 consta solamente de un ángulo de configuración. Debido a esto se tuvieron tres formatos distintos de capturas de pantalla, se presentó con el problema de que las capturas de pantalla iniciales de las juntas 2 y 3 cambiaban de formato con respecto a las de la junta 1. El formato de la junta 2 se puede apreciar en la Figura 16 y el formato de la junta 3 en la Figura 17.

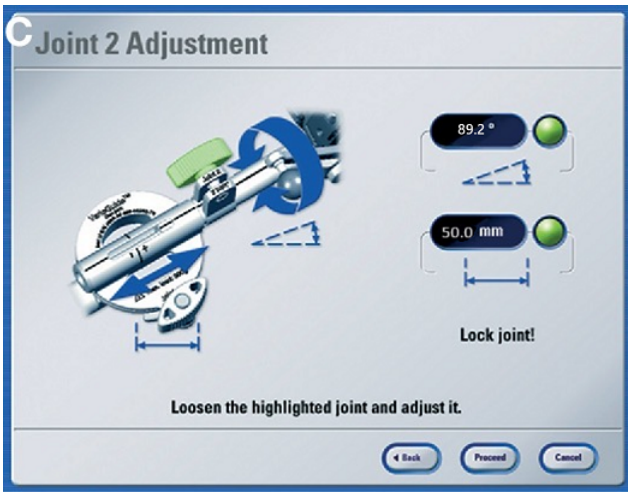


Figura 16: Formato de captura de junta 2

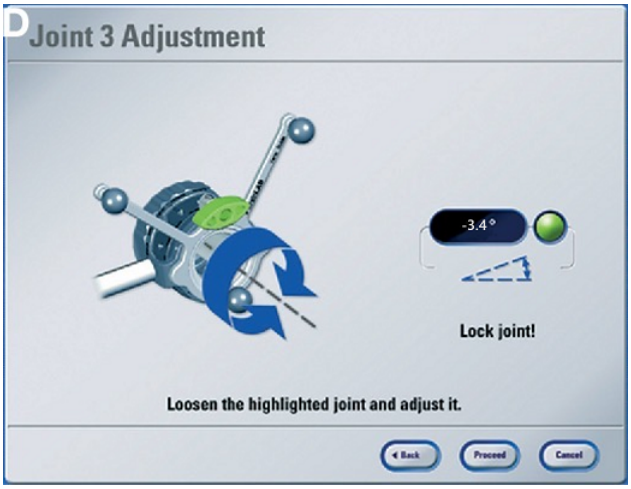


Figura 17: Formato de captura de junta 3

Dado que las capturas de pantalla se utilizan como la fuente principal de datos para la extracción de valores de las articulaciones del robot, cualquier error en esta fase podría llevar a una interpretación incorrecta de los datos y, consecuentemente, a un comportamiento no deseado del robot o simplemente no se haría el procesado de la imagen por lo que no se obtendrían datos. Por lo tanto, validar las capturas es esencial para garantizar la integridad y precisión del proceso.

8.1.1. Verificación de la región capturada

Se captura una región específica de la pantalla donde se muestran los ángulos de las articulaciones del robot. La captura de pantalla completa se puede apreciar en la Figura 18, la región especifica se puede apreciar en la Figura 19. Para validar que la captura corresponde a la zona correcta, se establecen coordenadas fijas que delimitan el área de interés. Este enfoque

garantiza que siempre se procese la misma parte de la pantalla, reduciendo la probabilidad de errores debidos a capturas incorrectas.

Figura 18: Formato de captura de pantalla completa

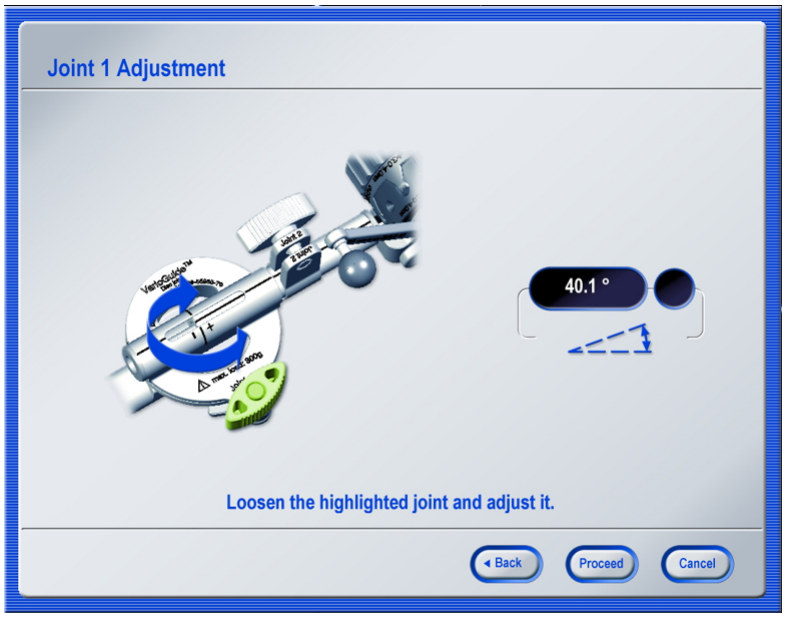


Figura 19: Región especifica capturada

Así mismo, esto ofrece una facilidad de ajuste. Estas coordenadas pueden modificarse fácilmente si cambia la resolución de la pantalla o la disposición de la interfaz gráfica, permitiendo una adaptación rápida sin alterar el resto del flujo de trabajo.

8.1.2. Limitaciones y desafíos

Aunque los métodos implementados funcionan bien en condiciones controladas, existen algunas limitaciones:

- **Dependencia de la configuración de la pantalla:** cambios en la resolución o en la disposición de la interfaz gráfica podrían requerir ajustes en las coordenadas de captura y en los parámetros de preprocesamiento. Aunque el ajuste de esto es fácil, presenta el problema que debe modificarse media vez se ejecute en un monitor distinto. Lo cual puede llegar a ser molesto en cierto aspecto.

Figura 20: Captura de junta a configurar capturada correctamente

Con relación a la dependencia de la configuración de la pantalla, podemos observar en la Figura 20 la captura correcta que se realiza para identificar la junta a configurar, esto depende de la configuración de la pantalla ya que al funcionar con la relación de píxeles de la pantalla e utilizar estos como coordenadas, al momento de cambiar de pantalla, la configuración no será la misma. Si se mantiene la misma configuración para capturar en una pantalla distinta, es posible toparse con capturas erróneas como la que se puede observar en la Figura 21.

Figura 21: Captura de junta a configurar capturada erróneamente

Algunos problemas presentados con la mala captura de la junta a configurar, ocurría al momento de procesar la imagen para obtener datos. Debido a que no se detectaba la junta a configurar, no era posible seguir con el OCR, pero esto no era presentado en la interfaz desarrollada en la fase previa, simplemente no ocurría nada. En el capítulo 10 se profundizará en las mejoras implementadas a la interfaz.

Teniendo en cuenta esto, se agregó un protocolo de programación defensiva para estos errores, que nos advierta que la imagen está mal ubicada y que por ende no era posible seguir con el procesado. Con esto el usuario puede saber el error ocurrido y reubicar la imagen. El mensaje que se despliega se puede apreciar en la Figura 22

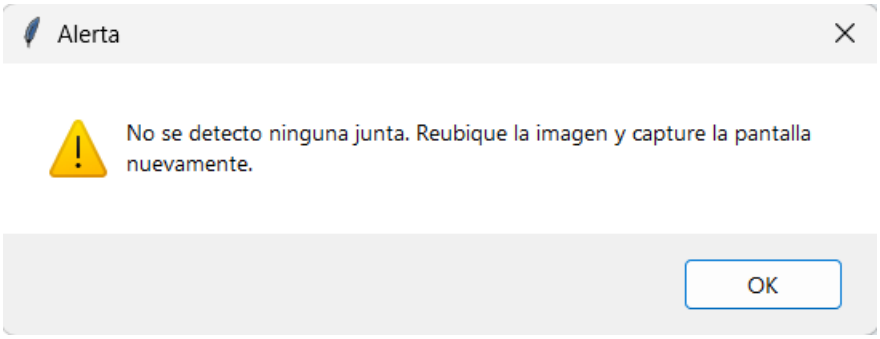


Figura 22: Mensaje de advertencia para junta no identificada

- **Sensibilidad a la calidad de la imagen:** la precisión del OCR puede verse afectada si la calidad de la captura de pantalla es baja o si hay elementos visuales que interfieren con la claridad de los caracteres. Se tuvo principalmente problemas al capturar la pantalla con los signos matemáticos, al ser pequeños eran los que más afectados se veían porque bajaba un poco la calidad.

8.2. Procesamiento de la imagen

El procesamiento de imágenes es un componente fundamental en sistemas que dependen de la extracción precisa de información visual, como es el caso de este proyecto, donde se utiliza el OCR para extraer datos numéricos de capturas de pantalla. En este contexto, el objetivo principal del procesamiento de imágenes es transformar lo obtenido de la validación de captura de pantalla en una representación que maximice la precisión del OCR, asegurando que los datos extraídos sean lo más exactos y fiables posible.

8.2.1. Escala de grises

El primer paso en el procesamiento de la imagen capturada es la conversión a escala de grises. Este proceso transforma la imagen de color en una imagen monocromática donde cada píxel representa un valor de intensidad luminosa (entre 0 y 255). Este cambio es fundamental porque:

- **Simplifica la imagen:** elimina la información de color, que no es relevante para el OCR, y reduce el ruido visual.
- **Facilita el procesamiento subsecuente:** muchos algoritmos de procesamiento de imágenes, como la umbralización, funcionan mejor con imágenes en escala de grises [26].

En esta etapa, la imagen resultante es más fácil de manejar y de procesar para el OCR ya que reduce la cantidad de daos y se enfoca únicamente en la luminancia.

La conversión a escala de grises de la imagen ocurre inmediatamente después de realizar la captura de pantalla, por ende, se puede apreciar ya la captura realizada y convertida a escala de grises como se muestra en la Figura 23.

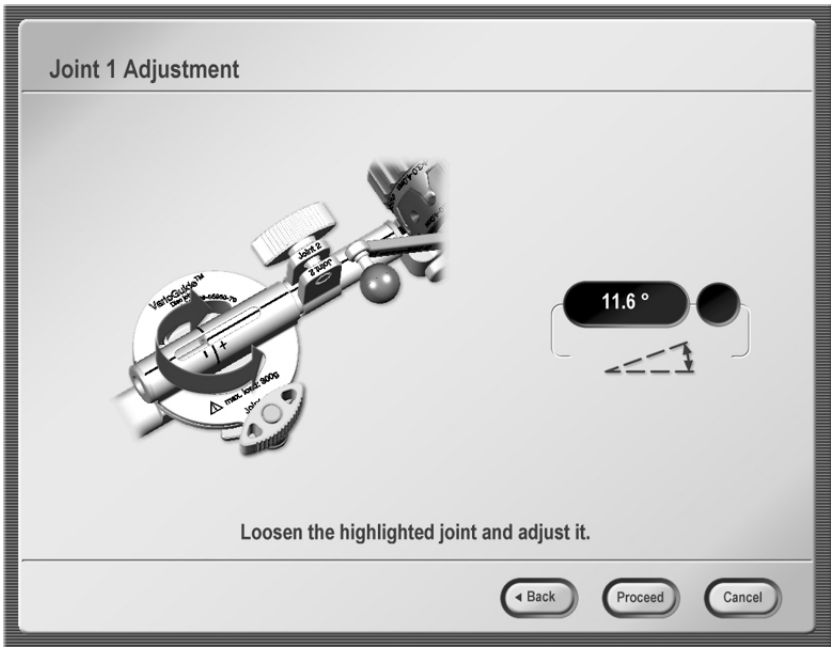


Figura 23: Captura de junta 1 convertida a escala de grises

8.2.2. Reducción de ruido

Para asegurar que la imagen esté lo más limpia posible, se aplica un filtro de reducción de ruido. El ruido en la imagen puede provenir de varias fuentes, como la calidad de la pantalla capturada o los artefactos introducidos durante la captura. Un filtro de desenfoque Gaussiano es utilizado para:

- **Suavizar la imagen:** reduce las variaciones de intensidad de pixel que no corresponden a los bordes reales de los caracteres, eliminando detalles no deseados que podrían intervenir con el OCR [27].
- **Preservar detalles:** mientras que suaviza la imagen, el filtro también preserva las estructuras relevantes como lo son los bordes de los caracteres.

Este paso asegura que la imagen final utilizada por Tesseract sea de alta calidad, minimizando los errores de OCR. En la figura 24 se puede apreciar una captura de la junta 1 con reducción de ruido.

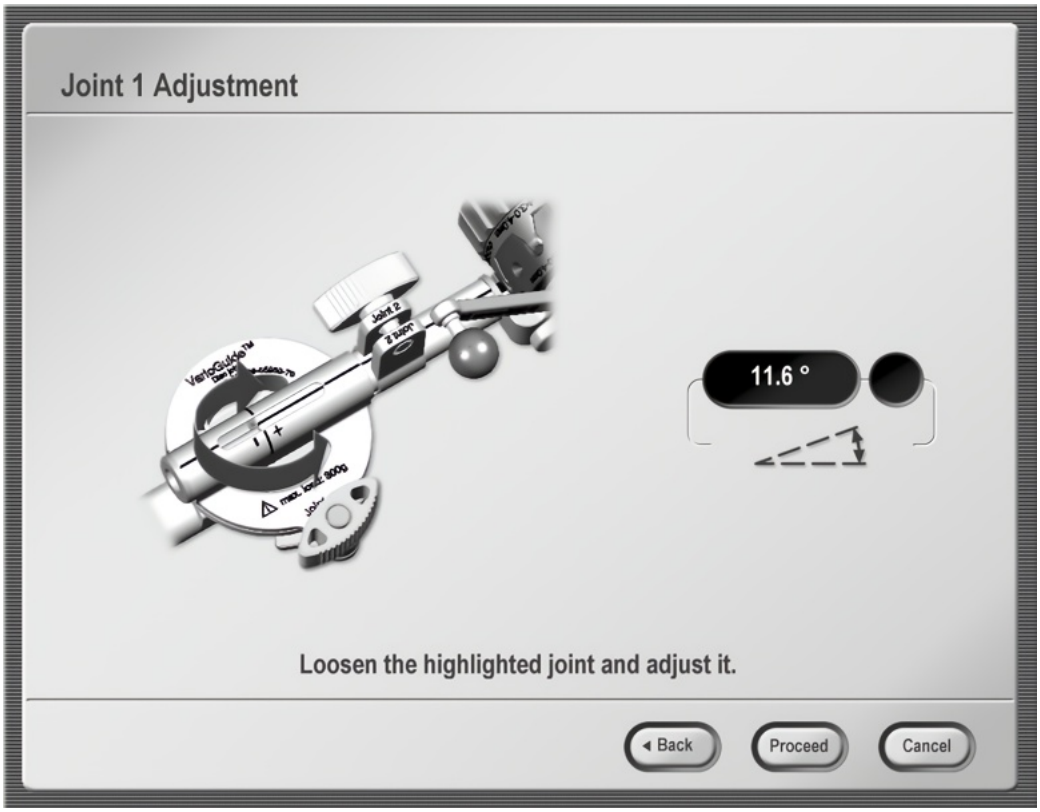


Figura 24: Captura de junta 1 con reducción de ruido

8.2.3. Umbralización binaria

Después de la conversión a escala de grises, la imagen se somete a un proceso de umbralización binaria. Este método convierte la imagen en una representación de solo dos valores: blanco (255) y negro (0). El propósito de la umbralización es:

- **Resaltar los caracteres:** separa claramente el texto del fondo, haciendo que los caracteres sean más nítidos y fáciles de reconocer.
- **Eliminar información irrelevante:** filtra el ruido visual que podría confundir al OCR, enfocándose solo en los elementos clave de la imagen.

