CONTROL DE COMPUTADORA VÍA VISIÓN ARTIFICIAL CON LABVIEW

J.D. Amaya-Ramírez[1*], V. Manuel-Ramírez[1], y M. Mejía Carlos[1]
[1]Instituto de Investigación en Comunicación Óptica, Universidad Autónoma de San Luis Potosí, Av. Karakorum, San Luis Potosí, 78210, México
*Contacto: A322556@alumnos.uaslp.mx

ABSTRACT—Se diseña e implementa un sistema alternativo para el control de un mouse de una computadora a través de la detección de movimientos de una mano en un área reducida. El desarrollo del sistema se hace en el software de Labview con ayuda de sus librerías enfocadas a la visión artificial (Machine Vision) para el control de las operaciones. Además, para la ejecución de las operaciones se usará la utilidad de línea de comandos NirCmd. El algoritmo responde correctamente a los comandos establecidos, aun así, el sistema mantiene el potencial disminuir los tiempos de respuesta y mejorar la precisión de la detección de colores minimizando las perturbaciones en la lectura.

Palabras Clave—Control, Detección, LabVIEW, Sistema de gestos, Visión Artificial, *Vision Assitant*

I. OBJETIVO

El objetivo de este proyecto es el diseño e implementación de un sistema alternativo para el control de un mouse de una computadora a través de la detección de movimientos de una mano en un área reducida mediante el uso de LabView y su librería de visión artificial *Vision.*

II. INTRODUCCIÓN

Implementar algoritmos que usan visión artificial es una operación compleja independientemente del campo. Estos deben ser diseñados para ser robustos, precisos y tener un bajo tiempo de respuesta para hacer frente a los escenarios que el problema presente.

Entre todas las líneas de desarrollo para nuevas aplicaciones aparecen los sistemas de control. Específicamente, el dotar a las máquinas de un sistema de visión capaz de convertir los movimientos y/o gestos del cuerpo humano en sus mecanismos de control.

En este trabajo se plantea y desarrolla el uso de la visión artificial para controlar ciertos parámetros que existen dentro de una computadora como el control del ratón con sus respectivas funciones junto con el control de volumen y brillo, usando gestos y movimientos de una mano, y la detección de colores presentes en la imagen muestreada. Lo anterior busca ser una alternativa para las personas con lesiones físicas y/o neuronales que limitan la movilidad total de las extremidades superiores y por ende también se ve limitada su capacidad de adaptar posturas adecuadas para el trabajo en un escritorio. De ahí que, uno de las cosas que siempre estuvo presente durante el desarrollo del proyecto es el control en un área pequeña de trabajo. La sección *IV* y *V* explica el los materiales usados y el desarrollo del programa, respectivamente. Mientras que en la sección *VI* plantea la configuración de pruebas a realizar con el programa desarrollado. Finalmente, las secciones *VII* y *VIII* muestran los resultados y conclusiones, junto con comentarios finales.

III. MARCO TEÓRICO

La visión artificial o visión por computador intenta emular la capacidad de algunos seres vivos para ver una escena (imagen), entenderla y actuar en consecuencia. Sus algoritmos se basan en entrenar máquinas para realizar funciones, se pretende que la función permita hacer una tarea óptica en mucho menos tiempo usando cámaras, datos y algoritmos en lugar de retinas, nervios ópticos y una corteza visual humana. [1]

La visión artificial funciona de manera muy similar a la visión humana, excepto que los humanos tenemos una ventaja inicial: el contexto informativo que hemos adquirido a lo largo de nuestra vida. Esto es a lo computacionalmente llamamos “entrenamiento” [2].

La implementación de sistemas computacionales basados en visión artificial es una de las novedades de los tiempos actuales, se han presentado múltiples softwares que han evolucionado su interfaz con redes neuronales basadas en la interpretación de imágenes, aunque existe una gran variedad de opciones para diseñar estos softwares, hay algunos que están más desarrollados tecnológicamente, como lo es el caso de LabVIEW [3].

Este cuenta con una extensión especializada en la visión artificial llamada *Vision Development* Module que proporciona una biblioteca extensa de funciones que permite acceder a cientos de algoritmos de procesamiento de imágenes y funciones para mejoramiento de imágen, verificación de presencia o ausencia de objetos, detección de patrones, entre otras. Adicionalmente, incluye una herramienta de diseño de algoritmos que simplifica el diseño del sistema de *visión* llamada *Vision Assistant*, donde solo es necesario seleccionar el procesamiento que se desea realizar desde una lista de opciones y configurar los parámetros correspondientes de manera interactiva en lugar de escribir líneas de código [4-6].