La umbralización se implementa con un umbral fijo, por ejemplo, 128, que se establece según la luminancia de la imagen.

Este proceso es crucial para mejorar la calidad del OCR, ya que permite que los caracteres se destaquen de manera clara sobre el fondo [28]. El resultado de la umbralización binaria en la captura de la junta 1 se puede ver en la Figura 25.

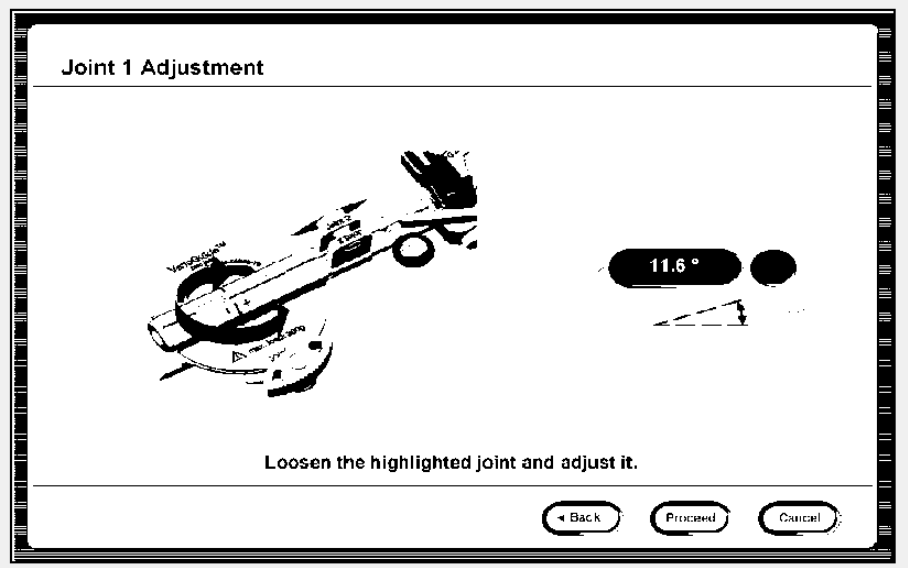


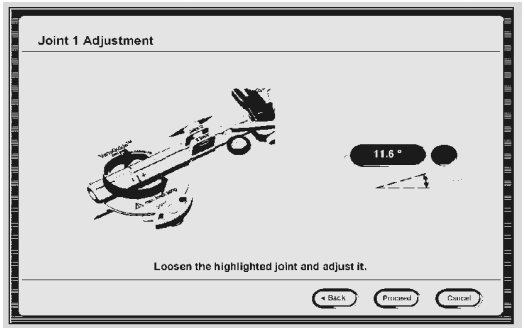
Figura 25: Captura de junta 1 umbralizada

8.2.4. Aumento de contraste

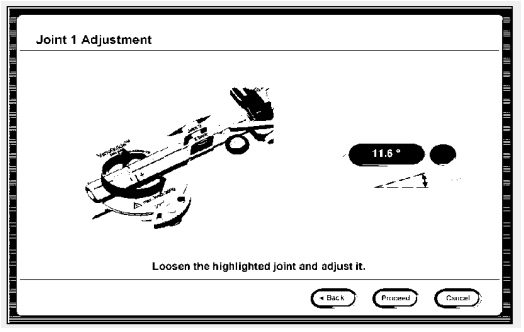
Para mejorar aún más la diferenciación entre los caracteres y el fondo, se aplica una técnica de aumento de contraste mediante la igualación del histograma [29]. Este proceso redistribuye los niveles de intensidad de la imagen para utilizar toda la gama de valores posibles, logrando:

- **Claridad mejorada:** hace que los caracteres sean más distinguibles del fondo al maximizar el contraste.
- **Optimización para OCR:** mejora la legibilidad de la imagen, lo que es crucial para que Tesseract pueda identificar correctamente los caracteres.

Este ajuste asegura que incluso los caracteres que están ligeramente difuminados o poco visibles sean realzados, mejorando la precisión del OCR. En la Figura 26 se puede apreciar la diferencia entre dos capturas de la junta 1 con diferentes niveles de contraste, en la primera siendo un contraste bajo y en la segunda un contraste alto.



(a) Captura con contraste bajo



(b) Captura con contraste alto

Figura 26: Diferencia de contrastes para captura de la junta 1

8.2.5. Verificación y evaluación

El procesamiento de imágenes implementado ha demostrado ser efectivo en maximizar la precisión del OCR en este proyecto. A través de la combinación de escala de grises, umbralización binaria, reducción de ruido, y aumento de contraste, la calidad de las imágenes se mejoró significativamente, lo que permitió a Tesseract reconocer los caracteres con un alto grado de precisión.

- **Reducción de errores:** los errores de OCR se minimizaron gracias al preprocesamiento, resultando en datos de ángulos que son confiables para el control del robot.
- **Velocidad de procesamiento:** a pesar de los pasos adicionales, el procesamiento es lo suficientemente rápido para permitir casi una actualización en tiempo real.

En las siguientes Figuras 27, 28 y 29, se puede apreciar el resultado final de los caracteres después de todos los procesos aplicados a la captura de pantalla. A pesar de que se mostraron como afectan los procesos utilizando capturas de la junta 1, estos mismos funcionan de la misma manera para la junta 1, 2 y 3.

Figura 27: Resultado de ángulo extraído

Figura 28: Resultado de ángulo negativo extraído

Figura 29: Resultado de desplazamiento lineal de junta 2 extraído

8.3. Módulo Tesseract

La necesidad de extraer información textual desde imágenes surge de la interfaz del entorno de simulación y control robótico, donde los ángulos de las articulaciones del robot se muestran visualmente. Para automatizar el proceso de captura y envió de estos datos al sistema robótico, había necesidad de crear una solución capaz de interpretar con precisión los valores numéricos mostrados en la pantalla. Tesseract fue elegido por su capacidad comprobada de realizar OCR en múltiples idiomas y formatos, su fácil integración con Python, y su flexibilidad para configurarse según las necesidades específicas del proyecto.

La configuración de Tesseract fue crucial para optimizar su rendimiento en el contexto del proyecto. Se personalizó para que se enfocara únicamente en los caracteres relevantes mediante el uso de un **"whitelist"** que incluye números, signos de grados y símbolos matemáticos.

$$'0123456789.^\circ - +' \tag{1}$$

Además, se utilizó la opción de **"–psm 6"** para especificar el modo de segmentación de página, optimizando Tesseract para el reconocimiento de bloques de texto orientados verticalmente, lo que corresponde a la presentación de los ángulos en la interfaz.

El uso de Tesseract en este contexto resultó en una extracción precisa de los ángulos de las articulaciones. Algunos aspectos que fueron clave para los resultados brindados por Tesseract fueron los siguientes:

- **Precisión:** Tesseract logró una alta precisión en el reconocimiento de caracteres con una tasa de error mínima en la interpretación de los valores numéricos, lo cuál es critico para el control correcto del robot. Esta mejoría en cuanto a la precisión se puede apreciar en el Cuadro 1.

Porcentaje de acierto en reconocimiento (50 corridas)			
	Junta 1	Junta 2	Junta 3
Algoritmos viejos	78	82	90
Algoritmos nuevos	100	100	94

Cuadro 1: Porcentaje de aciertos en cuanto al reconocimiento de caracteres

- **Velocidad de procesamiento:** el tiempo de procesamiento para cada imagen fue optimizado a través del preprocesamiento, resultando en un OCR rápido y eficiente que permitió una actualización casi en tiempo real de los valores capturados.

  Como se puede apreciar en los Cuadros 2 y 4 se tienen los tiempos de algoritmos pasados, nuevos y la reducción de tiempo entre estos.

Promedio de tiempo de procesamiento de 50 corridas			
	Junta 1	Junta 2	Junta 3
Algoritmos viejos	198.02 ms	247.61 ms	255.18 ms
Algoritmos nuevos	135.84 ms	141.87 ms	137.93 ms

Cuadro 2: Promedio de tiempos de procesamiento

Desviación estándar de tiempo de procesamiento de 50 corridas			
	Junta 1	Junta 2	Junta 3
Algoritmos viejos	6.95 ms	19.91 ms	2.87 ms
Algoritmos nuevos	3.51 ms	4.38 ms	3.98 ms

Cuadro 3: Desviaciones estándar de tiempos de procesamiento

Reducción de tiempo de procesamiento		
Junta 1	Junta 2	Junta 3
62.17 ms	105.74 ms	117.25 ms

Cuadro 4: Reducción de tiempos de procesamiento

Estos muestran que los tiempos han disminuido significativamente en las tres juntas cuando se utilizaron los nuevos algoritmos. Los promedios de tiempo de procesamiento de la fase anterior, mostrados en el Cuadro 2 son más altos que los tiempos correspondientes de los nuevos algoritmos. Esta mejora en la eficiencia es particularmente notable para la junta 2, donde el tiempo promedio se redujo de 247.61 milisegundos a 141.87 milisegundos.

En cuanto a las desviaciones estándar, el Cuadro 3 nos permite analizar la variabilidad o dispersión de tiempos de procesamiento respecto al promedio para los algoritmos de la fase pasada. En cuando a la junta 1 tiene una desviación estándar relativamente baja, lo que indica que los tiempos de procesamiento de esta junta están bastante cercanos al promedio. En cuanto a la junta 2, se tiene una desviación estándar más alta, lo que sugiere que los tiempos de procesamiento en esta junta varían más respecto a su promedio, es decir, hay más variación en los tiempos que toma el algoritmo en la fase anterior para esta junta, esto puede ser debido a que para esta junta se extraen 2 datos.

En el Cuadro 3 también podemos ver las desviaciones estándar para los algoritmos nuevos, estas desviaciones también son bajas, lo que indica que los tiempos de procesamiento son bastante consistentes y no hay variación con respecto a los promedios.

Comparando las desviaciones estándar de los algoritmos de la fase anterior, podemos decir que las desviaciones estándar de los algoritmos nuevos son, en general, más bajas que las de los algoritmos anteriores, lo que siguiere que los nuevos algoritmos no solo son más rápidos, sino que también más consistentes en su tiempo de procesamiento. Aunque la junta 2 sigue teniendo la desviación estándar más alta, esta es considerablemente más baja que en la fase anterior, indicando una mejora en la consistencia.

Cabe mencionar que, para el calculo de las estadísticas, se añadió la librería de pandas para estructurar y manejar los datos generados por el sistema. Esto facilitó el análisis estadístico de los resultados, así como la exportación de datos en formatos compatibles

con otras plataformas. Se complementó esta mejora con la inclusión de *matplotlib*, logrando así un análisis más claro y detallado del rendimiento del sistema.

El tiempo de procesamiento es crucial en sistemas que dependen de actualizaciones en tiempo real, como en este caso es el del control de un robot. La reducción acumulada en los tiempos de procesamiento entre las tres juntas tiene un impacto directo en la velocidad general del sistema, lo que permite una respuesta más rápida para el usuario en la interfaz. Estas mejoras sugieren que el sistema es ahora más eficiente, lo que podría permitir una mayor frecuencia de actualizaciones de los ángulos de las articulaciones y, por lo tanto, una mayor precisión y rapidez en la manipulación robótica.

- **Robustez en diversos escenarios:** se probaron diferentes configuraciones visuales, como variaciones en el brillo, contraste y color de fondo, así como diferentes niveles de aumento (*zoom*) en las imágenes. A pesar de estos cambios, Tesseract se mantuvo robusto ante dichas variaciones, demostrando su capacidad de adaptarse a diferentes condiciones de entrada y manteniendo un rendimiento consistente en el reconocimiento de caracteres.

8.3.1. Limitaciones y consideraciones de Tesseract

Aunque Tesseract funcionó bien en la mayoría de los casos, hubo algunas limitaciones observadas:

- **Sensibilidad a la calidad de la imagen:** la precisión de Tesseract puede disminuir si la calidad de la imagen es demasiado baja o si hay un ruido visual significativo, lo que subraya la importancia del preprocesamiento adecuado. Para este proyecto se trabajo con capturas de pantalla proporcionadas por el sistema de Brainlab directamente, por lo que la calidad de la imagen con la que se trabajo era bastante buena, aun así, se menciona esto ya que, con las versiones anteriores de capturas de pantalla, fue donde se tuvo esta dificultad, debido a que la calidad de las capturas era más baja que la de las imágenes recientes.
- **Errores en caracteres complejos o dañados:** aunque la configuración del **"whitelist"** ayudó a reducir errores, en algunos casos, caracteres complejos o parcialmente visibles no se reconocieron correctamente. Este problema se notó con los signos negativos o con valores algo parecidos, como lo pueden ser el número **"3"** con el número **"8"**. Debido a que para hacer posible el reconocimiento de caracteres se debía realizar una escala al fragmento a analizar, se consideró que con una escala demasiado grande los valores podían llegar a verse demasiado detallados que incluso se omitían ciertos aspectos del dígito como se puede ver en la Figura 30.

Figura 30: Dato de junta con una escala grande para el OCR

Por otra parte, si se daba una escala demasiado pequeña se tenía la confusión que un valor al verse demasiado pequeño se podía interpretar como otro, un claro ejemplo

sería el número **“3”** que al verse en una escala muy pequeña puede verse como un número **“8”** ya que no se logra apreciar su separación. Esto puede verse en la Figura 31.

Figura 31: Dato de junta con una escala pequeña para el OCR

- **Dependencia del entorno:** tesseract requiere configuraciones específicas dependiendo del entorno operativo, como la ubicación del ejecutable de Tesseract, lo cual puede necesitar ajustes adiciones en diferentes plataformas.

CAPÍTULO 9

Protocolo de comunicación

Se desarrolló un protocolo que permite la transferencia de datos de manera confiable y eficiente, asegurando que los comandos generados por el procesamiento de imágenes sean recibidos y ejecutados correctamente por el robot myCobot 280 M5. La estructura de este protocolo está diseñada para optimizar la velocidad de transmisión, la robustez ante posibles errores y la flexibilidad para futuras expansiones.

Para la comunicación con el sistema robótico, basándose en lo validado en fases anteriores, se optó por utilizar el protocolo de comunicación Universal Asynchronous Receiver-Transmitter (UART). Algunos factores que hicieron que se seleccionara esta tecnología fue su simplicidad y efectividad en sistemas embebidos. UART permite la comunicación en serie, lo que es ideal para la integración entre la computadora en serie y el microcontrolador que contiene el brazo robótico.

9.1. Implementación y validación del protocolo en Python

Para implementar este protocolo, se utilizaron las librerías **"PySerial"** y **"Tkinter"** en Python. **"PySerial"** permitió gestionar la comunicación en serie entre la computadora y el myCobot 280 M5, mientras que **"Tkinter"** fue clave para integrar esta comunicación con la interfaz gráfica de usuario (GUI), desde la cual se puede controlar el robot.

Para validar la implementación del protocolo de comunicación, se realizaron diversas pruebas donde se verificó la confiabilidad de la transmisión y recepción de datos bajo diferentes condiciones de operación. Las pruebas incluyeron:

- **Pruebas de integridad de datos:** se verificó que los datos enviados llegaran sin errores al robot, utilizando los códigos de ejemplo que nos proporciona la librería del

myCobot. Se verificó que con estas pruebas el robot de moviera correctamente, para posteriormente implementarlo en la unidad de procesamiento y que pasara a tomar los valores de los ángulos obtenidos por medio del OCR.

- **Pruebas de latencia:** se tomó en cuenta el tiempo de respuesta del sistema, el cual no presento ninguna tardanza, inmediatamente cuando se ejecutaba el código de prueba, el robot pasaba a adoptar las poses configuradas.

9.2. Estructura de la trama de datos

La comunicación entre el sistema y el robot se realiza en tramas de datos estructuradas. La estructura de la trama de datos para la comunicación con el myCobot 280 M5 mediante la librería dedicada en Python facilitó considerablemente el proceso de envío de información. La librería encapsula y abstrae gran parte de la gestión de los datos, lo que permite que, como usuario, uno se centre en los comandos y parámetros relevantes.

A pesar de que la librería myCobot maneja internamente los aspectos de bajo nivel, fue importante comprender los principales componentes de la trama que se envía al robot.

9.2.1. Establecimiento de la conexión

Antes de enviar cualquier comando al myCobot, es necesario establecer una conexión serial con el robot. La librería gestiona la identificación del puerto y la configuración de la velocidad de transmisión (baudaje). Este paso es equivalente a la inicialización del canal de comunicación en el protocolo UART.

9.2.2. Código de comando

En este caso, la librería se encarga de proporcionar el código de comando necesario para el tipo de acción que se va a ejecutar, en este caso, mover las articulaciones del robot a ángulos específicos. Cada comando está predefinido en la librería, y solo fue necesario llamar a la función correspondiente. El código de comando se relaciona con las funciones de alto nivel como **"send_angles"**, que es la encargada de enviar un conjunto de ángulos a las articulaciones del robot.

Datos

Los datos en la trama de comunicación son los ángulos de las articulaciones que el robot debe adoptar. Estos ángulos se pasan en forma de una lista de números flotantes (grados), que representan los valores de rotación de cada articulación. En este caso como el myCobot tiene 6 grados de libertad, el comando nos pide mandar 6 datos de configuración, uno para cada grado de libertad. Debido a que el sistema Brainlab de HUMANA únicamente dispone de 4 configuraciones, se utilizaron únicamente 4 grados de libertad del robot, los restantes se dejaron para que no cambien de su posición inicial.

El segundo parámetro en la función **"send_angles"** es la velocidad con la que deben ejecutarse los movimientos. Esta velocidad es un valor entero, en este caso, se dejó una velocidad de 90 %.

9.2.3. Validación de datos y confirmación de ejecución

Después de enviar los comandos, es importante verificar que los datos fueron enviados correctamente y que el robot ha ejecutado bien los movimientos. La librería proporciona funciones para leer la respuesta del robot. Los datos recibidos incluyen los ángulos actuales de las articulaciones, que se comparan con los ángulos enviados para asegurar que no hayan discrepancias entre estos.

En el código hay fragmentos que extraen los valores recibidos y los comparan con los enviados. Si los valores coinciden, se confirma que el comando ha sido ejecutado correctamente. Finalmente, si los valores recibidos coinciden con los esperados, se confirma la transmisión exitosa y la ejecución correcta del movimiento por parte del robot. Esto garantiza que no haya errores de transmisión o ejecución en el proceso.

9.2.4. Precauciones y alertas

Con relación a las pruebas realizadas en fases anteriores, se observó que cuando los datos no eran transmitidos correctamente o la comunicación con el robot fallaba, no se generaba ninguna advertencia visible, lo que dificultaba la identificación del problema. Este comportamiento llevó a la realización de pruebas adicionales para diagnosticar el origen de las fallas. Para abordar este problema y mejorar la fiabilidad del sistema, se implementaron diversas funciones de alerta y precaución que ayudan a detectar y manejar posibles errores de comunicación.

- **Sin conexión al myCobot 280 M5:** se implementó una verificación de conexión al puerto antes de enviar cualquier comando al robot. Si el sistema no detecta la conexión al myCobot 280 M5, se genera una alerta inmediata al usuario. Esta alerta se presenta en la interfaz gráfica, indicando que no hay un dispositivo conectado, y se bloquean las operaciones de envío de datos hasta que la conexión sea establecida. Esto previene la ejecución de comandos en vacío y evita pérdidas de tiempo innecesarias. Dicha alerta se puede observar en la Figura 32.

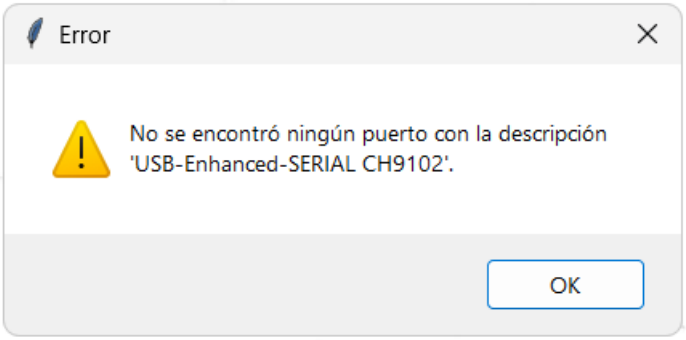


Figura 32: Alerta de conexión no establecida

- **Múltiples myCobot 280 M5 conectados:** durante las pruebas, se pensó que podrían haber entornos donde se tengan varios robots conectados simultáneamente al sistema, por lo que se identificó riesgo de enviar comandos al dispositivo equivocado. Para abordar este problema, se implemento una función que detecta y lista todos los dispositivos myCobot conectados a los puertos disponibles. Si se detecta más de un dispositivo, el sistema alerta al usuario mediante un mensaje en la interfaz gráfica solicitando la selección del puerto correspondiente al robot deseado. La alerta se puede ver en la Figura 33.

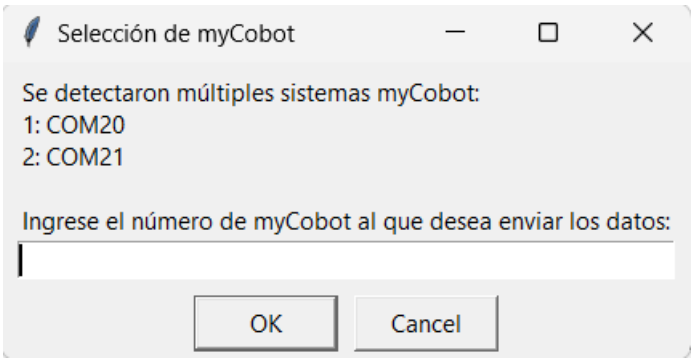


Figura 33: Alerta de varios sistemas robóticos conectados simultáneamente

Además, se bloquea el envío de comandos hasta que el usuario haya seleccionado manualmente el dispositivo adecuado desde la lista de puertos disponibles. Esto previene errores de control, como el envío accidental de movimientos a un robot no previsto, lo que podría ocasionar situaciones peligrosas o daños en el equipo. Esta precaución es crucial en entornos donde se utilizan múltiples robots simultáneamente para diferentes tareas. garantizando que el control se ejerza de manera precisa y segura sobre el robot correcto.

- **Datos incompletos:** para asegurar la integridad de las órdenes enviadas al myCobot 280 M5, se implementó una función de verificación que comprueba que todos los datos necesarios están presentes antes de realizar cualquier envío de comandos. Si los datos que se van a enviar, como los ángulos de las articulaciones o la velocidad, están incompletos o contienen valores inválidos, el sistema genera una advertencia al usuario. Esta

alerta aparece en la interfaz gráfica, indicando la falta de información. Esta alerta se puede apreciar en la Figura 34.

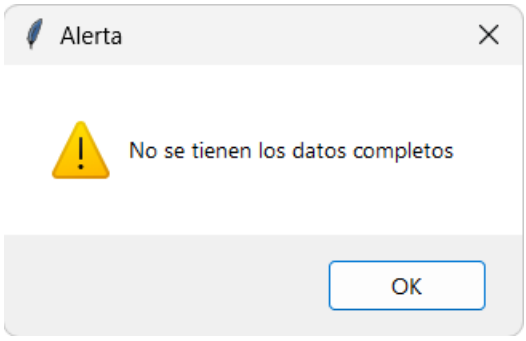


Figura 34: Alerta de datos incompletos para enviar al sistema robótico

El envío de datos se bloquea hasta que el usuario complete o corrija los campos faltantes. Esta medida previene que el robot reciba comandos incompletos, ya que podría causar movimientos inesperados, fallos en la ejecución de las tareas o daños al equipo.

CAPÍTULO 10

Interfaz gráfica

La interfaz gráfica (GUI) es un componente crucial y de lo más importante en la interacción entre el usuario el sistema, así mismo es una de las bases de este proyecto. En este trabajo, se ha diseñado una GUI para facilitar el control y monitoreo del sistema de procesamiento de imágenes y el sistema robótico asociado. La GUI actúa como medio a través del cual los usuarios realizan todo el procedimiento de procesado de la imagen, visualización de los caracteres obtenidos, envío de datos al robot, etc. Esto sin requerir conocimientos avanzados en programación o en control de robots, ya que se buscó hacer la GUI bastante intuitiva.

En términos de interacción con el usuario, se decidió mejorar la interfaz gráfica inicial basada en *tkinter*. Para ello, se incorporaron elementos como simpledialog y MessageBox, que facilitan la interacción dinámica con el usuario mediante cuadros de diálogo y mensajes informativos. Esta mejora no solo hace que el sistema sea más intuitivo y accesible, sino que también simplifica su uso por parte de operadores no técnicos.

En este capítulo se detallan el diseño, implementación y funcionalidades de la GUI, destacando las herramientas utilizadas para su desarrollo, así como la integración con el protocolo de comunicación y los módulos de procesamiento de imágenes. Aunque la interfaz de la fase anterior ya contenía ciertos aspectos, algunos se reubicaron y mejoraron para poder hacerla aun más intuitiva. La interfaz de la fase anterior se puede observar en la Figura 35.

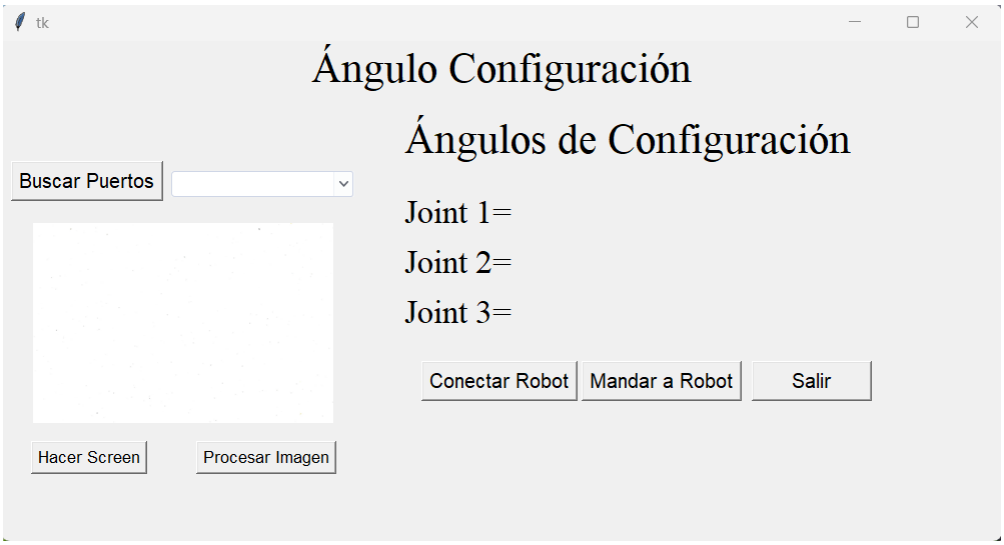


Figura 35: Interfaz gráfica (GUI) de la fase anterior [4]

10.1. Herramientas de desarrollo

La interfaz se diseñó de manera que para el usuario fuese intuitiva y sencilla de utilizar. En ella se implementaron herramientas que permiten interactuar con lo siguiente:

- Realizar captura de pantalla.
- Visualización de captura realizada.
- Realizar procesado de la imagen capturada.
- Visualización de ángulos de configuración.
- Configurar el robot en su posición inicial.
- Mandar ángulos de configuración al robot.
- Ver simulación del robot.
- Realizar proceso completo.

Para el desarrollo de la GUI también se utilizó el lenguaje de programación Python, en conjunto con ciertas librerías especializadas. La librería principal utilizada fue **"Tkinter"**, esta permite crear interfaces gráficas de manera sencilla y eficiente. **"Tkinter"** fue la elección primaria debido a su simplicidad y a la amplia documentación disponible, lo que facilitó su implementación.

En ciertos momentos, se considero el uso de **"PyQt"**, una librería más avanzada que ofrece mayor control sobre los elementos gráficos y permite un diseño más detallado y sofisticado para la GUI. Sin embargo, se optó por la simplicidad de Tkinter para garantizar una

experiencia de usuario más simple y accesible, aunque completa. Además, se integró **"PIL" (Python Imaging Library)**, que permitió manipular y mostrar las imágenes procesadas dentro de la interfaz, haciendo posible que el usuario visualice los resultados de los algoritmos de procesamiento de imágenes. El resultado de la estructura de la interfaz final se puede apreciar en la Figura 36.