Color	Azul	Amaillo	Verde	Rosa	Rojo	Naranja
Color Sensivity	Low	Medium	Low	Medium	High	High
Search Strategy	Agressive	Agressive	Agressive	Agressive	Agressive	Agressive
Min . Match Score	650	600-650	400	400	600-650	300
N. of Matches	1	1	1	1	1	1

Tabla 1: Valores de entrada para los subVIs de detección.

IV. MATERIALES Y MÉTODOS

El sistema consta de una cámara de celular, apuntada al área de detección, sostenida en un trípode y conectada vía alámbrica a una capturadora de video *elgato HD60 S,* que a su vez se encuentra conectada a la computadora y transmite la imagen al VI de *LabVIEW*. Para el apartado del control con la mano es usado un guante de color negro y 6 bandas de colores diferentes, por lo anterior, el área de detección consiste en una superficie de color negro para garantizar el máximo contraste entre el entorno y los colores.

Antes del uso del programa para el control de la máquina, es necesario evaluar la detección de cada uno de los colores por separado. Por lo que deben deshabilitarse los subVIs de control

Los detalles del funcionamiento de cada función de control se explican en la siguiente sección.

V. FIGURAS Y DIAGRAMAS

A. Creación de subvis de detección de colores

Los SubVIs fueron armados el con módulo de desarrollo de *Vision* (o sus siglas en ingles, VDM) de LabVIEW, el cual está diseñado para la asistencia en la creación de aplicaciones de visión artificial (*machine vision*) y con *Vision Assistant* que es una herramienta de ingeniería de algoritmos que simplifica el diseño de los sistemas de *Vision*.

En *Vision Assitant*, lo primero que se hizo es capturar en imágenes las bandas de colores (*Adquire Image*) para posteriormente hacer el *script* que procesará la imagen (*Process Image*). En este, únicamente se añadió la función de procesamiento *Color Location* la cual localiza el color especificado en una imagen a partir de una plantilla (*template*) que se crea conforme se configura la función. Finalmente, se creó el VI optimizado para ser optimizado en paralelo con la opción que ofrece *Vision Assitant: Create LabVIEW VI* y se le asignan los controles e indicadores que se muestran en la Figura 1.

B. Constitución de vi de detección de colores

El VI encargado del funcionamiento consta de un único ciclo *while* en el que está contenido el resto del programa. Antes de entrar al ciclo es necesario inicializar la cámara y configurarla adecuadamente, también se debe crear una localidad de memoria temporal para las imágenes que se vayan capturando y modificar su resolución a una deseada que en este caso es de 640 x 480 (Figura 2).

Conforme la imagen es capturada esta es enviada a los subVIs de detección de colores que cuentan con los parámetros de entrada y salida que se muestran en la Figura 1. Los valores de entrada de todos los subVIs de detección se muestran en la Tabla 1, aunque el valor de *Match Score* puede tener ligeras variaciones ($\mp$ 100 puntos) dependiendo de las condiciones de iluminación en las que se ejecute el programa. En la figura 3, se muestra el patrón de conexión de los 6 subVIs, cada uno de ellos se distingue por el color de fondo del ícono que es el color que detectarán (Figura 4).

La salida de la imagen del subVI (*Image*) se despliega en el panel frontal. Para la inicialización de cada operación de control de la computadora es necesario un valor booleano por lo que el número de coincidencias requeridas (una de cada color) es comparada con el número de coincidencias encontradas mediante el comparador *Equal?* y de ser iguales este devolverá un *TRUE*, de lo contrario entregará un *FALSE*.