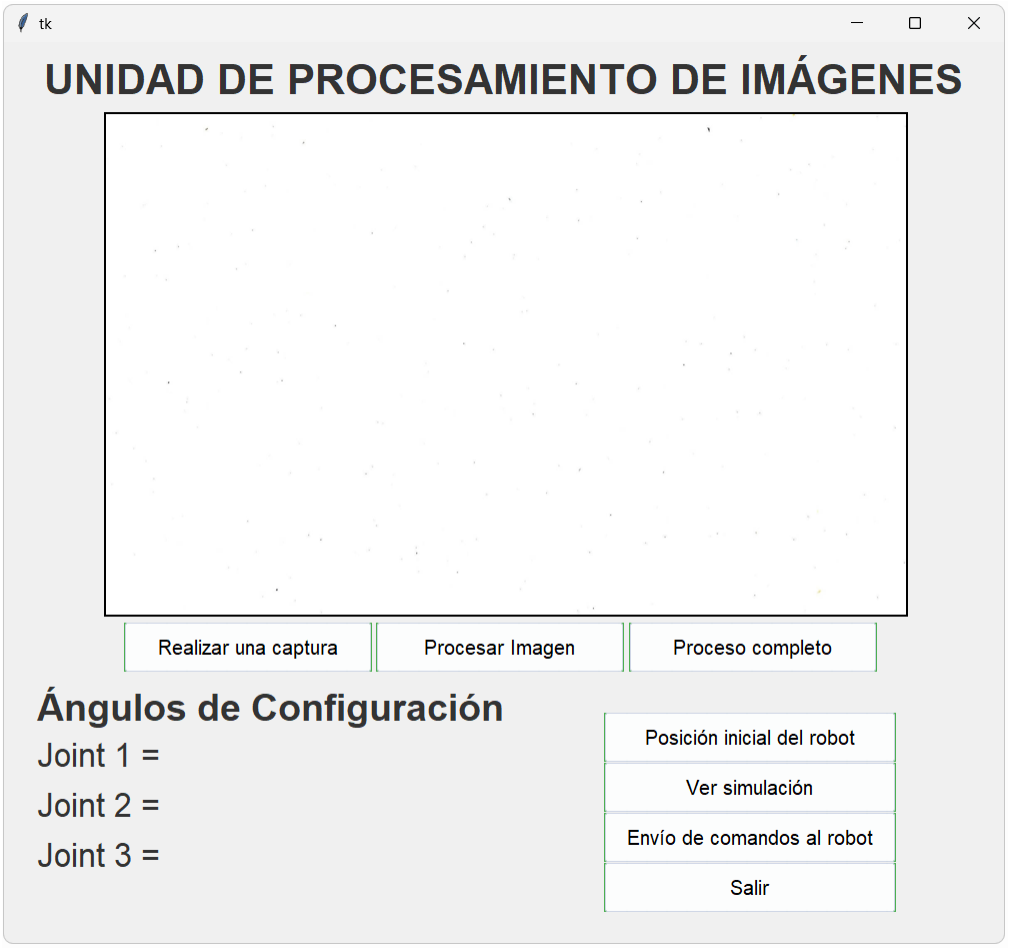


Figura 36: Interfaz gráfica (GUI)

10.2. Estructura de la interfaz

La estructura de la interfaz fue diseñada de manera que todas las funcionalidades del sistema estén realmente accesibles y fáciles de identificar para el usuario. la GUI se divide en varias secciones funcionales. En primer lugar se encuentra el panel de visualización que ocupa la mayor parte de la ventana en un lugar central, mostrando la imagen capturada, como se puede ver en la Figura 37.

En la parte de visualización podemos incluir también los espacios donde se muestran los valores de configuración de cada junta. Se tienen tres etiquetas que nos muestran qué configuración se tiene para cada junta. Estas etiquetas se van actualizando conforme se vaya

realizando el procesado de cada imagen. Para la junta 1 y 2, se tiene únicamente un valor de desplazamiento angular, mientras que, para la junta 2, se tiene un valor de desplazamiento angular y un valor de desplazamiento lineal en milímetros. La forma en como se muestran esta información se puede apreciar en la Figura 38.

El hecho de que se pueda tener la visualización inmediata del OCR es clave para que el usuario pueda monitorear el rendimiento del sistema y validar los datos obtenidos. Al visualizar que estos son correctos con los que se tienen en las capturas de pantalla, el usuario puede estar seguro de enviar los comandos al robot.

Figura 37: Interfaz gráfica (GUI) con imagen capturada

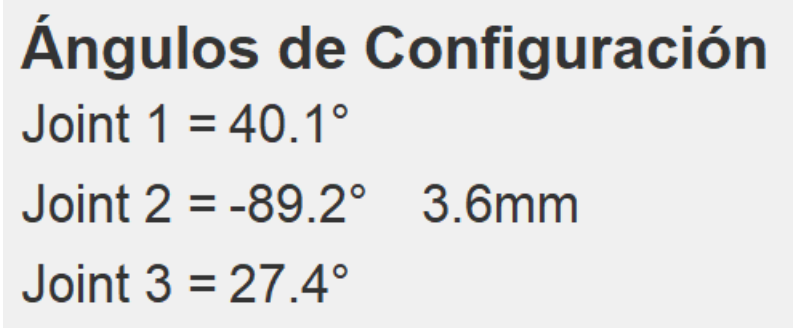


Figura 38: Apartado de etiquetas con valores de ángulos de configuración obtenidos a partir del OCR

En paralelo, se cuenta con un panel de control del robot, desde el cual el usuario puede emitir comandos para el movimiento del robot myCobot 280 M5. Este panel permite ajustar la posición y orientación del robot en función de los datos obtenidos del procesamiento de imágenes, haciendo uso de botones interactivos que simplifican la ejecución de comandos complejos. Esto se logra gracias a la integración con el protocolo de comunicación basado en UART, que permite una comunicación confiable sin retrasos significativos. De esta manera, el usuario puede enviar comandos desde la interfaz gráfica y ver los resultados reflejados en el comportamiento del robot casi instantáneamente. Los botones relacionados al control del robot son los siguientes:

- **Posición inicial del robot:** este botón, como su nombre lo indica, hace que el robot se configure en su posición inicial. Se implementó este botón, ya que, al inicio, el robot no tiene una posición definida, como se puede ver en la Figura 39, por lo que es fácil que este configurado de manera que no se tenga conocimiento de que movimiento puede realizar o puede ser confuso visualizarlo. Además, después de mandar al robot a cierta configuración, puede que no se desee tenerlo así después de concluida la tarea, por lo que, tenerlo en su posición inicial siempre y cuando no se realice algo más es lo indicado. La posición inicial del myCobot 280 M5 se puede apreciar en la Figura 40, se entiende como posición inicial cuando todas las juntas del myCobot 280 M5 no tienen ningún desplazamiento angular, es decir todas están en su posición de 0°.

Figura 39: myCobot 280 M5 sin configuración en sus juntas al momento de inicializarse

Figura 40: myCobot 280 M5 en su posición inicial

- **Envío de comandos al robot:** este botón desempeña una función crucial en la operación del sistema, ya que envía al robot las configuraciones específicas que debe adoptar, basadas en los valores obtenidos tras el procesamiento OCR. Es decir, los valores que se visualizan en las etiquetas asociadas a las juntas del robot son los que se enviarán directamente cuando se pulse este botón.

El envío de estos datos se realiza mediante un protocolo de comunicación previamente definido. Antes de realizar el envío, el sistema verifica que todos los campos de entrada contengan valores válidos, ya que para un análisis preciso del comportamiento del robot y la correcta ejecución de sus movimientos, es esencial que cada junta tenga un valor asignado. Esto asegura que el robot reciba la información completa y pueda ejecutar movimientos coordinados y coherentes en todas sus articulaciones.

Además, esta verificación previa evita el envío de configuraciones incompletas que podrían generar comportamientos indeseados o errores en la operación del robot. Si uno de los campos está vacío o contiene un valor no válido, el sistema bloqueará temporalmente el envío de datos y notificará al usuario para que corrija los errores (Figura 34).

Una vez todos los valores están validados, el sistema empaqueta la información y la transmite al robot, asegurándose de que los comandos lleguen de manera segura y sin errores. El robot, al recibir los datos, ajusta sus juntas a las posiciones indicadas por el OCR, permitiendo que el usuario observe los resultados en tiempo real como se puede ver en la Figura 42 donde el myCobot adopta las posiciones obtenidas en la Figura 41.

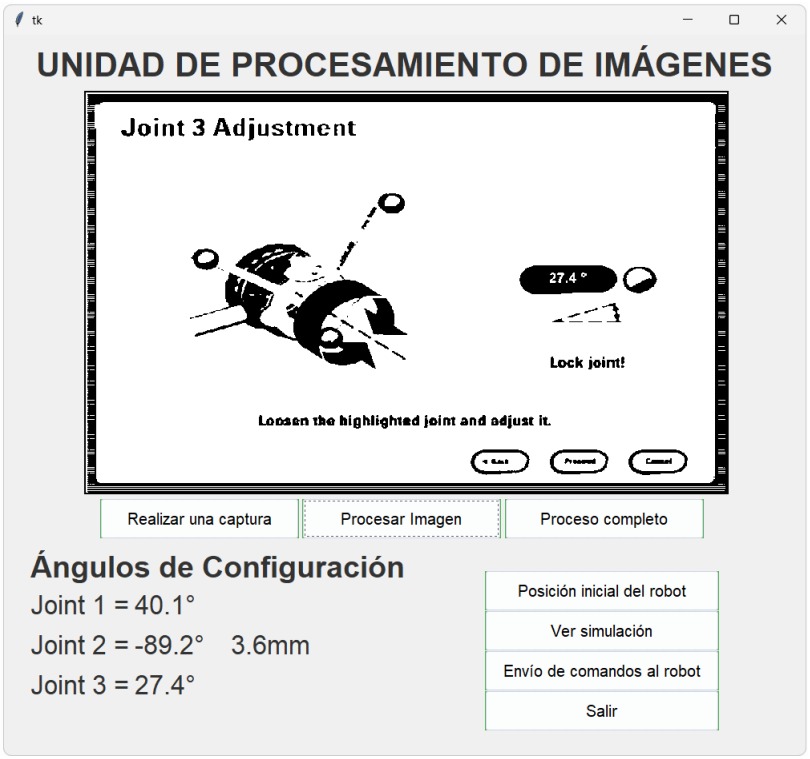


Figura 41: Configuración de datos para enviar a myCobot 280 M5

(a) Perspectiva 1

(b) Perspectiva 2

Figura 42: myCobot 280 M5 configurado con las posiciones de juntas de la Figura 41

10.3. Optimización de la interfaz gráfica

A lo largo del desarrollo, se realizaron varias iteraciones de la interfaz con el objetivo de optimizar la experiencia del usuario. Entre las mejoras implementadas, destaca la reducción de tiempos de respuesta. Durante las primeras pruebas, se destacaron retrasos entre el momento en que el usuario emitía un comando y la respuesta del sistema. Esto se solucionó optimizando el manejo de las llamadas a funciones y mejorando la actualización de la interfaz, así mismo, como el OCR también se mejoró con los distintos métodos de procesado de imagen, se requería menos tiempo de cómputo lo que también implico en la reducción del tiempo completo de la operación. Todo esto combinado resultó en una experiencia más fluida y fácil.

Para corroborar esto, se llevaron a cabo pruebas con usuarios finales para obtener retroalimentación sobre la facilidad de uso de la interfaz. Gracias a estas pruebas, se simplificaron algunos menús y botones, haciendo la GUI más intuitiva y accesible, incuso para usuarios con poca experiencia en el manejo de sistemas robóticos.

Algunas partes eliminadas con respecto a la versión anterior de la interfaz son:

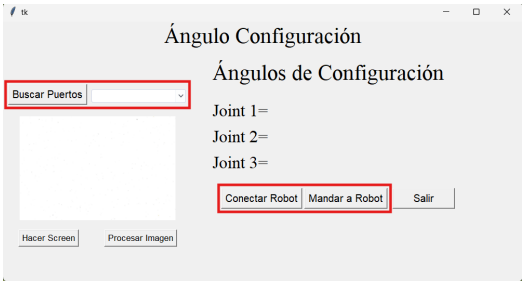
- **Búsqueda y selección de puertos**
- **Conexión al robot**

Como se buscaba que la interfaz fuera lo más intuitiva posible para usuarios que no tuvieran conocimientos previos de programación, se consideró que los conceptos **“Buscar**

puerto" y **"Puerto COM"** podrían confundir al usuario. Así mismo, aunque tuvieran conocimiento de esto, en las computadoras pueden haber más dispositivos conectados por medio de distintos puertos COM, por lo que al momento de conectar el robot y hacer la búsqueda de puertos, era bastante probable que salieran múltiples puertos COM sin identificar debidamente y que fuera complicado identificar cuál era el myCobot 280 M5.

Debido a esto, la parte de búsqueda y selección de puerto COM se eliminaron y en su lugar se implemento la lógica de búsqueda y selección de puerto de manera automática, directamente cuando se manden datos al robot.

Para lograr esto, primero se declaró una descripción de puerto, ya que al conectar el myCobot 280 M5, este tiene un nombre predeterminado. El nombre con el que se identifica al myCobot 280 M5 es **"USB-Enhanced-SERIAL CH9102"**. Al tener esto en cuenta, en el código se declaro como la descripción de puerto a buscar, con esto lo que hace el código es buscar entre los puertos disponibles, un puerto con esta descripción y al encontrarlo, establece la conexión y manda los datos. Si no se tiene un myCobot 280 M5 conectado, se verá una alerta en la pantalla (Figura 32) y si hay más de un myCobot 280 M5 conectado se verá una alerta en la pantalla y se pedirá que se ingrese a cual de las opciones disponibles se quieren enviar los datos (Figura 33).



(a) Botones necesarios para el envío de datos al sistema robótico en la fase anterior

(b) Botones necesarios para el envío de datos al sistema robótico en la fase nueva

Figura 43: Comparación de botones para el envío de datos entre fase previa y nueva

Como se puede apreciar en la Figura 43 se pasó de tener que realizar cuatro operaciones a solo tener que realizar una para el envío de datos al sistema robótico.

Ahora, las nuevas implementaciones que se hicieron en la GUI son:

Proceso completo

La función de proceso completo, como su nombre lo indica tiene como propósito realizar todo el proceso junto presionando un solo botón. Para el proceso de análisis, verificación,

etc. Es conveniente tener las funciones de captura, procesado de imagen y envío de datos por separado, pero pensando en las personas que necesitan directamente la interfaz para trabajar con el myCobot sin poner importancia a lo que sucede de por medio, es mucho más práctico tener un botón que realice todo el proceso junto.

La función de proceso completo contiene dentro de sí a las funciones de tomar la captura, procesar la imagen y mandar datos al robot. Cabe resaltar que la función de mandar datos al robot, por sí sola no funciona si no se tienen todos los datos completos, en este caso, la función de proceso completo, omíte esta parte y va configurando al robot conforme se vayan obteniendo los datos, es decir uno por uno. Al momento de capturar la pantalla de la junta 1, esta se procesa, se extrae el valor de configuración y se manda inmediatamente al robot, mientras las otras 2 juntas se mantienen sin configurar hasta que se procesen las imágenes con dicha información para estas.

Simulación

En este proyecto, la simulación se implementó para validar y evaluar el comportamiento del robot antes de implementarlo en un entorno físico. Este paso permite identificar posibles errores que puedan existir al momento de mandar las configuraciones al robot sin poner en riesgo el hardware.

El sistema de simulación está diseñado para replicar de manera sencilla, pero precisa, las características cinemáticas y dinámicas del robot myCobot 280 M5. Utilizando modelos virtuales, el entorno simula las juntas y enlaces del robot, así como los movimientos que este ejecutaría con base en los comandos generados por el sistema de procesamiento de imágenes y OCR. Esta simulación permite probar los movimientos y configuraciones del robot bajo condiciones controladas, verificando que los comandos enviados desde la interfaz de usuario sean correctos y que los movimientos resultantes sean los deseados.

Al estar integrado con la interfaz gráfica, permite al usuario iniciar y detener las simulaciones, cambiar configuraciones de parámetros y observar los resultados visuales sin necesidad de cambiar de entorno. Esto proporciona una experiencia de uso fluida, donde el usuario puede interactuar tanto con el modelo virtual del robot como con las herramientas de procesamiento de imágenes en una sola interfaz.

En las pruebas realizadas durante el desarrollo del proyecto, el hecho de tener la simulación fue clave para detectar y corregir errores en los algoritmos de procesamiento de imágenes y en el protocolo de comunicación con el robot. En algunos casos se tenían movimientos fuera de rango esperado por ciertas juntas, que en dado caso de no tener la simulación se hubieran enviado al robot y este no hubiera podido ejecutarlas debido al límite de movimiento o podría llegar a forzar los servomotores. Este tipo de pruebas previas reduce significativamente el riesgo de errores en la fase final de implementación, aumentando la confiabilidad del sistema.

Para el desarrollo de la simulación se utilizó como herramienta principal la "Robotics Toolbox de Peter Corke" [30], una librería ampliamente utilizada en la comunidad de robótica para simular y analizar robot. Esta toolbox, diseñada para MATLAB [31], pero con una versión en Python, proporciona un conjunto robusto de funciones que permitieron modelar

el comportamiento cinemático y dinámico de robots manipuladores. En este proyecto, se utilizó para modelar las características del robot myCobot 280 M5, como los grados de libertad de sus juntas y los enlaces que conectan las distintas partes del robot. En la Figura 44 se puede apreciar la simulación de un myCobot en el entorno de MATLAB.

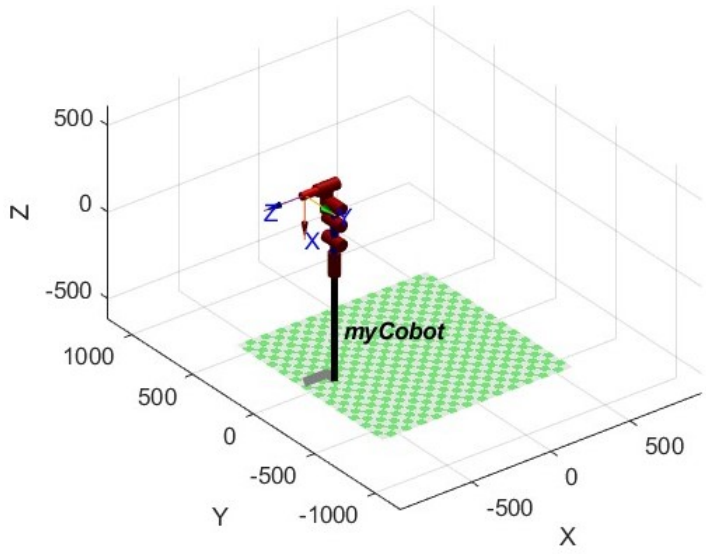


Figura 44: Simulación de un myCobot en MALTLAB utilizando la "Robotics Toolbox de Peter Corke"

Cabe mencionar que en esta parte de simulación también se tuvieron iteraciones, en la primera iteración simplemente se utilizó la "Robotics Toolbox" donde la simulación del myCobot 280 M5 se ve reflejada en la Figura 45.

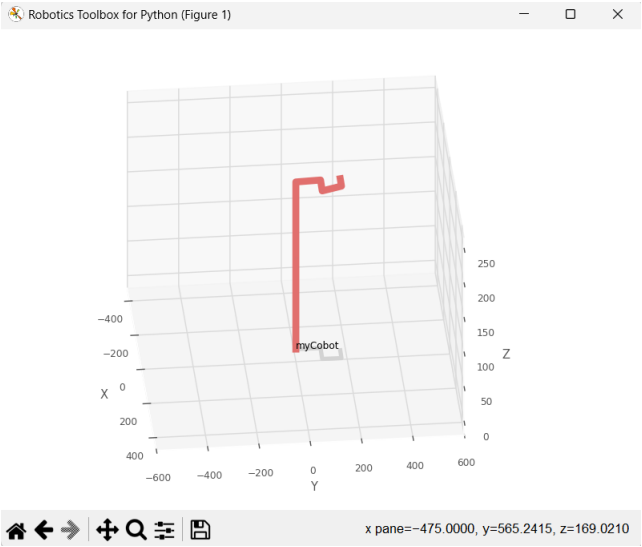


Figura 45: Primera iteración de simulación utilizando únicamente la "Robotics Toolbox"

El problema que se presentó en esta simulación es que el cuerpo del robot era de un mismo color y no era posible identificar la diferencia entre distancia de juntas, por lo que en tramos o configuraciones donde no se movía alguna junta costaba identificar si hubo un cambio o no. Se pensó en cambiar los colores de estas, pero directamente la "Robotics Toolbox" no permite un cambio de colores entre estas, por ende para hacer notar este cambio se optó por utilizar también "Matplotlib".

"Matplotlib" es una biblioteca de Python especializada en la creación de gráficos y visualizaciones, lo que permitió generar representaciones visuales del robot, sus movimientos y a diferencia de solamente la "Robotics Toolbox", la combinación con "Matplotlib" nos permitió el cambio de color entre juntas. El resultado de esta segunda iteración de simulación se puede observar en la Figura 46.

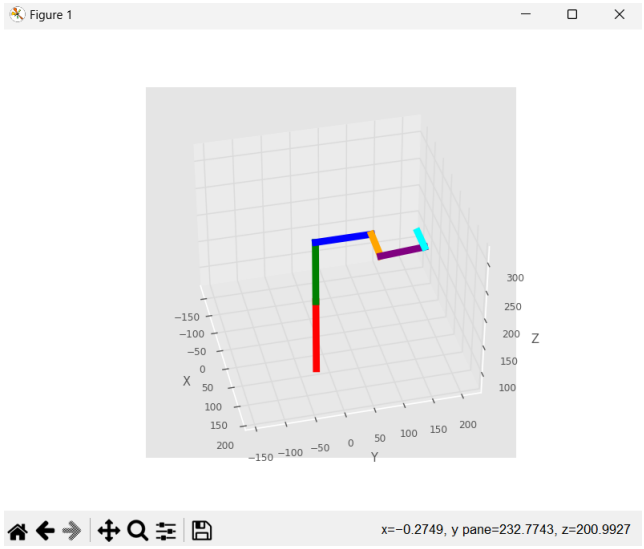
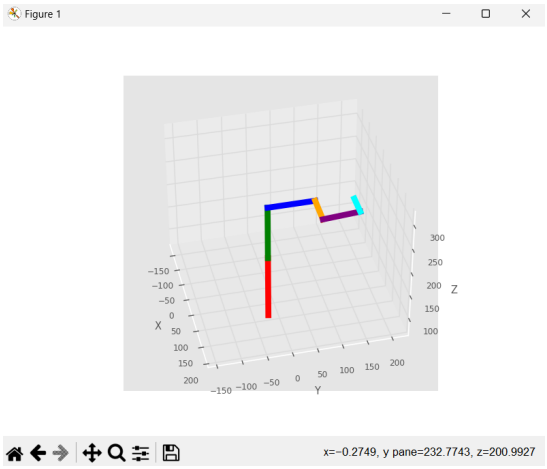


Figura 46: Segunda iteración de simulación

Teniendo definida la forma de simulación final, se procede a comparar la simulación con la posición real adoptada por el robot, esto se puede apreciar en la siguiente Figura 47. En la Figura 48 se puede ver una foto de todo el sistema junto físico, se puede observar el robot con la posición adoptada, la computadora con la interfaz y la simulación corriendo.



(a) Simulación de posición del myCobot 280 M5

(b) myCobot 280 M5 con configuración real

Figura 47: Comparación de configuración de pose del myCobot 280 M5 - simulación vs. realidad

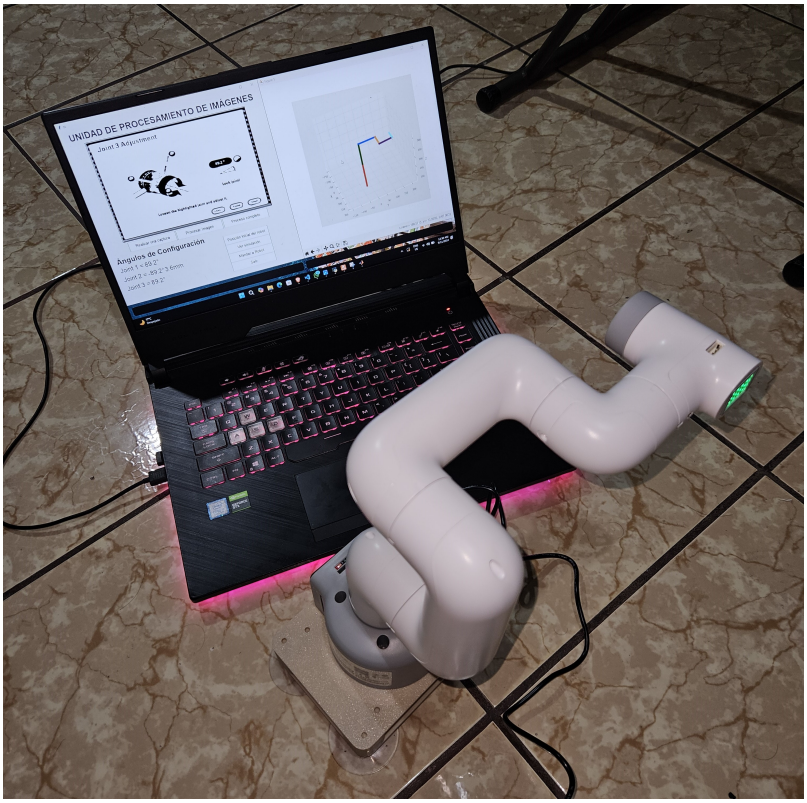


Figura 48: Visualización de todo el sistema conectado físicamente

CAPÍTULO 11

Validación del sistema completo

En este proyecto, se llevó a cabo la validación integral del sistema integrado, que incluye la herramienta de procesamiento de imágenes, el reconocimiento de caracteres, el protocolo de comunicación y la interacción con el sistema robótico físico. Este proceso tuvo como objetivo garantizar que cada componente funcione de manera correcta, tanto de forma independiente como en conjunto, bajo las condiciones previstas para su aplicación en el entorno de HUMANA.

El proceso de validación completo se detalla en el diagrama de flujo mostrado en la Figura 49, donde se describen las etapas clave de configuración y análisis. Este diagrama proporciona una visión general del flujo de trabajo, desde la inicialización del sistema hasta la evaluación final de los resultados, destacando cómo se integran y coordinan todos los componentes para cumplir con los requisitos del proyecto.

Como se mencionó anteriormente, el proyecto se divide en dos enfoques principales: el proceso de análisis y el flujo integrado. El proceso de análisis se centra en desglosar cada paso del sistema para observar su comportamiento con mayor detalle, permitiendo un examen minucioso de cada etapa. Por otro lado, el flujo integrado simula el comportamiento general del sistema al ejecutar todas las operaciones de manera continua, como ocurriría en un entorno real.

La ejecución de ambos enfoques puede observarse con mayor detenimiento en la Figura 49, donde se destacan las diferencias y complementos entre un análisis paso a paso y la integración completa de las operaciones para la validación del sistema.

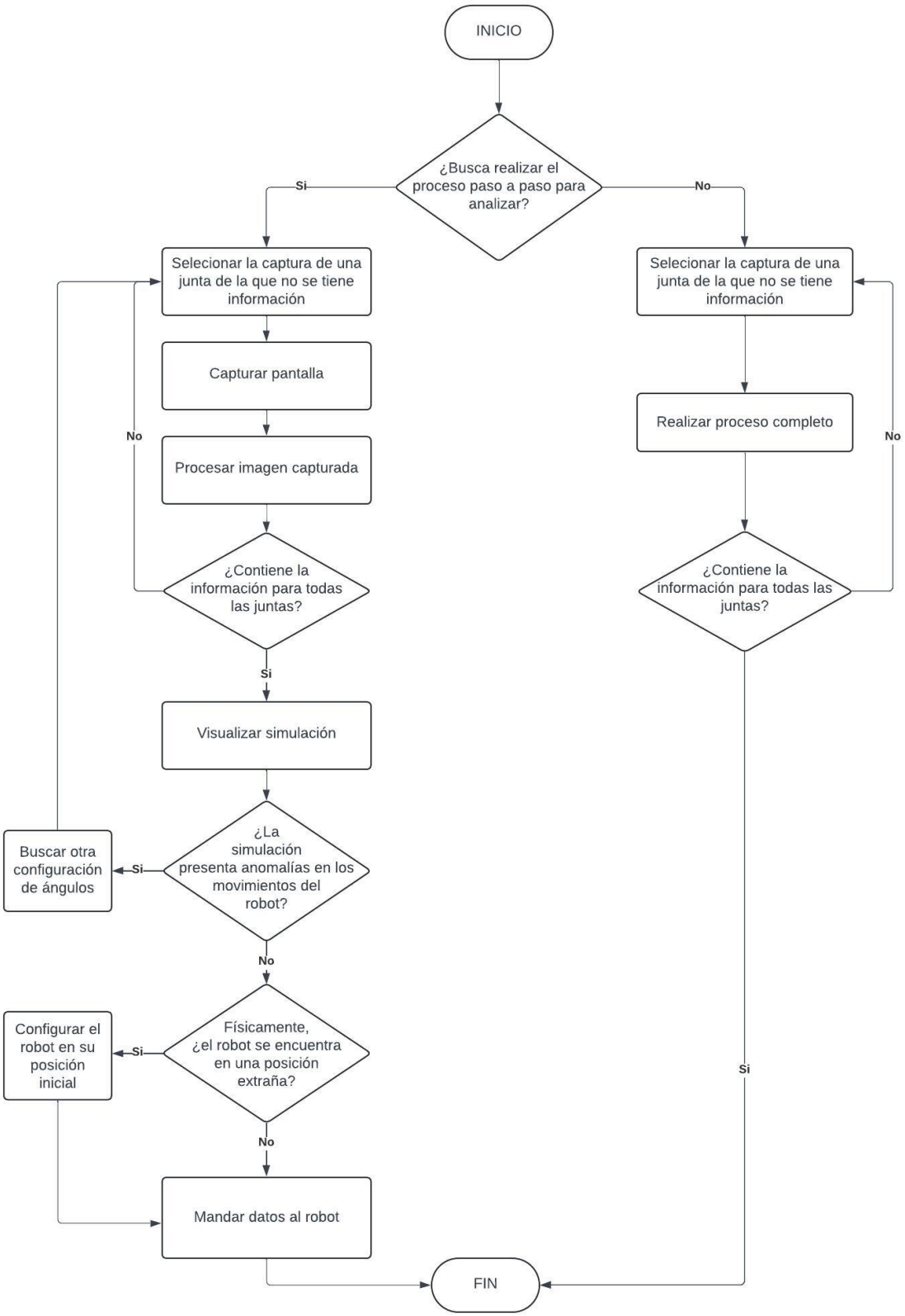


Figura 49: Diagrama de flujo para operar la unidad

Para evaluar el correcto desempeño del sistema, se realizaron pruebas exhaustivas con diferentes conjuntos de imágenes capturadas del sistema Brainlab. Estas pruebas buscaron validar la precisión del algoritmo de reconocimiento de caracteres, así como la capacidad del sistema para enviar y recibir comandos al robot de manera efectiva.