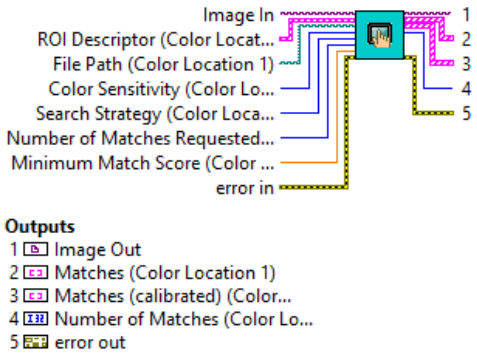


Figura 1: Configuración de entradas y salidas (Outputs) del SubVI para la detección de color

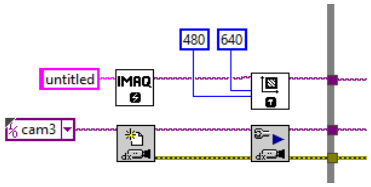


Figura 2: Inicialización de cámara

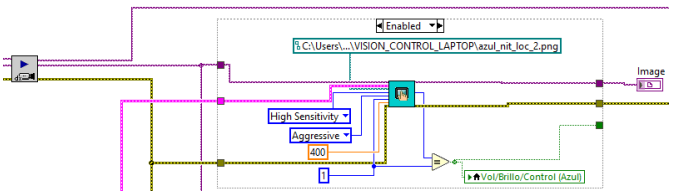


Figura 3: SubVI para deteccion de color, en este caso es para el color azul

Funcion	NirCMD (nircmd.exe)	[Argumentos | Opciones]
Mouse		
Botón Derecho	sendmouse	[right] [down | up | click | dblclick]
Botón Izquierdo	sendmouse	[left] [down | up | click | dblclick]
Rueda	sendmouse wheel	Velocidad contante de desplazamiento [0,100]
Mover Ratón	setcursor	Coordenadas [X] [Y]
Parametros de Computadora		
Brillo	setbrightness	Valor de Brillo [0,100]
Volumen	setvolume	Valor de Brillo [0,100]

Tabla 2: Funciones de NirCMD

C. NirCMD EN LabVIEW

NirCMD es una utilidad de línea de comandos la cual se ejecuta en la consola de la computadora y con ayuda de sus comandos es posible controlar funciones de la misma. En su uso para LabVIEW fue necesario la función (.vi) de *System Exec*, la cual, ejecuta comandos del sistema dentro del VI que se use. Posteriormente, para el armado de comandos para cada una de las funciones se utilizó únicamente la manipulación de cadenas. Las funciones que realizan los SubVIs con sus respectivos comandos de NirCMD en System Exec se muestran en la tabla 2.

Figura 4: Iconos de los SubVIs de detección de color

Control de Mouse (SubVI_mouse_nircmd.vi): Este subVI es el encargado del control del mouse y de sus funciones (Tabla 2). Como se explicó anteriormente, el núcleo del programa consiste en manipular cadenas para construir los comandos que serán enviados a la terminal. A nivel de implementación en el VI principal, dependiendo de la operación a realizar serán las entradas a usar (Figura 5) y que se active el switch para enviar la operación (*Enviar*).

En el caso de los botones (*boton*) es necesario especificar si es un clic izquierdo o derecho (*Der/Izq*) y si se "presiona" el botón o se suelta para los casos de una acción sostenida (*up/down*) o, si es un clic simple o doble de tratarse de una acción rápida (*clic/2clic*). Esto última se especifica mediante un *switch* booleano (*toque (T) / sostener(F)*) .

Para el caso, de la operación para mover el ratón (*raton mover*) únicamente es necesario enviar el par de coordenadas (x,y). Recordemos que el punto de origen del planto cartesiano de una computadora es la esquina superior izquierda.

Por último, en el caso de la rueda del mouse (*rueda mover*), esta solo recibirá el valor de la velocidad al que se desplazará hacia abajo o hacia arriba la pantalla.

Control de Brillo y Volumen (control_brillo.vi y control_volumen.vi): Estos subVIs son los encargados del control del brillo y del volumen. A diferencia del subVI anterior la manipulación de cadenas para construir los comandos es mucho mas simple pues solo es necesario modificar un valor (Figura 6 y 7). A nivel de implementación en el VI principal, únicamente es necesario un valor de entradas a usar en cada subVI y que se active su respectivo switch para enviar la operación (*Enviar*).

En todos los SubVI mencionados sus salidas, que únicamente son indicadores, no son utilizadas en el producto para el usuario pues carecen de importancia fuera de procesos de *debugging*.

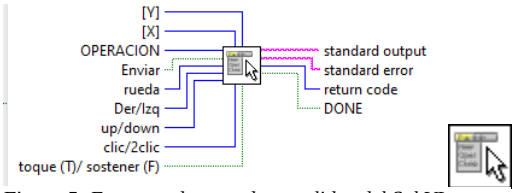


Figura 5: Esquema de entradas y salidas del SubVI para control de mouse (izquierda) y su ícono (derecha).

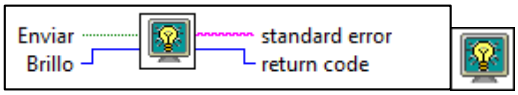


Figura 6: Esquema de entradas y salidas del SubVI para control de brillo (izquierda) y su ícono (derecha).

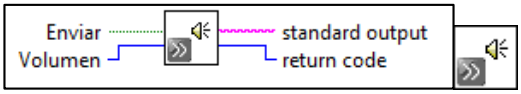


Figura 7: Esquema de entradas y salidas del SubVI para control de volumen (izquierda) y su ícono (derecha).

D. Programa principal

El funcionamiento general del programa se muestra en el Esquema 1. Como se explicó en la sección V-A, en cada subVI de detección de color se cuenta con un comparador booleano que indica si se detectó el color o no. La señal que este emita está directamente conectada a estructuras *case* en las que están contenidas las operaciones de control y, dependiendo de esta y de otras señales que a continuación serán descritas se determinará si se ejecuta la operación o no.

A partir de las funciones creadas y las necesidades del sistema diseñado se determinó que era necesario el uso de 6 colores, cada uno de ellos encargado de una tarea específica:

- Azul: Control de activación de botones de mouse.

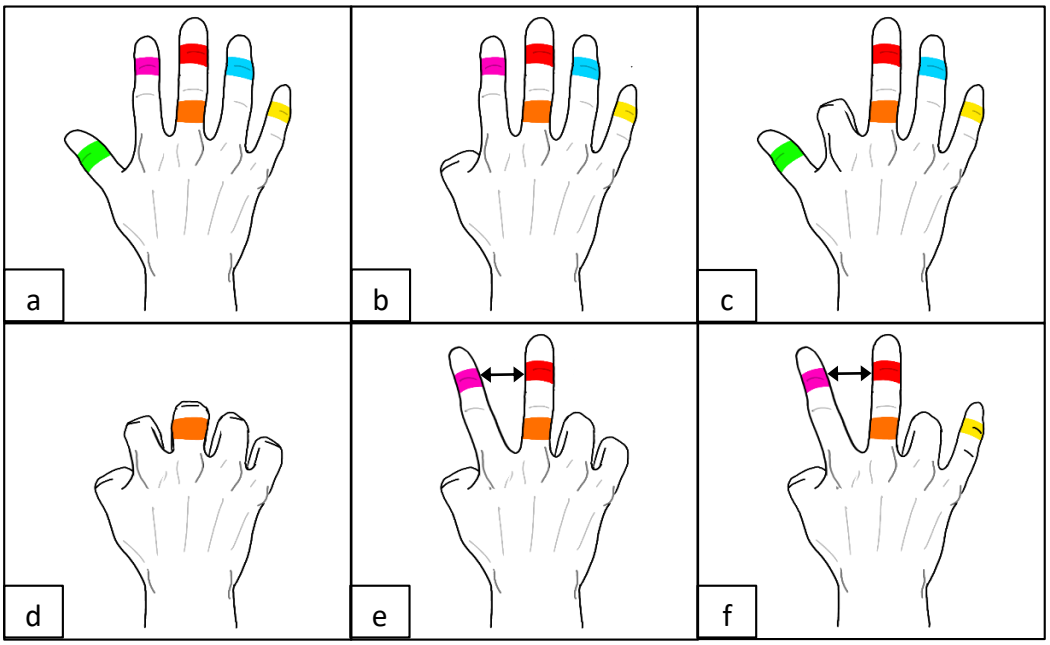
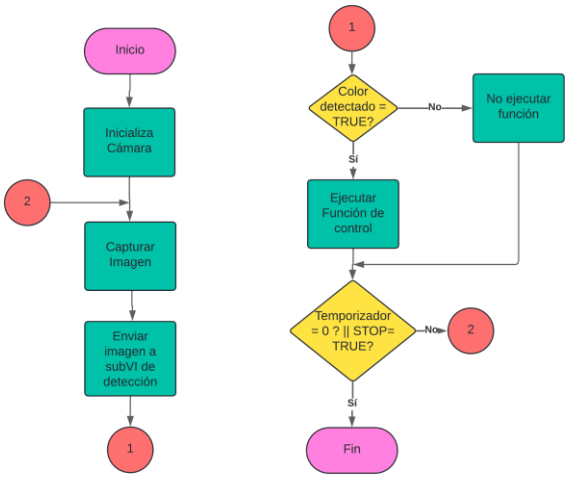


Ilustración 1: Sistema de gestos implementado. (a) Mano estática, ninguna función ejecutada más allá del control del movimiento del mouse. (b) Gesto para realizar un clic izquierdo. (c) Gesto para realizar un clic derecho. (d) Gesto para controlar la rueda. (e) Gesto para controlar el brillo. (f) Gesto para controlar el volumen.