En las siguientes figuras se aprecia el apartado de interfaz, simulación y finalmente la posición física del robot. En las Figuras 50 y 51 se pueden apreciar distintas configuraciones para la junta 1.

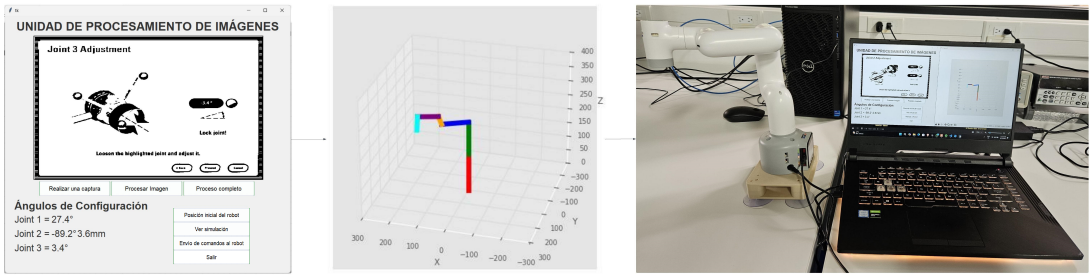


Figura 50: Primera configuración de proceso completo, primera variación de junta 1

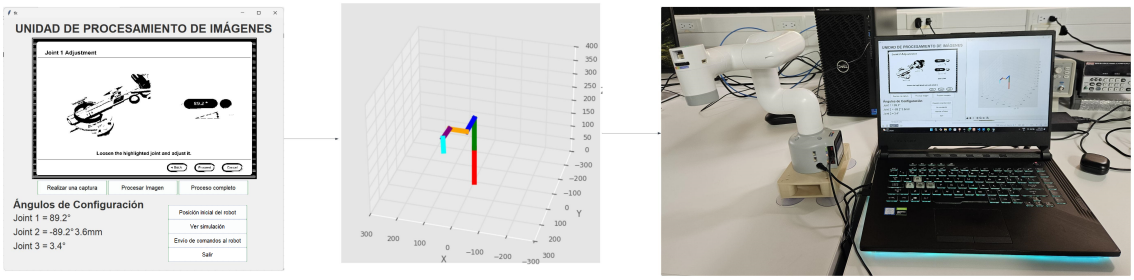


Figura 51: Segunda configuración de proceso completo, segunda variación de junta 1

Durante las variaciones, se midieron parámetros clave como el tiempo de procesamiento, la precisión en la identificación de caracteres y la respuesta del robot a los comandos generados. Los resultados indicaron que el sistema puede operar en tiempo real con un alto nivel de precisión, logrando identificar correctamente los ángulos y posiciones requeridos para el posicionamiento del robot.

Se realizó una validación funcional de la interfaz gráfica para asegurar que los usuarios puedan operar y configurar el sistema de manera intuitiva. La GUI fue probada en diferentes escenarios, incluyendo la selección y procesamiento de imágenes, el envío de comandos al robot y la visualización de resultados.

En las Figuras 52 y 53 se pueden apreciar distintas configuraciones para la junta 2.

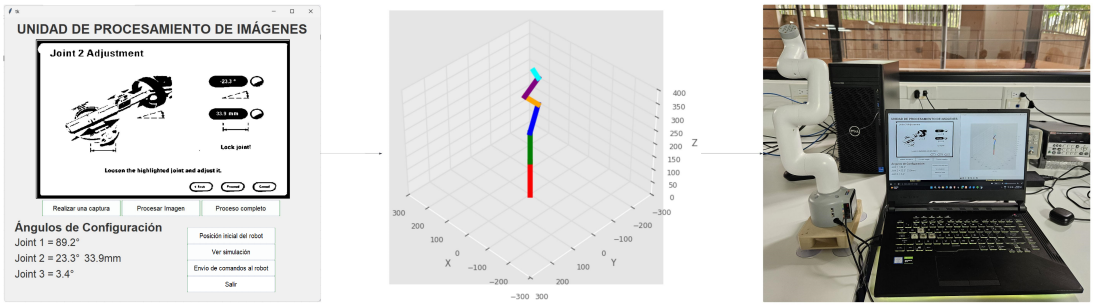


Figura 52: Tercera configuración de proceso completo, primera variación de junta 2

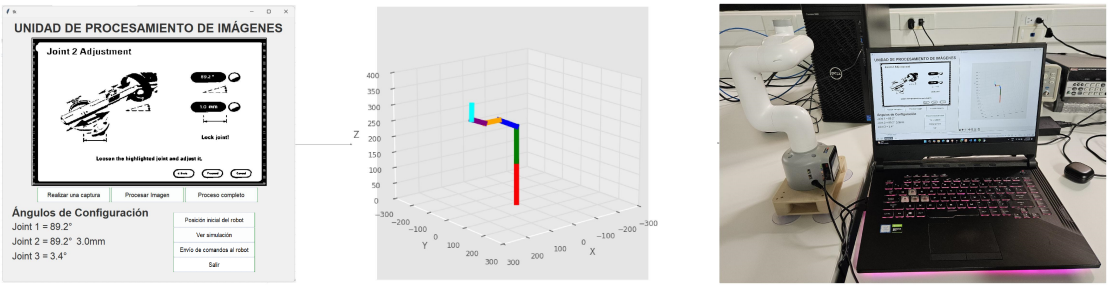


Figura 53: Cuarta configuración de proceso completo, segunda variación de junta 2

El protocolo de comunicación desarrollado se sometió a pruebas en un entorno controlado, donde se verificó la transmisión y recepción de datos entre la herramienta de procesamiento de imágenes y el robot myCobot. Los comandos fueron enviados a través de la GUI y se evaluó la respuesta del robot en función de las instrucciones recibidas. La comunicación se validó como fiable, ya que el robot ejecutó los movimientos con precisión y sin errores, evidenciando una integración exitosa entre los componentes del sistema.

En las figuras 54 y 55 se pueden apreciar distintas configuraciones para la junta 3.

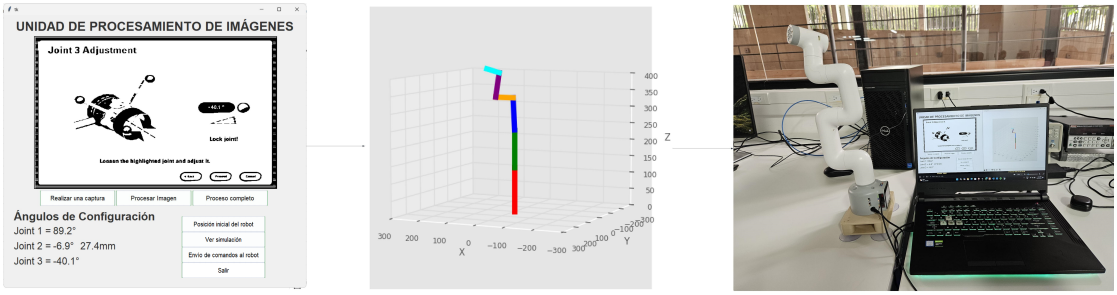


Figura 54: Quinta configuración de proceso completo, primera variación de junta 3

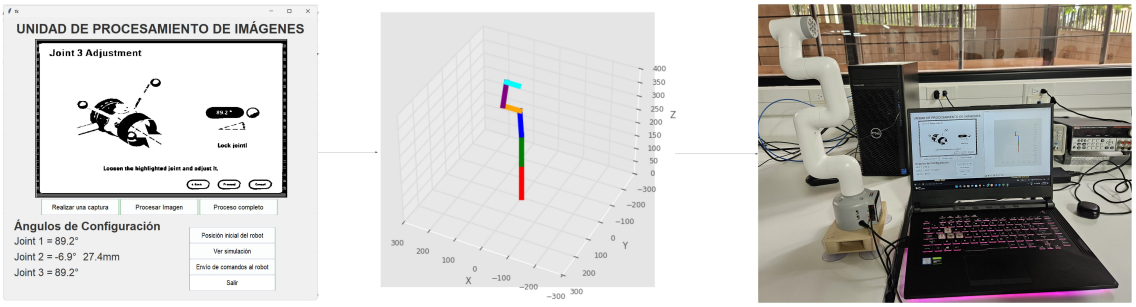


Figura 55: Sexta configuración de proceso completo, segunda variación de junta 3

La validación integral del sistema demostró que la herramienta de procesamiento de imágenes optimizada, en conjunto con la GUI y el protocolo de comunicación, es capaz de operar con alta precisión y confiabilidad. Durante las pruebas, el sistema robótico myCobot respondió adecuadamente a los comandos enviados, ajustando las posiciones de sus juntas de manera precisa y replicable. Estas pruebas incluyeron múltiples corridas, realizadas bajo diferentes condiciones de captura y procesamiento de datos, lo que permitió evaluar la robustez y consistencia del sistema.

En los ánexos 15 se incluyen los detalles completos de las corridas realizadas, mostrando resultados adicionales que respaldan la estabilidad del sistema. Estos datos incluyen gráficos que documentan el desempeño en términos de precisión, lo que evidencia la efectividad de la integración entre los algoritmos de procesamiento de imágenes, la GUI y el control robótico. Estos resultados refuerzan la conclusión de que el sistema es apto para aplicaciones prácticas.

CAPÍTULO 12

Conclusiones

- El uso de Tesseract para el OCR dentro de este proyecto fue un componente esencial para automatizar la extracción de datos textuales de la interfaz gráfica. A través de un preprocesamiento cuidadoso y una configuración personalizada, se logró un reconocimiento preciso y eficiente de los ángulos de las articulaciones del robot. Este enfoque no solo facilitó el control del robot, sino que también abrió posibilidades para futuras mejoras y aplicaciones en otros contextos donde la extracción de texto desde imágenes es requerida.
- La consistencia en la reducción de los tiempos en todas las juntas sugiere que los cambios realizados en los algoritmos no solo optimizan partes específicas del proceso, sino que han mejorado de manera global el sistema de captura y procesamiento de los ángulos. La reducción en los tiempos de procesamiento es un indicador de que se ha logrado un equilibrio adecuado entre la velocidad y la precisión, lo cual es esencial para mantener la fiabilidad del sistema sin comprometer la exactitud de los datos extraídos.
- La simplificación de la interfaz gráfica permitió que el sistema sea más accesible y eficiente. Este rediseño no solo mejoró la experiencia del usuario, sino que también redujo el margen de error humano, contribuyendo a una operación más confiable y a una curva de aprendizaje más corta. Además, las mejoras en la visualización de errores, como la notificación de capturas mal realizadas, aumentaron la robustez del sistema y facilitaron el diagnóstico y resolución de problemas durante su uso.
- La integración efectiva entre el sistema de procesamiento de imágenes y el robot myCobot 280 M5 fue uno de los logros más destacados del proyecto. Las pruebas realizadas demostraron que el sistema puede operar de manera autónoma y confiable, cumpliendo con los requisitos de aplicaciones prácticas. Este éxito también valida la flexibilidad y adaptabilidad del sistema para integrarse con otros robots en el futuro, especialmente con el brazo que se encuentra en desarrollo para simular el de HUMANA. Esto abre nuevas oportunidades de implementación en diferentes contextos.
- La validación exhaustiva del sistema bajo condiciones controladas aseguró su fiabilidad y precisión. A través de pruebas repetidas, se verificó que el sistema es capaz de manejar

variaciones en las imágenes y en las configuraciones del robot sin comprometer el rendimiento. Esto no sólo refuerza la calidad del desarrollo, sino que también asegura que el sistema esté listo para enfrentar desafíos en entornos reales.

CAPÍTULO 13

Recomendaciones

- Se recomienda implementar técnicas adicionales de preprocesamiento, como filtros adaptativos avanzados, transformaciones morfológicas o algoritmos basados en aprendizaje profundo para el mejoramiento de imágenes. Estas técnicas podrían abordar problemas como el bajo contraste, ruido o variaciones en las condiciones de captura, optimizando así la calidad de las imágenes antes de su análisis. En particular, el uso de redes neuronales convolucionales preentrenadas para el ajuste y limpieza de imágenes podría incrementar significativamente la precisión del OCR, especialmente, en escenarios donde las imágenes tienen elementos visuales complejos o signos matemáticos pequeños.
- Aunque Tesseract mostró resultados efectivos, se recomienda explorar e implementar otros motores OCR modernos, como Google Cloud Vision OCR, Microsoft Azure OCR o soluciones basadas en *deep learning* como CRNN (Convolutional Recurrent Neural Network). Estos motores podrían proporcionar mejores resultados en términos de velocidad y precisión, especialmente en configuraciones no ideales o imágenes con texto muy pequeño.
- Se recomienda realizar pruebas exhaustivas utilizando el sistema robótico final desarrollado específicamente para su integración con el sistema Brainlab de HUMANA. Este paso es crucial para validar la funcionalidad completa del sistema en un entorno más representativo de su aplicación final. Las pruebas deben enfocarse en evaluar la precisión de los movimientos, la fiabilidad en la comunicación de datos, y la capacidad del sistema para operar bajo condiciones reales de uso. Además, estas pruebas permitirán ajustar y personalizar los algoritmos de procesamiento de imágenes y el protocolo de comunicación para maximizar el rendimiento y la compatibilidad con el sistema robótico definitivo.
- Actualmente, el sistema depende de la configuración manual de coordenadas y parámetros de captura. Se sugiere desarrollar un módulo de calibración automatizada que permita al sistema adaptarse dinámicamente a diferentes pantallas, resoluciones y disposiciones sin intervención del usuario. Esto mejoraría considerablemente la experiencia del usuario y reduciría el margen de error operativo.

- Para garantizar la robustez y aplicabilidad del sistema, se recomienda realizar pruebas con una mayor variedad de imágenes y escenarios.

CAPÍTULO 14

Bibliografía

[1] A. Aranguren y T. Vela, «Sistema de seguimiento de objetos mediante procesamiento digital de imágenes aplicado al control de robots autónomos,» Tesis de licenciatura, Universidad Peruana de Ciencias Aplicadas, 2012.

[2] HUMANA, *HUMANA GT*, Accedido el 15 de marzo de 2024. dirección: `https://humanagt.org/`.

[3] Brainlab, *Brainlab*, Accedido el 15 de marzo de 2024. dirección: `https://www.brainlab.com/es/acerca-de-brainlab/`.

[4] F. E. Portales, «Optimización de la herramienta de procesamiento de imágenes para el sistema Brainlab de HUMANA - Fase III,» Tesis de licenciatura, Universidad Del Valle de Guatemala, 2023.

[5] J. J. Cifuentes, «Diseño e implementación del mando de control para brazo robótico asistencial en cirugía estereotáctica del cerebro,» Tesis de licenciatura, Universidad Del Valle de Guatemala, 2019.

[6] M. S. Juárez, «Optimización e implementación de un sistema mecánico para la automatización de un sistema Varioguide-Brainlab,» Tesis de licenciatura, Universidad Del Valle de Guatemala, 2022.

[7] J. V. Vanegas, «Rediseño y validación de los últimos 4 grados de libertad finales del brazo robótico HUMANA para cirugía asistida basado en el sistema VarioguideBrainlab,» Tesis de licenciatura, Universidad Del Valle de Guatemala, 2023.

[8] S. S. Galicia, «Optimización de la herramienta de procesamiento de imágenes para el sistema Brainlab de HUMANA,» Tesis de licenciatura, Universidad Del Valle de Guatemala, 2022.

[9] N. Aguirre, «Procesamiento de imágenes,» Proyecto de fin de carrera, Universidad Universidad de Sevilla, 2012.

[10] «Image Processing: Techniques, Types, & Aplications.» (2023), dirección: `https://www.v7labs.com/blog/image-processing-guide`.

[11] R. Q. Juan y C. M. Mario, «Redes neuronales artificiales para el procesamiento de imágenes, una revisión de la última década,» *RIEE&C, Revista de Ingeniería Eléctrica, Electrónica y Computación*, vol. 9, n.º 1, págs. 7-16, 2011.

[12] H. Trussell, E. Saber y M. Vrhel, «Color image processing [basics and special issue overview],» *IEEE Signal Processing Magazine*, vol. 22, n.º 1, págs. 14-22, 2005. DOI: 10.1109/MSP.2005.1407711.

[13] A. Koschan y M. Abidi, *Digital color image processing*. John Wiley & Sons, 2008.

[14] G. Deng y L. Cahill, «An adaptive Gaussian filter for noise reduction and edge detection,» en *1993 IEEE Conference Record Nuclear Science Symposium and Medical Imaging Conference*, 1993, 1615-1619 vol.3. DOI: 10.1109/NSSMIC.1993.373563.

[15] L. E. Sucar y G. Gómez, «Visión computacional,» *Instituto Nacional de Astrofísica, Óptica y Electrónica. México*, 2011.

[16] «¿Qué es Python?» Extraído de Amazon Web Series del apartado de herramientas para desarrolladores. (2023), dirección: https://aws.amazon.com/es/what-is/python/.

[17] A. D. R N, S. Chinta, N. K. Ashili, B. S. Babu, R. R. Vydugula y R. S. VSL, «An Intelligent Invoice Processing System Using Tesseract OCR,» en *2024 International Conference on Advances in Data Engineering and Intelligent Computing Systems (ADICS)*, 2024, págs. 1-6. DOI: 10.1109/ADICS58448.2024.10533509.

[18] «Optical Character Recognition (OCR).» (2024), dirección: https://www.egnyte.com/guides/governance/optical-character-recognition.

[19] S. Fernández, C. Javier y V. S. Consuegra, «Reconocimiento óptico de caracteres (ocr),» *Univ. Carlo*, vol. 3, n.º 7, pág. 2008, 2008.

[20] «Tesseract OCR: ¿Qué es y por qué lo deberías elegir en el 2024?» (2024), dirección: https://www.klippa.com/es/blog/informativo/que-es-tesseract-ocr/.

[21] «¿Qué es UART?» (2024), dirección: https://www.rohde-schwarz.com/es/productos/test-y-medida/essentials-test-equipment/digital-oscilloscopes/que-es-uart_254524.html#:~:text=UART%20(universal%20asynchronous%20receiver%20%2F%20transmitter,y%20recibir%20en%20ambas%20direcciones..

[22] D. Pérez, «Sistemas embebidos y sistemas operativos embedidos,» *Lecturas en ciencias de la computación. Universidad Central de Venezuela, Vols. % i de % 2lSSN*, págs. 1316-6239, 2009.

[23] P. A. L. Ramírez y H. A. Sosa, «Aprendizaje de y con robótica, algunas experiencias,» *Revista Educación*, págs. 43-63, 2013.

[24] «myCobot: 6-axis collaborative Robotic Arm.» (2024), dirección: https://www.elephantrobotics.com/en/mycobot-en/.

[25] «Robotic Kits: Robotic Arm Edge.» (2024), dirección: https://owirobot.com/.

[26] Y. Wan y Q. Xie, «A Novel Framework for Optimal RGB to Grayscale Image Conversion,» en *2016 8th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC)*, vol. 02, 2016, págs. 345-348. DOI: 10.1109/IHMSC.2016.201.

[27] A. Sugiura, K. Yokoyama y H. Takada, «Study on improvement of signal detectability considering noise characteristics in medical X-ray images,» en *2013 35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, 2013, págs. 7184-7187. DOI: 10.1109/EMBC.2013.6611215.

[28] M. A. B. Siddique, R. B. Arif y M. M. R. Khan, «Digital Image Segmentation in Matlab: A Brief Study on OTSU's Image Thresholding,» en *2018 International Conference on Innovation in Engineering and Technology (ICIET)*, 2018, págs. 1-5. DOI: 10.1109/CIET.2018.8660942.

[29] S. S. Pathak, P. Dahiwale y G. Padole, «A combined effect of local and global method for contrast image enhancement,» en *2015 IEEE International Conference on Engineering and Technology (ICETECH)*, 2015, págs. 1-5. DOI: 10.1109/ICETECH.2015.7275011.

[30] P. Corke, «A robotics toolbox for MATLAB,» *IEEE Robotics & Automation Magazine*, vol. 3, n.º 1, págs. 24-32, 1996. DOI: 10.1109/100.486658.

[31] «Robotics Toolbox.» (2024), dirección: https://petercorke.com/toolboxes/robotics-toolbox/.

CAPÍTULO 15

Anexos

15.1. Repositorio en GitHub

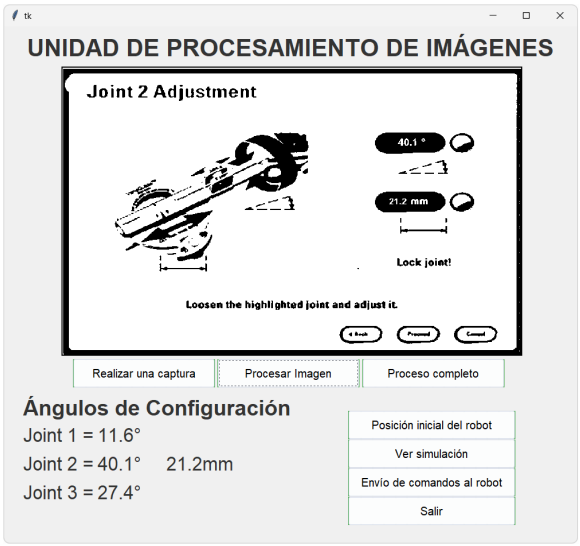
Para ver el código del proyecto y más información, visitar el siguiente repositorio de GitHub:

`https://github.com/SergioBoch/Trabajo_de_graduacion_Unidad_de_Procesamiento_de_imagenes`

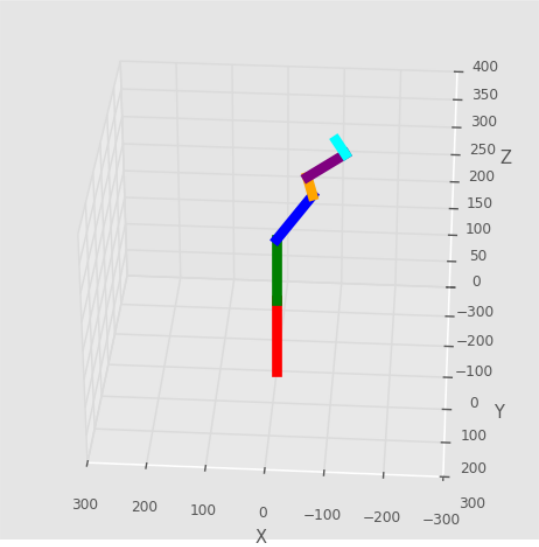
15.2. Distintas configuraciones de ángulos

En esta parte se pueden observar más configuraciones de ángulos para el sistema róbotico. Como tal, se verán tres subfiguras para cada figura principal, donde primero podremos ver la interfaz gráfica que nos muestra la configuración de ángulos extraída después de realizar el proceso de reconocimiento de caracteres. En la segunda subfigura se podrá ver la simulación de la configuración de ángulos obtenida y, finalmente, en la última subfigura, se podrá ver al robot físicamente adoptando la configuración de ángulos establecida.

Como se podrá ver en cada figura, la simulación y la versión física presentan el mismo comportamiento para el myCobot 280 M5.



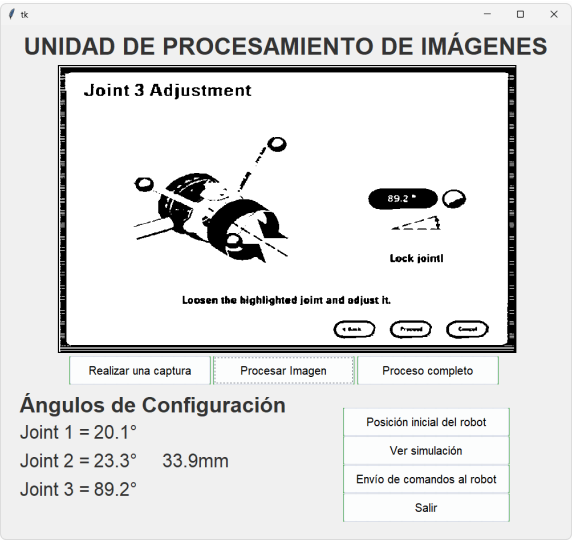
(a) Interfaz gráfica con información obtenida



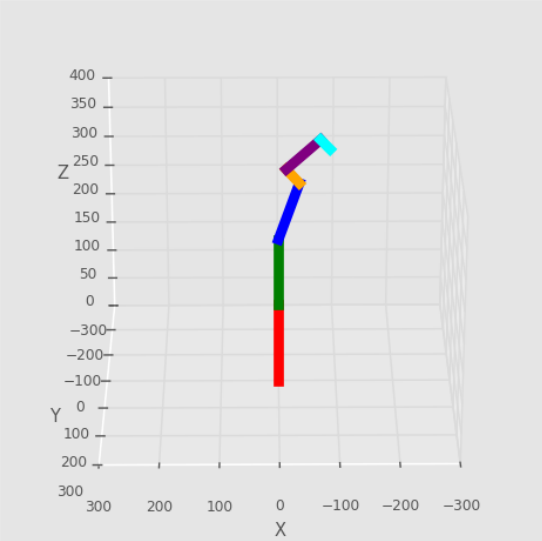
(b) Simulación de la configuración obtenida

(c) myCobot 280 M5 con configuración de ángulos

Figura 56: Primera configuración de ángulos



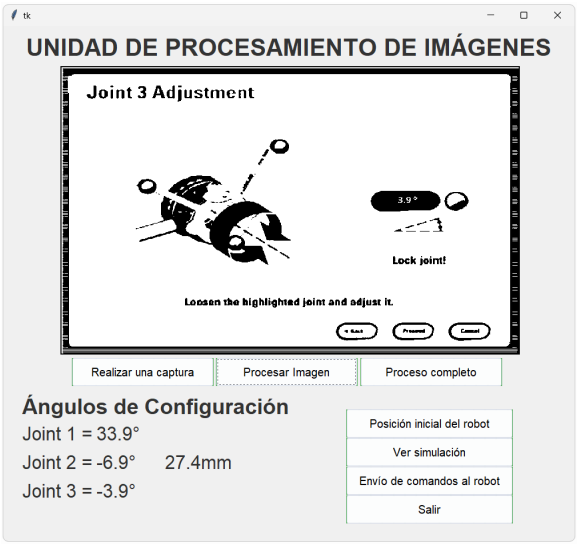
(a) Interfaz gráfica con información obtenida



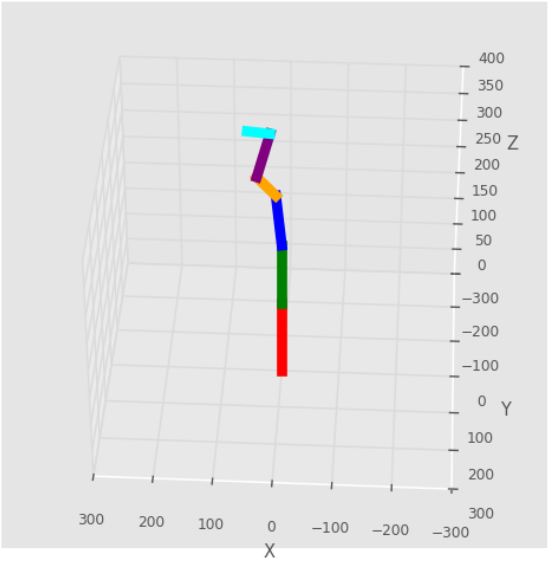
(b) Simulación de la configuración obtenida

(c) myCobot 280 M5 con configuración de ángulos

Figura 57: Segunda configuración de ángulos



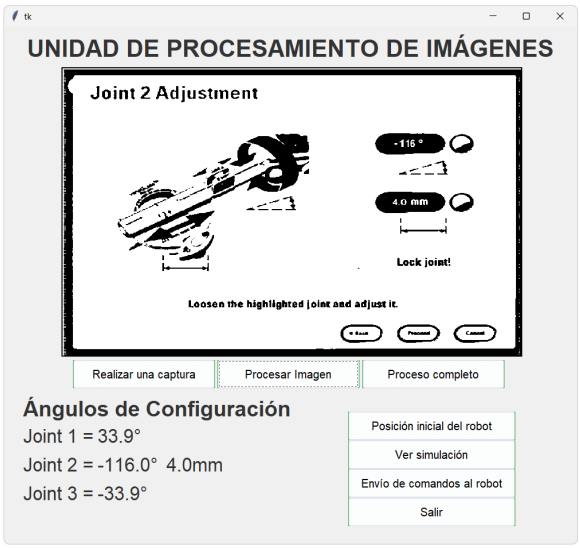
(a) Interfaz gráfica con información obtenida



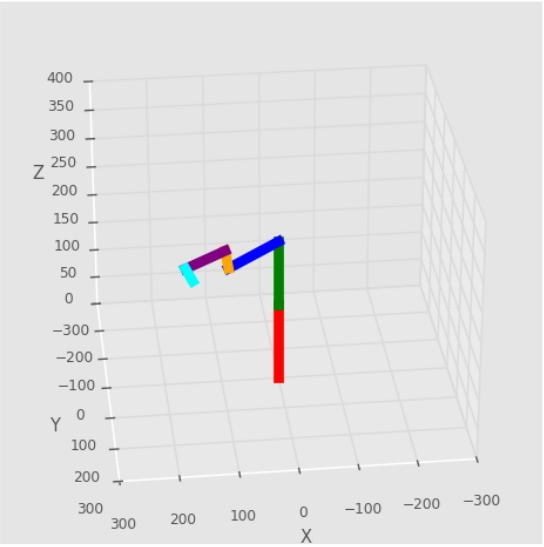
(b) Simulación de la configuración obtenida