- Amarrillo: Selector entre control de volumen y de brillo.
- Naranja: Control de rueda del mouse.
- Rojo: Control de posición de mouse.
- Rosa: Control de botón derecho.
- Verde: Control de botón izquierdo.

Aun así, se implementa un sistema de gestos para que el programa funcione de manera fiable y que a su vez sea de cómoda realización para el usuario (Ilustración 1).



Esquema 1: Diagrama de flujo del programa en general

A continuación, se describirán las funciones a nivel de programa basándose en el sistema de gestos mostrado previamente. Para los casos de funciones fuera de mover el ratón y de activador de botones, hecho de que el color sea detectado constantemente es un indicativo de que ningún equivalente a su acción mecánica se está llevando a cabo, es decir, es como si no se hubiese accionado algún botón o *switch* (Ilustración 1a) o que se "desactive" la realizada.

Control de botones y switch de funciones: Para evitar que se pierda el control total de la computadora en caso de algún error en el programa se añadieron múltiples controles a manera de seguros y "frenos de emergencia". El color azul funciona como un habilitador de los botones del mouse, es decir, si este color no es detectado, no se ejecutarán las funciones para ambos botones del mouse. Mientras que el color amarillo funciona como un *switch* entre las funciones de control de brillo y control de volumen como se explica más adelante (Ilustración 1f).

Botones del mouse: Esta parte tiene los colores verde y rosa asignados para los botones izquierdo y derecho, respectivamente. En el momento en que el color no sea detectado, esto se considerará como un clic del botón que se le tenga asignado (Ilustración 1b y 1c). En el caso específico de la acción mecánica de un clic izquierdo sostenido, es necesario que el color verde no sea detectado por más de 2 segundos para que se active esta función (Ilustración 1b y 1c).

Ratón del mouse: La función tiene asignado el color rojo, el cual, al contrario del resto de colores este debe ser detectado constantemente pues las coordenadas en las que sea detectado son enviadas a la función para mover el mouse en la pantalla. Aquí surge el problema de que la imagen transmitida no cuenta con suficiente densidad de píxeles para que el movimiento abarque toda la pantalla por lo que se deben normalizar las coordenadas antes de que sean ingresadas al subVI para que ambas sean equivalentes ambos sistemas (Ilustración 1a). La normalización puede variar dependiendo de la configuración y montaje del sistema.

Rueda del mouse: La función tiene asignado el color naranja. Para su activación, únicamente debe ser detectado su color, mientras que el resto no. Si el color se encuentra en la mitad superior del área de detección, la acción será como si la rueda fuera desplazada hacia adelante o arriba. En cambio, si el color se encuentra en la mitad inferior del área de detección, la

acción será como si la rueda fuera desplazada hacia atrás o abajo (Ilustración 1d).

Brillo y Volumen: La función trabaja con 3 colores al mismo tiempo. En el caso del brillo, para su activación se necesita que únicamente se detecten los colores rojo y rosa (el naranja resulta despreciable en esta operación). A continuación, se realizará una resta entre las coordenadas X en donde se localicen los colores, y si el resultado se encuentra dentro de un rango de 60 a 160 unidades, entonces se le restará 60 unidades a este valor antes de ser enviado al subVI de control de brillo. En el caso del volumen, esta dinámica se repite con la diferencia de que es necesario que también sea detectado el color amarillo (Ilustración 1e y 1f).

VI. CONFIGURACIÓN DE PRUEBAS

En base a todo el planteamiento y desarrollo realizado, se realizaron pruebas de funcionamiento continuo en las que se usó el sistema para controlar la computadora simulando acciones de un usuario común como cambiar entre ventas, leer documentos, dibujar en *paint*, subir o bajar el volumen de un archivo multimedia, cambiar el brillo de la pantalla, etc.

VII. RESULTADOS Y DISCUSIONES

El programa resultante se muestra la plataforma de *GitHub* [7]. En la ejecución del programa, lo primero que sobresaltó es la importancia del tipo de iluminación que reciben las imágenes. Debido a que las *templates* que se generaron para los subVIs de *Vision Assistant* fueron configuradas con luz blanca que se proyectaba de manera homogénea sobre toda la banda de color, al surgir alteraciones en la iluminación en las zonas de prueba afectó la detección de los colores por lo que fue necesario recalibrarlos. Después de este proceso, el programa mostró una capacidad sobresaliente en su tiempo de respuesta de las funciones. En el control del movimiento del mouse apareció cierta inestabilidad que se ve reflejada en "saltos" que realiza el mouse en la pantalla a pesar de que el color esté en una posición estática. Esto es consecuencia de la densidad de píxeles de la cámara, la resolución de la pantalla y el proceso de normalización entre estos dos aunque estas pueden ser minimizadas con pequeñas correcciones en este último que dependerán de los parámetros de los primeros dos.

La estabilidad del control del brillo y volumen resultó ser mayor de la esperada pues esperaban inestabilidades similares a las del control del mouse pero, durante las pruebas los "saltos" resultaron ser lo suficientemente pequeños como para que los controles sean considerados estables.

En el resto de funciones y colores estos tipos de inestabilidades que se reflejan en las coordenadas resultan ser despreciables pues solo es importante que se detecte la presencia o ausencia del mismo, pues ninguna función opera con sus posiciones.

Los resultados de las pruebas, tras las correcciones en los procesos previamente descritas, muestran un funcionamiento óptimo del programa que además, no muestra errores o retrasos en la respuesta a pesar de estar operando con múltiples subVIs de detección y de funciones en paralelo.

VIII. CONCLUSIÓN

Este trabajo presenta y evalúa el desempeño de una alternativa para el control de la computadora a través de los algoritmos la visión artificial que ofrece *LabVIEW*. Los tiempos de respuesta sigue siendo un problema a pesar de que LabVIEW intrínsecamente trabaja las tareas en paralelo debido a que las funciones establecidas requieren de una gran cantidad de procesamiento.

En el sistema resultante, la velocidad de análisis de una imagen completa en busca de colores, específicamente, la comparación con los estándares establecidos que rigen un determinado color RGB es directamente proporcional a la cantidad de pixeles de información contenidos en la muestra final; no se puede tomar a la ligera el tamaño de la imagen pues la precisión de los resultados corresponden a la cantidad de información registrada. Se espera que usando una técnica de compresión de imagen con una pérdida de datos tolerable pueda ser la mejor de las opciones para la optimización del algoritmo.

Además, el sistema de lectura de los gestos tiene el potencial de ser mejorado para la participación en la rehabilitación de pacientes con alguna dificultad en sus manos, sustituyendo este por un sistema basado en la detección y seguimiento de objetos encargado de buscar la forma de la mano e identificar los dedos. Esta es una actualización recomendada para futuras implementaciones que buscan un estándar muy alto de eficiencia, precisión y practicidad.

IX. REFERENCIAS

[1] García, I., & Caranqui, V. (Enero – Diciembre 2015). La visión artificial y los campos de aplicación. *Tierra Infinita* (1), 98-108 https://doi.org/10.32645/26028131.76

[2] Beatriz, B. P. (2022, September 1). *Introducción a la visión artificial: procesos y aplicaciones*. https://docta.ucm.es/entities/publication/072ca3fb-540f-4dc9-8a16-0241cb5cd986

[3] *Index of /Manuals/National instruments*. (n.d.). https://neurophysics.ucsd.edu/Manuals/National%20Instruments/

[4] *Descargar Vision Development Module*. (2023, August 2). NI. https://www.ni.com/es/support/downloads/software-products/download.vision-development-module.html#521682

[5] *What is Computer Vision? | IBM*. (n.d.). https://www.ibm.com/es-es/topics/computer-vision

[6] *Output calibrated contour points from Vision Assistant - NI*. (2022, May 4). https://knowledge.ni.com/KnowledgeArticleDetails?id=kA03q000001E0IUCA0&l=es-MX

[7] *danimaya19/CONTROL-DE-COMPUTADORA-V-A-VISI-N-ARTIFICIAL-CON-LABVIEW*. (n.d.). GitHub. https://github.com/danimaya19/CONTROL-DE-COMPUTADORA-V-A-VISI-N-ARTIFICIAL-CON-LABVIEW