(c) myCobot 280 M5 con configuración de ángulos

Figura 58: Tercera configuración de ángulos



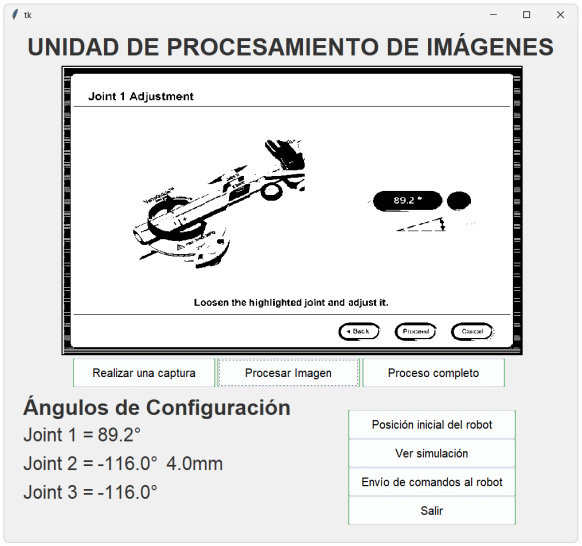
(a) Interfaz gráfica con información obtenida



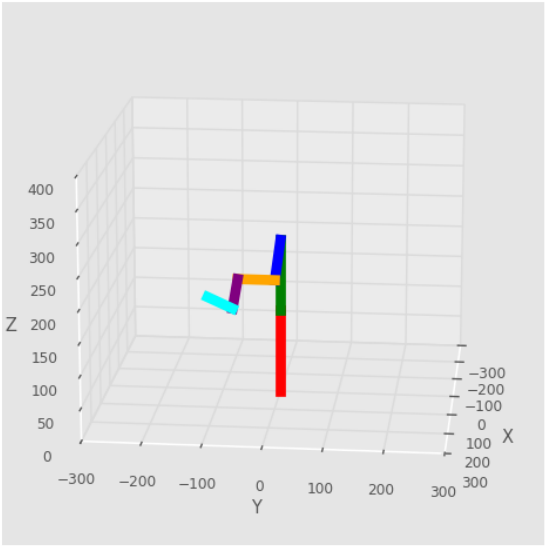
(b) Simulación de la configuración obtenida

(c) myCobot 280 M5 con configuración de ángulos

Figura 59: Cuarta configuración de ángulos



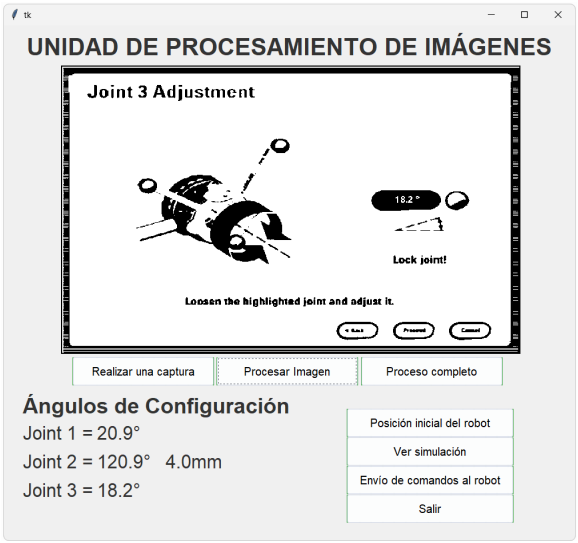
(a) Interfaz gráfica con información obtenida



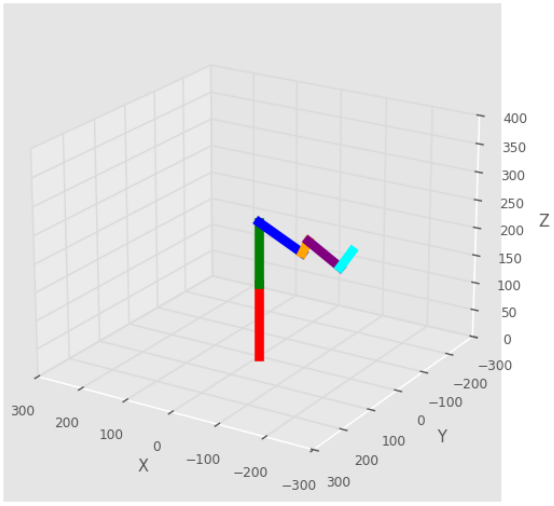
(b) Simulación de la configuración obtenida

(c) myCobot 280 M5 con configuración de ángulos

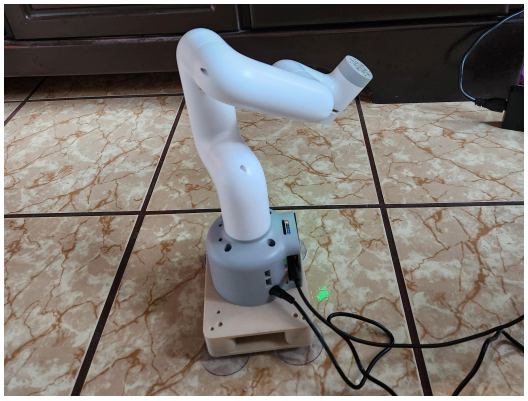
Figura 60: Quinta configuración de ángulos



(a) Interfaz gráfica con información obtenida

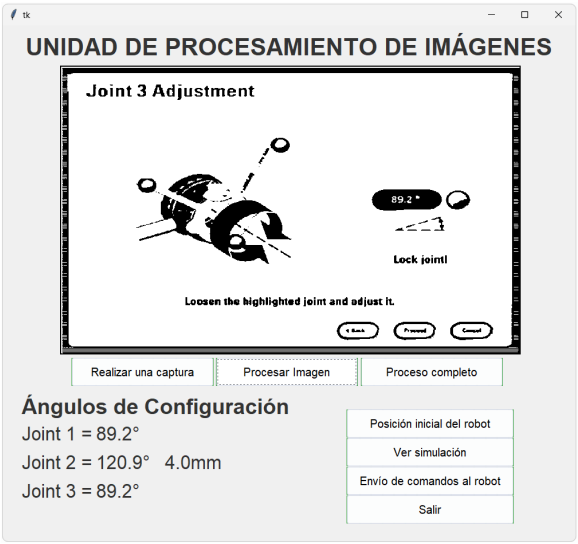


(b) Simulación de la configuración obtenida

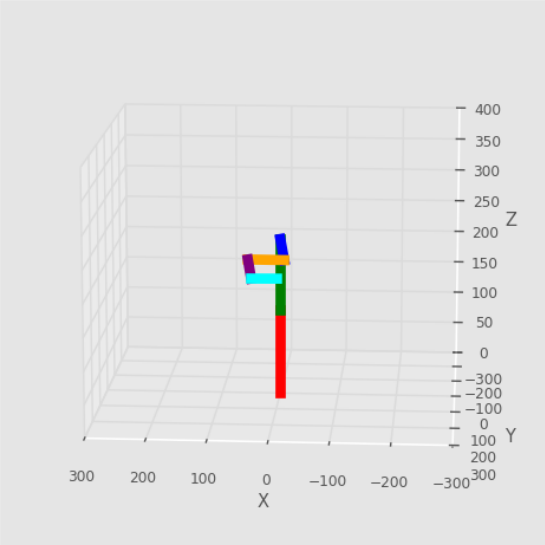


(c) myCobot 280 M5 con configuración de ángulos

Figura 61: Sexta configuración de ángulos



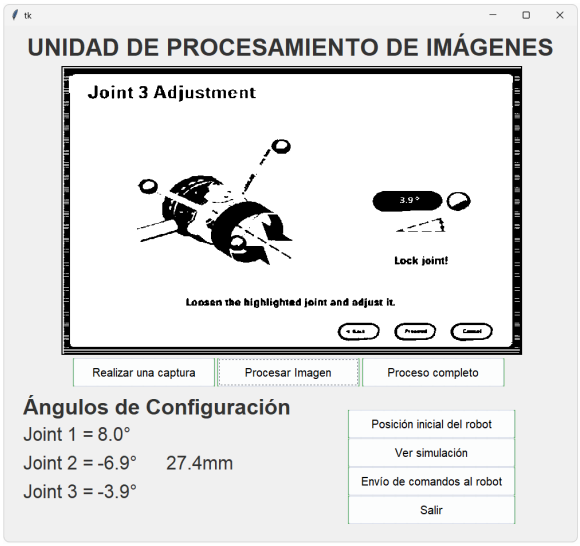
(a) Interfaz gráfica con información obtenida



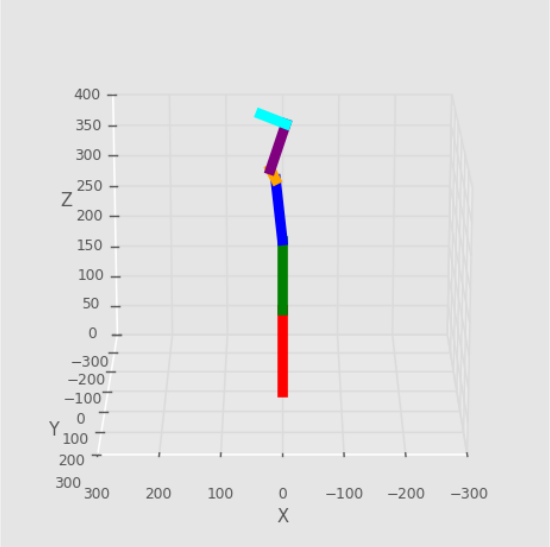
(b) Simulación de la configuración obtenida

(c) myCobot 280 M5 con configuración de ángulos

Figura 62: Séptima configuración de ángulos



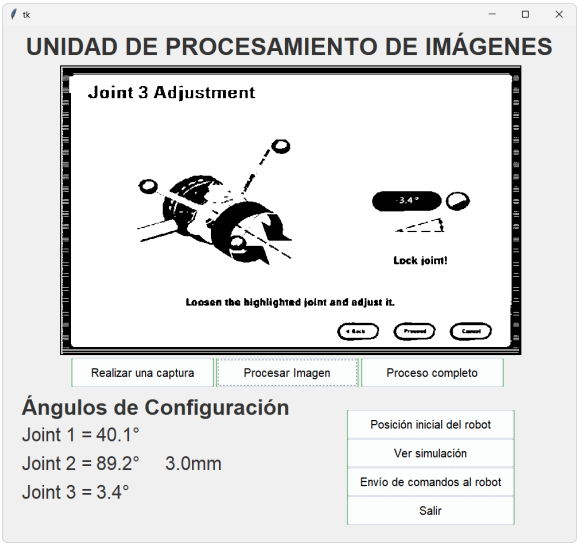
(a) Interfaz gráfica con información obtenida



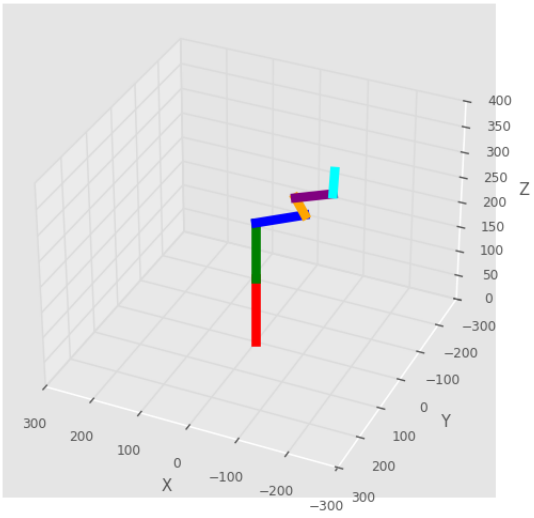
(b) Simulación de la configuración obtenida

(c) myCobot 280 M5 con configuración de ángulos

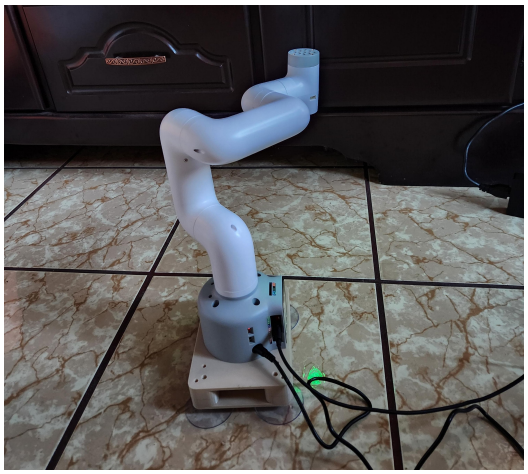
Figura 63: Octava configuración de ángulos



(a) Interfaz gráfica con información obtenida

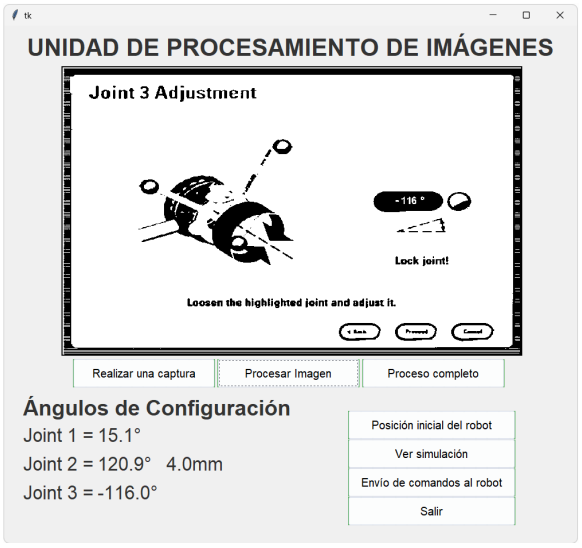


(b) Simulación de la configuración obtenida

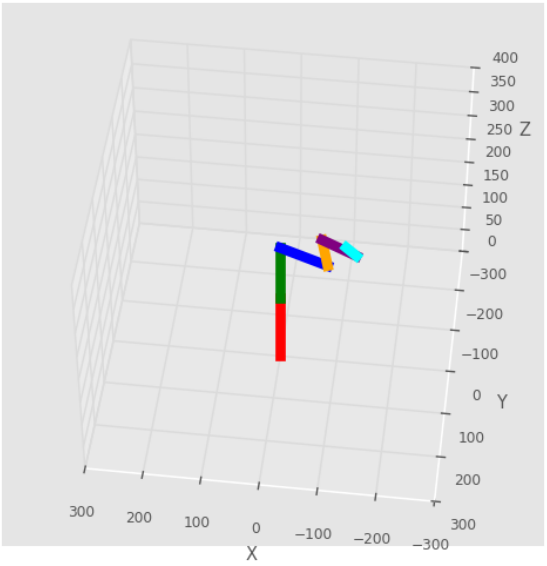


(c) myCobot 280 M5 con configuración de ángulos

Figura 64: Novena configuración de ángulos



(a) Interfaz gráfica con información obtenida



(b) Simulación de la configuración obtenida

(c) myCobot 280 M5 con configuración de ángulos

Figura 65: Décima configuración de ángulos

CAPÍTULO 16

Glosario

Juntas: es el punto de conexión entre dos partes de un robot que permite movimiento relativo entre ellas. 23

Programabilidad: es la capacidad de un sistema, dispositivo o software para ser programado o configurado para realizar tareas específicas. 22

Repetibilidad: es la proximidad de la concordancia entre los resultados de mediciones sucesivas del mismo mensurando realizadas en las mismas condiciones de medición. 22